

Table of Contents

PyGuide	7
مقدمة.....	8
Chapter 1 (Programming Concepts).....	10
Chapter 2 (Introduction to Python).....	12
Chapter 3 (Basics).....	14
List.....	15
List Comprehension.....	18
Tuples.....	19
Strings.....	20
UserString.....	22
Dictionaries	26
Conditions.....	29
Loops.....	32
while.....	32
for.....	33
Don't break up with me.....	35
Our dance will continue.....	36
Chapter 4(Functions & Procedures).....	38
DRY: Don't Repeat Yourself.....	38
To *args or not to *args.....	44
To **kwargs or not to **kwargs	45
Going global.....	46
Lambda/Anonymous Functions.....	48
Iterable.....	50
Generator.....	51
Iterator.....	51
Chapter 5 (OOP).....	56
مقدمة.....	56
Fields.....	56
Methods.....	56
Properties.....	56
Encapsulation.....	57
__init__.....	57
get/set.....	59
properties.....	61
Magic methods.....	63
Operator Overloading.....	67
BTW, ..Enums	70
Chapter 6 (Inheritance).....	74
Multiple Inheritance.....	81
Chapter 7 (Exceptions, Errors).....	84
Chapter 8 (IO).....	89

Basic IO.....	89
os, os.path.....	92
File Pointer.....	96
Chapter 9 (Modules/Packages:Charging a Battery).....	101
Finding Nemo.....	102
First Module	102
Packages	104
Platform.....	106
Chapter 10 (Databases).....	110
Python/MySQL.....	110
PySQLite.....	116
ORMs.....	123
Storm.....	124
SQLObject.....	126
SQLAlchemy.....	128
Chapter 11 (Parsing Data).....	130
XMLing with Python.....	130
1- minidom.....	132
2- SAX	136
3- Expat undercover.....	141
HappyMapper.....	143
ElementTree.....	147
HTMLing with Python.....	148
Beautiful Soup.....	149
Parsing CSV Files.....	150
ConfigParser.....	153
Regex:.....	155
التفاعل مع برامج أخرى.....	156
Gimme usage!.....	157
GIMME optparser!.....	157
os.system.....	160
subprocess.....	161
Replacer.....	162
Chapter 12 (Networking).....	171
Simple Server.....	171
Simple Client.....	173
SocketServer.....	175
App with a BUG.....	182
FTPing	203
XMLRPC what?.....	205
DocXMLRPCServer.....	208
Quote of the Day.....	212
Chapter 13 (Python on the WEB)	218
Grok.....	218
Webpy.....	230
The Big Three.....	234

Pylons.....	234
TurboGears.....	242
Chapter 14 (Extending Python).....	251
Pure C.....	251
Swig.....	260
ctypes.....	261
Chapter 15 (Functional Python).....	266
functools	267
itertools.....	269
Chapter 16 (PyInotify).....	274
مقدمة.....	274
Watches.....	275
Events.....	275
Going on.....	275
PDir.....	276
Event.....	276
Chapter 17 (GUI).....	278
PyGTK.....	278
Gladizer.....	316
Rad with Glade (Gqamoos).....	319
HowTo GladizerguiTK ?.....	334
الخاتمة.....	340
لقد فعلتها!.....	340
الى اين ؟.....	340
مصادر اخرى.....	340
شكرا!.....	341
اهداء.....	341

PyGuide

مقدمة

لماذا هذا الكتاب ؟

السبب الرئيسى هو أنه كتاب **مجاني** ومتكامل ولا اعتقد انك بعد قرائته ستواجه صعوبات مع Python

ماذا تتوقع بعد قراءتك لهذا الكتاب؟

بعد قرائتك للكتاب ستكون قد علمت الكثير حول بايثون وكيفية توظيف اللغة بصورة جيدة وجدية، ستصبح متأقلم مع تطبيقات قواعد البيانات والشبكات -ربما تفكر فى كتاب خادم ويب خاص بيك؟- والتطبيقات الرسومية والكثير والكثير من طرق معالجة البيانات وغيرها الكثير.. ستكتشف هذا بنفسك!

المتطلبات

- جهاز عليه احد نظم GNU/Linux او Windows (يفضل XP or better)
- معرفة جيدة بكيفية إستخدام Text Editor رفة باشياء مثل
- معرفة Permissions, Groups, Users .. فى حال عدم المعرفة يفضل الإتجاه إلى [LinuxAC](#) ستجد العديد من المواضيع لتساعدك

الإصدارات القادمة والتحديثات

- ستجد اخر الإصدارات والتحديثات على Programming-Fr34ks.NET

المساهمين

- Ahmed Youssef

حقوق النسخ

مسموح بالنسخ والتوزيع وإحتمالية التعديل تحت بنود **GNU FDL** مع وجود الأقسام الثابتة "حقوق النسخ ، الإصدارات القادمة ،المساهمين)

الاقتراحات والانتقادات يرجى إرسالها على [guru.python\[at\]gmail.com](mailto:guru.python[at]gmail.com) كل الشعارات و العلامات التجارية و الرموز المستخدمة في هذا الكتاب تعود ملكيتها إلى أصحابها.

Chapter 1 (Programming Concepts)

مامعنى كلمة **Programming** ؟

هى بتعنى القدرة على التخاطب مع الكمبيوتر وتنفيذ الهدف (حل المشكلة) علي ارض الواقع .. الكمبيوتر لايفهم اى شئ سوا 0 و 1 وصعب على البشر تعلمها إن لم يكن مستحيلا فلجأ لخيارات بديلة وهى استخدام لغات البرمجة

مامعنى **Programming Language** ؟

بكل بساطة هى وسيلة للتخاطب مع الكمبيوتر .. ولكننا قلنا إن الكمبيوتر لايفهم اى شئ سوا ال 0 وال 1 ومستحيل على الإنسان تعلمها! .. إذا الحل هو استخدام لغات وسيطة .. على سبيل المثال واحد عربى وواحد فرنسى والعربى مش يفهم فرنسى ولا الفرنسى يفهم عربى .. فالحل هو إنهم يتكلمو إنجليزى مثلا... او يجيبو مترجم بين الإثنين مش كذا ؟
فهنا الحل إننا هنحصل على مترجم يترجم افكارنا للغة الكمبيوتر 0 و 1 ويقوم المترجم بنفس الدور بتحويل رد فعل الكمبيوتر الى لغتنا المفهومة (:
وهنا دور ال Programming Language انت هتتعلم اللغة وكيفية التعامل معاها عشان تقدر تفهم المترجم "المقدم من اللغة" ماتريده وهو يفهمه للكمبيوتر بدوره

ماهو ال **Source Code** ؟

بكل اختصار هو حلك لمسألة رياضيات وتفكيرك وإستنتاجاتك لما تكتبها فى ورقة ولكن هنا هو حلك لبرنامج مطلوب منك على ملف Text

مامعنى ال **Debugging** ؟

على فرض إنك بتحل مسألة رياضيات و إكتشفت خطأ فى طريقة حلك .. فإنت بتتبع المشكلة اللى حصلت وتشوف إزاي تصححها وهو دا معنى ال Debugging اى تصحيح الأخطاء (:

Compiled vs Interpreted

كثير منا إشتغلو على نظم Windows وكان ديمنا بيشفوف ملفات إمتدادها exe. فايه معنى ال exe ؟ معناها Executable او قابل للتنفيذ ..

فى لغات برمجة مثل ال C و Pascal بيتوافر الناتج النهائى بتاع برنامجك على صورة ملف exe وهو عبارة عن تعليماتك اللى إديتها للمترجم عشان يفهمها للكمبيوتر ولكن فى صورتها النهائية (الكلام اللى قاله المترجم لل كمبيوتر) فمستحيل على الإنسان إنه يقرأ الملف دا .. وهنا معنى ال compiled فهى ملف ال exe يشمل التعليمات اللى كتبتها ولكن بلغة الكمبيوتر وهو وحده القادر على فهمها

وإذا نظرنا من جانب آخر إلى لغات مثل Python, Perl هنجد إن الملف بيكون إمتداده .py او .pl ولكنك تقدر تفتحه فى اى Text Editor وتقرأه -لفهمه لازم تكون عارف اللغة- والملف دا هو ال Source Code بتاعك نفسه بدون اى تحويلات ولا شئ ولكن لتنفيذه بنستدعى ال المفسر (Interpreter) فى كل مرة بحيث إنه يقرأ ال Source Code ويبلغه للكمبيوتر ويتم التنفيذ

من مميزات ال Compiled Languages مثل ال C هى السرعة ومن القصور هو إنك لازم تعمل Compile لل Source Code بتاعك على النظام اللذى تريد ان تنفذ البرنامج عليه فبرنامج مكتوب على Windows محتاج يتعمله recompile على ال Linux وهكذا ..

من مميزات ال Interpreted Languages هى انك ال Source Code بتاعك القياسى لايتحتاج لعمل Recompile على مختلف النظم وال archS

ومن القصور البطء

ملحوظة:

لما نتكلم على كلمة البطء فى ال Interpreted Languages بيكون المقصود البطء بالنسبة لل Compiled Language وليس البطء للمستخدم لأنك مش هتلاحظ الفرق لأن البطء فى شئ لا يكاد يذكر

Chapter 2 (Introduction to Python)

Python هي لغة برمجة عامة لمعظم المجالات ان لم تكن جميعها، وهي High Level Programming Language اي انها قريبة جدا من لغة الإنسان "الإنجليزية" بدأت في عام 1989 على يد Guido Van Rossum وهو عالم هولندي Python تتميز ب

- 1- سهولة التعلم
- 2- وضوح الكود وسهولة صيانتة
- 3- ال Portability المحمولة -لأنها بتعمل على اكثر من Platform-
- 4- Python تعطيك قوة ال Scripting Languages وبكل تأكيد اخفاء مشاكل ادارة الذاكرة وتوابعها

عندك

- 5- Open Source: فيقوم على تطويرها الآلاف من المطورين
 - 6- Python تقدم تكامل مع ال .NET وال Java من خلال IronPython, Jython
 - 7- بتدعم اكثر من paradigm ك Functional Programming, OOP
 - 8- Python لغة ممتعة!
- جدير بالذكر ان Python حاليا هي de facto في عالم الأوبن سورس حيث تفوقت على Perl من حيث الشعبية وهي لغة العام حسب تقرير [Tiobe](#)

ملفات Python سيكون امتداها .py, .pyc او .pyo

- .py => ملف بايثون
- .pyc => ملف بايثون مترجم
- .pyo => ملف كائن لبايثون

Downloading/Installing Python

ادخل على [/http://python.org/download](http://python.org/download)

لمستخدمي UNIX/UNIX-Like: ف Python غالبا مرفقة مع توزيعتك.. في حال لا قم بتحميل ال Source واعمل Build

```
./configure  
make  
make install
```

اي خطأ قم بمراجعة ملف ال README او INSTALL

لمستخدمي Windows: قم بتحميل ملف ال .msi

تشغيل بايثون

تقدر تشغل ال المفسر (Interpreter) كجلسه تفاعلية بمعنى انك تمرر ليه statement معينه وهو ينفذها

لمستخدمى Linux: افتح ال Terminal او ال Console واكتب Python
لمستخدمى Windows :

```
~Start -> Run -> Cmd  
~cd /Python_PATH/  
python.exe
```

او قم بدعم ال PYTHON_PATH فى ال Environment Variables -متغيرات البيئة- فى ويندوز كالتالى

set path=%path%;[C:\Python25](#)

او التالى

*Right Click on My Computer -> Properties(1
Advanced Tab(2
Environment Variables(3
in Variables for (UserName): Click on PATH - > Edit(4
; Add [C:\Python25](#)(5
لاتنسى الفاصلة المنقوطة*

اكتب اى statement جمله مثلا 2+1 او "Hello, World" واضغط Enter

Chapter 3 (Basics)

هنتناول فى الفصل دا مفاهيم اساسية زى ال variables وال Loops وال Conditions وهى جزء حيوى من كل اللغات

اولا ماهو ال variable (متغير)؟
هو قيمة متغيرة فى برنامج وبيهمك انك تتابعها وتتابع اى تغير يتم فيها.
مثلا حساب بنكى فيه فلوس "قيمة" قابلة للتغيير فلازم نعبر عنها بمتغير وليكن "money" ونشوف قيمته خلال برنامجنا سواء بالزيادة او النقص

افتح ال IDLE .. هشتغل بنظام تفاعلى فى الأكواد الصغيرة..

```
>>>balance=90000 #an integer
```

نطبع القيمة كالتالى

```
>>> print balance
90000
```

نضيف ليه قيمة ولتكن 100 (زيادة)

```
>>> balance = balance + 100
>>> print balance
90100
```

ننقص منه اى قيمة وليكن 890 (نقص)

```
>>> balance = balance - 890
>>> balance
89210
```

على فرض اننا عندنا variable بإسم name بيعبّر عن قيمة معينة لل name دا
ملحوظة ال variable ماهو الا alias لقيمة متغيرة فى برنامجك

```
>>> name="ahmed" #a string
>>> print name
ahmed
```

وهكذا تقدر تعمل متغير يعبر عن العمر مثلا age

```
>>> age=50
>>> print age
50
```

List

طب جميل موضوع المتغيرات هيدفعنا نتكلم عن ال Lists او القوائم هى عبارة عن Enhanced Array لمبرمجى ال C اقرب مثال هو طلاب الفصل او زملاء العمل هل تعتقد ان اذا عندنا 30 طالب اننا نعمل شئ مشابه للتالى

```
student_1="Ahmed"
student_2="Wael"
student_3="Ayman"
student_4="Tina"
.....
student_30="Youssef"
```

هل تتوقع اننا ننشئ 30 متغير ل 30 طالب بالصورة دى ؟ ممكن بس يفضل تكتب استقالتك بعدها D:
بالطبع لأ.. وهنا تيجى اهمية ال List وهى باختصار ال Grouping لل Data Types المشتركة بمعنى ان كل دول students صح ؟

```
students=["Ahmed", "Wael", "Ayman", "Tina"]
>>> type(students)
<type 'list'>
```

لاحظ ان اول عنصر فى ال list ال index بتاعه هو 0 والثانى هو 1 والثالث هو 2

القاعدة العامة

```
idx=n-1
```

مثلا عندنا list كالتالى

```
>>>friends=["St0rM", "Squall", "Tina", "Ayman"]
```

لاحظ موضوع ال indexing لأنه بسيط وهو بنستخدمه للوصول لعنصر فى ترتيب معين فى اى sequence

```
>>> friends[0] # 1st
'St0rM'
>>> friends[2] # 3rd
'Tina'
```

ال List بتقدمنا العديد من الوظائف او التسهيلات فى التعامل على فرض ان عندنا List كالتالى

```
>>> students=["Ahmed", "Ayman", "Tina", "Wael"]
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael']
```

.append(element)

لإضافة عنصر بنستخدم ال append ميثود كالتالى

```
>>> students.append("Gina")
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina']
```

-- تقدر تضيف عنصر كالتالى

```
>>> students += ['Marian']
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina', 'Marian']
```

فى الواقع تقدر تضيف عدة عناصر

```
>>> students += ["Omar", "Waleed"]
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina', 'Marian', 'Omar', 'Waleed']
```

او بإستخدام extend

.extend(iterable)

بتقوم بدمج ال iterable على ال list الحالية كالتالى مثلا

```
>>> students.extend(["Omar", "Waleed"])
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina', 'Marian', 'Omar', 'Waleed']
```

.remove(value)

يتم حذف اول ظهور لل value فى ال list مباشرة كما فى المثال

```
>>> students.remove("Ayman")
>>> students
['Ahmed', 'Tina', 'Gina', 'Marian', 'Omar', 'Waleed']
```

للحذف بنقوم بتحديد ال index (الترتيب) الخاص بالعنصر ليتم حذفه كالتالى مثلا

```
>>> del students[3]
>>> students
['Ahmed', 'Ayman', 'Tina', 'Gina', 'Marian', 'Omar', 'Waleed']
```

.insert(idx, item)

لإضافة عنصر معين فى ترتيب معين بنستخدم ال insert ميثود كالتالى مثلا هيثم إضافة Ayman فى الترتيب ال 3 (العنصر ال 4)

```
>>> students.insert(3, "Ayman")
```



```
>>> students
['Ahmed', 'Tina', 'Gina', 'Ayman', 'Marian', 'Omar', 'Waleed']
```

```
.pop(idx=-1)
```

يتقوم بحذف + إعادة عنصر في ال list بإستخدام ال idx وفي حال عدم توفيره ه يتم التطبيق على آخر عنصر بال list

```
>>> popped=students.pop()
>>> popped
'Waleed'
>>> popped=students.pop(2)
>>> popped
'Tina'
>>> students
['Ahmed', 'Ayman', 'Wael', 'Omar']
```

```
.reverse()
```

يتم عكس الترتيب من الآخر للأول كالتالى

```
>>> students
['Ahmed', 'Ayman', 'Wael', 'Omar']
>>> students.reverse()
>>> students
['Omar', 'Wael', 'Ayman', 'Ahmed']
```

```
.sort()
```

يتقوم بترتيب العناصر

```
>>> students.sort()
>>> students
['Ahmed', 'Ayman', 'Omar', 'Wael']
```

```
.index(value)
```

الحصول على الترتيب الخاص باول ظهور للعنصر

```
>>> students.index("Ayman")
1
```

```
.count(value)
```

للحصول على عدد مرات ظهور عنصر معين في ال list

```
>>> students.append("Ayman")
>>> students
```

```
['Ahmed', 'Ayman', 'Omar', 'Wael', 'Ayman']  
>>> students.count("Ayman")  
2
```

List Comprehension

إذا أخذت كورس رياضيات من قبل فربما تكون واجهت ال List Comprehension
$$S = \{ 2 \cdot x \mid x \in \mathbb{N}, x \leq 10 \}$$

هنا تنقسم ال LC الى جزئين جزء ما قبل ال بايب -انبوبة- | و جزء ما بعدها
ما قبلها يسمى function يتم تطبيقها على كل x
وما بعدها يسمى قائمة المدخلات وبعض الشروط

```
>>> [number*3 for number in range(20)]  
[0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57]
```

فيما سبق قمنا بتطبيق ال LC فى بايثون حيث قمنا بتحديد ال function (وهى ضرب كل عدد فى 3)
لكل عدد ينتمى الى الفترة من صفر ل 20

```
Prelude> [number*3 | number <- [1..20]]  
[3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,48,51,54,57,60]
```

نفس الشئ بالنسبة لكود ال haskell اذا حيث تقوم بتطبيق العملية بحذافيرها ما قبل وما بعد البايب
وربما تقوم بتحديد شرط ما (هنا لكل الأعداد الزوجية)

```
>>> [number*3 for number in range(20) if number%2==0]  
[0, 6, 12, 18, 24, 30, 36, 42, 48, 54]
```

كود ال haskell

```
Prelude> [number*3 | number <- [1..20], even number]  
[6,12,18,24,30,36,42,48,54,60]
```

طبعاً تقدر تستخدم الطرق العادية للفلتر ولكن دى ال on the fly solution

Tuples

1

ال Tuple هي Container -حاوية- اخرى زى ال List ولكن الفرق هي إنها غير قابلة للتغيير..

التعريف بيتضع العناصر بين ()

```
>>> t=(1, 2, 3, 4, 5)
>>> t
(1, 2, 3, 4, 5)
```

او تقدر تعرفها كالتالى مثلا

```
>>> t=1, 2, 3, 4, 5
>>> t
(1, 2, 3, 4, 5)
```

len(tup)

هتقوم بإعادة عدد عناصر ال tup (نفس السلوك مع اي كونتينر)
احنا قلنا انها غير قابلة للتغيير يعنى لو حاولنا نعدل اي عنصر المفروض يحصل مشكلة

```
>>> t=1, 2, 3, 4, 5
>>>t[0]=9 #Try to set the first element to 9
```

هنا حاولنا نخلي العنصر اللي فى الترتيب الأول قيمته تساوى 9
رد بايثون هيكون مشابه للتالى

```
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    t[0]=9 #Try to set the first element to 9
TypeError: 'tuple' object does not support item assignment
```

Strings

بايثون بتقديمك نوع من البيانات ليغير عن ال String ولكن اولا ماهو ال String ؟
هو بكل بساطة مجموعة من الحروف

لغات مثل سى لاتعترف بال String ولكن تعتبره مصفوفة (او قائمة) من الحروف

انشاء string

تقدر تعرفه كالتالى

```
>>> astring="Hello, World!"
```

هنا تم انشاء متغير باسم astring وتم تخزين القيمة Hello, World داخله
لاحظ لإنشاء أى سترينج استخدم علامات التنصيص " " او ' ' او "" "" هنشخ الفرق خلال الكتاب
زى مالايسست بتقديمنا خدمات تسهل علينا التعامل فكذلك ال strings
اولا اسم الداتاايب هو str (اختصار string)

```
>>> type(astring)  
<type 'str'>
```

len(string)

تقوم بحساب عدد الحروف الموجودة بال string

```
>>> len("Hello, World")  
12
```

.capitalize()

تقوم بإعادة كائن جديد تحول فيه اول حرف إلى حرف uppercase

```
>>> "hello".capitalize()  
'Hello'
```

.count(sub)

حساب عدد مرات تكرار مقطع معين

```
>>> "Hello".count('l')  
2
```

.lower()

اعادة كائن بيه كل الحروف lowercase

```
>>> "HELIO".lower()  
'hello'
```

.upper()

اعادة كائن بيه كل الحروف uppercase

```
>>> "helLo".upper()
'HELLO'
```

.swapcase()

بتقوم بإعادة كائن بعكس حالة الأحرف

```
>>> "hElLo".swapcase()
'HeLlO'
```

.title()

بتعيد كائن على صورة title -عنوان-

```
>>> "hello, world".title()
'Hello, World'
```

.startswith(sub)

بتختبر هل السترينج يبدأ بمقطع معين او لا

```
>>> "hello".startswith("he")
True
```

.endswith(sub)

بتختبر هل السترينج ينتهى بمقطع معين او لا

```
>>> "hello".endswith('lo')
True
```

سؤال: إيه اللي يخلينى استخدم tuple مكان list ؟ مع إنها بتقدملى امكانيات اقل؟
لأنك غالبا مش هتستخدمها ك list ولكن هتحتاج ميزة عدم التعديل عليها

.find(sub)

بتقوم بإعادة ال index الخاص بأول ظهور لمقطع معين

```
>>> s="Hello, World"
>>> s.find('W')
7
```

.strip([chars])

بتقوم بحذف ال chars من السترينج وفى حال عدم تحديدها هيعتبر انها المسافات

.lstrip([chars])

مثل الأولى ولكن من على اليسار فقط

rstrip([chars])

مثل الأولى ولكن من اليمين فقط

.isalpha()

بتختبر هل السترينج مكون من حروف الأبجدية أو لا

.isalnum()

بتختبر هل السترينج مكون من حروف من الأبجدية أو ارقام أو لا

.isdigit()

بتختبر هل الحروف المكونة للسترينج عبارة عن ارقام أو لا

.isupper()

بتختبر هل السترينج فى حالة uppercase أو لا

.islower()

بتختبر اذا كان السترينج فى حالة lowercase أو لا

.isspace()

بتختبر هل السترينج دا مسافة

.istitle()

بتختبر هل السترينج دا title أو لا

.splitlines()

تقوم بإعادة الأسطر فى قائمة List

ويوجد العديد بكل تأكيد تقدر تطلع على باقى الميثودز فى الوثائق الخاصة بالبيثون

ملحوظة هامة ال string مش mutable! -غير قابلة للتغيير-
لتستخدم mutable strings تابع قسم ال UserString

UserString

المشكلة:

```
somestring="hola"
print somestring, " ", id(somestring)

somestring += " guyZ"
print somestring, " ", id(somestring)
```

الناتج

```
hola , 3084138464
hola guyZ , 3084161896
```

لاحظ اختلاف ال id -المعرف- الخاصة بال somestring بعد اضافة مقطع ليه ، والسبب ان تم انشاء كائن جديد من ال str class وتم دمج القيمة القديمة + المقطع الجديد

الحل UserString Module

كما نعلم ان فى python ال Strings are immutable زى العديد من اللغات زى etc .. Java/C#
بمعنى ان اى تعديل هيتم على String مثل ال Concatenation -الدمج- يمحذف ال Variable -متغير- ويعمل Variable جديد كالتالى

```
>>> s="I'm a string"
>>> s
"I'm a string"
>>> id(s) # get the location
13511240
>>> s += ", and you ?"
>>> s
"I'm a string, and you ?"
>>> id(s) #get the location now!
13522272
```

لاحظ بعد ال Concatenation مع ال string الجديد إتغير ال location بتاعه .. لكن اللى حصل بالفعل هو ان اتعمل شئ مشابه للتالى

```
>>> s="I'm a string"
>>> del s
>>> s="I'm a string, and you?"
```

طب جميل ولكن على فرض إنك تريد تعمل mutable string او فى بعض اللغات إسمه StringBuffer/StringBuilder ؟

الحل هو إنك تستخدم class من UserString Module ودى Module -وحدة- موجودة عندك فى PythonPath/Lib/UserString.py

استدعى ال Module كالتالى

```
>>> import UserString as us
```

استخدمت as us لتعبر عن alias لل UserString module

ال Module بتتكون من 2 Classes هما UserString,MutableString
ال UserString هو ال base class و immutable تقدر تستخدمه بصورة مشابه لل str type
ال MutableString هو sub class من ال UserString و mutable

ملاحظة : ال MutableString معمول فيه override -اعادة تعريف- ل hash function لأنه طالما Mutable بيتقة unhashable ! -لاتقلق فى حال عدم فهمك للجملة السابقة-

```
def __hash__(self):  
    raise TypeError, "unhashable type (it is mutable)"
```

ال MutableString فيه method بإسم immutable ودى بتدى ل return immutable string !

```
>>> m_string=us.MutableString("Hello, ")
```

بنعمل Object بإسم m_string

تحديد ال location

```
>>> id(m_string)  
13511136
```

نعمل ref -مرجع- ليه

```
>>> ref_m_string=m_string
```

تحديد ال id ل ref هنلقية نفس ال id الخاص ب m_string

```
>>> id(ref_m_string)  
13511136
```

عمل دمج

```
>>> m_string += "CRUEL WORLD!"
```

نحدد ال id تانى هنلقية مازال هو هو

```
>>> id(m_string)  
13511136
```

ال id الخاص بال ref مازال زى ماهو

```
>>> ref_m_string  
'Hello, CRUEL WORLD!'  
>>> id(ref_m_string)  
13511136
```

عمل Object ولكن immutable يعنى ال id بتاعه هيتغير فى اى تغيير هيتم عليه

```
>>> im_string=m_string.immutable() # Return immutable string!  
>>> im_string
```



```
'Hello, CRUEL WORLD!'
```

تحديد ال id

```
>>> id(im_string)
13532856
```

عمل ref ليه وتحديد ال id

```
>>> ref_im_string=im_string
>>> id(ref_im_string)
13532856
```

عملية الدمج

```
>>> im_string += " blah blah blah"
```

تحديد ال id هنلاقيه إنه إتغير

```
>>> id(im_string)
13553416
```

ال ref مازال زى ماهو مش تم عليه اى تغيير لأنه بيشير لمكان Object تم الإستغناء عنه واصبح اسم im_string بيشير ل 'Hello, CRUEL WORLD! blah blah blah'

```
>>> ref_im_string
'Hello, CRUEL WORLD!'
>>> id(ref_im_string)
13532856
>>> im_string
'Hello, CRUEL WORLD! blah blah blah'
```

يفضل اعادة قراءة الفصل بعد قراءة جزئية الكائنات
ابحث عن موديلز مشابهه ل UserString

Dictionaries

القواميس: هي نوع من البيانات ونستخدمه في تخزين البيانات بصورة مفتاح وقيمة (key, value) كالقاموس بالفعل

انشائه

```
>>> d={}
```

اسم -نوع البيانات- هو dict

```
>>> type(d)
<type 'dict'>
```

```
>>> d['Name']='Ahmed'
>>> d['Age']=19
>>> d['Sex']='m'
```

هنا خزننا في القاموس الخاص بنا 3 keys وهما 'Name' , 'Age' , 'Sex'

.keys()

للحصول على ال Keys او الكلمات الدليلية في صورة ليست

```
>>> d.keys()
['Age', 'Name', 'Sex']
```

.values()

للحصول على ال Values في صورة ليست

.get(key)

بتعيد لك ال value الخاصة ب key معين

```
>>> d.get('Name')
'Ahmed'
```

في حال عدم وجوده هيثم إعادة None

```
>>> print d.get('Team')
None
```

Indexing

تقدر تحصل على القيمة الخاصة ب key معين من خلال ال Indexing -الفهرسة- كالتالى مثلا

```
>>> d['Name']  
'Ahmed'
```

او تقدر تضيف Key -مفتاح- جديد مثلا

```
>>> d['Lang']='Ruby'
```

او تقدر تعدل على قيمة موجودة

```
>>> d['Name']='Youssef'
```

.pop(key)

بتقوم بحذف ال Key وال Value الخاصة به من القاموس وبتعيدك ال value

.update(d)

بتقوم بعمل تحديث للقاموس ببيانات قاموس d

```
d.update({'Lang':'Python', 'Singer':'Delta'})  
>>> d  
{'Lang': 'Python', 'Age': 19, 'Singer': 'Delta', 'Name': 'Youssef'}
```

.has_key(key)

بتختبر وجود key معين فى القاموس

```
>>> d.has_key('Lang')  
True
```

.items()

بتعمل list مكونة من tuples بتشمل ال key, value فى صورة زوج

```
>>> print d.items()  
[('Lang', 'Ruby'), ('Singer', 'Delta'), ('Name', 'Youssef'), ('Country', 'EG'), ('Age', 19), ('Sex', 'm')]
```

.iteritems()

بتستخدم غالبا فى ال Iterations (سنتعرض ليها لاحقا)

```
for key, val in d.iteritems():  
    print key, " => ", val
```

```
Lang => Ruby
```

```
Singer => Delta  
Name => Youssef  
Country => EG  
Age => 19  
Sex => m
```

مثال اخر

```
>>> i=d.iteritems()  
>>> i.next()  
( 'Lang', 'Ruby' )  
>>> i.next()  
( 'Singer', 'Delta' )
```

Conditions

حياتنا مبنية على الإحتمالات والبرمجة مش خارج نطاقها.. مثلا اذا الخدمات اللى بتطلب تسجيل الدخول منك مبنية على احتمال "هل انت او لا"

الصورة العامة

```
if condition then  
    if_suite
```

مثلا "مع فارق التشفير وقواعد البيانات فى الإستخدام"

```
>>> if name=="ahmed" and password=="123456":  
    print "Welcome ahmed"
```

Welcome ahmed

لاحظ هنا اختبرنا هل الإسم قيمته مساوية "==" ل احمد **و** الباسورد قيمته "123456" فإذا الناتج True يتنفذ البلوك اللى بعدها وهو print Welcome ahmed فى حال ال ناتج الأساسى للشرط مش True يعنى False مش هيتنفذ حاجة. تمام.. طب اذا حيينا نعالج موضوع ان الشرط يكون False ؟ بمعنى اننا لو غيرنا مثلا الباسورد لأى قيمة مخالفة ل 123456 ال condition قيمته هتكون False وفى الحالة دى مش هيحصل شئ.. ولكن نريد ان نعرف اذا ال condition كان False مثلا؟

تابع المثال التالى وهو إستخدام if, else
الصورة العامة

```
if condition then  
    if_suite  
else then  
    else_suite
```

لاحظ المثال التالى

```
>>> name="ayman"  
>>> password=147859  
>>> if name=="ahmed" and password=="123456":  
    print "Welcome ahmed" #if_suite  
else:  
    print "Welcome, Who R U?" #else_suite
```

Welcome, Who R U?

حسنًا في حال وجود عدة احتمالات قد تكون سليمة أو لا نستخدم if, else if, else

```
if condition then
    if_suite
else_if condition then
    else if suite
else_if condition then
    else_if suite
else then
    else_suite
```

تابع المثال التالي

```
>>> if name=="ahmed" and password=="123456":
    print "Welcome ahmed" #if_suite
elif name=="tina" and password=="36987456":
    print "Welcome tina" #elif_suite
elif name=="ayman" and password==147859:
    print "Welcome, Ayman" #elif_suite
else:
    print "Who R U?"
```

Welcome, Ayman

الأول تم اختبار البلوك دا

if name=="ahmed" and password=="123456":

ولكنه مش True فيتعمل Escape للى بعده وهو

elif name=="tina" and password=="36987456":

ولكنه مش True برودو فيتعمل Escape للى بعده وهو

elif name=="ayman" and password==147859:

والبلوك دا قيمته True فيتنفذ البلوك تبعه وهو

print "Welcome, Ayman" #elif_suite

ملحوظة:

نستخدم == لإختبار عملية التساوي
نستخدم = لعملية الإسناد

if name=="ahmed" and password=="123456":

and = و (يجب تحقق جميع اقسام الشرط مثلا)

if name=="ahmed" or name=="ayman":

or = او (احد قسم الشرط يكفى لتنفيذ محتوى if)

if not name=="ahmed" :

not = ليس (لنفي الشرط)

!= تستخدم فى اختبار عدم المساواة

if name != "mahmoud"

Loops

while

مثل ال conditions معظم حياتنا مبنية على التكرار.. مثل كل يوم تصحى الصبح وتأخذ دش وتغفر وتنزل شغلك وهكذا لحد الأجازة مثلا تنام للمغرب D:

طالما انت فى دراسة او شغل "مش فى اجازة"
اصحى الساعة 6
خد دش
افطر
انزل شغلك

الصيغة العامة

```
while condition do  
    while_suite
```

تعالى نجرب ابسط لوب ممكن

```
>>> i=0  
>>> while i<10:  
    print "i: ", i  
    i += 1
```

```
i: 0  
i: 1  
i: 2  
i: 3  
i: 4  
i: 5  
i: 6  
i: 7  
i: 8  
i: 9
```

لاحظ ان الكود بيتنفذ كالتالى
الشرط هو ان ا تكون اقل من 10 وطالما الشرط دا حقيقى (صحيح) هيتنفذ البلوك التابع لل loop -الحلقة او الدوارة-

```
print "i: ", i
```



```
i += 1
```

السبب في أننا نزنود ال `l` أننا نخلي ال `l` توصل ل 10 بحيث ان ال `loop` تقف وإلا الشرط هيكون `True` للأبد.

for

```
Foreach element in container do  
    for_suite
```

نطبقها على Containers -الحاويات- مثل ال `string` او `list` او `tuple` مثلا

```
>>> string="Hello, World!"  
>>> for char in string:  
    print char
```

```
H  
e  
l  
l  
o  
,  
  
W  
o  
r  
l  
d  
!
```

معناها كالتالي : لكل (حرف) في ال `string` اطبع (الحرف) دا

على فرض ان عندنا `list` كالتالي بإسم `students`

```
>>> students=["Ahmed", "Tina", "StOrM", "Salma"]  
>>> for student in students:  
    print student
```

```
Ahmed  
Tina  
StOrM  
Salma
```

المعنى هنا: لكل Element -عنصر- او student موجود فى ال Container او ال students list اطبع ال عنصر دا
ملحوظة:

لطباعة العنصر الموجود فى ال Container بدون اضافة سطر جديد ضيف كومة لل print كالتالى

```
>>> string="Hello, World!"
>>> for char in string:
    print char,

Hello, World!
```

raw_input

زى ماشفنا print وعرفنا انها مختصة بالطباعة هنشوف المسئول عن الإدخال وهنا raw_input/input
الصيغة العامة

raw_input(prompt)

حيث ان ال prompt هى الرسالة اللى هتظهر للمستخدم
مثال

```
>>> name=raw_input("Enter your name: ")
Enter your name: ahmed
>>> print "Hola, ", name
Hola, ahmed
```

س: مالفرق بين raw_input, input ؟
استخدامك ل input بيساوى بالظبط التالى

eval(raw_input(prompt))

مش فى احسن من التجربة العملية تابع المثال التالى

```
>>> val=raw_input("Enter: ")
Enter: 2+13+541
>>> print val
2+13+541
```

لكن مع استخدامنا ل input هيتعمل eval -اختصار ل evaluate- للمدخلات ك Python Expression كالتالى

```
>>> val=input("Enter: ")
Enter: 2+13+541
```

```
>>> print val
556
```

تم تحقيق ال 2+13+541 من خلال python واعادة الناتج ليك

eval(expression)

```
>>> eval("1+2")
3
```

eval يتاخذ expression وتحاول اعادة الناتج ليك اذا كان فيه معنى

Don't break up with me

بنستخدم break للخروج من حلقة عند استيفاء شرط معين (مثلا تمت قراءة كل البيانات من ملف فلا داعي لمحاولة القراءة او تم قراءة 100 رقم فردى فلا داعي للإستمرار) لدينا كلمة مثل "hellopython" نريد ان نعرف موقع حرف ال t فيها فالفكرة ان نقوم بعمل حلقة على الأحرف ونختبر ماذا كان t او لا.. واذا كان t نقم بتخزين قيم المركز الحالى ونخرج من الحلقة "لعدم احتياجنا لها بعد الآن"

```
word="hellopython"

whereist=0
count = 0
while count < len(word):

    if word[count]=='t':
        whereist=count
        break
    #no need to keep going on
    count += 1
```

الناتج

```
striky@striky-desktop:~/workspace/pytut/src$ python tstbreak.py
Now count is 1
Now count is 2
Now count is 3
Now count is 4
Now count is 5
```

```
Now count is 6
Now count is 7
t was found at word[7]
```

Our dance will continue

تستخدم continue للهروب من الحلقة الحالية (ربما لعدم استيفاء عنصر الشروط المطلوبة للعمل عليه) واستكمالها على العنصر الذي يليه مثلاً

```
tstvars=['123mx', 'hello', 'acc', '9']

for var in tstvars:
    if var[0].isdigit():
        continue #no work will be done on this item, maybe the next?
    else:
        print var, " => ", "is valid."
```

هنا نختبر كل عنصر من عناصر tstvars ما إذا كان يصلح ان يكون اسم متغير في بايثون فنقوم بعمل حلقة على العناصر

```
for var in tstvars:
```

ونختبر ما إذا كان يبدأ برقم (احد الشروط عدم بدا تسمية المتغيرات في بايثون برقم) فإذا كان رقم نهرب من الحلقة الحالية ونستكمل على العنصر التالي في القائمة tstvar

```
if var[0].isdigit():
    continue #no work will be done on this item, maybe the next?
```

تدريب:

اكتب برنامج لتسجيل الدخول بالبيانات التالية

```
user_name="Ahmed"
user_pass="123456"
```

ل 3 محاولات وفي حالة الفشل تطلع رسالة ب

!Account Suspended

*أستخدم raw_input للحصول على الداتا من المستخدم

Chapter 4(Functions & Procedures)

ال Functions واستخدامها اترسمى عليها paradigm كامل Functional Programming -الوظيفية-او
Procedural Programming -الإجرائية- وكان/مازال شائع جدا للآن
لغات مثل السي وباسكال بتعتمد على ال Functions كحجر اساس

ايه هي ال Function او ال Procedure ؟
هي بلوك (قسم) من الأكواد اكتب لإمكانية استخدامه اكثر من مرة

مثلا نريد ان نطبع رسالة كالتالى

Python rocks!

لأكثر من مرة فى برنامجنا لسبب ما.. هل يعقل انى اكتب

```
print "Python rocks!"
```

فى كل مرة ؟ ولنفرض انى كتبها 20 مرة هل يعقل انى اذا حبيت اعدل التعبير بدل Python rocks ل Perl
rocks انى اعدل فيه 20 مرة؟
من هنا جت اهمية ال Functions وهى تستخدم لتوفير وقت ومجهود وتطبيق مبدأ DRY

DRY: Don't Repeat Yourself

كل اللى عليك هو انك تحط الرسالة فى function
حيث بنبدأ التعريف بالكلمة المفتاحية **def** وبعدها اسم الدالة rocks وبعدها قائمة المعاملات
الشرح الآن من خلال ال Python Shell بطريقة تفاعلية وبعد كذا فى جزئية انشاء الوحدات نتعرض لعمل
السكربت وكيفية استدعاءه

```
>>> def rocks():  
    print "Python rocks!"
```

وتستدعيها كل ماتحب

```
>>> rocks()  
Python rocks!  
>>> rocks()  
Python rocks!  
>>> rocks()  
Python rocks!
```

لاحظ انك اذا حبيت تعدل كلمة Python ل Perl مثلا مش هتحتاج تعدل فى اى شئ غير فى ال rocks
function فقط تحولها للتالى

```
>>> def rocks():  
    print "Perl rocks!"
```

لاحظت الفرق ؟ توفير وقت ومجهود ومش كررت نفسك فى مليون سطر
طب تمام لكن فيه مشكلة لحد الوقتى وهى اننا اضطررنا نعدل جوا ال function ونستبدل كلمة مكان كلمة

وهكذا فنريد ان نخليها ابسط فى الإستخدام بحيث انها تطيع اللى إحنا عايزنها تطيعه (ال دالة فيها متغير معين)
فالحل هو اننا نعيد تعريفها كالتالى مثلا

```
>>> def rocks(thing):  
    print thing, "rocks!"
```

كدا هيتم طباعة ال argument اللى هيتمرر + كلمة rocks! كالتالى

```
>>> rocks("Python")  
Python rocks!  
>>> rocks("Perl")  
Perl rocks!
```

جميل نفس الفكرة نريد ان نعمل function تجمع 3+2 وتطيع الناتج لينا

```
>>> def add():  
    print 2+3
```

استخدمها كالتالى

```
>>> add()  
5
```

لكن هل لاحظت شئ ؟ انها مقيدة بمعنى انى مش قادر استخدم غير 3+2 فقط طب افرض انا عايز احدد ارقام
من عندى ايه الحل ؟
هممم نفس فكرة المثال اللى قبله انك تمرر الأرقام اللى تعجبك ك arguments لل add function ودا هيتم
الأول
1- انك تعيد تعريف ال function كالتالى مثلا

```
>>> def add(first, second):  
    print first + second
```

2- تستخدمها

```
>>> add(2, 3)  
5  
>>> add(3, 7)  
10
```

ارسال ال arguments للدالة

```
def printArgs(first, second, third):  
    print "First: ", first  
    print "Second: ", second  
    print "Third: ", third
```

لاحظ لإستخدام الدالة هنستدعيها كالتالى

```
printArgs("Hello", "Cruel", "World")
```

والناتج

```
First: Hello
Second: Cruel
Third: World
```

ولكن على فرض انى عايز احدد قيم اساسية او حتى ادخل ال arguments -المعاملات- بطريقة عشوائية؟! بكل بساطة تقدر تستدعيها كدا

```
printArgs(third="World", second="Curel", first="Hello")
```

بحيث انك تحدد قيمة كل عنصر بإستخدام الإسناد (لأنهم متغيرات واضحة للدالة ;) ولتحديد قيم افتراضية ؟ مثلا عايز ادخل قيمتين او قيمة واحدة او حتى مش ادخل اى قيمة!!!؟؟ تقدر تحدد دا من خلال تعريف الدالة نفسها مثلا كالتالى

```
def printArgs(first="Hello", second="Cruel", third="World"):
    print "First: ", first
    print "Second: ", second
    print "Third: ", third
    print "-"*20
```

```
printArgs("Bye") #Changes the first..
printArgs(second="") #only the second is set to ""
printArgs()
```

والناتج

```
First: Bye
Second: Cruel
Third: World
-----
First: Hello
Second:
Third: World
-----
First: Hello
Second: Cruel
Third: World
-----
```

وهكذا.. دا مفيد فى موضوع ال overloading -التحميل الزائد- لل function (بحيث ان يتم تنفيذ اكثر من وظيفه لنفس ال function اعتمادا على ال arguments المرسله) تابع range لحد الآن ال Functions غير مفيدة لأنها مش بتعيد قيمة يعنى مش تقدر تستفيد منها فى برنامجك انك تعمل كالتالى مثلا


```
>>> val=add(2, 3)
5
>>> print val
None
```

*ايه دا ؟ انا كنت متوقع ان val هتكون قيمتها 5!
دا هيتم فى حالة واحدة ان الطرف اليمين من ال expression تكون قيمته 5 لكن 3, 2 add() قيمته None لأنه بيعمل print لمجموع الرقمين لكن مش بيعمل بيهم return

*ايه None ؟

None=null=nil=Nothing

جميل طب ازاي اخلى قيمة ال Function تساوى مجموع الرقمين ؟
بسيطة اعمل return بالمجموع!
كالتالى مثلا

```
>>> def add(first, second):
    return first+second

>>> val=add(2, 3)
>>> print val
5
```

هنا return عبرت عن قيمة ال Function

كثير للأسف مش يعرف الفرق بين ال Functions وال Procedures
على كل حال اعتبرها كالتالى
ال Procedure هو اى Function ليس لها return

--لمبرمجى السى++ والجافا اى Function ال return لها void بيقع اسمها Procedure

بايثون بتقدم ليك العديد من ال Functions الجاهزة مثل

raw_input(prompt)

للحصول على الداتا من المستخدم

input(prompt)

زى ماقلنا هى بتستدعى التالى

eval(raw_input(prompt))

eval(expression)

بتحقق قيمة ال expression

abs(number)

بتعيد ليك ال Absolute value -القيمة المطلقة- وهى العدد بدون اشارة

```
>>> abs(10)
10
>>> abs(-10)
10
```

*تدريب:

اكتب Function بإسم getABS وتأخذ argument واحدة بإسم number ومشابهه ل abs

max(iterable)

هى function تأخذ container فى الغالب -إى شئ ممكن يطبق عليه foreach- وتعيد اكبر قيمة فيه كالتالى
مثلا

```
>>> max([3, 4, 5, 6])
6
>>> max("Ahmed")
'm'
```

min(iterable)

هى العكس من max وهى بتعيد اصغر قيمة

```
>>> min("Ahmed")
'A'
>>> min([3, 4, 5, 6])
3
```

ملحوظة: اصغر قيمة للأحرف يتم بناء على قيمتها فى ASCII وتقدر تحصل عليها من خلال ord
وللحصول على الحرف من خلال القيمة بنستخدم chr

ord(char)

```
>>> ord("A")
65
>>> ord("a")
97
>>> chr(65)
'A'
>>> chr(97)
'a'
```

sum(seq)

بتستخدم للحصول على مجموع sequence ما كالتالى

```
>>> sum([1, 2, 3, 4, 5])
15
```

*تدريب اكتب function مشابهه ل sum وإسمه getSum وتأخذ sequence ك argument

oct(num)

بتعيد القيمة من النظام الثمانى لل num

```
>>> oct(15)
'017'
```

hex(num)

بتعيد القيمة من النظام الست عشرى لل num

```
>>> hex(15)
'0xf'
```

len(object)

فى الواقع len استخدامها يختلف حسب نوع ال argument اللى هتتمرر ليها يعنى مثلا إذا كان string هيتم إعادة عدد الحروف وإذا كانت list هيتم إعادة عدد العناصر المكونة ليها وهكذا

```
>>> len("Ahmed")
5
>>> len([1, 2, 3, 4, 5, 6])
6
```

هنتعلم قريب ازاي نحدد الطريقة اللى هنتعامل بيها len مع ال objects بتاعتنا D:

round(f_num, digits)

هى function بتعمل تقريب ل f_num بعدد معين من الأرقام بيساوى digits كالتالى مثلا

```
>>> round(2678.367789)
2678.0
>>> round(2678.367789, 4)
2678.3678
```

copyright()

```
>>> copyright()
Copyright (c) 2001-2008 Python Software Foundation.
All Rights Reserved.

Copyright (c) 2000 BeOpen.com.
All Rights Reserved.

Copyright (c) 1995-2001 Corporation for National Research Initiatives.
All Rights Reserved.
```

Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
All Rights Reserved.

بتعرضلك ل copyright الخاص بيثون

credits()

بتعرضلك ال credits

```
>>> credits()
Thanks to CWI, CNRI, BeOpen.com, Zope Corporation and a cast of thousands
for supporting Python development. See www.python.org for more information.
```

range(end)

بتعيد ليك list من 0 لحد end بزيادة قيمتها 1

```
>>> range(10) #0 to 10
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

range(start, end)

بتعيد ليك List من start لحد end بزيادة قيمتها 1

```
>>> range(1, 5) #1 to 5
[1, 2, 3, 4]
```

range(start, end, step)

بتعيد ليك list من start لحد end بزيادة مقدارها step

```
>>> range(1, 20, 3) #1 to 20 with step=3
[1, 4, 7, 10, 13, 16, 19]
```

*تدريب اكتب function تقوم بعرض جدول ال ASCII

To *args or not to *args

عنوان عجيب
مثال على دالة sum

```
def mysum(alist):
    total=0
    for i in alist:
        total += i

    return total
```

لإستخدامها هنمرر list من الأرقام الي الدالة ك argument كالتالى

```
print mysum([1, 2, 3, 4, 5])
```

مثال بإستخدام *args - تعنى اننا سنمرر عدد غير محدد من المعاملات- فيتم تخزين كل المعاملات التى سيتم تمريرها فى المتغير args وهو قائمة

```
def newsum(*args):
    total=0
    for i in args:
        total +=i
    return total
```

لإستخدامها: هنمرر الأرقام ك arguments للدالة كالتالى

```
print newsum(1, 2, 3, 4, 5)
```

انشاء دالة مشابهة ل printf

```
def printf(fmt, *args):
    print fmt%args

printf("Name: %s, Age: %d", "Ahmed", 20)
#converted to print "Name %s, Age: %d"%("Ahmed", 20)
```

لاحظ ان هيتم التحويل *args إلى دالة print لإستبدال ال (%s, %d) place holders) بالقيم المدخلة *تدريب "اكتب sprintf -دالة تقوم بتجهيز سلسلة نصية واعادتها لمتغير بدلا من طباعتها- بإستخدام بايثون

To **kwargs or not to **kwargs

بتمرر مفاتيح keys -تقدر تقول named arguments معاملات مسماه - وقيمها للدالة مثال

```
def newprintf(fmt, *args, **kwargs):
    print "fmt: ", fmt
    print "*args: ", args
    print "***kwargs: ", kwargs
    #do something usefu
    if kwargs.has_key("verbose"):
        print "Verbose..."
```

```
newprintf("This is some FMT", 1, 2, 3, 4, 5, 6, verbose=True, cleanup=True, use_ssl=False)
```

الناتج

```
fmt: This is some FMT
*args: (1, 2, 3, 4, 5, 6)
**kwargs: {'use_ssl': False, 'cleanup': True, 'verbose': True}
Verbose...
```

Going global

انك توثق الدالة وتعمل اية ويتاخذ معاملات ايه شئ ممتاز فى اى برنامج لان اهمية التوثيق من اهمية الكود

```
def newprintf(fmt, *args, **kwargs):
    """Simple function to learn out howto use *, ** trick in functions."""
    print "fmt: ", fmt
    print "*args: ", args
    print "**kwargs: ", kwargs
    #do something usefu
    if kwargs.has_key("verbose"):
        print "Verbose..."
```

لاحظ ان اول سطر بعد التعريف يعبر عن ال doc او وثيقة الدالة (ملف المساعدة بتاعها: d)

تقدر تستدعى ال __doc__ من الدالة ليعرضلك جزئية المساعدة الخاصة بيها او تستدعى دالة help عليها
__doc__

```
>>> print newprintf.__doc__ #STR
```

```
Simple function to learn out howto use *, ** trick in functions.
Help on function newprintf in module __main__:
```

help

```
>>> help(newprintf)
```

```
newprintf(fmt, *args, **kwargs)
    Simple function to learn out howto use *, ** trick in functions.
```

على فرض ان عندنا متغيرات عامة فى الملف الخاص بنا ومتاجين نستخدمها فى دالة معينة هنعمل اية ؟
استخدم global كالتالى

```
__DEBUG=True

def isdebug():
    global __DEBUG
    return __DEBUG

print "Debug? ", isdebug()

#output:
Debug? True
```

كل المطلوب انك تعرف الدالة اللى هتستدعى فيها متغيرك العام بانه global باستخدام global ويليها اسم المتغير
او تقدر تستخدم globals() كقاموس للمتغيرات العامة كالتالى مثلا

```
__DEBUG=False

def isdebug():
    x, y, z=range(3)
    print locals()
    return globals()["__DEBUG"]

print isdebug()

#output:
#{'y': 1, 'x': 0, 'z': 2}
#False
```

لاحظ استخدام locals() هنا بتعيد قاموس ايضا يعبر عن المتغيرات المحلية فى سياقها مثل x,y,z محليين فى السياق الموجودة فيه وهو الدالة isdebug

```
hndy.py - C:\Python25\hndy.py
File Edit Format Run Options Windows Help

#Handy Functions.

#-execfile ->Runs a file passed as an argument.

execfile('ch1.py')

#-eval -> Evaluates an expression.

eval('1+2+3+4+5') # equiv to 1+2+3+4+5 ->Result = 15

#-exec -> Executes a string containing arbitrary Python code.

List=[1, 2, 3, 4, 5]
exec "b=[x for x in List]"

#print b returns -> [1, 2, 3, 4, 5]
```

execfile(filepath)

تقوم بتنفيذ ملف بايثون

exec expr

تقوم بتنفيذ تعبير بايثون

Lambda/Anonymous Functions

قراءتك لهذه الجزئية اختيارية

بايثون بتتيحك استخدام الدوال المجهولة ودا باستخدام lambda مستعارة من لغة lisp

```
def getName(name):
    return name

anonyFunc=lambda name: name

print anonFunc("Mido")
print getName("Mido")

#Output:
#Mido
#Mido
```

مابعد lambda هو ال args ومايليها هو ال return


```
def getSum(*args):  
    return sum(args)  
  
anonySum=lambda *args: sum(args)  
  
print getSum(1, 2, 3, 4, 5)  
print anonysum(1, 2, 3, 4, 5)  
  
#Output:  
15  
15
```

map

بتطبيق function معينة على مجموعة من العناصر

```
>>> print map(lambda w: w.upper(), ["ahmed", "mostafa", "omar"])  
['AHMED', 'MOSTAFA', 'OMAR']
```

هنا هيتعمل ريترن -اعادة- بنسخة من العناصر بعد التعديل (التحويل للحروف الكبيرة)

استخدام lambda مش واضح بالنسبة ليك مش مشكلة استبدالها كالتالى

```
def toupper(w):  
    return w.upper()  
  
users=["ahmed", "mostafa", "omar"]  
print map(toupper, users)  
  
#output:  
#['AHMED', 'MOSTAFA', 'OMAR']
```

صحيح ايه ال w اللي كانت فى lambda وموجود فى toupper ك parameter دى ؟
ال w دى بتعبر عن كل عنصر فى ال sequence هتطبق عليه الدالة

filter

بنستخدمها لتصفية sequence معينة هنشوف مثال

```
numbers=range(20)  
print filter(lambda i: i&1, numbers) #odds.  
print filter(lambda i: not i&1, numbers) #evens  
  
#output:  
#[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]  
#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

لاحظ هنا في المثال عندنا مجموعة ارقام من 0 ل 19
هنستخدم قاعدة 1 & n لإختبار هل الرقم فردي او زوجي (او اي طريقة تعجبك) زي ماشايفين في المثال
بإستخدام lambda وبالفعل سيتم التصفية بناءا على القاعدة اللي حاطينها في ال function

المثال بدون استخدام lambda

```
def isodd(i):
    if i&1:
        return True
    return False

def iseven(i):
    return not isodd(i)

print filter(isodd, numbers) #odds
print filter(iseven, numbers)#evens

#output:
#[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

اماكن استخدام lambda ؟ يفضل تستخدمها مع ال properties كديل لل getters (تابع الفصل القادم او في دالة ب سطر واحد)

ملحوظة

```
لإختبار هل الرقم زوجي او لا اختبر ناتج باقى القسمة على 2 هل يساوى 0 او لا ويتم ذلك بإستخدام %
num % 2 == 0 ?
```

Iterable

هى حاوية container -مثل ال list, tuple, string-

Generator

بكل بساطة هى دالة تعطينا iterator

Iterator

هى كائن عبارة عن موجة يمثل بيانات ال iterable له طريقة next عندما يتم استدعاؤها يتم اعادة العنصر التالى فى ال iterable -هل تذكر for loop؟-

```
>>> lst=range(5)
```

```
>>> lst
```

```
[0, 1, 2, 3, 4]
```

```
>>> for item in lst:
```

```
...     print item
```

```
...
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

```
>>> s="Hello"
```

```
>>> for char in s:
```

```
...     print char
```

```
...
```

```
H
```

```
e
```

```
l
```

```
l
```

```
o
```

بالفعل احسنت

```
>>> iterator=iter(s)
```

هنا انشئنا موجة تعبر عن محتوى الحاوية s -الحروف-

```
>>> iterator
<iterator object at 0x22e83d0>
```

هذا هو الكائن

```
>>> iterator.next()
'H'
```

استدعاء next يعطيني العنصر التالي في الموجة

```
>>> iterator.next()
'e'
```

العنصر التالي

```
>>> iterator.next()
'o'
```

العنصر التالي

```
>>> iterator.next()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

تم رفع استثناء لإنهاء العناصر التالية في الموجة

على فرض لدينا التالي

```

class Student(object):
    def __init__(self, id, name):
        self.id=id
        self.name=name
    def __str__(self):
        return "<Student %d: %s>"%(self.id, self.name)

```

صف يمثل الطالب

```

class Students(object):
    def __init__(self):
        self._studentslist=[]

    def add(self, studentobject):
        self._studentslist.append(studentobject)

```

صف يمثل مجموعة الطلاب

```

fakedata=[(1, 'ahmed'),(2, 'khaled'), (3, 'ayman'), (4, 'tina')]
students=Students()
for id, name in fakedata:
    s=Student(id, name)
    students.add(s)

```

نملء مجموعة الطلاب ببيانات غير حقيقية الآن مالمسلوك المتوقع اذا قمنا بعمل

```

for student in Students:
    print student

```

ان يقوم بطباعة كل طالب فى مجموعة الطلاب اليس كذلك ؟

Traceback (most recent call last):

File "tstitors.py", line 27, in <module>

```
for s in students:
```

TypeError: 'Students' object is not iterable

ماهذا ؟؟ لجعل الصف الخاص بك iterator يجب عليك اعادة تعريف __iter__ و next

```
siter=iter(students)
```

```
print siter
```

انشئنا iterator

```
print siter.next()
print siter.next()
print siter.next()
print siter.next()
print siter.next()
```

الناتج

```
<Student 1: ahmed>
<Student 2: khaled>
<Student 3: ayman>
<Student 4: tina>
```

Traceback (most recent call last):

File "tstiters.py", line 44, in <module>

```
    print siter.next()
```

File "tstiters.py", line 23, in next

```
    raise StopIteration
```

بالطبع تستطيع استخدامها مع for

```
for s in students:
```

```
    print s
```

ليعطيك الناتج السابق حتى ال StopIteration

نعود لل generators

```
def ifind(self, sname):
    for student in self._studentslist:
        if student.name==sname:
            yield student
```

هنا نقوم بالبحث في ال self._studentslist عن اسم طالب يساوي sname وإذا وجدناه نقوم بعمل yield له

```
gen=students.ifind('ahmed') #0, 4
```

```
print gen
```

```
print gen.next()
print gen.next()
```

```
<generator object ifind at 0x7f43f1f73af0>
```

```
<Student 1: ahmed>
```

```
<Student 5: ahmed>
```

لاحظ هنا ان الكود lazy بمعنى ان لن يتم تنفيذه الا اذا حددت انت ذلك
كيف تستخدم next؟؟ نعم فعلا ال generator نفسه هو iterator

```
def isearch(string, c):
    for i, char in enumerate(string):
        if char==c:
            yield i
```

```
s="Hello World! bye world!"
```

```
gen=isearch(s, "o")
```

```
print list(gen)
```

اذا قمت بعمل list على المولد إنه يقوم بإعادة جميع العناصر التي سيتم عمل yield لها في صورة
[4, 7, 18]

تستطيع ان تقوم بالحصول على العنصر التالي بإستخدام next

ملحوظة لاتستطيع الحصول على عناصر تالية بعد استدعاء list عليها! لأنه سيعطيك StopIteration

حاولت ايصال الموضوع بطريقة سهلة بناء على المقالة
[/http://www.learningpython.com/2009/02/23/iterators-iterables-and-generators-oh-my](http://www.learningpython.com/2009/02/23/iterators-iterables-and-generators-oh-my)

التي نشرت في [Python Magazine](#)

Chapter 5 (OOP)

مقدمة

الموضوع هيتحول لصورة ممتعة اكثر بمراحل عن الصفحات اللي فاتت
الدنيا حولينا زى ماهى كلها احتمالات ولوبس فهى كلها OOP X OOP

اولا يعنى ايه OOP ؟

هى اختصار ل Object Oriented Programming وهى اسلوب مختلف فى البرمجة عن اسلوب ال
Procedural Programming زى اللي فى السى مثلا وهو بكل بساطة = سهولة وكفاءة اكثر بمراحل من ال
Procedural Programming وبكل تأكيد اكثر تنظيم!

اذا بصيت حوايك هتلقه الدنيا كلها عبارة عن Objects عربية انسان عصفورة طيارة قطعة كل دول Objects

هندرس على فرض انك بتحدد تصميم لإنسان بتيجى بورقة وقلم كدا وتكتب

Fields

ال fields هى (كل مايستخدم فى وصف الإنسان)

اسم
سن
نوع
لون
طول
وزن

Methods

ال methods هى (كل مايؤديه الإنسان)

يتحرك
ياكل
يشرب
ينام

Properties

ال properties هى عبارة عن Encapsulation (تغليف) لل fields لمجرد حماية الصفات الخاصة بالإنسان
فهتكون كالتالى

اسم
سن
نوع
لون
طول
وزن
--
-

Encapsulation

حول التغليف

إذا هناك مريض وبأخذ كبسولة -تعالجه من البرد- فهو يتعامل مع الكبسولة ولا يتعامل مع المواد التي تغلفها الكبسولة ، تحتاج **لصلاحيات وصول** وفهم للتعامل ربما في المختبر؟

هنتعرض ليها بالتفصيل ان شاء الله فى مثالنا

الصورة المبدئية للصف class الخاص بنا هتكون كالتالى

```
class Human(object):

    def __init__(self, name, color, sex, height, weight):
        self.name=name
        self.color=color
        self.sex=sex
        self.height=height
        self.weight=weight

    def eat(self):
        #code to eat.
        pass

    def drink(self):
        #code to drink.
        pass

    def sleep(self):
        #code to sleep.
        pass

    def play(self):
        #code to play.
        pass
```

*لكتابة اي صف class بنبدأ التعريف بكلمة class

*اسم الصف class يكون بادئ Uppercase

*الصف class بتاعك لازم يورث من Object (هنتكلم عن الوراثة) ولكن يكفيك ان اى انسان ماهو إلا object و اى class فى الدنيا ماهو إلا object فييفة اكيد هيورث صفات ال Object.

__init__

بدأنا ب ال Constructor -مشيد او بناء- وهنا فى بايثون بيكون عبارة عن function بإسم __init__
*لاحظ اى method تبدأ بمعاملاتها ب self .. تقدر تستبدل self بأى كلمة تناسبك ولكن مجتمع بايثون مرتبط ب self فإلتزم بالقواعد.

نبدأ نجهز ال fields بتاعت ال class زى name, sex, color,.. etc

بإتنا نعمل field بكل إسم self.name, self.sex, self.color

ونسند له القيم اللي تم تمريرها من ال constructor كالتالى

```
def __init__(self, name, color, sex, height, weight):
    self.name=name
    self.color=color
    self.sex=sex
    self.height=height
    self.weight=weight
```

جميل تعالى نختبر ال class الخاص بنا

```
ahmed=Human("Ahmed", "White", "M", 178,70)
```

- 1- اتنا ننشئ object من ال Human class ونمرر الداتا اللي هنوصف بيها ال object دا
- 2- جملة print بسيطة لعرض ال fields

```
print ahmed.name
print ahmed.sex
print ahmed.height
print ahmed.weight
print ahmed.color
```

جميل الكود الخاص بنا 10/10 ولكن فيه مشاكل!

```
ahmed.name=17777777
print ahmed.name
```

تخيل حد يبقه اسمه عبارة عن ارقام !!! كدا الكود الخاص بنا فيه مشكلتين

- 1- انه مكشوف
 - 2- انه غير آمن اى حد يقدر يحط اى قيم على مزاجه ودا اسمه تهريج
- فقدامنا حل جميل جدا وهو اتنا نستخدم اسلوب ال get/set ودا اسلوب شهير جدا لحماية الكود ولكن ازاي نحمل الكود وهو مكشوف ؟ جميل جدا كدا انت بقيت ماشى معايا صح لازم نمنع الاكسس للمستخدم على ال fields بتاعتنا بس ازاي؟!

get/set

بسيطة اسبق اسم كل field ب 2 underscores كالتالى مثلا

```
def __init__(self, name, color, sex, height, weight):
    self.__name=name
    self.__color=color
    self.__sex=sex
    self.__height=height
    self.__weight=weight
```

كدا ال fields بقت private مش فى حد يقدر يتعامل معاها غير ال class نفسه فى عملياته الداخلية لكن المستخدم الخارجى لأ
ضيف 2 methods لكل field واحدة get والثانية set كالتالى

```
#Getters
def get_name(self):
    return self.__name

def get_color(self):
    return self.__color

def get_sex(self):
    return self.__sex

def get_height(self):
    return self.__height

def get_weight(self):
    return self.__weight

#Setters
def set_name(self, new_name):
    self.__name=new_name

def set_color(self, new_color):
    self.__color=new_color

def set_height(self, new_height):
    self.__height=new_height

def set_weight(self, new_weight):
```

```
self.__weight=new_weight

def set_sex(self, new_sex):
    self.__sex=new_sex
```

1- تعالى نختبر موضوع حماية ال fields كالتالى مثلا

```
>>> print ahmed.__name
AttributeError: 'Human' object has no attribute '__name'
```

هتلقه error كالتالى

2- تعالى نجرب نتعامل مع ال fields من خلال ال getters/setters كالتالى مثلا

```
ahmed=Human("Ahmed", "White", "M", 178,70)
print ahmed.get_name()
print ahmed.get_color()

ahmed.set_name("Youssef")
print ahmed.get_name()

ahmed.set_name(141241) #Wut?
print ahmed.get_name()

#Output:
Ahmed
White
Youssef
141241
```

جميل جدا ولكن برودو set_name مش عملت حاجة سمحت ان name ياخذ قيمة رقم!
طب وإيه المشكلة ؟ عدل الكود كالتالى مثلا..

```
def set_name(self, new_name):
    if isinstance(new_name, str):
        self.__name=new_name
    else:
        print "new_name ain't a string!" #and do nothin
```

هنا بنختبر هل ال new_name عبارة عن كائن من str او لا
باستخدام isinstance

او تقدر تعدل اسلوب ال icheck ليكون كالتالى

```
if type(new_name)==str:
    self.__name=new_name
```

“يعنى هل هو string او لا”
باستخدام ال type function بدل isinstance

اذا كان string يبقية عادى نعدل الإسم اذا لأ تطلع رسالة new_name ain't a string ومش يتم اى تعديل!
تقدر تصنيف ال rules اللى تناسبك مثلا الطول بتاعها وهل يشمل ارقام او لأ وهكذا... مع باقى ال fields
وهى دى عملية ال encapsulation! وهى باختصار حماية ال data members او fields او ال attributes
الخاصة بال class باستخدام getters/setters

جميل الإسلوب دا صح مش كدا ؟

properties

لكن بايثون بتقدملك اسلوب ابسط لهندسة ال class بتاعك واستخدامه وهو استخدام ال Properties
فاكر لما قلنا انها عبارة عن encapsulation لل fields ؟ تمام ال properties بتعمل class لل
getter/setter مباشرة

```
ahmed.name=new_name
```

بدلا من

```
ahmed.set_name(new_name)
```

```
ahmed.name
```

بدلا من

```
ahmed.get_name()
```

الله! ايه دا انت رجعت تانى لموال ال fields مش كنا قلنا اننا لازم نخليها private ونبعدها عن المستخدم؟!
تمام انا قلت كدا بس شكلك مش مركز لأننا بتتكلم عن ال properties (:
هيكون فى property مثلا بإسم name تقوم بشغل ال get_name و ال set_name كالتالى مثلا

```
name=property(fget=get_name, fset=set_name, doc="Gets/Sets the name.")
```

fget بتعبر عن الميثود المسئولة عن ال get وهنا هتكون ال get_name وهى بتستدعى فى حال

```
object.name
```

fset بتعبر عن الميثود المسئولة عن ال set وهنا هتكون ال set_name وهتستدعى فى حال

```
object.name=new_name
```

doc بتعبر عن وصف ال property

```
ahmed=Human("Ahmed", "White", "M", 178,70)
print ahmed.get_name()
print ahmed.name
ahmed.name="Youssef"
print ahmed.name
ahmed.name=979878 #uses the get_name rules!
```

طلب كذا في حاجة ايه الهدف من التكرار في ؟ ليه يكون عندي name, get_name, set_name اسمج
للمستخدم انه يستخدمهم ؟
تمام بكل بساطة اعمل ال get_name, set_name ك private method واتعامل بيها داخل الصف وخلي
ال name ك property ظاهرة للمستخدم ويتعامل معاها بدل مايتعامل مع 2 methods
ازاي اخليهم private ؟ بكل بساطة اسبق اساميهم ب 2 underscores

Magic methods

ندخل فى ال Magic Methods

بداية هى كل method مبدئية ومنتھية ب __ مثلا __init__
الفائدة: هى بتيح ليك تعريف سلوك التعامل مع ال Builtin Functions زي len مثلا!
بتتيح ليك دعم ال Operator Overloading هنشوف كل الكلام دا بالتفصيل

__init__ -1
هى ميثود مسؤلة عن تجهيز ال fields فى حال انشاء ال object
--منتشرة بإسم Constructor

```
>>> class Human(object):
    def __init__(self, name):
        #Initialize the fields.
        self.name=name
        self.hands=2
```

يتم استدعائها فى حال الإنشاء ل object

```
>>> h1=Human("sami")
>>> h1.name
'sami'
>>> h1.hands
2
```

فى تعليق اضافى لما ندخل فى ال Inheritance

2- __getitem__(self, key), __setitem__(self, key, val)

على فرض ان عندنا صف بإسم MyDict

```
>>> class MyDict(object):
    def __init__(self, d={ }):
        self.__d=d
    def __getitem__(self, key):
        if key in self.__d.keys():
            return self.__d[key]
    def __setitem__(self, key, val):
        self.__d[key]=val
```

احنا مثلا لانريد نتعامل مباشرة مع ال self.__d ولكن نريد ان نتعامل معاه من خلال الصف او عملية ال indexing

```
>>> md=MyDict({'Name':'Ahmed', 'Sex':'m'})
```

نستخدم __getitem__

```
>>> md['Name'] #Call __getitem__('Name')
'Ahmed'
```

نستخدم __setitem__

```
>>> md['Name']='Youssef'
>>> md['Name']
'Youssef'
```

3- .__len__(self)

بيها بنحدد سلوك الصف الخاص بنا مع الدالة الشهيرة len

```
>>> class Lener(object):
    def __init__(self, s, alist):
        self._s=s
        self._list=alist
    def __len__(self):
        return len(self._s)
```

هنا بتعرفنا لل __len__ قمنا بتحديد السلوك فى حال استخدام len مع اى كائن مع الصف دا وهنا هنخليها تعيد عدد حروف ال سترينج self._s

```
>>> l=Lener("Ahmed Youssef", ["Tina", "Salma"])
>>> len(l) #Calls __len__
13
```

إذا اعدنا تعريفها لتكون كالتالى مثلا

```
def __len__(self):
    return len(self._list)
```

فعند استدعاء len على اى كائن من النوع Lener هيثم اعادة عدد عناصر ال self._list

```
>>> l=Lener("Ahmed Youssef", ["Tina", "Salma"])
>>> len(l)
2
```


4- `__iter__(self)`

بتحدد سلوك الصف من خلال لتعريفك ل generator وال iterations وتحديد التعامل مع for loop

```
>>> class Tech(object):
...     def __init__(self, langs, nums):
...         self._langs=langs
...         self._nums=nums
...     def __iter__(self):
...         for lang in self._langs: yield lang

>>> t=Tech(['Python', 'Ruby', 'Rebol'], [1500, 1414, 12515])
>>> for lang in t:
...     print lang
```

للتوضيح اكثر

```
>>> i=iter(t)
>>> i.next()
'Python'
>>> i.next()
'Ruby'
>>> i.next()
'Rebol'
```

*ملحوظة: لازم تتعمل على container

5- `__str__`, `__repr__`

لتحديد سلوك الكائن عند استدعاءه كعامل للدول str, repr

س: ما الفرق بين str, repr

Str تعطى تمثيل -سلسلة حرفية- للكائن مقروء للبشر Human readable
repr تعطى تمثيل للكائن-سلسلة حرفية- يمكن عمل eval عليه -لإعادة انشاء الكائن من تلك السلسلة الحرفية العائدة!-

```
>>> class Human(object):
...     def __init__(self, name):
...         self.name=name
...     def __str__(self):
...         return "<Human object: %s>"%self.name
...     def __repr__(self):
...         return "Human('%s')"%self.name
... 
```

```
>>> h=Human("ahmed")
>>> h
Human('ahmed')
>>> str(h)
'<Human object: ahmed>'
>>> s=repr(h)
>>> s
"Human('ahmed')" #string
>>> res=eval(s)
>>> res
Human('ahmed') # human object!
>>> type(res)
<class '__main__.Human'>
```

Operator Overloading

التحميل الزائد

وهو تعريف سلوك الكائنات لتقوم بتنفيذ فعل معين عندما تكون معاملا في عمليات (مثل الجمع والطرح والضرب والقسمة وغيرها)

1+4 دى إستخدام ال+ Operator وهو إن يجمع عددين
2*1 إستخدام ال* Operator هنا إنه يضرب عددين
1-2 إستخدام ال- Operator هنا إنه يطرح عددين
ولكن ماذا عن human + human
هل ينفع يكون ل Operator أكثر من إستخدام ؟
أها مثلا + Operator بيستخدم فى عمل دمج بين ال Strings

```
>>> s1='Hello, '  
>>> s2='World!'  
>>> s=s1+s2  
>>> s  
'Hello, World!'
```

يعنى إستخدمنا ال + Operator فى وظيفة اخرى غير الجمع وهى الدمج دى بإختصار هى ال Overloading Operators .. يعنى يكون ل Operator أكثر من إستخدام.

فى Special Methods او بتسمى احيانا بال Magical Methods هى اللى بتوفرلنا موضوع ال Operator Overloading دا + بعض الأشياء الأخرى

__add__ للجمع
__sub__ للطرح
__mul__ للضرب وهكذا

فلنفترض إن عندى class وليكن Worker مثلا

```
class Worker(object):  
  
    def __init__(self, name, work_hours):  
        self.name=name  
        self.work_hours=work_hours
```

وانت عايز تعمل زيادة لساعات العمل work_hours او نقصان او مضاعفة ؟!
فى عدة حلول زى إنك تعمل 3 Methods كالتالى مثلا

```
def increment_workinghours(self, hours):  
    self.work_hours += hours
```

```
return self.work_hours

def decrement_workinghours(self, hours):
    self.work_hours -= hours
    return self.work_hours

def mul_workinghours(self, hours):
    self.work_hours *= hours
    return self.work_hours
```

حل آخر : هو إنك تعمل Overload لل Operators ال + و - و * كالتالى

```
def __add__(self, hours):
    self.work_hours += hours
    return self.work_hours
def __sub__(self, hours):
    self.work_hours -= hours
    return self.work_hours

def __mul__(self, hours):
    self.work_hours *= hours
    return self.work_hours
```

هيكون صورة الصف كالتالى

```
class Worker(object):

    def __init__(self, name, work_hours):
        self.name=name
        self.work_hours=work_hours

    def increment_workinghours(self, hours):
        self.work_hours += hours
        return self.work_hours

    def decrement_workinghours(self, hours):
        self.work_hours -= hours
        return self.work_hours

    def mul_workinghours(self, hours):
        self.work_hours *= hours
        return self.work_hours

    def __add__(self, hours):
        self.work_hours += hours
        return self.work_hours
```

```
def __sub__(self, hours):
    self.work_hours -= hours
    return self.work_hours

def __mul__(self, hours):
    self.work_hours *= hours
    return self.work_hours
```

اعمل Object من ال Class وليكن w

```
>>> w=Worker('EVAN', 4)
>>> w.increment_workinghours(3)
7
>>> w.decrement_workinghours(2)
5
>>> w.mul_workinghours(2)
10
```

انا شايف إن الإسلوب دا ممل جدا مع إنه احيانا بيكون اءمن بعض الشئ ولكنه ممل!

اعمل Object تانى وليكن w1

```
>>> w1=Worker('ANN', 5)
>>> w1+2
7
>>> w1-4
3
>>> w1*5
15
```

جدول بكل المعاملات + ال Magic Methods الخاصة بيهم لتعريفهم

```
+ __add__, __radd__
- __sub__, __rsub__
* __mul__, __rmul__
/ __div__, __rdiv__, __truediv__ (for Python 2.2),
__rtruediv__ (for Python 2.2)
// __floordiv__, __rfloordiv__ (for Python version 2.2)
% __mod__, __rmod__
** __pow__, __rpow__
<< __lshift__, __rlshift__
>> __rshift__, __rrshift__
& __and__, __rand__
```

```
^ __xor__, __rxor__
| __or__, __ror__
+= __iadd__
-= __isub__
*= __imul__
/= __idiv__, __itruediv__ (for Python version 2.2)
//= __ifloordiv__ (for Python version 2.2)
%= __imod__
**= __ipow__
<<= __ilshift__
>>= __irshift__
&= __iand__
^= __ixor__
|= __ior__
== __eq__
!+, <> __ne__
> __gt__
< __lt__
>= __ge__
<= __le__
```

BTW, ..Enums

لاحظ استخدامنا فى الفصل السابق لأوامر فى التعامل بين السرفر والعملاء مثل `setSharedFiles/` او `query/` الخ
ولكن لفصل افضل للتطبيق تستطيع انشاء `Enums`

احيانا بنحتاج اننا نجمع مجموعة من البيانات تحت اسم معين مثلا ايام الأسبوع (احد اثنين ثلاثاء .. الخ) والألوان (ابيض ازرق اخضر .. الخ)

مثلا فى باسكال

```
type
TDay = (Saturday=1, Sunday=2, Monday=3, Tuesday=4, Wednesday=5, Thursday=6, Friday=7);
```

الكود المشابه ليه فى بايثون ممكن يكون

```
(Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday=range(1, 7
```

ففى كذا طريقة فى بايثون
وممكن تعرفهم كالتالى

```
class MyDays(object): #as rawenum
    sunday, monday, tuesday=range(3)
class MyKVDays(object):
    sunday, monday, tuesday=0, 9, 2
```

مممكن نعمل class كالتالى

```
class RawEnum(object):
    def __init__(self, start, *enum): #do we need to set a start, end, step? very fancy...
        self._kw=dict(zip(enum, range(start, start+len(enum))))
        counter=start
        for arg in enum: #assuming it doesn't exist.
            self.__setattr__(arg, start)
            counter += 1
    __str__=lambda self: str(self._kw) #Can be solved with an ordered dict.
```

وهنا بيتحدد فيه قيمة البداية لل enum فقط

```
Colors=RawEnum(5, 'white', 'black', 'blue')
print Colors.white
print Colors
```

او اتنا نستخدم **kwargs

```
class KVEnum(object):
    def __init__(self, **kwargs):
```

```
self.__kw=kwargs
for k, v in kwargs.items():
    self.__setattr__(k, v)
__str__=lambda self: str(self.__kw)
```

بحیث نقدر نستخدمها كالتالى

```
Days=KVEnum(sunday=0, monday=9, tuesday=2)
print Days
print Days.monday
```


Chapter 6 (Inheritance)

العلم كله مبني على الوراثة والإكمال من حيث انتهى الآخرون حيث لاداعي لتكرار ملايين الأسطر لأستخدامها مرة أخرى

تخيل ان عندنا صف Human كالتالي

```
class Human(object):

    def __init__(self, name, sex):
        self._name=name
        self._sex=sex

    def _set_name(self, name):
        self._name=name

    def _set_sex(self, sex):
        self._sex=sex

    name=property(fget=lambda self:self._name, fset=_set_name)
    sex=property(fget=lambda self:self._sex, fset=_set_sex)
```

وعندنا صف Employer كالتالي

```
class Employer(object):

    def __init__(self, name, sex, salary):
        self._name=name
        self._sex=sex
        self._salary=salary

    def _set_name(self, name):
        self._name=name

    def _set_sex(self, sex):
        self._sex=sex

    def _set_salary(self, salary):
        self._salary=salary

    name=property(fget=lambda self:self._name, fset=_set_name)
    sex=property(fget=lambda self:self._sex, fset=_set_sex)
    salary=property(fget=lambda self: self._salary, fset=_set_salary)
```

اكيد لاحظت ان ال Employer هو Human ولكن مش فيه زيادة عن ال Human غير ال salary attribute و

ال salary setter وال salary property

يعنى نقدر نقول ال

Employer *is-a* human

فكل اللى عليك انك تحسن الكود بحيث ان ال Employer يورث كل الصفات + الميثودز الموجودة بال Human ويضيف عليه المميزات الخاصة بيه زي ال salary كالتالى مثلا

```
class Employer(Human):
    def __init__(self, name, sex, salary):
        Human.__init__(name, sex)
        self._salary=salary

    def _set_salary(self, salary):
        self._salary=salary

salary=property(fget=lambda self: self._salary, fset=_set_salary)
```

تعالى نتكلم بمثال اوضح وقريب من العالم الحقيقى بعض الشئ وعلى المثال السابق
إنسان وموظف ومدير

الموظف ماهو إلا إنسان والمدير ماهو إلا انسان مش كذا ؟ تعالى الأول نعرف ال Human class الخاص بنا

```
class Human(object):

    def __init__(self, name, color, sex):

        #Data members..
        self.__name=name
        assert sex in Gender.Options #Male or Female only.
        self.__sex=sex
        self.__color=color

    #getters/setters...
    def getName(self):
        return self.__name

    def setName(self, value):
        self.__name = value

    def getSex(self):
        return self.__sex

    def setSex(self, value):
        self.__sex = value
```

```

def getColor(self):
    return self.__color

def setColor(self, value):
    self.__color = value

#properties...
name = property(getName, setName, None, "Gets/Sets name.")

sex = property(getSex, setSex, None, "Gets/Sets sex.")

color = property(getColor, setColor, None, "Gets/Sets color.")

__str__=lambda self: "<Human object: %s >"%self.__name

__unicode__=__str__

#methods..
def eat(self):
    #Eating
    pass

def drink(self):
    #Drinking
    pass

def sleep(self):
    #Sleeping
    pass

```

هنا عرفنا صنف جديد بيعبر عن الإنسان وليه متغيرات داخلية زي الإسم واللون والنوع وبعض الميثودز لمعالجتهم
وميثودز اخرى مثل eat, drink, sleep

لاحظ ان النوع لازم يكون موجود فى Gender.Options المعرفة كالتالى

```

class Gender(object):
    Male, Female="Male", "Female" #0, 1 whatever!
    Options=(Male,Female)

```

الإسلوب دا يقدر يفيدك لما تيجى عايز تعمل type لنوع صامت او جامد مثلا النوع او الألوان وهكذا
لاحظ ال `__str__` دي magic method بيتم استدعائها عند استدعاء `print` او حتى ال casting بإستخدام `str()`
لاحظ اننا خلينا `__unicode__` نفس المعنى من `str` (وبفضل انك تستخدم ال `unicode` على طول الأبلكيشن
بتاعك)
تقدر تعرفهم اكيد كالتالى

```
def __str__(self):
    ....

def __unicode__(self):
    return self.__str__()
```

نيجى للموظف

```
class Employer(Human):

    def __init__(self, name, color, sex, salary, firm):
        #Construct the human with (name, color, sex)
        Human.__init__(self, name, color, sex)
        #Superize it :)
        #super(Employer, self).__init__(name, color, sex)

        self.__salary=salary

        self.__firm=firm

    def getFirm(self):
        return self.__firm

    def setFirm(self, value):
        self.__firm = value

    firm = property(getFirm, setFirm, None, "Gets/Sets the firm.")

    def getSalary(self):
        return self.__salary

    def setSalary(self, value):
        self.__salary = value

    salary = property(getSalary, setSalary, None, "Gets/Sets salary.")
```

```

__str__=lambda self: "<Employer object:(%s, %d) >"%(self.getName(), self.salary)
#super(Employer, self)

def eat(self):
    print "This is my break (eat or having fun hummm?"

def sleep(self):
    print "Hi, it's me sleeping!"

```

هنا بننشئ صف جديد مشتق من ال Human وبنجهزه بالبيانات الخاصة بال Human (اسم ولون ونوع) والبيانات الخاصة بالموظف (المرتب والشركة) ومتغيرات تعالج الحالة

- اعدنا تعريف بعض الميثودز (الطرق) الخاصة بال Human بمعنى انهم اصبحت overridden

نيجى للمدير ومالمدير الا موظف ولكن له صلاحيات اعلى زى مثلا اعطاء علاوة او فصل موظف وهكذا

```

class Manager(Employer):

def __init__(self, name, color, sex, salary, firm):
    #Employer.__init__(self, name, color, sex, salary, firm)
    super(Manager, self).__init__(name, color, sex, salary, firm)
    #print self.__dict__
    #{'_Human__color': 'white', '_Employer__salary': 200000, '_Human__name': 'Wael',
'_Employer__firm': 'Sun', '_Human__sex': 'Male'}

def raiseSalaryOf(self, emp, theraise):
    assert (isinstance(emp, Employer) and emp.firm==self.firm)
    emp.salary += theraise

```

لاحظ انك لو طلبت __dict__ لآى صف هيعرضلك قاموس dictionary فيه القيم اللى وراثتها من المتغيرات الداخلية والمقطع الأول منها يعبر عن منين موروثه

تعالى ننفذ عالمنا الصغير دا كتطبيق سريع

```

if __name__=="__main__":

    ahmed=Employer("ahmed", "white", Gender.Male, 50000, "Google")
    omar =Employer("omar", "black", Gender.Male, 40000, "Sun")
    tina =Employer("christina", "white", Gender.Female, 50000, "Google")
    emps=(ahmed, omar, tina)
    wael=Manager("Wael", "white", Gender.Male, 200000, "Sun")
    wael.raiseSalaryOf(omar, 9000)

```

```
print issubclass(Employer, Human)
print isinstance(ahmed, Human) #Ahmed is a Human..
print Human.__bases__ #inherits object.
print Manager.__bases__ #Inherits Employer..
```

لاحظ ان `issubclass(C, B)` يتسأل هل الصف C مشتق من B او لا
لاحظ ان `isinstance(object, Type)` يتسأل هل ال object دا تم انشاءه من الصف Type او لا
لاحظ ان `__bases__` بتعبر عن الأب (او الأباء لو وراثت من اكثر من صف)
احنا استخدمنا `assert` فى كذا جزئية لكن ايه هى `assert` ؟
assert بتفيد فى عمل تصحيح سريع بحيث انك تضمن عدم التنفيذ فى حال وجود خطأ

امتة استخدم `ASSERT` ؟ استخدامها مقترن بحالة التطوير والمحل مثلا مش هتعمل كود فيه `assert`
`statement` للمستخدم النهائي ، لكن ممكن تحطها فى مكتبة هيسخدمها مطور (هيراعى المتطلبات مضبوط)
غير كذا استخدم ال `Exceptions, Errors`

Multiple Inheritance

مقدمة:

الوراثة المتعددة هي ميزة موجودة في بعض اللغات الكائنية تسمح لصف بان يورث اكثر من صف كما في المثال التالي سنجد ان هناك المستذئب يرث الإنسان والذئب ويجمع بين ميزاتهم

```
class Wolf(object):
    def __init__(self):
        self.__bite=True

    can_bite=lambda self:self.__bite

    def fullmoon(self):
        print "Woooooooooooooooooooooooooooo"

    def bite(self, h):
        print "I'm cursed"

class Werewolf(Human, Wolf):

    def __init__(self, name, color, sex):
        Human.__init__(self, name, color, sex)
        Wolf.__init__(self)
yasser=Werewolf("yasser", "black", Gender.Male)
print yasser.__dict__
print "Can bite? ", yasser.can_bite()
yasser.bite(ahmed)
print Werewolf.__bases__
```

```
class Wolf(object):
    def __init__(self):
        self.__bite=True

    can_bite=lambda self:self.__bite

    def fullmoon(self):
```

```

print "Woooooooooooooooooooooooooooo"

def bite(self, h):
    print "I'm cursed"

class Werewolf(Human, Wolf):

    def __init__(self, name, color, sex):
        Human.__init__(self, name, color, sex)
        Wolf.__init__(self)

```

تابع مثالنا السابق هنا عندنا صف بإسم wolf وصف بإسم مستدئب Werewolf بيورث الإنسان والذئب

```

yasser=Werewolf("yasser", "black", Gender.Male)
print yasser.__dict__
print "Can bite? ", yasser.can_bite()
yasser.bite(ahmed)
print Werewolf.__bases__

```

النتج

```

{'_Human__color': 'black', '_Wolf__bite': True, '_Human__name': 'yasser', '_Human__sex': 'Male'}
Can bite? True
I'm cursed
(<class '__main__.Human'>, <class '__main__.Wolf'>)
(<class '__main__.Human'>, <class '__main__.Wolf'>)

```

لاحظ الآباء للمستدئب زى ماشفت فى ناتج __bases__ (<class '__main__.Human'>, <class '__main__.Wolf'>)

تعرف على ال MRO - لن يتم تغطيتها فى هذا الكتاب -
<http://www.python.org/download/releases/2.3/mro>

Chapter 7 (Exceptions, Errors)

مش فى شئ كامل وطالما كتبت كود توقع انك ستجد فيه مشكلات! مثلا انقطاع الإتصال مع الداتايز او مشكلات فى التتورك او مكتبيات غير متوافرة او كود خاطئ SyntaxError او بيانات دخلها المستخدم بصورة خاطئة او خطئ رياضى او او
فنحن لانقوم بكتابة software writing ولكن اعادة كتابة software rewriting لماذا نستخدم text editor وليس text writer فى رأيك P?

اولا ايه هى ال exceptions ؟ هى شئ حصل يتسبب فى تغيير مسار برنامجك "الى انت خططه"
المفتاح للموضوع دا 4 حاجات (مش شرط يكون خطأ) ولكن شئ غير محسوب

- 1- try
- 2- except
- 3- finally
- 4- raise

try مش فيها شئ عجيب كل المطلوب منك هو انك تضيف الجزء اللى هتشك ان ممكن يحصل فيه exception

except: بتصطاد فيها ال exception وتعالجه

finally : الكود المرتبط بيها هيثم تنفذ دائما حتى لو مش عاجت ال exception بتكون غالبا لإغلاق المصادر المفتوحة (مثلا فايل او كونيكتشن معين)
raise: هى المسئولة عن اطلاق ال exceptions دى

تعالى نشرحها بمثال شهير جدا
هنحاول نقسم عدد على صفر

```
>>> 1/0
```

رد بايثون

```
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    1/0
ZeroDivisionError: integer division or modulo by zero
>>>
```

لاحظ السطر ZeroDivisionError: integer division or modulo by zero

بيتكلم فيه عن حدوث Error للقسمة على صفر + وصف ال Error
طب جميل جدا انا كذا عرفت ان ممكن يتقسم على صفر طب انا عايز اصطاد ان حد حاول يعمل كذا
1- ضع الكود بتاعك فى try بلوك
2- هندل -تعامل مع- ال exception فى except بلوك

```
try:
    print 1/0
except ZeroDivisionError, e:
    print e.message
```

```
#output: integer division or modulo by zero
```

مثال آخر ادخال قيم غير سليمة او منطقية

```
>>> def sayHi(name):
    if not isinstance(name, str):
        raise ValueError("name ain't string.")
    else:
        print "Hi, %s"%name
```

هنا عرفنا function بإسم sayHi يتاخذ بارمتر واحد بإسم name (منطقيا لازم يكون str) وإلا مثلا مش هنقول hi لرقم؟! فى حالة ان المستخدم هيمرر رقم او اى شئ غير str هنعمل raise لإپرور بإسم ValueError (معرف مسبقا) وبرسالة name ain't string.

```
>>> sayHi(9)

Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    sayHi(9)
  File "<pyshell#26>", line 3, in sayHi
    raise ValueError("name ain't string.")
ValueError: name ain't string.
```

تعالى نعمل مثال لإنشاء exception خاص بيك
نريد ان نكتب فنكشن معينة تقرا رقم من المستخدم وتعمل بيه return لكن لو المستخدم دخل قيمة مخالفة
يتم عمل raise ل Exception بإسم IntOnly

1- انشئ الصف الجديد واشتقه من نوع Exception او Error مناسب

```
>>> class IntOnly(ValueError):
    def __init__(self, msg):
        self.message=msg
    def __str__(self):
        return repr(self.message)
```

2- اكتب الفنكشن واعمل raise لل IntOnly فى حال عدم التوافق

```
>>> def readInt():
    inp=raw_input("Enter a num: ")
    try:
        i=int(inp) #cast to integer.
        return i
```

```
except Exception:
    raise IntOnly("Integers only are allowed.")
```

3- اكتب كود برنامجك

```
>>> try:
    j=readInt()
except IntOnly, e:
    print e.message
```

هنا بنستدعى readInt ونسند قيمتها ل z في ال try block
في حالة حدوث ايورر(خطأ) من النوع IntOnly هنعمل منه كائن (اللى هو e) ونشوف e.message
ودا الأسلوب المفضل
تقدر تتعامل مع كذا اكسبشن بإستخدام except حسب ماتحب طالما الكود بتاعك فيه مشاكل

مثال على finally

```
#!/usr/bin/python

f = None #out of try block. As finally doesn't have try' context

try:
    f = file('somefile, 'r')
    lines = f.readlines()
    for line in lines:
        print line, #avoid \n\n !
except IOError, e:
    print 'IOErrorError'
finally: #cleaning up
    if f:
        f.close()
```

كدا وصلنا للصيغة العامة وهى

```
try:
    suite
except EX1, ex1: suite

except EX2, ex2: suite

except EX3, ex3: suite

finally: suite
```

يوجد بعض الصفوف المجهزة للتعامل مع الإستثناءات/الأخطاء

BaseException: هو الأب

Exception: هو الأب المشتق منه الإستثناءات المعتادة

ImportError: محاولة استدعاء موديل

KeyboardInterrupt: عندما يقاطع المستخدم التنفيذ (غالبا ب C^)

NameError: محاولة استدعاء identifier غير موجود

SyntaxError: كود بايثون خاطئ

IndexError: الوصول لترتيب غير موجود في sequence معينة

KeyError: مفتاح غير موجود في قاموس معين

IOError: مشاكل في الدخول او الخرج IO ملف غير موجود مثلا

OverflowError: تعدى الحجم المسموح به لنوع معين

OSError: نظام التشغيل

AssertionError: خطأ نتج بسبب فشل في assert expression

```
>>> 1==0
False
>>> assert 1==0
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    assert 1==0
AssertionError
```

أكتب اكواد افضل وعالجها بصورة افضل!

Chapter 8 (IO)

Basic IO

هنتكلم فى الفصل دا عن التعامل مع ال files وتحديد ال IO اختصارا ل Input/Output
ال file type يعبر عن صف مسئول عن التعامل مع الملفات (ممكن يكون سو كيت او غيره بما ان كل شئ عبارة عن file)

فى اكثر من طريقة لإنشاء file object ودا عن طريق open او file class
open هتعمل ريترن ب file object وهى معرفة كالتالى

open(name, mode, buffering)

حيث name هو مسار الملف
mode هو يعبر عن حالة الوصول (الملف مفتوح للقراءة ، للكتابة ، للإضافة ؟) الافتراضى هو r للقراءة

r → قراءة فقط

w → كتابة فقط (بيتم محو كل البيانات الموجودة)

a → للإضافة من عند النهاية ، مع عدم محو البيانات

r+ → قراءة وكتابة

w+ → كتابة وقراءة

اى من ال modes السابقة اذا لحقته ب b اختصارا ل binary سيتم التعامل مع binary read, binary write, .. etc

ال buffering لتحديد هل سيتم عمل اى buffering فى حال التعامل مع الملف، الافتراضى هو 1- (بيتم النقل لنظام التشغيل)

.close()

لغلق ال file object

.read(num=None)

بتقوم بقراءة عدد معين من البايتات وفى حال عدم تحديده بيتم قراءة الملف بكامله

.readline()

قراءة سطر واحد

.readlines()

قراءة كل السطور (على صورة list)

.tell()

بتخبرنا بالمكان الحالى

.fseek(offset, whence=0)

بتنقل المكان الحالى الى offset معين بعد ال whence
whence ربما تكون 0 (اى بداية الملف) او 1 (المكان الحالى) او 2 (نهاية الملف)
عندنا ملف سميناه iotest.xcd فيه التالى 0123456789

.write(s)

يقوم بكتابة s فى الملف

.writelines(seq_of_strings)

يقوم بكتابه كل عناصر seq_of_strings فى الملف
--كقيامك بإستدعاء write على كل عنصر فى هذه ال sequence

.fileno()

الحصول على File Descriptor خاص بال file

.flush()

لتأكيد نقل ال buffer الداخلى كله على الملف

.name

للحصول على مسار الملف

.mode

للحصول على ال access mode

.encoding

للحصول على الإنكودينج

.closed

للتحقق من تحقق اغلاق الملف

```
FNAME="iotest.xcd"
txt="""
line 1
line 2
line 3
line 4
line 5
line 6
line 7
some text
yada yada yada!
"""
f=open(FNAME, "w")
print f.fileno()
print f.name
print f.mode
print f.closed
f.write(txt)
f.close()
print f.closed
```

فى المثال السابق قمنا بفتح ملف iotest.xcd للكتابة، وكتبنا فى داخله محتويات المتغير txt الناتج

```
3
iotest.xcd
w
False
True
```

القراءة

```
f=open(FNAME, "r")
lines=f.readlines()
for line in lines:
    print "LINE = > ", line, #avoid printing a new line.
```

الناتج

```
LINE = >  
LINE = > line 1  
LINE = > line 2  
LINE = > line 3  
LINE = > line 4  
LINE = > line 5  
LINE = > line 6  
LINE = > line 7  
LINE = > some text  
LINE = > yada yada yada!
```

معاملات سطر الأوامر لبرنامجك

بكل بساطة مش هتحتاج غير ال argument vector ودى موجودة فى ال sys.argv لو فاكّر من السى

```
int main(int argc, char** argv){  
  
}
```

فللوصول لل argv استخدم ال argv الموجودة بال sys module وللحصول على عددهم استخدم len(argv)

```
striky@striky-desktop:~$ python myecho.py hello world 123 "78 yay"  
Number of arguments: 5  
myecho.py  
hello  
world  
123  
78 yay
```

myecho.py

```
#!/bin/python  
  
from sys import argv # arguments vector.  
  
print "Number of arguments: ", len(argv)  
  
for arg in argv:  
    print arg
```

os, os.path

فى عندنا 2 modules مهمين هما os, os.path
ضيفهم كالتالى

```
import os
import os.path as op
```

`os.uname(...)`
بتعيد tuple مكونة من (sysname, nodename, release, version, machine)

```
>>> print os.uname()

('Linux', 'striky-desktop', '2.6.24-21-generic', '#1 SMP Tue Oct 21
23:43:45 UTC 2008', 'i686')
```

`os.getcwd()`

بتعيد المسار الحالى

```
>>> print os.getcwd()

/home/striky/workspace/pytut/src
```

`os.getcwdu()`

مثل سابقتها ولكن بتعمل ريترن ب unicode

`os.environ`

هى dictionary مخزن فيه متغيرات البيئة environment مثل ال HOME, LOGNAME, PATH,.. etc للإطلاع عليهم

```
for key, val in os.environ.items():
    print key, " => ", val
```

للحصول على مفتاح بعينه

`os.getenv(key, default)`

للحصول على قيمة مفتاح ما في `environ`

```
print os.getenv("HOME")
```

ال `default` سيتم اعادةها في حالة عدم وجود المفتاح

انا root ؟

```
def isroot():  
    return os.getenv("USER")=="root"
```

طبعا تقدر تستخدم ال `keys, values methods` بال `environ` dictionary

`os.putenv(key, value)`

إضافة `key` لل `environment` بقيمة `value`

`os.unsetenv(key)`

حذف `key` ما

`os.chdir(to)`

بتقوم بنقل المسار الحالي `cwd` الى `to`

```
print os.getcwd()  
os.chdir("/home/striky")  
print os.getcwd()  
os.chdir("Music")  
print os.getcwd()  
  
#output  
/home/striky/workspace/pytut/src  
/home/striky  
/media/s3/Music
```

لاحظ ان `Music` هنا `symbolic link` تحت `home/striky/` ويشير ل `media/s3/Music/`

`os.listdir(path)`

بتقوم بإعادة `list` من مكونات ال `path`

```
>>> p=os.getcwd() #/home/striky/workspace/pytut/src  
>>> print os.listdir(p)  
['userstringtest.py', 'iosess.py', 'gcombo.py', 'iohelpers.py',  
'oopsample.py', 'iotest.xcd', 'gtk1.py', 'complpath.py']
```

`os.link(src,dest)`

بتنشئ `hard link` من `src` الى `dest`

<code>os.symlink(src, dest)</code>	بتنشئ symbolic link من src إلى dest
<code>os.unlink(path)</code>	حذف path
<code>os.remove(path)</code>	مثل <code>os.unlink</code>
<code>os.rmdir(path)</code>	لحذف مجلد معين
<code>os.rename(src, dest)</code>	اعادة التسمية
<code>os.removedirs(path)</code>	بتقوم بالحذف من اسفل لأعلى مثلا
<code>os.removedirs('foo/bar/baz')</code>	ستقوم بحذف مجلد baz اولا ثم bar ثم foo
<code>chmod(path, mode)</code>	بتقوم بتعديل ال mode على path
<code>chown(path, uid, gid)</code>	تحديد ال uid, gid على path معين
<code>print os.sep # /</code>	الفاصل العناصر المسار
<code>/home/striky</code>	
<code>print os.curdir #.</code>	المجلد الحالي وهى ال "."
<code>print os.altsep#None</code>	حرف فاصل بديل
<code>print os.pardir#..</code>	المجلد الأب وهى ال ".."
<code>print os.extsep#.</code>	الفاصل للإمتدادات وهو ال "."
<code>print os.pathsep#:</code>	الفاصل فى متغير ال PATH وهنا:
<code>print repr(os.linesep)#\n</code>	الفاصل بين السطور وهنا هو ال "\n"

```

p=os.getcwd() #/home/striky/workspace/pytut/src
F=p+r'/'+"iohelpers.py"
print op.basename(F)

print op.isfile(F)
print op.islink(F)
print op.isabs(F)
print op.isdir(F)

```

```

print op.isdir(p)
print op.ismount("/media/s3")
print op.abspath(F)
print op.dirname(F)
print op.split(F)
print op.splitdrive(F)
print op.splitext(F)
print op.exists(F+"xx")#Nope!
print op.getatime(F) #last access time
print op.getmtime(F) #last modification time
print op.getsize(F) #file size.

print op.join("/home", "striky", "Music")

```

النتائج

```

iohelpers.py
True
False
True
False
True
True
/home/striky/workspace/pytut/src/iohelpers.py
/home/striky/workspace/pytut/src
('/home/striky/workspace/pytut/src', 'iohelpers.py')
('', '/home/striky/workspace/pytut/src/iohelpers.py')
False
1226557143.0
1226557142.0
521
/home/striky/Music
('/home/striky/workspace/pytut/src/iohelpers', '.py')

```

exists(path)

هل المسار موجود ؟

isfile(path)

هل ال path ملف؟

isdir(path)

هل ال path مجلد؟

islink(path)

هل ال path عبارة عن link ؟

ismount(path)

هل هو عبارة عن نقطة ضم ؟

isabs(path)

هل هو المسار بالكامل ؟

basename(path)

القاعدة فى المسار

abspath(path)

المسار المطلق

dirname(path)

اسم المجلد

getatime(path)

الحصول على توقيت ال last access

getmtime(path)

الحصول على توقيت ال last modification

getctime(path)

الحصول على توقيت ال last change او ال last creation اذا كان على windows

getsize(path)

الحصول على مساحة path

join(a)

لدمج مكونات ال path بإستخدام الفاصل المناسب

split(path)

تقوم بإعادة tuple مكونة من ال dirname وال basename

splitdrive(path)

تقوم بإعادة tuple مكونة من ال drive, وباقى المسار

splittext(path)

تقوم بإعادة tuple مكونة من المسار كامل بدون الإمتداد و الإمتداد

expanduser(path)

تقوم بإستبدال ال tilde (~) بمسار المنزل للمستخدم

```
expanduser("~/Desktop")
'/home/ahmed/Desktop'
```

File Pointer

يمكن تكون مليت من استخدام FileHandler.write

```
>>>F = open(fileName, 'w')
>>>print >> F, 'Hola' #It will write the Hola word to the file that we
opened
>>>F.close()
>>>F=open(fileName, 'r')
>>>for line in F.readlines(): print line
```


Hola

على فرض إنك هتعمل File ودا معناه إنك هتستخدم ال 'w' permission

```
>>> f = open('C:\\2.txt', 'w') #Open 2.txt for writing mode.
>>> print >> f, 'Hola!' #Add 'Hola' to it
>>> print >> f, 'Hello!' # same
>>> print >> f, 'Using File Pointer !' #the same
>>> f.close() #closing the file handler.
>>> f = open('C:\\2.txt', 'r') # Open in reading mode.
>>> for line in f.readlines(): #iterates through the file lines
    print line #printing each line.
Hola!
Hello!
Using File Pointer !
```

```
>>>f.close() #closing the handler.
```

طب تمام .. هنتفتح ال File مرة ثانية ولكن فى ال Append mode -وضع الإضافة

```
>>> f=open('C:\\2.txt', 'a')
>>> print >> f, 'Programming Fr34ks r0x!'
>>> print >> f, 'File pointers are', #Note : this comma is used to avoid
printing a new line.
>>> print >> f, ' so great' # added to the line 'File pointers are'
>>> f.close()
```

نقرا اللى مكتوب فى ال File بإتنا نعمل Iteration بسيطة على ال filehandler.readlines method

```
>>> f=open('C:\\2.txt', 'r')
>>> for line in f.readlines():
    print line
```

Hola!

Hello!

Using File Pointer !

Programming Fr34ks r0x!

File pointer are so great

Chapter 9 (Modules/Packages:Charging a Battery)

بايثون مشهورة بعبارة batteries included فاللغة نفسها لاتقدم سوى ال syntax ولكن لإستخدامها تم تقسيم الخدمات الى ملفات خارجية بإسم modules وإذا كانت modules مترابطة تم تخزينها فى package على سبيل المثال بايثون لاتقوم بتوفير الدوال الخاصة بالرياضيات مباشرة

```
>>> cos(30)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
```

معنى السابق وجود خطأ فى السطر الأول من ال module "أبسط وحدة لتطبيق بايثون" ونوع الخطأ هو عدم وجود دالة ال cos فى مساحة البرنامج
إذا ؟ كيف اتعامل مع الدوال الرياضية ؟
يوجد فى مكتبات بايثون الأساسية modules لعمل معظم انواع التطبيقات قواعد بيانات شبكات حسابات معالجة بيانات .. الخ الخ

Importing a Module:

قم بإستدعاء ال module المسماة math وهى وحدة تشمل العديد والعديد للقيام بالعمليات الحسابية بإستخدام import

```
import math
```

هكذا قمنا بإستدعاء الوحدة لمساحة البرنامج
لاحظ قد لاتكون ال module موجود فينتج ImportError

```
>>> try:
...     import mymath
... except ImportError, e:
...     print e
...
No module named mymath
```

تقدر تستخدم ال magical import لإستدعاء الموديلز أيضا

```
>>> msys=__import__('sys')
```

تقدر تستخدم as لعمل تسميه مختلفة (لموديل او احد عناصر موديل الخ) هنا للموديل مثل المثال السابق msys كبديل ل sys

```
>>> import sys as msys
```

وهكذا

Finding Nemo

اين توجد هذه الوحدات؟ هل يوجد شروط لإستدعاءها ام ماذا ؟
المفسر لايعلم عن كل ملف بايثون على جهازك ولكن هناك بعض الأماكن التي يبحث فيها قبل ان يرسل لك ال
ImportError مثل المجلد الحالى او مجلد بايثون الافتراضى او مجلد site-packages (يفضل استخدامه عند اضافة
اي وحدات خارجية لبايثون)
للحصول على القائمة كاملة التى يبحث فيها المفسر استدعى sys.path

```
>>> import sys
>>> sys.path
['', '/usr/lib/python2.5/site-packages/Tempita-0.2-py2.5.egg', '/usr/lib/python2.5/site-packages/Mako-0.2.2-
py2.5.egg',
....
....]
```

First Module

يفضل دائما عند انشاء موديل او اى سكربت انك تنشئ هيدر مشابه للتالى

```
#####
# Author: Ahmed Youssef
# License: GPL V3
# Module: firstmodule
# Purpose: Learning modules
# Date: 12-20-2008
#####
```

انشئ ملف firstmodule.py وعرف فيها مجموعة من الدوال كالتالى

```
def aloha():
    print "Aloha!"

def adios():
    print "Adios!"
```

ايه الهدف من ال modules قلنا ؟ فصل التطبيق لأكثر من جزء لإمكانية استخدام خدماته اكثر من مرة وايضا
تسهيل تقسيم العمل مثال شخص يكون مسئول عن جزئية التعامل مع قاعدة البيانات وشخص اخر مسئول عن
الواجهة وهكذا

اوکی کتبنا المودیل ماذا الآن ؟
انشئ ملف جديد وليكن greeter.py هيثم فيه استخدام الدوال aloha, adios الموجودة في firstmodule.py

```
import firstmodule

firstmodule.aloha()
firstmodule.adios()
```

عند التنفيذ

```
striky@striky-desktop:~/workspace/pytut/src$ python greeter.py
Aloha!
Adios!
```

وهكذا

using from

ربما لاتريد ان تكتب اسم الموديل في كل مرة او ربما ماتريده او دالة معينة فقط الخ

```
from firstmodule import aloha, adios

aloha()
adios()
```

او ربما تريد استدعاء جميع محتويات module ما قم فقط بتنفيذ جملة الإستدعاء
*from somemodule import **
حيث * تعنى جميع المحتويات

Reloading

ربما تستخدم بيئة تفاعلية او ربما نظام معين يكون عملية اعادة التشغيل فيه مكلفة او غيرها وقمت بتعديل module معينة قم بتنفيذ الدالة reload لإعادة تحميل ال module بالتعديلات الجديدة

Walk on the main

احترس من اختبار الكود في ال module وإلا سيتم تنفيذه بمجرد استدعائها
إذا عدلت كود ال firstmodule.py الى

```
def aloha():
    print "Aloha!"

def adios():
    print "Adios!"
```

```
aloha()
```

وقمت بتنفيذ ال greeter.py ستجد الناتج كالتالى

```
striky@striky-desktop:~/workspace/pytut/src$ python greeter.py
Aloha!
Aloha!
Adios!
```

مع اننا استدعينا aloha مرة واحدة ولكن تم تنفيذها مرتان وذلك بسبب استدعائها فى ال firstmodule والحل ؟ الاستطيع اختبار الكود ؟

ليس الأفضل ان تختبر اولا اذا كانت ال module هى التى يتم تنفيذها كملف رئيسى او مجرد مستدعاه فى ملف اخر ؟ كيف اعلم هذا ؟

بكل بساطة توفر لك ال modules متغير خاص بإسم __name__ يحوى اسم ال module الحالية وهو "__main__"

قم بإضافة شرط فى نهاية ال firstmodule كالتالى

```
if __name__=="__main__":
    #Testing...
    aloha()
```

وتم حل المشكلة عند تنفيذك لل greeter.py سيتم طباعة Aloha! و Adios! وعند تنفيذك لل firstmodule فتكون هى السكرتير الرئيسى الذى سيتم تنفيذه او ال __main__ فيتم تنفيذ كود الإختبار او ايا كان (:

It's all about __all__

للتحكم فيما يمكن استدعاه من وحدة ما تستطيع استخدام متغير خاص بإسم __all__
["__all__"="aloha__

Packages

للآن جيد ماذا لو زاد عدد ال modules تستطيع بالتأكيد انشاء المئات من تلك الملفات ولكن يجب عليك تحريم ال modules المترابطة مثلا اذا كنا ننشئ مشروع عن التعامل مع قواعد بيانات مختلفة oracle, sqlite, mysql وغيرهم ليس الأفضل تجميعهم فى حزمة ما بإسم databases مثلا ؟ ويتم استدعائها

```
import databases.mysql
```

او مثلا


```
from databases import mysql
```

بكل تأكيد هذا اكثر تنظيما

```
myfirstpackage/  
  |-- __init__.py  
  |-- mysql.py  
  |-- oracle.py  
  `-- sqlite.py
```

لدينا حزمة بإسم myfirstpackage وتشمل 4 ملفات
1- ال __init__ وفيه يتم تحديد الوحدات التي نريد تحميلها مباشرة وربما بعض المتغيرات الأساسية ؟
2- 3 وحدات بإسم mysql.py, oracle.py, sqlite.py معرفين كالتالين

الملف mysql.py

```
def about():  
    print "mysql module."
```

الملف oracle.py

```
def about():  
    print "oracle module."
```

الملف sqlite.py

```
def about():  
    print "sqlite module."
```

الملف __init__

```
print "__init__ myfirstpackage"  
import mysql
```

```
VERSION="1.42.0"
```

لاحظ اننا قمنا بعمل import ل mysql مباشرة وحددنا متغير بإسم VERSION

انشئ ملف dbtester.py

```
import myfirstpackage

print dir(myfirstpackage)
print myfirstpackage.VERSION
myfirstpackage.mysql.about()
myfirstpackage.sqlite.about()
```

ستجد الناتج مشابه للتالى

```
striky@striky-desktop:~/workspace/pytut/src$ python dbtester.py
__init__ myfirstpackage
['VERSION', '__builtins__', '__doc__', '__file__', '__name__', '__path__', 'mysql']
1.42.0
mysql module.
Traceback (most recent call last):
  File "dbtester.py", line 6, in <module>
    myfirstpackage.sqlite.about()
AttributeError: 'module' object has no attribute 'sqlite'
```

وبكل تأكيد لن سيتم رفع استثناء AttributeError بسبب عدم استدعاء sqlite للساحة إلا اذا قمت بإضافتها يدويا

```
import myfirstpackage.sqlite
```

Platform

كثيرا ما نحتاج للحصول على معلومات عن النظام الذي يعمل عليه البرنامج (لإختبار التوافقية ، الإعتماديات او ربما العلم بالشئ)
تقدم لنا بايثون وحدة بإسم platform

مثال

```
striky@striky-desktop:~/workspace/pytut/src$ python platformreport.py
[architecture => ('32bit', 'ELF')]
[dist => ('debian', 'lenny/sid', '')]
[java_ver => ('', ('', ('', '')), ('', ''))]
[libc_ver => ('glibc', '2.4')]
[mac_ver => ('', ('', ''), '')]
[machine => i686]
[node => striky-desktop]
```

```
[platform => Linux-2.6.27-9-generic-i686-with-debian-lenny-sid]
[processor => ]
[python_build => ('r252:60911', 'Oct 5 2008 19:24:49')]
[python_compiler => GCC 4.3.2]
[python_version => 2.5.2]
[python_version_tuple => ['2', '5', '2']]
[release => 2.6.27-9-generic]
[system => Linux]
[uname => ('Linux', 'striky-desktop', '2.6.27-9-generic', '#1 SMP Thu Nov 20 21:57:00 UTC 2008', 'i686', '')]
[version => #1 SMP Thu Nov 20 21:57:00 UTC 2008]
[win32_ver => ('', '', '')]
```

تستطيع بكل تأكيد كتابة كل function مثلا

```
platform.dist()
platform.machine()
platform.uname()
```

الخ الخ
ولكن ربما نستخدم حيلة صغيرة لإستدعاءهم جميعا ؟

```
import platform
for s in dir(platform):
    if not s.startswith("_"): #If it does not start with an underscore.
        f=getattr(platform, s) #Fetch the attribute (should be a function..)
        try:
            print "[%s => %s]"%(s, f()) #Prints attr, returned value
        except:
            pass #Global catch for functions requires params(e.g popen).
```

هنا نقوم بعرض محتويات platform بإستخدام dir ونحصل على ال function object بإستخدام getattr من الوحدة وبإسم الدالة ونقوم بتنفيذها (مجرد استدعائها بعد الحصول عليها) وبس كذا

Chapter 10 (Databases)

Python/MySQL

MySQLdb هي Interface بتسمحك بالتعامل مع MySQL من خلال بايثون او كى القصة بدأت ان اتعمل wrap لل MySQL C APIs بصورة OO فى امثلة لشكل ال APIs

<http://mysql-python.sourceforge.net/MySQLdb.html#id5>

جميل احنا تعاملنا كله من خلال ال MySQLdb وهي عملت wrap ل mysql_ انترفيس لضمان التكافئ مع ال DB API specifications هتكون [PEP 249](#)

اولا ال connect

connect(...)

هي المسئولة عن انشاء الإتصال بقاعدة البيانات وتعمل ريترن ب Connection Object لازم عشان ننشئ اتصال نحتاج شوية معلومات زي ال

- host
ودا بيعبر عن الهوست اللى هيثم الإتصال عليه (الإفتراضى localhost)

- user
اسم المستخدم (الإفتراضى المستخدم الحالى)

- passwd
الباسورد الخاص بإسم المستخدم (افتراضى لا يوجد)

- db
قاعدة البيانات (الإفتراضى لآ)

- port
زي مانت عارف ال MySQL ليها server ودا البورت بيعبر عن ال TCP Port اللى بيستخدمه السرفر وافتراضيا 3306 (عدله لو قمت بتغييره!)

- ssl
لإنشاء SSL Connection (ملحوظة: throws exception: لو غير مدعم!)

- compress
لتفعيل ال compression (الإفتراضى لآ)

- connect_timeout
تحدد زمن ال timeout

- charset
إذا تم اضافتها هيثم تضمين use_unicode=True

- sqlmode
لتحديد ال sqlmode (يفضل تراجع MySQL documentation)

-
تقدر تحدد الكثير من الإعدادات كل اللى عليك تراجع ال MySQL Documentation

apilevel

بتحدد اى DB API مدعمة ؟ الحالى 2.0

1- اعمل import ل MySQLdb كالتالى

```
>>> import MySQLdb as ms
```

--انا خليت ms ك alias طبعاً انت حر فى كيفية الإستدعاء

```
>>> ms.apilevel  
'2.0'
```

```
>>> ms.threadsafety  
1
```

حسناً ايه معنى ال threadsafety اصلاً ؟

هى عبارة عن رقم بين [0, 3]

:0

يعنى ان ال threads مش تقدر تشارك فى ال module

:1

ان ال threads تقدر تشارك فى ال module ولكن مش ال connections

-2

ان ال threads تقدر تشارك فى ال module وال connections ولكن مش ال cursors (هنتكلم عنها)

-3

اعلى شئ وهى امكانية المشاركة الكاملة فى ال module, connections و ال cursors

paramstyle

سترينج يعبر عن طريقة التعامل مع ال queries من خلال المدخلات يعنى مثلاً احياناً فى ناس بتستخدم
format ؟ (علامة استفهام) او طريقة s% او حتى استخدام الأرقام 1: و 2: وهكذا (حسب الترتيب) فالإفتراضى هو format

```
>>> ms.paramstyle  
'format'
```

ال Exceptions/Errors المرتبطة هنا هما Errors مشتقين من Error

ينقسمو الى

1- InterfaceError ودا بيعبر عن ايرور(خطأ) فى الإنترفيس المستخدمة مش ال داتايز

2- DatabaseError بيعبر عن ايرور(خطأ) فى قاعدة البيانات وتنقسم لكذا شئ اهمهم

DataError مشاكل مع الداتا

OperationalError ايرور(خطأ) اثناء تنفيذ عملية معينة

ProgrammingError فشل فى تنفيذ sql command معين

NotSupportedError عملية غير مدعومة!

1- ال Connection Objects

هى كائن بيتم اعاتها عند الإتصال بقاعدة بيانات ولها عدة ميثودز

1- close()

لغلق الإتصال

2-commit()

لتنفيذ ال transaction الحالى (مش هتفرق فى حال مش فى تدعيم لل transactions اصلاً او ال auto commit مفعلة)

3-rollback()

الغاء ال transaction الحالى (نفس الملحوظة السابقة ولكن لاحظ فى حال انهاء الإتصال وعدم ال commit هيتم عمل rollback اتوماتيك!)

4- cursor()

5- set_sql_mode(sqlmode)

بتحدد ال sqlmode (يفضل تراجع ال MySQL documentation)

6- set_character_set(charset)

تحديد ال charset

للحصول على cursor (هنتعرض ليه)

2- ال Cursor Objects

بكل بساطة طالما عندك اتصال بقاعدة بيانات بيقه انت محتاج * تتفاعل * معاها عن طريق تنفيذ SQL statements معينة فال cursor بيقوم بدور الوسيط بينكم بيسمحلك تنفذ SQL statements ويسمحلك تتعامل مع ال rows الناتجة

ال Cursor ليه شوية fields و methods اهمهم

execute(sqlQuery, args)

بتقوم بتنفيذ Sql Statement على قاعدة البيانات ويتم تجهيزها قبل التنفيذ ب args فى حال لو انت قررت تعمل late binding

where name=?

او

where name=:name

وهكذا

rowcount

عدد الصفوف الناتجة من تنفيذ اخر امر

callproc(proc, args)

لإستدعاء stored procedure !

fetchone()

الحصول على صف واحد من الناتج

fetchmany()

للحصول على عدد معين من الصفوف تم تحديده من خلال arraysize (تبع ال cursor وتعتبر عن عدد الصفوف) arraysize

بتعتبر عن رقم الصفوف اللى هيثم اعادته من خلال ال fetchmany method

fetchall()

للحصول على كل ال الصفوف الناتجة

rownumber

ال index الخاص ب ال cursor !

نختم بمثال (جزء من برنامج حالى استخدمت فيه MySQL ك backend)

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL auto_increment,  
  `username` varchar(50) NOT NULL,  
  `password` varchar(50) NOT NULL,  
  `state` tinyint(2) NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `username` (`username`)  
) ;
```


محتاجين تطبيق crud على الجدول دا من خلال Python/MySQL

Create/Read/Update/Delete

1- استدعى MySQLdb

```
import MySQLdb as ms
```

2- انشئ صف DBMan

```
class DBMan(object):

    def __init__(self, dbname="pyim"):

        self._dbname=dbname
        self._sqlconnection=ms.connect(host="localhost",
                                       user="root",
                                       passwd="",
                                       db="pyim")

        self._sqlcursor=self._sqlconnection.cursor()
```

لاحظ في ال constructor محدين اسم ال database ك pyim
وحددنا البيانات وانشئنا Connection object باسم self._sqlconnection
وحصلنا على cursor منه باسم self._sqlcursor

إضافة مستخدم جديد

```
def addUser(self, username, pwd, state=State.Offline):
    #State.Offline=1

    md5edpass=self._md5(pwd)
    sqlstmt="INSERT INTO users VALUES(NULL, '%s', '%s', %d)"%(username, md5edpass, state)
    try:
        self._sqlcursor.execute(sqlstmt)
        self._sqlconnection.commit()
    except Exception, e:
        print e.message
```

حذف مستخدم

```
def deleteUser(self, username):

    sqlstmt="DELETE FROM users WHERE username='%s'"%username
    try:
        self._sqlcursor.execute(sqlstmt)
        self._sqlconnection.commit() #commit
    except Exception, e:
        print e.message
```

تسجيل دخول

```
def login(self, username, pwd):

    md5edpass=self._md5(pwd)
    sqlstmt="SELECT username, password FROM users WHERE username='%s' AND password='%s'"%(
username, md5edpass)
    self._sqlcursor.execute(sqlstmt)
    if self._sqlcursor.fetchone():
        self.setState(State.Online, username)
```

تغيير الحالة

```
def setState(self, state, username):

    sqlstmt="UPDATE users SET state=%d WHERE username='%s'"%(state, username)
    try:
        self._sqlcursor.execute(sqlstmt)
    except Exception, e:
        print e.message
```

عرض الكل

```
def getAllUsers(self):

    sqlstmt="SELECT username, state FROM users"
    self._sqlcursor.execute(sqlstmt)
    for row in self._sqlcursor.fetchall():
        yield row[0], row[1]
```

ملحوظة: انا هنا ناقشت MySQLdb من خلال مفهوم ال DB API بمعنى ان نفس المبادئ هتلقها ثابتة فى اى انترفيس هتستخدمها
ومازلنا منتظرين DB API 3

Refs:

[MySQLdb 1.2.2 docs](#)

[Python DB API Specifications v2](#)

PySQLite

ماهى PySQLite؟
هى interface ل SQLite من خلال ال Python

للتحميل اضغط هنا

للتثبيت مثل اى Lib

python setup.py install

لنبدأ

1- نحتاج نعمل import لل module
PySQLite هتعملها import كـ pysqlite2 ولكن ال lib دى بردو اللى يهمنى فيها هو sub-module بإسم dbapi2

الكتابة

2- نحتاج نعمل Connection مع DB تمام ؟ ال db نفسها عبارة عن file عادى جدا -فى حال عدم وجوده هيثم إنشاء file جديد- فلعمل ال Connection هحتاج نستخدم ال connect method الموجودة بال dbapi2

```
#!/bin/python  
  
from pysqlite2 import dbapi2 as SQLite
```

نعمل connect ال connect ميثود بتنشئ file فى حال عدم وجوده وإذا موجود هيتعمل return بيه

```
dbConnection=SQLite.connect("mydb.sqlite")
```

كدا انشأنا ال connection بنجاح

ملحوظة: تقدر تعمل Quick Access DB على ال Memory

```
memConnection=SQLite.connect(":memory:")
```

بعد ما أنشأنا ال Connection محتاجين نعمل Cursor عشان نستخدمه فى التعامل مع ال DB

```
cursor=dbConnection.cursor() #gets a cursor object..
```

نريد ان ننشئ Table وليكن بإسم Info ويشمل 3 Fields مثلا

id: integer, primary Key

name: varchar(50)
phone: varchar(10)

جميل بيقة هحتاج SQL Statement

```
sqlStmt='CREATE TABLE info (id INTEGER PRIMARY KEY, name VARCHAR(50), phone VARCHAR(10))'
```

ولتنفيذ ال SQL Statement نستخدم ال execute method الخاصة بال cursor object

```
>>> cursor.execute(sqlStmt)
<pysqlite2.dbapi2.Cursor object at 0x0128B230>
>
```

ندخل بعض ال داتا

```
>>> cursor.execute('INSERT INTO info VALUES(null, "ahmed youssef", "12345678")')
<pysqlite2.dbapi2.Cursor object at 0x0128B230>

>>> cursor.execute('INSERT INTO info VALUES(null, "3amer mohamed", "41234114")')
<pysqlite2.dbapi2.Cursor object at 0x0128B230>
```

نقدر ندخل ال fields كالتالى ..

```
>>> username="guru"
>>> phone ="36987452"
```

كل اللى عليك تمرر علامة إستفهام وفى ال 2nd argument تخليها tuple مكونة من ال vars اللى عايز تدخلها ..

```
>>> cursor.execute('INSERT INTO info VALUES(null, ?, ?)', (username, phone)) #replaced...
<pysqlite2.dbapi2.Cursor object at 0x0128B230>
```

بعد ماعدلنا او اضعنا لازم نستدعى ال Commit method لحفظ التعديلات دى ..

```
>>> dbConnection.commit()
```

ملحوظة: إذا حببت تخلى التعديلات يتم تنفيذها اوتوماتيك
ضعيف فى ال connect ميثود الخاصة بإنشاء ال connection التالى

autocommit=1

فى حالة قيامك بتعديل ما وحييت ترجع فيه بنستخدم ال rollback method

بعد إنتهائك افعل ال cursor, connection

```
cursor.close()
dbConnection.close();
```

القراءة

كالعادة لازم نعمل connect على db وننشئ ال connection
ونعمل cursor object بإستخدام cursor ميثود الموجودة بال connection object
ننفذ بعض ال sql statements ولكن هنا هنخليها عبارة عن إستعلامات بسيطة

ننشئ ال connection

```
>>> dbConnection=SQLite.connect("mydb.sqlite") #reopen the db..
```

ننشئ cursor

```
>>> cursor=dbConnection.cursor()
>>> #let's query the db..
```

sql statement ليتم تنفيذها

```
>>> sqlStmt='SELECT * from info'
```

تنفيذ ال sqlStmt

```
>>> cursor.execute(sqlStmt)
```

fetchall هى ميثود بتعيد كل ال rows على صورة tuples ف list

```
>>> cursor.fetchall()
[(1, u'ahmed youssef', u'12345678'), (2, u'3amer mohamed', u'41234114'), (3, u'guru',
u'36987452')]
```

او تقدر تعمل شئ مشابه لكدا بإنك ت iterate على كل الصفوف اللى موجودة بال result

```
>>> for row in cursor:
    #id, name, phone
```

```
print "-----"  
print "ID: ", row[0]  
print "Name: ", row[1]  
print "Phone: ", row[2]
```

```
-----  
ID: 1  
Name: ahmed youssef  
Phone: 12345678  
-----  
ID: 2  
Name: 3amer mohamed  
Phone: 41234114  
-----  
ID: 3  
Name: guru  
Phone: 36987452
```

لاحظ إنك تقدر تتعامل معاها ب next. لأنها iterator

```
>>> cursor.next()  
(1, u'ahmed youssef', u'12345678')  
>>> cursor.next()  
(2, u'3amer mohamed', u'41234114')
```

fetchmany(num)

بتعيد عدد معين من الصفوف

```
>>> ret=cursor.fetchmany(2)  
>>> ret  
[(1, u'ahmed youssef', u'12345678'), (2, u'3amer mohamed', u'41234114')]
```

fetchone()

بتعيد صف واحد

```
>>> one=cursor.fetchone()  
>>> one  
(3, u'guru', u'36987452')
```

جميل جدا .. طب وإذا حبيت اخزن user defined type ؟
بكل بساطة اعمل ال class بتاعك الأول

```
class Person(object):

    def __init__(self, name, phone):
        self.name=name
        self.phone=phone
```

ننشئ connection و cursor ولكن ننبه ال database انها تعمل parse لل declared types زي ال Person مثلا .. هنغير شوية ونتعامل مع ال memory

```
#create a connection.
memConnection=SQLite.connect(':memory:', detect_types=SQLite.PARSE_DECLTYPES)

#cursor
cursor=memConnection.cursor()
```

الآن ننشئ table بحيث إنه ياخذ 2 fields وهم ال ID, information

```
cursor.execute("CREATE TABLE test (id INTEGER PRIMARY KEY, p person)")
```

جميل جدا .. ناقص إننا نحدد إزاي ال object الخاص بنا يتحول ل string وازاي نجمع ال data بتاعته تاني من ال string دا
ملحوظة: إحنا بنتكلم على مجرد text بإستخدم toString method مثلا .. مش serializing objects او Pickling

```
def adaptPerson(person):
    return "%s;%s" %(person.name, person.phone)
```

وكيفية التجميع .. بكل بساطة إحنا حولنا ال fields بتاعت ال Person object ل string ودمجناهم ب ; .. بيقع نقدر نجمعهم بإننا نفصل ال ; ونمرر ال قيم الخاصة بال fields دي لل Constructor وننشئ object منها

```
def convToPerson(text):
    name, phone=map(str, text.split(";"))
    return Person(name, phone)
```

بعد ما عملنا الميثودز الخاصة بالتحويل والتجميع .. كل اللي ناقص اننا نبليغ SQLite بكدا

```
SQLite.register_adapter(Person, adaptPerson)
SQLite.register_converter("person", convToPerson)
```

ننشئ بعض الكائنات


```
p1=Person("ahmed", "12345678")
p2=Person("rul3z", "89745632")
```

ونضيفهم للجدول

```
cursor.execute('INSERT INTO test VALUES(null, ?)', (p1, ))
cursor.execute('INSERT INTO test VALUES(null, ?)', (p2, ))
```

نجرب نستعلم عن الموجودين

```
#select..
cursor.execute('SELECT * from test')
for row in cursor:
    print row

#output:
(1, (ahmed;12345678))
(2, (rul3z;89745632))
```

نقفل ال cursor, connection

```
#clean-up
cursor.close()
memConnection.close()
```

الكود النهائي

```
#!/bin/python

from pysqlite2 import dbapi2 as SQLite

#dbName='myobjDBTest.sqlite'
#create a connection.
#dbConnection=SQLite.connect(dbName, detect_types=SQLite.PARSE_DECLTYPES)
memConnection=SQLite.connect(':memory:', detect_types=SQLite.PARSE_DECLTYPES)
#cursor
cursor=memConnection.cursor()

class Person(object):
```

```

def __init__(self, name, phone):
    self.name=name
    self.phone=phone

def __repr__(self):
    return "(%s;%s)" %(self.name, self.phone)

#define a method to register it..

def adaptPerson(person):
    return "%s;%s" %(person.name, person.phone)

def convToPerson(text):
    name, phone=map(str, text.split(";"))
    return Person(name, phone)

SQLite.register_adapter(Person, adaptPerson)
SQLite.register_converter("person", convToPerson)

p1=Person("ahmed", "12345678")
p2=Person("rul3z", "89745632")

#create a test table..
cursor.execute("CREATE TABLE test (id INTEGER PRIMARY KEY, p person)")

#add
cursor.execute('INSERT INTO test VALUES(null, ?)', (p1, ))
cursor.execute('INSERT INTO test VALUES(null, ?)', (p2, ))

#select..
cursor.execute('SELECT * from test')
for row in cursor:
    print row

#clean-up
cursor.close()
memConnection.close()

```

وللمزيد راجع التالى :

<http://www.devshed.com/c/a/Python/Using-SQLite-in-Python/>
<http://www.initd.org/tracker/pysqlite/wiki/basicintro>
<http://www.initd.org/pub/software/python/used-functions>

ملحوظة: sqlite3 اصبحت وحدة اساسية فى بايثون تقدر تطبق نفس الفصل عليها

ORMs

ما معنى ORM ؟ هى اختصار ل Object Relational Mapping حيث تقوم بتمثيل بياناتك على هيئة objects من classes بدلا من صفوف من جداول
فبإختصار ال class يعبّر عن بنية الجدول وال object يعبّر عن كل صف فى الجدول
تستطيع ايضا من خلال ال ORMs ادارة العلاقات بين الجداول وبعضها يتيح الوراثة!!

Storm

هنبداً ب storm وهى ORM مقدم من canonical

اولا بعض الأساسيات

```
>>> from storm.locals import *
```

ننشئ صف ليمثل لنا جدول للكتب

```
>>> class Book(object):  
...     __storm_table__="book"  
...     id=Int(primary=True)
```

```
... name=Unicode()  
... npages=Int()
```

ننشئ قاعدة بيانات

```
>>> db=create_database("sqlite:")
```

ننشئ كائن مخزن (لنتعامل مع عناصر قاعدة البيانات)

```
>>> store=Store(db)
```

ننشئ الجدول المعبر عن الكتب

```
>>> store.execute("CREATE TABLE book (id INTEGER PRIMARY KEY, name VARCHAR,  
npages INTEGER)")
```

ننشئ كائن

```
>>> rbook.name=u"Introduction to Ruby"  
>>> rbook.npages=210  
>>> print rbook.id, rbook.name, rbook.npages  
None Introduction to Ruby 210
```

فلنضيفه الآن

```
>>> store.add(rbook)  
<__main__.Book object at 0xb78aecac>  
>>> print rbook.id, rbook.name, rbook.npages  
None Introduction to Ruby 210  
>>>  
>>> pybook.name=u"PyGuide"  
>>> pybook.npages=230  
>>> print pybook  
<__main__.Book object at 0xb78ae80c>  
>>> store.add(pybook)  
<__main__.Book object at 0xb78ae80c>
```

الحصول على سجل ما

```
>>> pythonbook=store.find(Book, Book.name==u"PyGuide").one()  
>>> pythonbook.name  
u'PyGuide'
```

الطريقة one تحصل على صف واحد
او ربما البحث بال primary key الخاص به

```
>>> store.get(Book, 1).name  
u'Introduction to Ruby'
```

الطريقة flush كالعادة لعمل flush

```
>>> store.flush()  
>>> store.get(Book, 1).id  
1  
>>> pythonbook=store.find(Book, Book.name==u"PyGuide").one()  
>>> pythonbook.id  
2
```

الطريقة commit تستخدم لتحقيق اى تعديل على اى كائن
التعامل مع الكائنات افضل كثيرا من جمل SQL المملة وتساعد على تجنب الكوارث وتسهل امكانية النقل من
قاعدة بيانات لأخرى بكل سهولة

للمزيد تابع [/https://storm.canonical.com](https://storm.canonical.com)

SQLObject

ايضا مثال رائع لل ORMs هو SQLAlchemy مشابه ل storm

1- استدعاء المكونات ل sqlalchemy

```
>>> from sqlalchemy import *  
>>>
```

2- انشاء Hub ليعالج الإتصال القادم من العنوان sqlalchemy://memory: وهى مسار قاعدة بيانات sqlalchemy مخزنة فى
الذاكرة

```
>>> sqlalchemyhub.processConnection=connectionForURI('sqlalchemy://memory:')
```

```
>>>
```

3- نكتب صف يعبر عن الجدول

```
>>> class Book(SQLObject):  
...     title=StringCol()  
...     npages=IntCol()
```

4- ننشئ الجدول

```
>>> Book.createTable()  
[]
```

5- ننشئ كائنات من الصف Book

```
>>> rbbook=Book(title="Introduction to Ruby", npages=230)  
>>> rbbook.title  
'Introduction to Ruby'  
>>> rbbook.npages  
230  
>>> rbbook  
<Book 1 title="Introduction to ..." npages=230>  
  
>>> pybook="PyGuide"  
>>> pybook=Book(title="PyGuide", npages=330)  
>>> pybook=Book(title="PyGuide", npages=330)  
>>> pybook  
<Book 2 title='PyGuide' npages=330>
```

للحصول على كتاب ما بإستخدام ال id استخدم الطريقة get

```
>>> Book.get(1)  
<Book 1 title="Introduction to ..." npages=230>  
>>> book=Book.get(2)  
>>> book  
<Book 2 title='PyGuide' npages=330>
```

للإستعلام استخدم الطريقة select

```
>>> books=Book.select()  
>>> list(books)  
[<Book 1 title="Introduction to ..." npages=230>, <Book 2 title='PyGuide'  
npages=330>]
```

هنا حصلنا على جميع الكائنات من الصف Book

```
>>> rbbooks=Book.select(Book.q.title=="Introduction to Ruby")
>>> list(rbbooks)
[<Book 1 title="Introduction to ..." npages=230>]
```

هناك selectBy ايضا بديلة ل select تابع المثال التالى للحصول على كائنات الكتب التى عدد صفحاتها 230

```
>>> pybook
<Book 2 title='PyGuide' npages=330>
>>> pybook.npages=230
>>> pages230=Book.selectBy(npages=230)
>>> list(pages230)
[<Book 1 title="Introduction to ..." npages=230>, <Book 2 title='PyGuide'
npages=230>]
```

للمزيد تابع [/http://www.sqlobject.org](http://www.sqlobject.org)

SQLAlchemy

TODO

Chapter 11 (Parsing Data)

XMLing with Python

ملفات ال xml من اهم الملفات اللى بنتعامل معاها بصورة شبه يومية وبايثون من انسب الحلول للتعامل معاها..
فى اكثر من باكيج للتعامل مع ال Markups
<http://docs.python.org/lib/markup.html>

على كل حال فى هذه الجزئية سنتحدث عن كيفية معالجة واستخلاص البيانات من ملفات ال XML

على فرض عندنا ملف كالتالى

```
<?xml version="1.0"?>

<!DOCTYPE books SYSTEM "books.dtd">
<?xml-stylesheet type="text/xsl" href="books.xsl"?>

<books>
  <book id="1">
    <name>Introduction to Python</name>
    <author>Ahmed Youssef</author>
    <price>80</price>
  </book>
  <book id="2">
    <name>Introduction to Java</name>
    <author>Wael Muhammed</author>
    <price>130</price>
  </book>
  <book id="3">
    <name>Introduction to Ruby</name>
    <author>Ahmed Youssef</author>
    <price>70</price>
  </book>
  <book id="4">
    <name>Introduction to Linux Programming</name>
    <author>Ahmed Mostafa</author>
    <price>90</price>
  </book>
</books>
```

فى root وهى ال books tag
وليها ابناء كل واحد بإسم book
كل book tag ليه attributes ؟ ايوة كل book ليه id معين
داخل كل book بيتشمل name, author, price tags لإسم الكتاب والكاتب والسعر
من الملف دا نريد ان نحصل على اسم كل كتاب ومجموع السعر بتاعهم

1- minidom

في implementation خفيفة ل DOM بإسم minidom هنستدعيها كالتالى

```
import xml.dom.minidom as md #(parse, parseString..)
```

في عندنا دالتين مهمين وهم parse, parseString
parse للتعامل مع file
parseString للتعامل مع string
والإثنين هيدولك ريترن ب document object

ال node object

هو يعتبر الأب لكل العناصر ملف ال xml وليه ميثودز/صفات مهمة

nodeType

بتعبر عن النوع هل هي text node, element, comment, document,... etc

parentNode

رفرنس للأب (ماعدا ال document root) ولل attrs هتكون ديما None

previousSibling

ال node السابقة لل node الحالية إلا اذا كانت هي الأولى

nextSibling

ال node التالية إلا اذا كانت ال node الحالية هي الأخيرة

childNodes

جميع ال nodes اللى داخل ال node الحالية

hasChildNodes()

هل في nodes داخلها ؟

firstChild

اول ابن

lastChild

آخر ابن

hasAttributes()

هل فيها attributes ؟

appendChild(child)

إضافة ابن جديد

insertBefore(child, before)

بتضيف child قبل ال before وفي حال عدم وجوده بيتم إضافته في النهاية

removeChild(child)

حذف ابن child

normalize()

ربط ال text nodes المتقاربة

ال document object بيعبر عن الملف وليه ميثودز/صفات مهمة زي

documentElement # used as a property

ودى بتعبر عن ال root element وفي مثالنا هنا هي books

getElementsByTagName(tagName) #tagName

بتدور على tagName معين في كل الأبناء وابنائهم وهكذا وتديلك ريترن ب element object

createElement(tagName)

لإنشاء tag جديد

createComment(comment)

لإنشاء تعليق داخلي

createAttribute(attr)

لإنشاء صفة attribute

ملحوظة في بعض الميثودز بنفس الاسم ولكن اخرها NS ودي لربطها مع namespace ما وبتاخذ nsURI كمعامل ليها.

ال Element Object يعبر عن عنصر معين في الملف وليه ميثودز مهمة زي

tagName #used as a property

بتعيد الاسم المجرد لل element

getElementsByTagName*

مشابه للموجودة بال document object

hasAttribute(attrName)

هل بيحتوي على attribute ؟

getAttribute(attrName)

بيعيدلك قيمة attribute معينة باسم attrName

setAttribute(attrName, val)

يبربط attribute معينة attrName ليها قيمة val بالعنصر

removeAttribute(attrName)

لحذف attribute معينة attrName (مش بيرفع اي exception !)

ملحوظة في بعض الميثودز بنفس الاسم ولكن اخرها NS ودي لربطها مع namespace ما وبتاخذ nsURI كمعامل ليها.

مجموعة ال exceptions

<http://docs.python.org/lib/dom-exceptions.html>

طيب تمام

1- استدعى ال minidom

```
import xml.dom.minidom as md #(parse, parseString..)
```

2- انشئ ال document object سواء باستخدام parse او parseString حسب تخزينك لملف ال xml

```
doc=md.parse("books.xml")
```

3- احصل على ال document root و اعرضه واحصل على كل tag قيمته book واطبعه

```
def inspectBooks():
    global doc
    print "Root Element: ", doc.documentElement.tagName
    books=doc.getElementsByTagName("book")
    for book in books:
```

```

if book.hasAttribute("id"): #id and it should have one!
    print "ID: ",book.getAttribute("id")
for child in book.childNodes:
    if child.nodeType==child.ELEMENT_NODE:
        if child.tagName=="name":
            child.normalize()
            print "Book: ",child.firstChild.data

```

تمام ال doc هنا -متغير عام- global variable

global doc

الحصول على ال document root هنا جالنا ريترن ب Element object واحنا نريد ال tagName
doc.documentElement.tagName

نحصل على كل العناصر اللي tagName بتاعها book

books=doc.getElementsByTagName("book")

نعمل loop على كل عنصر فيها

for book in books:

إذا كان فيه id attribute (لمجرد عرض المثال)

```

if book.hasAttribute("id"): #id and it should have one!
    print "ID: ",book.getAttribute("id")

```

طيب ولطباعة اسم الكتاب؟ لاحظ انه متخزن فى ال tag name
بسيطة جدا نعمل loop على كل الأبناء فى لا book element ونشوف النوع إذا كان ELEMENT NODE و ال tagName بتاعه هو name نطبعه

```

if child.tagName=="name":
    child.normalize()
    print "Book: ",child.firstChild.data

```

ملحوظة لل nodes انواع كثير etc .., text, comment, element

ال firstChild دا بيعبر عن ال text node اللي فى ال tag name وال data بتدى ريترن بال string اللي جواها

<name> text node ... </name>

الحصول على الثمن الكلى

```

def getTotalSum():
    global doc
    thesum=0
    prices=doc.getElementsByTagName("price")
    for price in prices:
        price.normalize()
        thesum += int(price.firstChild.data) #TO int.
    return thesum

```

نحصل على كل ال price elements

<price>numeric_value</price>

ونحول القيمة ل int وبس ونضيفها على ال thesum وبعد ما نخلص نعمل ال ريترن بيها

ناتج التنفيذ ل

```
inspectBooks()
print "Total Sum: ", getTotalSum()
#output

Root Element: books
ID: 1
Book: Introduction to Python
ID: 2
Book: Introduction to Java
ID: 3
Book: Introduction to Ruby
ID: 4
Book: Introduction to Linux Programming
Total Sum: 370
```

2- SAX

يعتمد على ال events بمعنى انه يبدلك خبر كل ما يبدأ عنصر او يبدأ ال content الى داخله وهكذا يمكن تشوفه اعقد شوية لكن انا عن نفسى من محبى استخدامه

1- استدعى اللى هنستخدمه

```
from xml.sax import make_parser, parseString
from xml.sax.handler import ContentHandler
```

ال ContentHandler هو مفتاحنا السحري فيه ميثودز event handlers بيتعمل ليها استدعاء عند حدوث حدث معين
startDocument()

يتم استدعائها مرة واحدة عند بداية الملف

endDocument()

يتم استدعائها مرة واحدة عند نهاية الملف

startElement(name, attrs)

يتم استدعائها عند بداية قراءة كل عنصر el

<el [attr1=val1, attr2=val2, ... attrN=valN]>CONTENT</el>

characters(content)

يتم استدعائها عن بداية قراءة محتوى العنصر

<el [attr1=val1, attr2=val2, ... attrN=valN]>CONTENT</el>

endElement(el)

يتم استدعائها عند نهاية قراءة عنصر el

<el [attr1=val1, attr2=val2, ... attrN=valN]>CONTENT</el>

فى بعض الميثودز بتنتهى ب NS ودى فى حالة التعامل مع namespace

ال Attributes ماهى الا mapping او dictionary مضاف ليها بعض الميثودز مثل

getLength()

للحصول على عددهم

getNames()

الحصول على اسم كل attribute

getType()

للحصول على النوع وهى عادة CDATA

getValue(attrName)

الحصول على القيمة المرافقة لل attribute المسماة attrName

نرجع للمثال

1- هنستدعى الميثودز/الصفوف المستخدمة

```
from xml.sax import make_parser, parseString
from xml.sax.handler import ContentHandler
```

ال make_parser هنعيد ليها XML reader

parseString لقراءة ال xml من string

parse هى ميثود تبع ال XML reader object بتاخذ مسار ملف (نفس parse, parseString اللى تحدثنا عنهم فى

(minidom
ContentHandler صف مختص بمعالجة المحتوى (لاحقا)
2- ملف ال xml ك string مخزن

```
xmlDoc=""<?xml version="1.0"?>

<books>
  <book id="1">
    <name>Introduction to Python</name>
    <author>Ahmed Youssef</author>
    <price>80</price>
  </book>
  <book id="2">
    <name>Introduction to Java</name>
    <author>Wael Muhammed</author>
    <price>130</price>
  </book>
  <book id="3">
    <name>Introduction to Ruby</name>
    <author>Ahmed Youssef</author>
    <price>70</price>
  </book>
  <book id="4">
    <name>Introduction to Linux Programming</name>
    <author>Ahmed Mostafa</author>
    <price>90</price>
  </book>
</books>
""""
```

3- انشئ صف جديد مشتق من ال ContentHandler

```
class BooksHandler(ContentHandler):
```

ملحوظة اى ميثود مش هتعملها override مش تكتبها فى معالج المحتوى - ContentHandler - ..

```
def __init__(self):

    self._total=0 #Sum of prices.
    self._curel=None
    self._curid=None
    self._booksInfo=[]
    self._authors=[]
```

ايه المتغيرات دي كلها ؟
self._total للتخزين المجموع الكلى للأسعار
self._curel لتخزين اسم العنصر اللي بيتم معالجته
self._curid لتخزين اخر id تم قرائته
self._booksInfo تخزين معلومات عن الكتاب مكونة من ال name, id
self._authors تخزين أسماء الكتاب

```
def getTotal(self):  
    return self._total  
  
def getBooksInfo(self):  
    return self._booksInfo  
  
def getAuthors(self):  
    return self._authors
```

عرفنا getters للوصول للمتغيرات الداخلية
ملحوظة يفضل تستخدم properties مع lambda

```
booksinfo=property(fget=lambda self: self._booksInfo)  
authors=property(fget=lambda self: self._authors)  
total=property(fget=lambda self: self._total)
```

```
def startDocument(self):  
    #print "Starting Document."  
    pass
```

لو حبيت تضيف اى رسالة او اى حاجة على هواك يتم تنفيذها عند بداية قراءة الملف

```
def endDocument(self):  
    #print "Ending Document."  
    pass
```

نفس السابقة ولكن عند انتهاء القراءة

```
def startElement(self, el, attrs):  
  
    #print "Starting ", el  
    self._curel=el  
    if el=="book":  
        #get the id..  
        self._curid=attrs.getValue("id") #attrs["id"]
```

هنا هيتم الإستدعاء عند بداية قراءة كل عنصر el والصفات الخاصة بيه attrs
1- نخزن العنصر الحالي فى ال self._curel

```
self._curel=el
```

2- نختبر اذا كان العنصر الحالي هو book فليه attribute بإسم id فنحصل عليها ونخزنها كآخر id لآخر كتاب تم قرائته فى ال reader

```
if el=="book":  
    #get the id..  
    self._curid=attrs.getValue("id") #attrs["id"]
```

طبعا تقدر تحصل عليها كأنك بتعامل مع dict مش بإستخدام getValue. ميثود

```
def characters(self, content):  
    if content.strip():  
  
        if self._curel=="price":  
            #print "In Price.."  
            try:  
                self._total += int(content)  
            except:  
                pass  
        elif self._curel=="name":  
            self._booksInfo +=[(content, self._curid)]  
        elif self._curel=="author":  
            self._authors +=[content]  
        else:  
            pass
```

هنا هيتم استدعائها عن قراءة المحتوى للعنصر الحالي
وبناء على العنصر الحالي هنتعامل سواء اذا كان ثمن او اسم الكتاب او الكاتب

4- ننشئ كائن من معالج المحتوى الجديد BooksHandler

```
bh = BooksHandler()
```

ننشئ XML reader ونمرر ليه ال handler الجديد والفايل اللي هيتعالج او نستخدم parseString ونمرر ليه كائن معالج المحتوى (bh)

```
p = make_parser( )
```

```
p.setContentHandler(bh)
p.parse(open("books2.xml"))
```

او نستخدم parseString نحدد ال string اللى هيتعالج وال هاندلر (bh)

```
parseString(xmlDoc, bh)
```

للمزيد عن
<http://docs.python.org/lib/module-xml.sax.html>
ومش تنسى <http://www.saxproject.org>

ايهما استخدم ؟

همم DOM بيتعمد على انشاء tree للملف ودا شاق جدا للملفات اللى حجمها كبير!
من ناحية اخرى SAX بيتعمد على ال events ودا اسلوب فعال جدا

3- Expat undercover

Expat مكتبة سى سريعة لمعالجة ملفات ال XML وتم عمل wrapper ليها فى بايثون

1- استدعى الموديلز اللازمة

```
import xml.parsers.expat as exp
```

2- انشئ صف جديد بنفس فكرة ال ContentHandler

```
class ParsingHandler(object):

    def __init__(self, xml):
        self._curel=None
        self._curattrs=None
        self._inbook=False
        self._books=[]
        self._thesum=0

        self._p=exp.ParserCreate()
        self._p.StartElementHandler=self.__startElement
        self._p.EndElementHandler=self.__endElement
        self._p.CharacterDataHandler=self.__charsDataHandler
        self._p.Parse(xml)
```

لاحظ عندنا متغيرات لمتابعة العنصر الحالى والصفات الحالية له وهل احنا داخل ال book tag او لا واسماء الكتاب والمجموع الكلى

القسم الثانى متعلق بال parser

1- انشئ XMLParserType object باستخدام ParserCreate

2- اربط ال handlers المختصين ببداية كل عنصر ونهايته والمحتوى ب handlers انت هتجهزهم لاحقا

3- عالج ال xml باستخدام ال Parse method

انشئ getters

```
def getTotalSum(self):
    return self._thesum

def getBooksInfo(self):
    return self._books

def printBooksInfo(self):
    for book in self._books:
        print book
```

عرف ال handlers الخاصين بنا اللى اسندناهم للمعالجينا الأساسيين لل self._p parser

```
def __startElement(self, el, attrs):
    print "Starting: ",el, attrs
    if el=="book":
        self._inbook=True
        self._curel=el
        self._curattrs=attrs

def __charsDataHandler(self, data):
    if data.strip():
        if self._inbook and self._curel=="name" :
            self._books += [data]
        elif self._curel=="price" :
            self._thesum += int(data)
        else:
            pass

def __endElement(self, el):
    if el=="book":
        self._inbook=False
        self._curel, self._curattrs=None,None
```

الإستخدام

```
if __name__=="__main__":
    p=ParsingHandler(xmlldoc)
    print "Total sum: ", p.getTotalSum()
    p.printBooksInfo()
```

لاحظ ان ال xmlldoc هو string يعبر عن ملف ال xml اللى هيتم معالجته

ناتج التنفيذ

```
Total sum: 370
Introduction to Python
Introduction to Java
Introduction to Ruby
Introduction to Linux Programming
```

ضع فى حسابانك ملف كالتالى

```
<?xml version="1.0"?>
<computer>
  <library>
    <books>
      <book id="1">
        <name>Introduction to Python</name>
        <author>Ahmed Youssef</author>
        <price>80</price>
      </book>
      <book id="2">
        <name>Introduction to Java</name>
        <author>Wael Muhammed</author>
        <price>130</price>
      </book>
      <book id="3">
        <name>Introduction to Ruby</name>
        <author>Ahmed Youssef</author>
        <price>70</price>
      </book>
      <book id="4">
        <name>Introduction to Linux Programming</name>
        <author>Ahmed Mostafa</author>
        <price>90</price>
      </book>
    </books>
  </library>
</computer>
```

فى الجزئيات السابقة تحدثنا عن DOM, SAX وكلام كثير اخر .. مارأيك فى هذا الكود ؟

```
for book in computer.library.books:
    print book.id, book.name, ", by ", book.author

totalsum=sum([int(str(book.price)) for book in computer.library.books])
print "SUM: ", totalsum
```

اليس اسهل كثيرا ؟
الناتج

```
1 Introduction to Python , by Ahmed Youssef
2 Introduction to Java , by Wael Muhammed
3 Introduction to Ruby , by Ahmed Youssef
4 Introduction to Linux Programming , by Ahmed Mostafa
SUM: 370
```

ماذا عن ال attributes - السمات او الصفات-؟
تستطيع الوصول اليها ايضا من خلال اسمها مباشرة

او ملف من amazon مشابه للتالى

```
<?xml version="1.0" encoding="UTF-8"?>
<ItemSearchResponse xmlns="http://webservices.amazon.com/AWSECommerceService/2005-10-05">
  <OperationRequest>
    <HTTPHeaders>
      <Header Name="UserAgent">
      </Header>
    </HTTPHeaders>
    <RequestId>16WRJBVEM155Q026KCV1</RequestId>
    <Arguments>
      <Argument Name="SearchIndex" Value="Books"></Argument>
      <Argument Name="Service" Value="AWSECommerceService"></Argument>
      <Argument Name="Title" Value="Ruby on Rails"></Argument>
      <Argument Name="Operation" Value="ItemSearch"></Argument>
      <Argument Name="AWSAccessKeyId" Value="dontbeaswoosh"></Argument>
    </Arguments>
    <RequestProcessingTime>0.064924955368042</RequestProcessingTime>
  </OperationRequest>
  <Items>
    <Request>
      <IsValid>True</IsValid>
      <ItemSearchRequest>
        <SearchIndex>Books</SearchIndex>
        <Title>Ruby on Rails</Title>
      </ItemSearchRequest>
    </Request>
    <TotalResults>22</TotalResults>
    <TotalPages>3</TotalPages>
    <Item>
      <ASIN>0321480791</ASIN>
      <DetailPageURL>http://www.amazon.com/gp/redirect.html%3FASIN=0321480791%26tag=ws%26code=xm2%26cID=2025%26ccmID=165953%26location=/o/ASIN/0321480791%253FSubscriptionId=dontbeaswoosh</DetailPageURL>
      <ItemAttributes>
        <Author>Michael Hartl</Author>
        <Author>Aurelius Prochazka</Author>
        <Manufacturer>Addison-Wesley Professional</Manufacturer>
```



```

        <ProductGroup>Book</ProductGroup>
        <Title>RailsSpace: Building a Social Networking Website with Ruby on Rails (Addison-
Wesley Professional Ruby Series)</Title>
        </ItemAttributes>
    </Item>
</Items>
</ItemSearchResponse>

```

وتريد الحصول على بعض العناصر اليس كذلك ؟

```

print rt.OperationRequest.HTTPHeaders.Header.Name
print rt.OperationRequest.Arguments[0].Name
print rt.OperationRequest.RequestProcessingTime
print rt.Items.TotalPages
print rt.Items.TotalResults
print rt.Items.Item.ASIN
print rt.Items.Item.DetailPageURL

```

اليس اسهل كثيرا ؟

```

UserAgent
SearchIndex
0.064924955368042
3
22
0321480791
http://www.amazon.com/gp/redirect.html%3FASIN=0321480791%26tag=ws
%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/ASIN/0321480791%253FS
ubscriptionId=dontbeaswoosh

```

او ربما مهتم ب Twitter ؟
على فرض لدينا هذا الملف

```

<?xml version="1.0" encoding="UTF-8"?>
<statuses type="array">
  <status>
    <created_at>Sat Aug 09 05:38:12 +0000 2008</created_at>
    <id>882281424</id>
    <text>I so just thought the guy lighting the Olympic torch was falling
when he began to run on the wall. Wow that would have been catastrophic.</
text>

```

```

<source>web</source>
<truncated>>false</truncated>
<in_reply_to_status_id>1234</in_reply_to_status_id>
<in_reply_to_user_id>12345</in_reply_to_user_id>
<favorited></favorited>
  <user>
    <id>4243</id>
    <name>John Nunemaker</name>
    <screen_name>jnunemaker</screen_name>
    <location>Mishawaka, IN, US</location>
    <description>Loves his wife, ruby, notre dame football and iu
basketball</description>

<profile_image_url>http://s3.amazonaws.com/twitter_production/profile_images/53781608/Photo_75_normal.jpg</profile_image_url>
    <url>http://addictedtonew.com</url>
    <protected>>false</protected>
    <followers_count>486</followers_count>
  </user>
</status>
</statuses>

```

وتريد الحصول على كل ما تحت ال user

```
statuses.status.user.inspect_me()
```

ستجد الناتج

```

Attrs:
Tags:
  id => 4243
  name => John Nunemaker
  screen_name => jnunemaker
  location => Mishawaka, IN, US
  description => Loves his wife, ruby, notre dame football and iu
basketball
  profile_image_url =>
http://s3.amazonaws.com/twitter_production/profile_images/53781608/Photo_7
5_normal.jpg
  url => http://addictedtonew.com
  protected => false
  followers_count => 486

```

اكيد مازالت تحتاج للعديد من الإختبارات وبإستخدامك هيثم تحسينها

كيفية الإستخدام
1- استدعاء happymapper

```
import happymapper
```

2- استخدام الدالة `get_root` التي تأخذ معامل `xmlfile` وهو مسار ملف ال `xml` وتعيد لك ال `root tag` طيب دا بالنسبة للملفات الخارجية ماذا عن النصوص الداخلية ؟ قرأت الصفحة فى متغير داخلى او ماشابه ماذا تفعل ؟

تستطيع استخدام الدالة `get_root_document` التي تأخذ معامل `doc` وهو محتوى ملف `XML` وتعيد لك ال `root tag` ملحوظة الملفات المستخدمة فى الشرح `amazon.xml` و `twitter.xml` مأخوذه من هنا

<http://railstips.org/2008/11/17/happymapper-making-xml-fun-again>

للحصول على HappyMapper

[/http://bitbucket.org/xmonader/happymapper](http://bitbucket.org/xmonader/happymapper)

ElementTree

TODO

HTMLing with Python

على فرض أنك قرأت الفصل السابق قم بتنفيذ البرنامج التالي

--سكربت يقوم بقراءة صفحة من الإنترنت ويقوم بالحصول على جميع اللينكات فيها اعتمد على النموذج التالي

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

from HTMLParser import HTMLParser as HP
import urllib2 as ulib
import sys

def fetchdatafrom(url):
    return ulib.urlopen(url).read()

def as_unicode(data):
    return data.decode("cp1256").encode("utf-8")

class PageParser(HP):

    def __init__(self):
        self._ina=False
        self._links=[]

    links=lambda self: self._links

    def handle_start_tag(self, tag, attrs):
        pass

    def handle_data(self, data):
        pass

    def handle_endtag(self, tag):
        pass

def getlinks(url):

    htmlsrc=fetchdatafrom(url)
    p=PageParser()
    p.feed(htmlsrc)
    return p.links()
```

طريقة الإستخدام مشابهه لتلك مع SAX

حيث تعيد تعريف الطرق handle_starttag و handle_data و handle_endtag للتعامل مع وسوم ال HTML

الدالة fetchdatafrom تقوم بإعادة كود الصفحة اليك على صورة string
الدالة as_unicode تقوم بتحويل ال cp1256 الى unicode (ربما اذا اردت ان تعالج ال data تستطيع الإستفادة منها)
ال PageParser هو صف يشتق ال HTMLParser ويتم التعامل داخله مثلما تعاملنا مع الصفوف المشتقة
ContentHandler
ولإطعامه السورس نستخدم الطريقة feed
الدالة getlinks تقوم بالحصول على الروابط من الطريقة links التي تعيد لنا الروابط التي تم قرائتها

Beautiful Soup

هى HTML/XML parser بايثونية وتعالج ايضا الملفات المكتوبة بطريقة سيئة ولا تجعلك تقلق من الإنكودينج
لمعالجة ملفات ال HTML استخدم الصف BeautifulSoup واذا اردت معالجة ملفات XML استخدم
BeautifulStoneSoup

حل المطلوب السابق بإستخدام BeautifulSoup

```
#!/bin/python

import BeautifulSoup as bs
import urllib2 as ulib

def fetchdatafrom(url):
    return ulib.urlopen(url).read() or ""

def getzetcodemain():
    return fetchdatafrom('http://zetcode.com')

soup=bs.BeautifulSoup(getzetcodemain())
for el in soup.findAll('a'):
    # [0][0] is href.
    print "[url=%s]%s[/url]"%(el.attrs[0][1], el.contents)
```

تنتظرك رحلة رائعة مع الوثائق الخاصة بيها
<http://www.crummy.com/software/BeautifulSoup/documentation.html>

تدريب

استفيد من السكريبتات السابق فى تنفيذ التالى
انشاء مفهرس للمنتديات يأخذ قسم معين كبداية وتقوم بتحديد عدد الصفحات المطلوبة ويقوم بفتحها
واستخلاص اللينكات والعناوين لها

Parsing CSV Files

CSV هي اختصار ل comma separated values من الأسم واضح انها تستخدم في تمثيل قيم مع فصلها باستخدام الفاصلة (ال comma) في صفوف (تستخدم عادة في استيراد او تصدير بيانات ما ربما قاعدة بيانات مثلا؟) لاحظ ممكن يكون الفاصل مجرد مسافة او سلاش / او او او ولكن الأشهر هو ال فاصلة مثال

```
ahmed, 19, m
ayman, 20, m
```

وقد تحتوي على صف اول يمثل الهيدر (يشمل عناوين الأعمدة)

```
name, age, sex
ahmed, 19, m
ayman, 20, m
```

على فرض لدينا ملف بإسم somefile.csv وفيه البيانات التالية

```
ahmed, m, 19
wael, m, 20
radwa, f, 19
gina, f, 21
ayman, m, 20
```

استدعى ال csv module كالتالى

```
import csv
```

قم بإنشاء reader object "مسئول عن القراءة للملف ومعالجته" باستخدام `csv.reader()`

```
reader=csv.reader(open("somefile.csv", "rb")) #default dialect. #b as a catch for win32.
```

تقدر تستخدم ال for loop مع ال reader فهي تقوم بعمل yield لكل صف يتم قراءته وللحصول على رقم الصف استخدم ال line_num

```
for row in reader:
    print row, " at: ", reader.line_num
```

فلنقم بتحسين المثال بعض الشيء

```
import csv

f=open("somefile.csv", "rb")
try:
    reader=csv.reader(f) #default dialect. #b as a catch for win32.
```

```

for row in reader:
    print row, " at: ", reader.line_num

except Exception, ex:
    print ex.message
finally:
    f.close()

```

إذا أردت ربط الصف بقاموس وذلك بتحديد أسماء الأعمدة كالتالي

```

reader=csv.DictReader(f, fieldnames=("name", "age", "sex")) #default dialect. #b as a catch for win32.

for row in reader:
    print row, " at: ", reader.line_num
    print row["name"] #the name column

```

للكتابة الموضوع سهل أيضا
 تنشئ writer (كاتب) من csv.writer
 تكتب ال header (أسماء الأعمدة) باستخدام الطريقة writer.writerow
 وتعمل دالة على المدخلات لكتابة كل صف على سبيل المثال

```

import csv

f=open("somefile1.csv", "w")
try:

    writer=csv.writer(f)
    inputrows=(
        (1, "ahmed", "ahmedf1@gmail.com"),
        (2, "ayman", "aymanf2@gmail.com"),
        (3, "smsm", "smsm@yahoo.com")
    )

    headers=("id", "user", "email")
    writer.writerow(headers)

    for row in inputrows:
        writer.writerow(row)

except Exception, ex:
    print ex.message

finally:

```

```
f.close()
```

للمزيد حول ال CSV راجع وثائق بايثون

ConfigParser

موديل ConfigParser لمعالجة ملفات ال .ini "لإعلاقة لها بالرجيسترى!"
ملف ال .ini نوع قديم من وصف البيانات ومستخدم بكثرة فى التطبيقات القديمة نسبيا
مثال

```
[program]
name = SVM
version = 0.2.4
license = GPLv3

[author]
name = Ahmed Youssef
email = guru.python@gmail.com
```

هنا فى الملف يوجد 2 sections او قسمين الأول program والثانى author كل منهم يحوى keys/values فى
بيسموها اوبشنز "براكتك"
مثلا ال name هو key تحت ال program section وقيمه SVM
احفظ الملف السابق وليكن tst.cfg

1- استدعى ال ConfigParser موديل

```
from ConfigParser import *
```

2- انشئ كائن

```
cp=SafeConfigParser() #create an object of SafeConfigParser
```

لاحظ ان فى كذا صف RawConfigParser, ConfigParser, SafeConfigParser فال RawConfigParser هو الأب
واشتقه ال ConfigParser وهو الأب ل SafeConfigParser لذا قم بإستخدامه دائما

3- قم بقراءة الملف

```
cp.read("tst.cfg") #read by filename.
```

ال configparser object الى انشئناه cp يقوم بقراءة الملف بإستخدام ال read method
لإضافة سكشن جديد
لإضافة key جديد تحت القسم section وله قيمة value
.add_section(section)
.set(section, key, value)
.sections()

للحصول على جميع الأقسام

`.has_section(section)`

هل يوجد قسم بإسم section ؟

`.get(section, key)`

للحصول على قيمة ل key تحت section

`.options(section)`

للحصول على كل ال options تحت section معين

`.has_option(section, option)`

هل section يحوى option بإسم option ؟

`.items(section)`

الحصول على list مكونة من tuples يتضمن key, value مثلا

```
[('name', 'Ahmed Youssef'), ('email', 'guru.python@gmail.com')]
```

`.write(fp)`

كتابة الملف سواء على ال stdout او فى ملف ما.. الخ

فى مجموعة من ال Errors مثلا NoSectionError, ParsingError, DuplicateSectionError, NoOptionError ودى فى حال محاول الوصول لقسم او اختيار غير موجود او محاولة التكرار او خطأ فى معالجة الملف "كتابة بصورة غير سليمة" وغيرهم..

من الحاجات اللى تهتمك.. السكاشن -الأقسام- وال Options

```
SECTCRE = re.compile(
    r'\['                               # [
    r'(?P<header>[^\]]+)'               # very permissive!
    r'\]'                               # ]
)
OPTCRE = re.compile(
    r'(?P<option>[^\s][^:=]*)'          # very permissive!
    r'\s*(?P<vi>[:=])\s*'              # any number of space/tab,
    # followed by separator
    # (either : or =), followed
    # by any # space/tab
    r'(?P<value>.*)$'                  # everything up to eol
)
```

راجع ConfigParser.py للإطلاع على المزيد

Regex:

TODO

التفاعل مع برامج اخرى

للآن تستطيع كتابة سكريبتات جميلة بالبايثون ولكن ايضا قد نحتاج لإدخال بعض البيانات لبرنامج معين من خلال سطر الأوامر

sys.argv هي list تشمل كل المعاملات التى تم ارسالها لبرنامجك

```
#echo .py
from sys import argv

print "ARGV: ", argv

for i, arg in enumerate(argv):
    print "Argv[%d]: %s"%(i, arg)
```

لاحظ ان اول معامل فى ال argv سيكون دائما هو اسم البرنامج
يتم الفصل بين كل معامل بإستخدام مسافة
لدمج اكثر من معامل ضعهم بين علامتى تنصيص

```
striky@striky-desktop:~/workspace/pytut/src$ python echo.py Hello
ARGV: ['echo.py', 'Hello']
Argv[0]: echo.py
Argv[1]: Hello
striky@striky-desktop:~/workspace/pytut/src$ python echo.py "Hello World"
```

```
ARGV: ['echo.py', 'Hello World']
Argv[0]: echo.py
Argv[1]: Hello World
```

ماذا عن enumerate ؟

هى دالة تقوم بإعادة index (لعدد الدورات) وقيمة من container (مثلا list ك argv)

Gimme usage!

تطبيق جيد ايضا ان تضع دالة باسم usage توضح كيفية استخدام البرنامج
قم دائما باختبار عدد المعاملات التى تم ارسالها للسكربت فأى عدد غير مقبول قم بعرض ال usage

```
def usage():
    """My fancy usage helper"""
    .....

def consoleMain():
    if len(argv) != DEFINED_LENGTH:
        GIMME_USAGE

if __name__=="__main__":
    consoleMain()
```

GIMME optparser!

بايثون كالعادة توفر لك العديد والعديد لمساعدتك فتوفر لك اكثر من وحدة لمعالجة معاملات سطر الأوامر لاحظ
الإستخدام التالى

```
striky@striky-desktop:~$ python mufhrs.py -f http://linuxac.org/forum/forumdisplay.php?f=23 -l 1 -u 3 -s 1 -t vb > pgfihrsx2.txt
```

ياإلهي كيف تدير كل هذه المعاملات الغير معقولة؟
انتبه جيدا لأن هذا هو جزء من حل السكربت المطلوب منك سابقا

هنا مثلا لفهرسة قسم فى منتدى نريد اقل ترتيب للصفحة واكثر ترتيب ومقدار الزيادة (ستفيدك كثيرا اذا قررت محاولة فهرسة منتدى SMF) ونوع المنتدى نخبر السكربت بهذا عن طريق تحديد اسم للمعامل وقيمة له مثلا لمقدر الزيادة -s

وتكون قيمتها هى المعامل التالى لها 1

عدد الصفحة المطلوب الإنتهاء عندها -u

وقيمتها 3

لنوع المنتدى -t

وقيمته vb

وهكذا ، او ربما استخدام الصيغة المطولة

--step=1

--upper=3

--ftype=vb

لاحظ ان الترتيب ليس هاما!!

اكيد اخذ العديد من الشروط والإختبارات (شكرا لبايثون والوحدة optparse لقد اخذت الكثير من الجهد عن عاتقنا) هذا الجزء من حل السكربت المطلوب

```
def consoleMain():
```

```
    optparser=OptionParser()
    optparser.add_option("-f", "--forum", dest="forumlink", help="Forum Section")
    optparser.add_option("-l", "--lower", dest="lower", help="Lowest page")
    optparser.add_option("-u", "--upper", dest="upper", help="Upper page")
    optparser.add_option("-s", "--step", dest="step", help="Step")
    optparser.add_option("-t", "--type", dest="ftype", help="Forum type (e.g vb)")
    options, args=optparser.parse_args() #defaulted to sys.argv[1:]
    #print options, "====",args
    forumlink=optparser.values.forumlink
    lower=int(optparser.values.lower)
    upper=int(optparser.values.upper)
    forumtype=optparser.values.ftype.lower()
    step=int(optparser.values.step)
```

1- يجب استدعاء ال الوحدة optparse او الصف OptionParser مباشرة للإختصار

```
from optparse import OptionParser
```

2- انشاء كائن من OptionParser

```
optparser=OptionParser()
```

3- اضافة اسماء المعاملات باستخدام الطريقة add_option والتي تأخذ معاملات عديدة اهمها

1- الصورة المختصرة لإسم الاختيار f-

2- الصورة الطويلة (الكاملة) لإسم الاختيار forum-

3- اسم من اختيارك للحصول على قيمته وليكن forumlink مثلا

4- قسم المساعدة الخاص بال option

```
optsparser.add_option("-f", "--forum", dest="forumlink", help="Forum Section")
optsparser.add_option("-l", "--lower", dest="lower", help="Lowest page")
optsparser.add_option("-u", "--upper", dest="upper", help="Upper page")
optsparser.add_option("-s", "--step", dest="step", help="Step")
optsparser.add_option("-t", "--type", dest="ftype", help="Forum type (e.g vb)")
```

عند استدعاءك للبرنامج فى وضع المساعدة help يتم تنفيذ شئ مشابه للتالى

Options:

-h, --help show this help message and exit

-f FORUMLINK, --forum=FORUMLINK

Forum Section

-l LOWER, --lower=LOWER

Lowest page

-u UPPER, --upper=UPPER

Upper page

-s STEP, --step=STEP Step

-t FTYPE, --type=FTYPE

Forum type (e.g vb)

striky@striky-desktop:~/workspace/pytut/src\$ python mufhrs.py --help

Usage: mufhrs.py [options]

Options:

-h, --help show this help message and exit

-f FORUMLINK, --forum=FORUMLINK

Forum Section

-l LOWER, --lower=LOWER

Lowest page

-u UPPER, --upper=UPPER

Upper page

-s STEP, --step=STEP Step

-t FTYPE, --type=FTYPE

Forum type (e.g vb)

رائعة اليس كذلك ؟
للحصول على قيم الاختيارات

```
forumlink=optsparser.values.forumlink
lower=int(optsparser.values.lower)
```

```
upper=int(optsparser.values.upper)
forumtype=optsparser.values.ftype.lower()
step=int(optsparser.values.step)
```

ما هذا ؟ كيف جعلت بايثون اسماء الإختيارات كمتغيرات خاصة بالكائن ؟
ج: بعض سحر setattr (:)

تدريب:

*قم بكتابة اداة مشابهة ل cat بإستخدام بايثون علما بأن ال stdin, stdout, stderr ستجدهم فى الوحدة sys

os.system

لتنفيذ اوامر خاصة بالنظام توجد الدالة system فى الوحدة os والتي تعيد ايضا ال exit status (التي يعيدها البرنامج عند انتهاءه لتشير لنجاح او حدوث خطأ اثناء التنفيذ)
على فرض لدينا هذا السكريبت exitstatus.py

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

def who():
    name=raw_input("Name: ")
    if name != "Ahmed":
        print "Not Ahmed"
        exit(1)
    else:
        print "Welcome"

who()
```

وقمنا بتنفيذ

```
striky@striky-desktop:~/workspace/pytut/src$ python exitstatus.py
Name: Ahmed
Welcome
striky@striky-desktop:~/workspace/pytut/src$ echo $?
0
```

المتغير \$? يشمل ال exit code الخاص بالبرنامج

تنبيه:

subprocess

كثير من الأحيان نحتاج لتنفيذ أوامر والحصول على الخرج الخاص بها
 فى هذا المثال سنقوم بتنفيذ الأمر cat على الملف السابق ونقوم بقراءته داخل السكربت
 قم بإستدعاء الوحدة subprocess الخاصة بتنفيذ برامج فرعية داخل البرنامج (كبديل ل system و مشابهها)

```
>>> import subprocess as sb
>>> ret=sb.call(['cat', 'exitstatus.py'])
#!/usr/bin/env python
#-*- coding:utf-8 -*-

def who():
    name=raw_input("Name: ")
    if name != "Ahmed":
        print "Not Ahmed"
        exit(1)
    else:
        print "Welcome"

who()
>>> ret
0
```

الطريقة call تقوم بتنفيذ امر ما فى list حيث اول عنصر هو الأمر والباقى هو المعاملات التى يأخذها البرنامج

```
>>> catoutput=sb.Popen(["cat", "exitstatus.py"], stdout=sb.PIPE).communicate()[0]
>>> print catoutput
#!/usr/bin/env python
#-*- coding:utf-8 -*-

def who():
    name=raw_input("Name: ")
    if name != "Ahmed":
        print "Not Ahmed"
        exit(1)
    else:
        print "Welcome"

who()
```

نبدأ من اليمين ليسار الطريقة communicate تعيد لنا tuple تشمل الخرج والخطأ فنأخذ العنصر الأول وهو الخرج (ناتج تنفيذ العملية)
Popen هو صف يأخذ اول عنصر args والأمر والمعاملات
stdout, stdin, stderr هي الخرج والإدخال والخطأ الخاصين بالعملية ويأخذو قيم None او PIPE (قيمة خاصة لتعلم بوجود فتح انبوبة pipe) او file object او رقم ليعبر عن file descriptor موجود
shell قيمة منطقية (True لتعبر عن تنفيذ البرنامج من خلال الشيل او False ليتم التنفيذ من خلال excevp) (لا تهتم الآن)

Replacer

ربما لديك مجموعة كبيرة من الملفات تحتاج القيام بتعديل سريع عليها جميعا ؟
فى هذه الجزئية سننشئ سكربت للتعديل على تلك الملفات بإستخدام
1- القواميس dict
2- ال recursion (وهى استدعاء الدالة لنفسها)

حسنا هاهو السكربت

```
#!/bin/python

#Imports#
import os
import os.path as op
import time
from sys import argv

class Replacer(object):

    def __init__(self, parentDir, dic={}, exts=[]):
        self.__parentDir=parentDir
        self.__dic=dic
        self.__exts=exts
        self.__replacingTimes=0

    def __getExt(self, s):
```

```

#idx=s.rfind(".")
#ext=s[idx:]
#return ext
    return op.splittext(s)[1]

def replace(self):
    start=self.__parentDir
    for e in os.listdir(start): #foreach entry in os.listdir..
        path=start+op.sep+e
        if os.path.isfile(path):
            ext=self.__getExt(path)
            if not ext in self.__exts:
                continue #Re-Loop..
            #replace..
            f=file(path, "r")
            src=f.read()
            f.close()
            for key in self.__dic:
                self.__replacingTimes += src.count(key)
                src=src.replace(key, self.__dic[key])

            f=file(path, "w")
            f.write(src)
            f.close()
        elif os.path.isdir(path):
            #RECURSE..
            rep=Replacer(path, self.__dic, self.__exts)
            rep.replace()
        else:
            continue

```

الفكرة هي اعطاء مجلد لتعديل الملفات التي بداخله
وإذا كان داخله مجلد يتم فتح ذلك المجلد للتعديل على ما في داخله وهكذا

نبدأ بـ

```

def __init__(self, parentDir, dic={}, exts=[]):
    self.__parentDir=parentDir
    self.__dic=dic
    self.__exts=exts
    self.__replacingTimes=0

```

تحديد مجلد الأب parentDir
تحديد الكلمات القديمة والجديدة في قاموس ليتم استبدالهم
تحديد الإمتدادات القابل العمل عليها

الحصول على امتداد ملف

كما ذكرنا تستطيع استخدام os.path.splitext اللتي تعيد لنا قائمة مكونة من اسم الملف والإمتداد

```
def __getExt(self, s):
    #idx=s.rfind(".")
    #ext=s[idx:]
    #return ext
    return op.splitext(s)[1]
```

تستطيع ايضا كتابة ذلك يدويا بإستخدام الطريقة rfind للحصول على ترتيب ال نقطة "." من اليمين - حتى لاتقع في مشكلة مع ملفات مثل h1.ext1.ext2 - وتحسب الحروف من ذلك الترتيب الى النهاية تأتي الى الطريقة replace

```
def replace(self):
    start=self.__parentDir
    for e in os.listdir(start): #foreach entry in os.listdir..
        path=start+op.sep+e
        if os.path.isfile(path):
            ext=self.__getExt(path)
            if not ext in self.__exts:
                continue #Re-Loop..
            #replace..
            f=file(path, "r")
            src=f.read()
            f.close()
            for key in self.__dic:
                self.__replacingTimes += src.count(key)
                src=src.replace(key, self.__dic[key])

            f=file(path, "w")
            f.write(src)
            f.close()
        elif os.path.isdir(path):
            #RECURSE..
            rep=Replacer(path, self.__dic, self.__exts)
            rep.replace()
        else:
            continue
```

فى هذه الطريقة نعمل كالتالى
1- الحصول على قائمة بالمدخلات فى المجلد الرئيسى

```
for e in os.listdir(start): #foreach entry in os.listdir..
    path=start+op.sep+e
```

2- اختبار ما إذا كان المسار ملفا

```
if os.path.isfile(path):  
    ext=self.__getExt(path)
```

3- اختبار اذا كان ذلك الملف يحوى امتداد مقبولا (محدد للإستبدال) وإلا نعود الى بداية الدوارة بإستخدام
continue

```
if not ext in self.__exts:  
    continue #Re-Loop..
```

5- اذا كان ملفا يحوى امتداد مقبول للإستبدال يتم فتحه لقراءة محتواه وغلقه واستبدال القيم القديمة بالجديدة

```
#replace..  
f=file(path, "r")  
#Note: Not reading line by line as I've never met a more than 1MB text file!  
src=f.read()  
f.close()  
for key in self.__dic:  
    self.__replacingTimes += src.count(key)  
    src=src.replace(key, self.__dic[key])
```

6- فتح الملف للكتابة وكتابة ذلك المحتوى مرة اخرى وغلقه

```
f=file(path, "w")  
f.write(src)  
f.close()
```

7- اذا كان المسار مجلدا فيتم فتحه (على اساس انه المجلد الرئيسى واعادة تنفيذ ماسبق بإستدعاء الطريقة
replace وهذا مايسمى بال recursion

```
elif os.path.isdir(path):  
    #RECURSE..  
    rep=Replacer(path, self.__dic, self.__exts)  
    rep.replace()  
else:  
    continue
```

التعامل مع المستخدم

```

def consoleMain():
    ##python replacer.py root [old] [new] [exts]

    root, oldones, newones, exts=argv[1:5]
    oldones=oldones.split(",") #use comma in between.
    newones=newones.split(",") #use comma in between.
    exts=exts.split(",")      #use comma in between.
    if len(oldones)==len(newones):
        dic=dict(zip(oldones, newones))
        #print "DIC:", dic
        #print "EXTS:", exts
        start=time.time()
        rep=Replacer(root, dic, exts)
        rep.replace()
        end=time.time()
        print "Time: ", (end-start)
    else:
        print "len(keys)!=len(values)"
        exit(1)

    start=time.time()
    rep=Replacer(root, {old:new}, [exts])
    rep.replace()
    end=time.time()
    print "Time: ", (end-start)

```

تابع لمزيد من التوضيح جزئية استخدام ال optparse

حسننا البرنامج يعمل الآن ولكن هل يعمل جيدا ؟ هل نحتاج لكل هذه الأكواد هل هناك طريقة افضل ؟ بالطبع تستطيع استخدام os.walk كالتالي مثلا

```

for root, dirs, files in os.walk(self.__parentDir):
    for f in files:
        if ("*" or self.__getExt(f)) in self.__exts:
            fullpath=op.join(root, f)
            .....
            .....

```


Chapter 12 (Networking)

فصل اكبر من ان يغطيه كتاب مثل هذا و لمقدمة عن ال sockets راجع مقالة ويكيديا
http://en.wikipedia.org/wiki/Internet_socket

Simple Server

```
#simplserver.py
import socket

class EchoServer(object):

    def __init__(self, host="", port=51002):
        self._host, self._port=host, port
        self._endpoint=(host, port) #host, addr
        self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

    def start(self):
        self.sock.bind(self._endpoint)
        self.sock.listen(1)
        print "Server running on: ", self._port
        self.handle_request()

    def handle_request(self):

        while True: #Waits for a client.
            clientsock, addr=self.sock.accept()
            #clientfile=clientsock.makefile('rw', 0) #Create a file-like object.
            print "Connection from: ", addr

            clientsock.sendall(str(addr)+" you are connected to server...")

            while True: #communication loop

                #clientfile.write(str(addr)+" you are connected to server.\n")
                msg=clientsock.recv(8092)
                if msg:
                    print ">> ", msg
                    clientsock.sendall(msg)
                    #msg=clientfile.readline().strip() #clean it up.
                    #print "Recieved: ", msg
```



```

        #clientfile.write("Got: "+msg+"\n")

        #Cleaning UP
        #clientfile.close()
        clientsock.close()

if __name__=="__main__":
    try:
        es=EchoServer()
        es.start()
    except KeyboardInterrupt:
        exit()

```

فى هذا الكود انشأنا صف جديد بإسم EchoServer

```

class EchoServer(object):

    def __init__(self, host="", port=51002):
        self._host, self._port=host, port
        self._endpoint=(host, port) #host, addr
        self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)

```

- 1- قمنا بتحديد ال endpoint (الهوست والبورت)
 - 2- انشأنا TCP Socket
 - 3- قمنا بتفعيل استخدام SO_REUSEADDR لإلغاء شغل البورت عنده إيقاف السرفر
- قمنا بتعريف طريقة start لبدأ السرفر

```

def start(self):
    self.sock.bind(self._endpoint)
    self.sock.listen(1)
    print "Server running on: ", self._port
    self.handle_request()

```

- 1- عمل bind (ربط بال endpoint التى تم تحديدها)
 - 2- تجهيز وتشغيل ال tcp listener باستخدام الطريقة listen
 - 3- نستدعى الطريقة handle_request التى سيتم فيها التعامل مع العميل
- مكونة من حلقتين الأولى للتعامل مع العملاء المنتظرين والثانية لمعالجة عميل ما

```

def handle_request(self):

```

```

while True: #Waits for a client.
    clientsock, addr=self.sock.accept()
    #clientfile=clientsock.makefile('rw', 0) #Create a file-like object.
    print "Connection from: ", addr

    clientsock.sendall(str(addr)+" you are connected to server...")

    while True: #communication loop

        msg=clientsock.recv(8092)
        if msg:
            print ">> ", msg
            clientsock.sendall(msg)

    #Cleaning up
    clientsock.close()

```

الطريقة accept تقبل اتصالاً وتعيد لنا كائن socket وعنوان
 تستطيع استخدام الطريقة makefile لإنشاء file-like object للتعامل مع ال socket
 الطريقة sendall لإرسال رسالة
 الطريقة recv للحصول على الرسالة القادمة (ويتم تحديد حجمها عن طريق معامل ال bufsize)

Simple Client

```

#simpleclient.py
import socket

class SimpleClient(object):

    def __init__(self, endpoint=('127.0.0.1', 51002)):
        self._endpoint=endpoint
        self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.connect(self._endpoint)

    def start(self):

        while True:

            data=self.sock.recv(8096)
            if not data:
                break
            print data

            msg=raw_input("> ")
            if not msg:

```

```

        break
    self.sock.send(msg)
self.sock.close()

if __name__=="__main__":
    try:
        sc=SimpleClient()
        sc.start()
    except KeyboardInterrupt:
        exit()

```

هنا انشأنا صف جديد SimpleClient

```

class SimpleClient(object):

def __init__(self, endpoint=('127.0.0.1', 51002)):
    self._endpoint=endpoint
    self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.sock.connect(self._endpoint)

```

نقوم بعمل اتصال مع السرفر (تحديد ال endpoint) باستخدام الطريقة connect

الطريقة start تقوم بعمل حلقة الإتصال مع السرفر

```

def start(self):

    while True:

        data=self.sock.recv(8096)
        if not data:
            break
        print data

        msg=raw_input("> ")
        if not msg:
            break
        self.sock.send(msg)
    self.sock.close()

```

الخ الخ

SocketServer

هى موديل فيها تجميع للصفوف الشائعة فمثلا لاجابة لكتابة الأكواد السابقة لمجرد انشاء TCPServer او UDPServer وهكذا ولكن الأساس ثابت وهناك بعض المتغيرات التى يمكن اعادة تعريفها (التعامل مع العميل على سبيل المثال) سرفر

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

from SocketServer import TCPServer, StreamRequestHandler

class MyStreamRequestHandler(StreamRequestHandler):

    def handle(self):
        print "Got connection from: ", self.client_address
        self.wfile.write(str(self.client_address)+" you are connected to server.")
        #Communication loop...

        while True:

            msg=self.request.recv(1024)
            if not msg:
                break
            print ">> ", msg
            #Send it back...
            self.request.send(msg)
            print "Done handling..."

def go(endpoint=(", 52002")):
    addr=endpoint
    tcpServer=TCPServer(addr, MyStreamRequestHandler)
    tcpServer.allow_reuse_address=1
    print "Server started..."
    tcpServer.serve_forever() #inf. loop

if __name__=="__main__":
    try:
        go()
    except KeyboardInterrupt:
```

```
exit()
```

عمل

```
import socket

class SimpleClient(object):

    def __init__(self, endpoint=('127.0.0.1', 52002)):
        self._endpoint=endpoint
        self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.connect(self._endpoint)

    def start(self):
        #self.sock.sendall("Hey you!!")
        while True:

            data=self.sock.recv(1024)
            if not data:
                break
            print data

            msg=raw_input("> ")
            if not msg:
                break

            self.sock.sendall(msg)
        #self.sock.close()

if __name__=="__main__":
    try:
        sc=SimpleClient()
        sc.start()
    except KeyboardInterrupt:
        exit()
```

MixIns

تستطيع بكل سهولة ان تجعل سرفرك يعالج اكثر من عمل سواء باستخدام ال Threading او ال Forking وذلك بإشتقاقك لل ThreadingMixIn او ال ForkingMixIn الموجودة في SocketServer module

```
class MyServer(ThreadingMixIn,TCPServer):
```

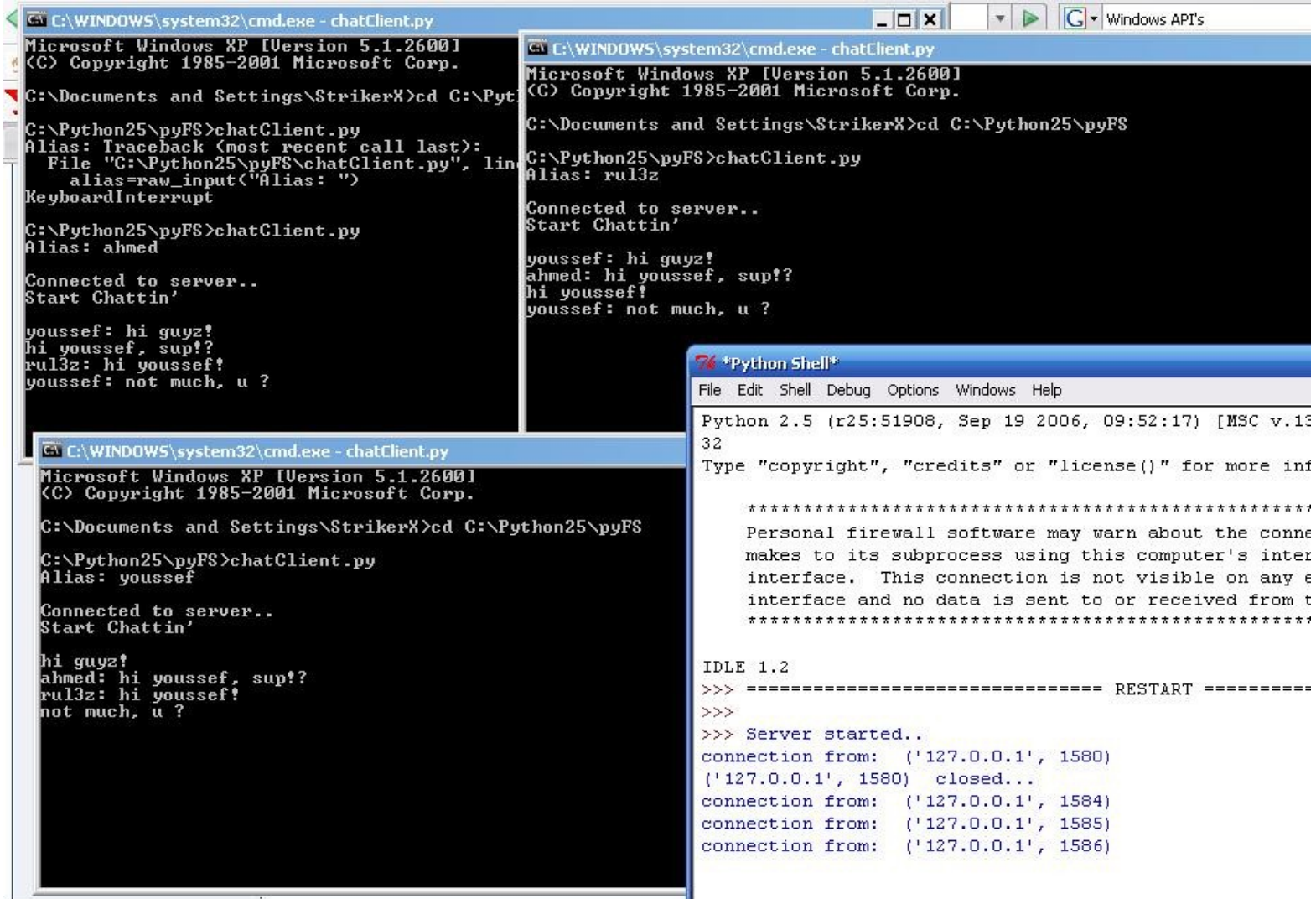
```
pass #Done!
```

او هكذا

```
class MyServer(ForkingMixIn, TCPServer):  
    pass #Done!
```

الفرق ان ال Forking يتم معالجة كل عميل فى بروسيس -عملية- جديدة بينما ال Threading يتم معالجتها داخل نفس العملية ولكن بخيط جديد (تعدد مهام مثل محرر النصوص الذى تكتب فيه ويقوم بالترقيم وتصحيح الأخطاء الإملائية والعديد من هذه العمليات فى ان واحد)

تطبيق دردشة



```
C:\WINDOWS\system32\cmd.exe - chatClient.py  
Microsoft Windows XP [Version 5.1.2600]  
(C) Copyright 1985-2001 Microsoft Corp.  
C:\Documents and Settings\StrikerX>cd C:\Python25\pyFS  
C:\Python25\pyFS>chatClient.py  
Alias: Traceback (most recent call last):  
  File "C:\Python25\pyFS\chatClient.py", line 1  
    alias=raw_input("Alias: ")  
KeyboardInterrupt  
C:\Python25\pyFS>chatClient.py  
Alias: ahmed  
Connected to server..  
Start Chatten'  
youssef: hi guyz!  
hi youssef, sup!?  
rul3z: hi youssef!  
youssef: not much, u ?  
C:\WINDOWS\system32\cmd.exe - chatClient.py  
Microsoft Windows XP [Version 5.1.2600]  
(C) Copyright 1985-2001 Microsoft Corp.  
C:\Documents and Settings\StrikerX>cd C:\Python25\pyFS  
C:\Python25\pyFS>chatClient.py  
Alias: youssef  
Connected to server..  
Start Chatten'  
hi guyz!  
ahmed: hi youssef, sup!?  
rul3z: hi youssef!  
not much, u ?  
C:\WINDOWS\system32\cmd.exe - chatClient.py  
Microsoft Windows XP [Version 5.1.2600]  
(C) Copyright 1985-2001 Microsoft Corp.  
C:\Documents and Settings\StrikerX>cd C:\Python25\pyFS  
C:\Python25\pyFS>chatClient.py  
Alias: rul3z  
Connected to server..  
Start Chatten'  
youssef: hi guyz!  
ahmed: hi youssef, sup!?  
hi youssef!  
youssef: not much, u ?  
Python Shell  
File Edit Shell Debug Options Windows Help  
Python 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.13  
32  
Type "copyright", "credits" or "license()" for more info  
*****  
Personal firewall software may warn about the connection  
makes to its subprocess using this computer's internet  
interface. This connection is not visible on any other  
interface and no data is sent to or received from the  
*****  
IDLE 1.2  
>>> ===== RESTART =====  
>>>  
>>> Server started..  
connection from: ('127.0.0.1', 1580)  
('127.0.0.1', 1580) closed..  
connection from: ('127.0.0.1', 1584)  
connection from: ('127.0.0.1', 1585)  
connection from: ('127.0.0.1', 1586)
```

السرفر

```
#!/bin/python
```

```

import socket
import threading

class ChatServer(object):
    """Indexer..."""

    def __init__(self, port):

        self.port=port
        addr=("", self.port)
        self._bufsize=2048

        self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) #Quick restarts.
        self.listener.bind(addr)
        self.alSocks=[]
        #self.tListening=threading.Thread(target=self.listeningHandler, args=[])
        #self.tListening.start()

        self.listeningHandler() #Start listening...


    def listeningHandler(self):
        self.listener.listen(5)
        print "Server started.."
        while True:
            clientSocket, clientAddr=self.listener.accept()
            #Handle the client in a new thread...
            self.tHandleClient=threading.Thread(target=self.clientHandler, args=[clientSocket])
            self.tHandleClient.start()


    def clientHandler(self, clientSocket):

        self.alSocks += [clientSocket]
        print "connection from: ", clientSocket.getpeername()
        self._bufsize=2048
        try:
            while True:

                data=clientSocket.recv(self._bufsize)
                if not data:
                    break
                #handle sending all recieved in another thread..
                #serverToAll=threading.Thread(target=self.serverToAll, args=[clientSocket, data])
                #serverToAll.start()
                self.serverToAll(clientSocket, data)

```

```

except Exception:
    #don't act
    print clientSocket.getpeername(), " closed..."
finally:
    self.alSocks.remove(clientSocket)
    clientSocket.close()

def serverToAll(self, currentClient, data):
    try:
        for sock in self.alSocks:
            if not sock == currentClient:
                sock.send(data)
            else:
                pass
    except Exception, e:
        print e

if __name__=="__main__":

    chatServer=ChatServer(8030)

```

ملحوظة لإنشاء خيط جديد فى برنامجك قم بإستخدام الصف threading.Thread لإنشاء كائن وقم بتحديد ال target وهى الميثود التى سيتم تنفيذها بصورة خارجية فى ذلك الثريد و args هى عبارة عن list تحوى المعاملات (args) الخاصة بتلك الميثود

العميل

```

#!/bin/python

import socket
import threading

class Peer(object):

    def __init__(self, serverAddr=('localhost', 8030), alias="anonymouse"):

        self.serverAddr=serverAddr
        self.tcpClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.alias=alias
        self._bufsize=2048
        self.tcpClient.connect(self.serverAddr)

        print "\nConnected to server.."
        #self.tClientToServer=threading.Thread(target=self.clientToServerHandler, args=[])
        #self.tClientToServer.start()

```



```

        self.clientToServerHandler()

def clientToServerHandler(self):
    print "Start Chattin' \n"
    while True:

        data=raw_input()
        msg=alias+": "+data
        if not data:
            break
        serverToClient=threading.Thread(target=self.serverToClientHandler, args=[])
        serverToClient.start()
        #self.serverToClientHandler()

        self.tcpClient.send(msg)


def serverToClientHandler(self):

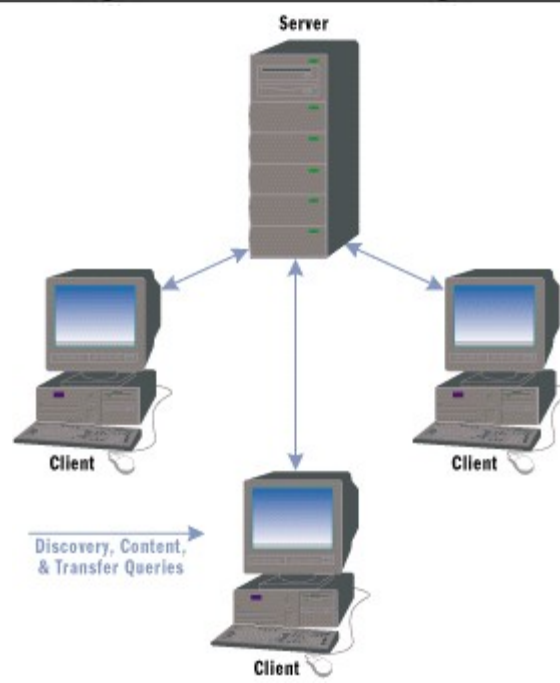
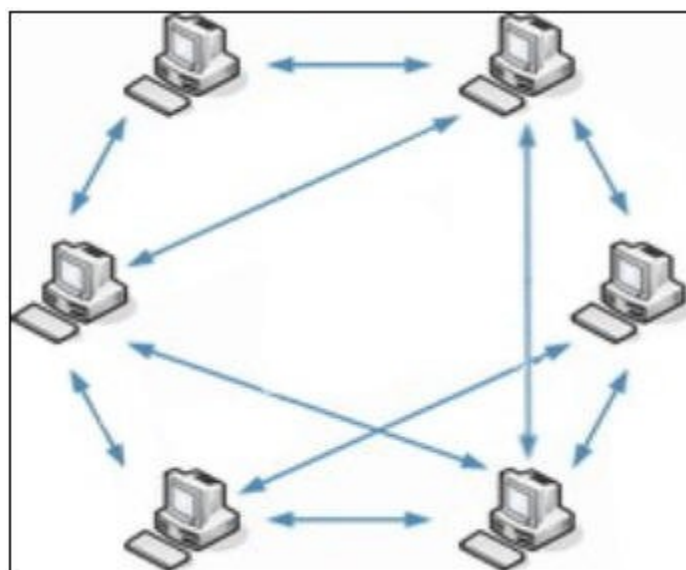
    while True:
        data=self.tcpClient.recv(self._bufsize)
        if not data:
            break
        print data


if __name__=="__main__":

    alias=raw_input("Alias: ")
    peer=Peer(alias=alias)

```

App with a BUG



Server-Based Network وهو يعتمد إن يكون فى Server يعمل بدور وسيط بين الأجهزة وينطلق عليه Discovery Server أو Indexer لأنه سيكون على Database بتتخزن فيها معلومات كل Peer من حيث ال Shared Files و ال Alias و .. إلخ

ملحوظة: ال Model الثانى مش يعتبر P2P بصورة كاملة

اولا هنعمل Module بسيطة نعمل بيها Encapsulation لما سبق

```
class PeerInfo(object):
    def __init__(self, alias, sharedFiles, listeningPort):
        self.alias=alias
        self.sharedFiles=sharedFiles
        self.listeningPort=listeningPort
    def __str__(self):
        sb="Alias: " + self.alias
        sb += "\nFiles: " + str(self.sharedFiles)
        sb += "\nListens At:" + str(self.listeningPort)
        return sb

def serialize(obj):
    """Serialize an object to a string..."""
    return marshal.dumps(obj)
def deserialize(objString):
    """Deserialize an object string..."""
    return marshal.loads(objString)
if __name__=="__main__":
    p=PeerInfo("ahmed", [1, 2, 3 ,4], 80)
    print p
    print "alias: ",p.alias
    print "files: ", p.sharedFiles
    print "port : ", p.listeningPort
```

ال class peerInfo هو class هنخزن فيه بيانات المستخدم عشان تكون اسهل فى التعامل والإستعلام وهى ال alias, sharedFiles, ListeningPort
هنعمل 2 methods بسيطين جدا لمعالجة ال Objects اللى هتتبع على السوكيت وهما serialize و deserialize

بصورة مبدئية لما نفكر فى ال Discovery Server لازم يكون فيه شوية مميزات
1- ان يتم تسجيل بيانات اى حد يعمل Connect والبيانات دى بتشمل Alias, SharedFiles, ListeningPort
2- نقدر نستعلم عن الملفات الموجودة على ال Clients الآخرين

3- كل Client يقدر يعدل على بياناته
4- يقدر يعرض كل ال Clients الآخرين
5- يقدر يعدل على ال SharedFiles
6- اقصى عدد يقدر يتصل بالسرفر
7- وطبعاً لازم يقدر يتعامل مع أكثر من Client فى نفس الوقت
8-
وهكذا

ملحوظة: ممكن تعمل Clients ليهم صلاحيات اعلى "مثلا اللى مشترين pro للمنتج بتاعك"

اولا إحنا مش هنعمل DB لأننا مش منتج كبير .. احنا يدوب بنعرض الفكرة .. فال DB بتاعتنا هتكون عبارة عن Dictionary او HashTable او غيرهم حسب اللغة اللى إنت جاي منها

```
{'clientIP:port' : peerInfo Object}
```

اولا هنعمل socket object ونعمل set لل options بتاعته "اهم شئ REUSEADDR وهى بتسمحك بإستخدام ال port مرة اخرى مباشرة فى حال إنك قفلت السرفر لسبب ما "وغالباً هنا الإختبار للبرنامج"
ملحوظة : مع إننا هندعم ال Chat بين المستخدمين ولكننا هنستخدم TCP مش UDP -لأننا هنتعامل مع نقل لملفات-

```
(self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM  
(self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1
```

ال Class هيبداً بالصورة دى .. وهنحدد فيه ال port اللى هيشغل عليه السرفر .. واقصى عدد للمستخدمين والأوامر المدعومة
register/ : لتسجيل ال client
setSharedFiles/ : لتعديل ال ملفات اللى معمولها share
setNick/ : لتغيير ال Nick او ال alias
showall/ : لعرض جميع المستخدمين
query/ : للإستعلام عن ملف ما

```

class DiscoveryServer(object):
    '''Indexer...'''
    def __init__(self, port, maxPeers=5):
        self.port=port
        addr=('', self.port)
        self.maxPeers=maxPeers
        self.supportedCommands=["/register", "/setNick", "/setSharedFiles",
"/showall", "/query"]
        self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)
        self.listener.bind(addr)
        self.tListening=threading.Thread(target=self.listeningHandler,
args=[])
        self.tListening.start()
        self.alSocks=[]
        # {clientAddr:peerInfo Object}
        self.db={}
        self.log=[]

self.BUF=2048

```

لاحظ منش هسجل log فى البرنامج .. هسييهاك واجب او فرصة إنك تعدل شئ لشيئ احسن وهكذا .. بحيث إنك تبقة مشارك
self.alSocks هى List هنضيف فيها كل ال Sockets المفتوحة "هتعرف بعد شوية"
لاحظ فى السطر دا

```
self.tListening=threading.Thread(target=self.listeningHandler, args=[])
```

إننا انشأنا thread للميثود listeningHandler وال args التى سنمررها [] List فاضية
لأنها منش بتاخذ args
ولبدأ ال thread بنستخدم start method
ال implementation الخاص بال method دى

```

def listeningHandler(self):
    self.listener.listen(self.maxPeers)
    print "Server Started..."
    while True:

```

```

clientSocket, clientAddr=self.listener.accept()
print "Gotta a connection from", clientAddr
tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
tClientHandling.start()
clientSocket.close()

```

بنخلي السرفر يبدأ فى عملية ال listening وبعد كذا بنعمل forever loop بنتعامل فيها مع اى client بيعمل connect وبمجرد مايعمل Connect نهندله فى thread جديد بإستخدام method بإسم clientHandler

```

tClientHandling=threading.Thread(target=self.clientHandler,
args=[clientSocket])

```

ملحوظة accept method بتدى return ب tuple مكونة من ال clientSocket, clientAddress

```

def clientHandler(self, clientSocket):
    self.alSocks += [clientSocket]
    formattedAddress=clientSocket.getpeername()[0]+":"+str(clientSocket.getpeername()[1])
    objString=""
    try:
        while True:
            objString=clientSocket.recv(self.BUF)
            if not objString:
                break
            data=deserialize(objString)
            #print data
            tAnalyzeData=threading.Thread(target=self.analyzeData, args=[data, clientSocket])
            tAnalyzeData.start()

            objString=""

    except Exception, e:
        print "E: ", e
        print clientSocket.getpeername(), " closed.."
        self.alSocks.remove(clientSocket)
        del self.db[formattedAddress]

```

ال method دى خاصة بالتعامل مع ال Client وكل اللى يتعمله فى الحقيقة هى إنها بتضيف ال ClientSocket إلى ال alSocks list اللى بال server class وتبدأ thread جديد تحلل فيه الداتا الجاية من ال Client وطالما هتتحلل data وفى thread جديد يبقا لازم نعمل method تبقة مسؤلة عن عملية ال تحليل او ال parsing لل دانا واكيد ال method دى هتاخذ argument وهى ال data واللى ارسل الداتا وهو ال clientSocket

```
tAnalyzeData=threading.Thread(target=self.analyzeData, args=[data, clientSocket])
tAnalyzeData.start()
```

جميل جدا .. تحليل الداتا فى ال analyzeData method

```
def analyzeData(self, data, clientSocket):
    formattedAddress=clientSocket.getpeername()
    [0]+" ":""+str(clientSocket.getpeername()[1])
    try:
        if isinstance(data, tuple): #registering...
            pInfo=PeerInfo(data[1], data[2], data[3]) #(register,
alias, files, port)
            print "Registering: ", pInfo.alias
            print pInfo
            self.db[formattedAddress]=pInfo #peerInfo object..
            print self.db
        if isinstance(data, list):
            try:
                #split the sender's alias..
                #recvd=['tina: /showall']
                recvd=data[0].split(": ")[1]
                cmd=recvd.split(" ")[0]
                # test cmd...
                if not cmd in self.supportedCommands:
                    self.sendToAll(data, clientSocket)
                else:
                    if cmd=="/showall":
                        self.showAllTo(clientSocket)
                    if cmd=="/query":
                        fileName=recvd.split(" ")[1]
                        self.queryFile(fileName, clientSocket)
                    if cmd=="/setNick":
                        self.setNick(formattedAddress, recvd.split(" ")
[1])
```

```
except Exception,e :
    print "Error: ", e
```

```
except Exception, e:
    print "Data: ", data
    print "Error: ", e
    self.alSocks.remove(clientSocket)
```

انا اتبع طريقة بسيطة شوية هنا وهو انى بعث تسجيل البيانات على صورة tuple مكونة من (register", files=[], listeningPort/") وحولتها ل peerInfo object وبعد كذا بعض الاختبارات لل command نفسه لو كان query نعمل كذا لو كان كذا نعمل كذا وإذا مش كان موجود فى الأوامر المدعومة بال Server تبقة مجرد رسالة تتبع لى لكل الأهل والأحباب فى حال إن حصل أى ايروور(خطأ) يتم حذف ال Client من ال alSocks لنعرف إنه غير active او متصل بالسرفر حاليا

طب فى حال لو إن الداتا اللى إتبعنت مجرد مسح عادية ؟ يعنى هحتاج نبعثها لكل المستخدمين ماعدا بالطبع اللى ارسلها مش كذا ؟

نحصل على كل المستخدمين منين ؟

اها تمام من ال alSocks اللى بتعبر عن كل ال Clients المتفاعلين مع السرفر حاليا

قشطة

```
def sendToAll(self, msg, clientEx):
    print "Message Recieved: ", msg
    try:
        for sock in self.alSocks:
            if not sock==clientEx:
                sock.send(serialize(msg))
            else:
                pass
    except Exception, e:
        print "Error: ", e
```

جميل جدا.. فى حال لو المستخدم عايز يستعرض المستخدمين الموجودين "اكيد رد السرفر هيكوّن ليه لوحده مش كذا ؟" super

فهتكون ال showallTo method كالتالى


```
def showAllTo(self, clientSocket):

    data="\n-----\nOnline Users:\n"
    for addr, pInfo in self.db.items():
        data += pInfo.alias + " -> " +addr +"\n"
    data += "\n-----\n"
    print data

    clientSocket.send(serialize(data))
```

ملحوظة انا فضلت اعمل serialize لكل الداتا اللى هتتبع حتى لو strings عشان مش اقعد اتعب نفسى فى ال de-bugging واعملهم de-serialize فى الطرف التانى

لمعالجة امر تغيير ال Nick وهو setNick/

```
def setNick(self, to, newNick):
    self.db[to].alias=newNick
    print "Nick Changed..."

    print self.db[to]
```

جميل جدا ناقص ال Querying files اللى هيطلب الإستعلام هو واحد مش كل الناس فلازم نمرر ال socket بتاعه فى ال method + نبعتله ال داتا

```
def queryFile(self, fileName, clientSocket):
    print "Querying: ", fileName
    data=""
    for addr, pInfo in self.db.items():
        if fileName in pInfo.sharedFiles:
            data += "\n"+addr + " | " + pInfo.alias + " => " +
            fileName
            data += "\n\t" + pInfo.alias + " Listens at: " +
            str(pInfo.listeningPort)
    print data

    clientSocket.send(serialize(data))
```

كدا انهينا ال Discovery Server او ال Indexer

ننقل على ال Client او ال Peer
ال Peer لازم يتفاعل مع ال Server و يقدر يحمل ملفات من ال Peers التانيين و ان يكون فى peers يقدر و يحملو
منهم بردو فكذا هنعمل client ليتعامل مع السرفر ونعمل internal server يكون مسئول عن عملية ال دونلود او ال
fetching

ال Constructor بتاعه هياخد
alias -1
list of sharedFiles -2
-3 ال endPoint الخاصة بالسرفر
ال endPoint هى ال Server IP + port

بمجرد ما يتم ال connection مع ال server نهنده فى thread جديد

```
def __init__(self, alias, serverAddr=(), sharedFiles=[]):  
  
    self.alias=alias  
    self.serverAddr=serverAddr  
    self.sharedFiles=sharedFiles  
    self.tcpClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    self.tcpClient.connect(self.serverAddr)  
    self.BUF=1024  
    self.listeningPort=rnd.randint(8081, 10000)  
    self.pInfo=PeerInfo(self.alias, self.sharedFiles, self.listeningPort)  
  
    print "\nConnected to server..."  
    self.tClientToServer=threading.Thread(target=self.clientToServerHandler, args=[])  
    self.tClientToServer.start()
```

رائع جدا
بس بردو لازم نجهز لل Peers اللى هيحاولو يعملو fetch ل files من عندنا فهنعمل server object وبردو نشغله فى
thread جديد صح كذا ؟

```
#listen for connections in background..  
self.addr=("", self.listeningPort)  
self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  
self.listener.bind(self.addr)  
self.tListening=threading.Thread(target=self.listeningHandler, args=[])
```

```
self.tListening.start()
```

جميل بما إنى على جهاز واحد فتهتصل مشكلة انى مش هقدر اعمل set ل port معين اربطه بالسرفر لأن بكل بساطة كل ال Peers هيحاولو يربطو على السرفر دا .. فبكل بساطة هنستخدم randint method الموجودة ب random module ونحصل على اى random int بس لاحظ إنه يكون فوق ال 1024 وبردو نبلغ بيه ال Discovery Server

```
self.listeningPort=rnd.randint(8081, 10000)
```

اولا ال registerAtServer Method

```
def registerAtServer(self):
    msg=("/register", self.alias, self.sharedFiles, self.listeningPort)
    self.tcpClient.send(serialize(msg))
```

وهى اول ميثود هيتم إستدعائها بمجرد إنى اعمل connect على السرفر

```
def clientToServerHandler(self):
    print "Start Chatting.."
    #first register the files...
    self.registerAtServer()
    while True:
        tServerToClient=threading.Thread(target=self.serverToClientHandler, args=[])
        tServerToClient.start()
        data=raw_input()
        if not data: continue
        if data.lower=="exit": exit()

        if data.split(" ")[0]=="/fetch":
            fileneeded=data.split(" ")[1]
            addr=data.split(" ")[2]
            tFetchFile=threading.Thread(target=self.fetchFile, args=[addr, fileneeded])
            tFetchFile.start()
        else:
            msg=self.alias+": "+data
            self.tcpClient.send(serialize([msg]))
```

بما إن ال client مش ليه غير وظيفة واحدة بس اللى مش هتتعامل مع ال server وهى ال fetching فهختبرها الأول إذا هى المطلوبة او لا.. فى حال آه نبدأها فى thread جديد خاص بالتعامل معاها لو لا تبقة مجرد رسالة او امر زى إستعلام او تغيير بيانات فهنبعتة للسرفر والسرفر يحلله

نهدل رسائل ال server فى thread هستخدم ال serverToClientHandler method

```
def serverToClientHandler(self):
    while True:
        data=deserialize(self.tcpClient.recv(self.BUF))
        if not data: break
        if isinstance(data, list): #data ['tina: hi']
            print data[0]
        else:
            print data
```

عملية ال fetching بكل بساطة هنتعامل معاها كالتالى

```
def fetchFile(self, addr, fileneeded):
    #addr is formatted => addr:listeningPort
    endPoint=addr.split(":")[0], int(addr.split(":")[1])
    fetchTCPClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    fetchTCPClient.connect(endPoint)
    fetchTCPClient.sendall(serialized("/fetch", fileneeded))
    tDownloadFile=threading.Thread(target=self.downloadFile, args=[fetchTCPClient, fileneeded])
    tDownloadFile.start()
```

ونبدأ فيها thread جديد لل download يستخدم ال downloadFile method و هيمرر ال tcpClient, الفايل المطلوب ك args ليها

```
def downloadFile(self, fetchTCPClient, fileneeded):
    f=file(fileneeded, "wb")
    while True:
        try:
            buf=fetchTCPClient.recv(self.BUF)
            if not buf:
                break
            f.write(buf)
        except EOFError, eofErr:
            print "EOFError: ", eofErr
        except Exception, e:
            print "Error: ", e
            break
    print "File Downloaded!"
    f.close()
    fetchTCPClient.close()
```

جميل جدا احنا كدا شبه خلصنا .. ناقص شئ واحد بس وهو التعامل مع ال clients اللى نريد ان يحملو فايلات مننا -خليك فاكّر إنّنا already مشغلين thread خاص بال listening للطلبات

```
self.tListening=threading.Thread(target=self.listeningHandler, args=[])
self.tListening.start()
```

جميل ال thread دا بيستخدم method listeningHandler وال method دى مش هتأخذ args

```
def listeningHandler(self):
    self.listener.listen(5)
    while True:
        clientSocket, clientAddr=self.listener.accept()
        tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
        tClientHandling.start()
```

محتاجين إنها تكون Multi-threaded عشان مش نربط نفسنا مع مستخدم واحد بس
فهنعمل thread جديد يعالج كل client يعمل connect على ال internal server

```
def clientHandler(self, clientSocket):
    rcvd=clientSocket.recv(self.BUF)
    data=deserialize(rcvd)
    if isinstance(data, tuple):
        if data[0]=="/fetch": #go on..
            fileneeded=data[1] #(/fetch, fileneeded, from)
            print "File Request: ", fileneeded
            f=file(fileneeded, "rb")
            while True:
                try:
                    buf=f.read(self.BUF)
                    if not buf:
                        break
                    clientSocket.send(buf)
                except Exception, e:
                    print "Error: ", e
                    break

            f.close()
            clientSocket.close()
            print "Copied!"
```

ويس كدا

```
if __name__=="__main__":
```

```
alias=raw_input("Alias: ")
sharedFiles=os.listdir(os.getcwd())
peer=Peer(alias, ('localhost', 8080), sharedFiles)
```

طبعاً تقدر تطبطها بحيث إنك تمرر ال addr الخاص بال server من ال command line او حتى من prompt الأكواد

Discovery Server

Code:

```
#!/bin/python
import socket
import sys
import os
import threading
from utils import serialize, deserialize, PeerInfo
class NotSupportedCommand(Exception):
    pass
class DiscoveryServer(object):
    "Indexer..."
    def __init__(self, port, maxPeers=5):
        self.port=port
        addr=("", self.port)
        self.maxPeers=maxPeers
        self.supportedCommands=["/register", "/setNick", "/setSharedFiles", "/showall", "/query"]
        self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.listener.bind(addr)
        self.tListening=threading.Thread(target=self.listeningHandler, args=[])
        self.tListening.start()
        self.alSocks=[]
        # { clientAddr:peerInfo Object}
        self.db={}
        self.log=[]
        self.BUF=2048

    def listeningHandler(self):
        self.listener.listen(self.maxPeers)
        print "Server Started..."
        while True:
            clientSocket, clientAddr=self.listener.accept()
```

```

    print "Gotta a connection from", clientAddr
    tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
    tClientHandling.start()
    clientSocket.close()
def clientHandler(self, clientSocket):
    self.alSocks += [clientSocket]
    formattedAddress=clientSocket.getpeername()[0]+":"+str(clientSocket.getpeername()[1])
    objString=""
    try:
        while True:
            objString=clientSocket.recv(self.BUF)
            if not objString:
                break
            data=deserialize(objString)
            #print data
            tAnalyzeData=threading.Thread(target=self.analyzeData, args=[data, clientSocket])
            tAnalyzeData.start()

            objString=""

    except Exception, e:
        print "E: ", e
        print clientSocket.getpeername(), " closed.."
        self.alSocks.remove(clientSocket)
        del self.db[formattedAddress]

def analyzeData(self, data, clientSocket):
    formattedAddress=clientSocket.getpeername()[0]+":"+str(clientSocket.getpeername()[1])
    try:
        if isinstance(data, tuple): #registering...
            pInfo=PeerInfo(data[1], data[2], data[3]) #(register, alias, files, port)
            print "Registering: ", pInfo.alias
            print pInfo
            self.db[formattedAddress]=pInfo #peerInfo object..
            print self.db
        if isinstance(data, list):
            try:
                #split the sender's alias..
                #recvd=['tina: /showall']
                recvd=data[0].split(": ")[1]
                cmd=recvd.split(" ")[0]
                # test cmd...
                if not cmd in self.supportedCommands:
                    self.sendToAll(data, clientSocket)
                else:

```

```

        if cmd=="/showall":
            self.showAllTo(clientSocket)
        if cmd=="/query":
            fileName=recvd.split(" ")[1]
            self.queryFile(fileName, clientSocket)
        if cmd=="/setNick":
            self.setNick(formatedAddress, recvd.split(" ")[1])

    except Exception,e :
        print "Error: ", e

except Exception, e:
    print "Data: ", data
    print "Error: ", e
    self.alSocks.remove(clientSocket)

def queryFile(self, fileName, clientSocket):
    print "Querying: ", fileName
    data=""
    for addr, pInfo in self.db.items():
        if fileName in pInfo.sharedFiles:
            data += "\n"+addr + " | " + pInfo.alias + " => " + fileName
            data += "\n\t" + pInfo.alias + " Listens at: " + str(pInfo.listeningPort)
    print data
    clientSocket.send(serialize(data))

def showAllTo(self, clientSocket):

    data="\n-----\nOnline Users:\n"
    for addr, pInfo in self.db.items():
        data += pInfo.alias + " -> " +addr +"\n"
    data += "\n-----\n"
    print data
    clientSocket.send(serialize(data))

def sendToAll(self, msg, clientEx):
    print "Message Recieved: ", msg
    try:
        for sock in self.alSocks:
            if not sock==clientEx:
                sock.send(serialize(msg))
            else:
                pass
    except Exception, e:
        print "Error: ", e

```



```

def setNick(self, to, newNick):
    self.db[to].alias=newNick
    print "Nick Changed..."
    print self.db[to]

if __name__=="__main__":
    discoveryServer=DiscoveryServer(8080)

```

Utils:

```

import marshal

class PeerInfo(object):
    def __init__(self, alias, sharedFiles, listeningPort):
        self.alias=alias
        self.sharedFiles=sharedFiles
        self.listeningPort=listeningPort
    def __str__(self):
        sb="Alias: " + self.alias
        sb += "\nFiles: " + str(self.sharedFiles)
        sb += "\nListens At:" + str(self.listeningPort)
        return sb

def serialize(obj):
    "Serialize an object to a string..."
    return marshal.dumps(obj)
def deserialize(objString):
    "Deserialize an object string..."
    return marshal.loads(objString)
if __name__=="__main__":
    p=PeerInfo("ahmed", [1, 2, 3 ,4], 80)
    print p
    print "alias: ",p.alias
    print "files: ", p.sharedFiles
    print "port : ", p.listeningPort

```

Peer

```
#!/bin/python
```

```

import socket
import sys
import os
import threading
from utils import serialize, deserialize, PeerInfo
import random as rnd

class Peer(object):
    def __init__(self, alias, serverAddr=(), sharedFiles=[]):

        self.alias=alias
        self.serverAddr=serverAddr
        self.sharedFiles=sharedFiles
        self.tcpClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.tcpClient.connect(self.serverAddr)
        self.BUF=1024
        self.listeningPort=rnd.randint(8081, 10000)
        self.pInfo=PeerInfo(self.alias, self.sharedFiles, self.listeningPort)

        print "\nConnected to server..."
        self.tClientToServer=threading.Thread(target=self.clientToServerHandler, args=[])
        self.tClientToServer.start()
        #listen for connections in background..
        self.addr=("", self.listeningPort)
        self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.listener.bind(self.addr)
        self.tListening=threading.Thread(target=self.listeningHandler, args=[])
        self.tListening.start()

    def registerAtServer(self):
        msg=("/register", self.alias, self.sharedFiles, self.listeningPort)
        self.tcpClient.send(serialize(msg))

    def clientToServerHandler(self):
        print "Start Chatting.."
        #first register the files...
        self.registerAtServer()
        while True:
            tServerToClient=threading.Thread(target=self.serverToClientHandler, args=[])
            tServerToClient.start()
            data=raw_input()
            if not data: continue
            if data.lower=="exit": exit()

            if data.split(" ")[0]=="/fetch":

```

```

        fileneeded=data.split(" ")[1]
        addr=data.split(" ")[2]
        tFetchFile=threading.Thread(target=self.fetchFile, args=[addr, fileneeded])
        tFetchFile.start()
    else:
        msg=self.alias+": "+data
        self.tcpClient.send(serialize([msg]))
def serverToClientHandler(self):
    while True:
        data=deserialize(self.tcpClient.recv(self.BUF))
        if not data: break
        if isinstance(data, list): #data ['tina: hi']
            print data[0]
        else:
            print data
def fetchFile(self, addr, fileneeded):
    #addr is formatted => addr:listeningPort
    endPoint=addr.split(":")[0], int(addr.split(":")[1])
    fetchTCPClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    fetchTCPClient.connect(endPoint)
    fetchTCPClient.sendall(serialize("/fetch", fileneeded))
    tDownloadFile=threading.Thread(target=self.downloadFile, args=[fetchTCPClient, fileneeded])
    tDownloadFile.start()
def downloadFile(self, fetchTCPClient, fileneeded):
    ## try:
    ##     f=file(fileneeded, "wb")
    ##     while True:
    ##         dataRcvd=fetchTCPClient.recv(self.BUF)
    ##         if not dataRcvd: break
    ##         f.write(dataRcvd)
    ##
    ##     print "File Downloaded!"
    ##
    ## except Exception, e:
    ##     print "Error: ", e
    ##
    ## finally:
    ##     print "Closing the file.."
    ##     fetchTCPClient.close()
    ##     f.close()
    f=file(fileneeded, "wb")
    while True:
        try:
            buf=fetchTCPClient.recv(self.BUF)
            if not buf:
                break

```

```

        f.write(buf)
    except EOFError, eofErr:
        print "EOFError: ", eofErr
    except Exception, e:
        print "Error: ", e
        break
    print "File Downloaded!"
    f.close()
    fetchTCPClient.close()

def listeningHandler(self):
    self.listener.listen(5)
    while True:
        clientSocket, clientAddr=self.listener.accept()
        tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
        tClientHandling.start()

def clientHandler(self, clientSocket):
    rcvd=clientSocket.recv(self.BUF)
    data=deserialize(rcvd)
    if isinstance(data, tuple):
        if data[0]=="/fetch": #go on..
            fileneeded=data[1] #(/fetch, fileneeded, from)
            print "File Request: ", fileneeded
            f=file(fileneeded, "rb")
            while True:
                try:
                    buf=f.read(self.BUF)
                    if not buf:
                        break
                    clientSocket.send(buf)
                except Exception, e:
                    print "Error: ", e
                    break

            f.close()
            clientSocket.close()
            print "Copied!"

if __name__=="__main__":
    alias=raw_input("Alias: ")
    sharedFiles=os.listdir(os.getcwd())
    peer=Peer(alias, ('localhost', 8080), sharedFiles)

```

جميل كذا انهيت البرنامج ولكن فى trick اكيد لأننا كاتبين عنوان الفصل App with a Bug فى عدة مشاكل فى التطبيق

1- استخدام marshal (يفضل استخدم pickle لعمل serialize للكائنات) ولكن بما ان التطبيق فورى فمش ليها تأثير
2- الإستخدام المكثف للثريدينج (يفضل انشاء الثريدات كالتالى

1- ثريد معالجة كل عميل
وتحليل الداتا فى نفس الثريد وليس ثريد آخر (ربما لن تشعر بالفرق الآن ولكن عند محاولة نقل التطبيق الى قاعدة بيانات)

وبالنسبة للعميل كذلك لاتقم بتحليل الداتا فى ثريد منفصل الا اذا كنت تعلم ماذا تفعل

يفضل دائما استخدام ال stdlib لضمان عدم الوقوع فى مشكلات فمثلا للثريدينج تستطيع استخدام ForkingMixIn او ThreadingMixIn

إضافات للتطبيق

- استخدام قاعدة بيانات (ربما sqlite3 مثلا او MySQL)

2- وفر خاصية ال resume للملفات اللى بتتحمل

3- فعل ال LOG "استفيد من الرسائل اللى بيتعملها print" على السرفر

4- استخدم ال Discovery Server ك Web Service

5- واجهة رسومية بسيطة

6- اوامر اكثر لهندلة ال PeerInfo وامتيازات خاصة

FTPing

بايثون بتقديمك موديلز عديدة لمهام كثيرة مثل ftplib (للقيام بعمليات العميل الخاصة ببروتوكول FTP) استخدامها مباشرة
1- استدعاء الموديل

```
import ftplib
```

2- بيانات الدخول

```
HOST="YOUR_HOST"  
USER="YOUR_USERNAME"  
PASSWD="YOUR_PASSWORD"
```

3- انشاء كائن من الصف FTP

```
ftp=ftplib.FTP()
```

4- انشاء الإتصال باستخدام الطريقة connect التي تأخذ معاملات ال host و رقم البورت (افتراضيا 21)

```
(ftp.connect(HOST, 21
```

5- الدخول باستخدام الطريقة login وتأخذ معاملات user, password

```
ftp.login(USER, PASSWD)
```

6- التعامل الخاص بيك

بعض الطرق

getwelcome()	لعرض رسالة الترحيب
rename(old, new)	لإعادة تسمية old ب new
cwd(path)	تغيير مجلد العمل الحالي current working directory
pwd()	مسار مجلد العمل الحالي
mkd(path)	انشاء مجلد path
delete(f)	حذف الملف f
rmd(d)	حذف المجلد d
size(f)	الحصول على مساحة ملف f
quit()	ارسال رسالة QUIT
close()	لإنهاء الإتصال
set_pasv(boolean)	هل نوع الإتصال سلبى ام لا ؟ passive mode
retrbinary(command, callback[, maxblocksize[, rest]])	للحصول على ملف RETR f واستدعاء callback على كل block عند اكتمال تحميلها
storbinary(cmd, file[, block])	STOR_ تخزين ملف file ما مع تحديد مساحة قطع النقل لكل مرة
abort()	الغاء عملية نقل ملف
dir(p)	عرض ال listing الخاصة ب p
للمزيد راجع الوثائق الرسمية او راجع سورس الموديل ftplib.py	

XMLRPC what?

إذا لم تكن مهتما ب XML-RPC فكن حرا للإنتقال للفصل القادم
في أبسط الصور XML-RPC هو بروتوكول Remote Procedure Call عبر بروتوكول HTTP
للمزيد <http://en.wikipedia.org/wiki/XML-RPC>

نوضح الموضوع بمثال بسيط

```
#!/bin/python

from SimpleXMLRPCServer import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler
from SocketServer import ThreadingMixIn

class Greeter(object):

    def hi(self):
        """Returns hi message"""
        return "Hi"

    def bye(self):
        """Returns bye message"""
        return "Bye"

    def say(self, what):
        """Returns Simone says message"""
        return "Simone says: "+ what

class MyServer(ThreadingMixIn, SimpleXMLRPCServer):

    pass
```

```
def test():
    addr=("", 40002)
    srvr=MyServer(addr, SimpleXMLRPCRequestHandler)
    srvr.register_instance(Greeter())
    srvr.register_introspection_functions()
    srvr.serve_forever()
    #print "Started..."

if __name__=="__main__":
    test()
```

للوهلة الأولى سيتبادر لذهنك اننا ننشئ سرفر متعدد الخيوط (ThreadingMixIn) واستخدمنا بدل ال SocketServer صف مشابه وهو SimpleXMLRPCServer ؟

نعم بالفعل

ولدينا صف عادى جدا

```
class Greeter(object):

    def hi(self):
        """Returns hi message"""
        return "Hi"

    def bye(self):
        """Returns bye message"""
        return "Bye"

    def say(self, what):
        """Returns Simone says message"""
        return "Simone says: "+ what
```

كل ما هنالك اننا نريد ان نتيج امكانية استخدام طرق ذلك الصف عبر ال HTTP لكائن SimpleXMLRPCServer بعض الطرق مثل register_instance التى تقوم بتسجيل ذلك الكائن على السرفر و register_introspection_functions تقوم بتسجيل بعض الدوال لمعرفة مايتعلق بذلك الكائن مثل system.listMethods (للحصول على الطرق الخاصة به) و system.methodSignature للحصول على توقيع الطريقة (اسمها والمعاملات وال retrun) و methodHelp للحصول على نص المساعدة الخاص بالطريقة وذلك بوضع اسمها كمعامل لهذه الطريقة

```
srvr.register_instance(Greeter())
srvr.register_introspection_functions()
```

قم بتشغيل ال سرفر ونأى للعميل

```
import xmlrpclib

s = xmlrpclib.ServerProxy('http://localhost:40002')
print s.system.listMethods()
print s.hi()
print s.bye()
print s.say("Something")

print s.system.methodHelp('say')
```

1- ننشئ كائن ServerProxy نحدد فيه عنوان السرفر ورقم البورت الذى ينصت اليه

```
s = xmlrpclib.ServerProxy('http://localhost:40002')
```

2- للحصول على قائمة الطرق المتاحة على السرفر نستخدم system.listMethods
3- استغلال الطرق المتاحة مثل استدعاء الطرق hi او bye

```
print s.hi()
print s.bye()
```

4- ايضا استغلال الطرق التى قد تأخذ معاملات ما مثل say

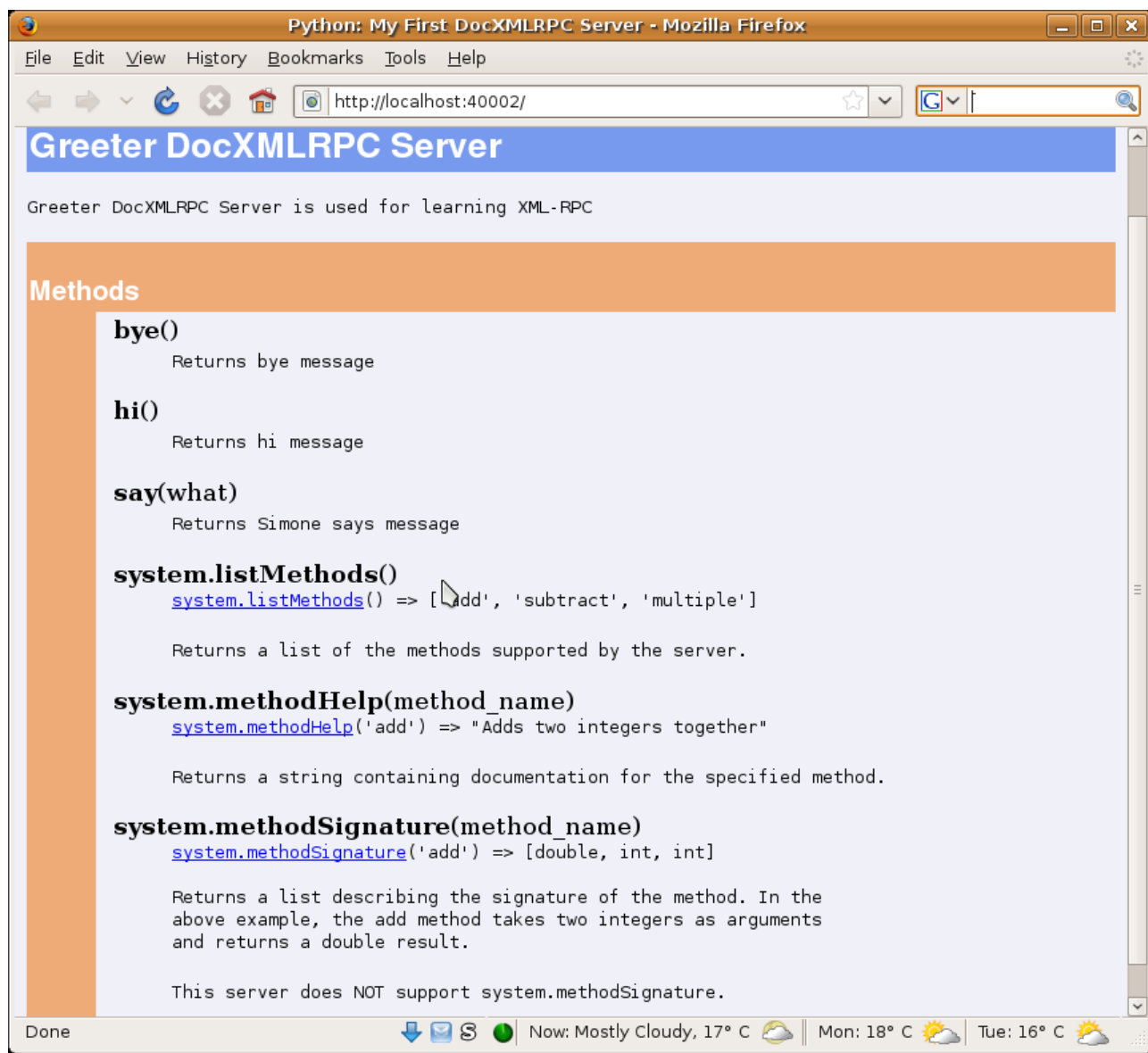
```
print s.say("Something")
```

سيكون الناتج

```
striky@striky-desktop:~/workspace/pytut/src$ python xmlrpcclient1.py
['bye', 'hi', 'say', 'system.listMethods', 'system.methodHelp', 'system.methodSignature']
Hi
Bye
Simone says: Something
Returns Simone says message
```

يوجد صف اخر وهو المفضل استخدامه DocXMLRPCServer وذلك للمساعدة فى عرض ال documentation ايضا

DocXMLRPCServer



مثل سابقه ولكن له بعض الطرق الإضافية مثل `set_server_title` لوضع العنوان و `set_server_name` لوضع اسمه في أعلى الصفحة و `set_server_documentation` لوضع وثيقة خاصة به

```
from DocXMLRPCServer import DocXMLRPCServer, DocXMLRPCRequestHandler
from SocketServer import ThreadingMixIn

class Greeter(object):

    def hi(self):
        """Returns hi message"""
        return "Hi"

    def bye(self):
        """Returns bye message"""
        return "Bye"

    def say(self, what):
        """Returns Simone says message"""
        return "Simone says: "+ what

class MyServer(ThreadingMixIn, DocXMLRPCServer):

    pass

def test():
    addr=("", 40002)
    srvr=MyServer(addr, DocXMLRPCRequestHandler)
    ##server methods...
    srvr.set_server_title("My First DocXMLRPC Server")
    srvr.set_server_name("Greeter DocXMLRPC Server")
    srvr.set_server_documentation("Greeter DocXMLRPC Server is used for learning XML-RPC")
    srvr.register_instance(Greeter())
    srvr.register_introspection_functions()
    srvr.serve_forever()
    #print "Started..."

if __name__=="__main__":
    test()
```

تستطيع الإستفادة من هذا السرفر فى لغات اخرى ايضا مثل روبي مثلا

```
##Ruby
striky@striky-desktop:~$ irb
irb(main):001:0> require "xmlrpc/client"
=> true
irb(main):002:0> require "pp"
=> true
irb(main):004:0> s=XMLRPC::Client.new2("http://localhost:40002")
=> #<XMLRPC::Client:0xb7aa2a50 @user=nil, @proxy_port=nil, @auth=nil, @cookie=nil, @create=nil,
@port=40002, @http=#<Net::HTTP localhost:40002 open=false>, @proxy_host=nil, @http_last_response=nil,
@parser=nil, @timeout=30, @path="/RPC2", @password=nil, @http_header_extra=nil, @use_ssl=false,
@host="localhost">
irb(main):005:0> s
=> #<XMLRPC::Client:0xb7aa2a50 @user=nil, @proxy_port=nil, @auth=nil, @cookie=nil, @create=nil,
@port=40002, @http=#<Net::HTTP localhost:40002 open=false>, @proxy_host=nil, @http_last_response=nil,
@parser=nil, @timeout=30, @path="/RPC2", @password=nil, @http_header_extra=nil, @use_ssl=false,
@host="localhost">
irb(main):006:0> s.call('hi')
=> "Hi"
irb(main):007:0> s.call('bye')
=> "Bye"
irb(main):008:0> s.call('say', "Hello, World!")
=> "Simone says: Hello, World!"
```

Quote of the Day

هتنبشئ فى المثال دا سرفس مشابهه ل "اقتباس اليوم"
وهو سرفر يعمل فى الخلفية ويرسل اقتباس عشوائى لكل عميل ثم يغلق الإتصال معه

1- ملف ال quotes.txt

```
Never pretend to a love which you do not actually feel, for love is not  
ours to command. --Alan Watts  
To love deeply in one direction makes us more loving in all others.  
--Anne-Sophie Swetchine  
There is always some madness in love. But there is also always some reason  
in madness. --Friedrich Nietzsche  
Life is wasted on the living. --Douglas Adams  
Life is just a chance to grow a soul. - A. Powell
```

2- ال quoter وهو موديل للحصول على الإقتباسات من ملف ما

```
#!/bin/python  
  
from __future__ import with_statement  
import random as rnd  
  
def get_quotes(f="quotes.txt"):      
    """quotes are separated by \n  
    with open(f) as fh:  
        lines=fh.read()  
        quotes=lines.split("\n")  
    return quotes  
  
def get_random_quote(quotes=get_quotes()):  
    return rnd.choice(quotes)  
  
if __name__=="__main__":  
    print get_random_quote()
```

كما نرى السطر الأول بيستدعى ال with_statement من المستقبل d: الإمكانيات التى تم تقرير اضافتها
للإصدارات الأحدث من بايثون والموديل random للحصول على اختيار عشوائى
*الدالة get_quotes متغيرة فى f وهو مسار الملف الذى يحوى الإقتباسات وجعلناها quotes.txt افتراضيا
لاحظ فى إستخدام with تم استبدال try/except لأنها تتم داخليا
نحصل على الإقتباسات "كل سطر يحوى اقتباس" (ممكّن تعمل strip للتأكيد على عدم وجود اسطر زائدة فى
الملف

*الدالة get_random_quote متغيرة فى quotes وهو قائمة list بالإقتباسات وجعلناها افتراضيا قيمة get_quotes () ،
تستخدم للحصول على اختيار عشوائى من quotes ويتم ذلك بإستدعاء الدالة choice التى تعيد لنا اقتباس
عشوائى متتابعة (قائمة) وهنا فى المثال هى quotes

3- السرفر quotd

```
from SocketServer import TCPServer, StreamRequestHandler, ThreadingMixIn
import threading
import quoter
from django.utils import daemonize

class MyServer(ThreadingMixIn,TCPServer):
    pass

class MyStreamRequestHandler(StreamRequestHandler):

    def handle(self):

        self.request.send(self.get_quote()+"\r\n")

    def get_quote(self):
        return quoter.get_random_quote()

def go(endpoint=(", 58000")):
    addr=endpoint
    tcpServer=MyServer(addr, MyStreamRequestHandler)
    tcpServer.allow_reuse_address=1
    print "Server started..."
    tcpServer.serve_forever() #inf. loop

if __name__=="__main__":
    try:
        daemonize.become_daemon()
        go()
    except KeyboardInterrupt:
        exit()
```

الكود بسيط جدا عند اتصال اى عميل يتم الحصول على اقتباس عشوائى ويتم ارساله
بس دا كود عادى جدا زى اللى سبق؟

بالفعل ولكن يختلف فى وجود الوحدة daemonize من django.utils
وتنفيذ الدالة become_daemon منها
بكل بساطة هذا هو الكود المطلوب منك للتحويل الى !daemon
تعالى نلقى نظرة على daemonize.become_daemon

```
import os
import sys

if os.name == 'posix':
    def become_daemon(our_home_dir='.', out_log='/dev/null',
                      err_log='/dev/null', umask=022):
        "Robustly turn into a UNIX daemon, running in our_home_dir."
        # First fork
        try:
            if os.fork() > 0:
                sys.exit(0) # kill off parent
        except OSError, e:
            sys.stderr.write("fork #1 failed: (%d) %s\n" % (e.errno, e.strerror))
            sys.exit(1)
        os.setsid()
        os.chdir(our_home_dir)
        os.umask(umask)

        # Second fork
        try:
            if os.fork() > 0:
                os._exit(0)
        except OSError, e:
            sys.stderr.write("fork #2 failed: (%d) %s\n" % (e.errno, e.strerror))
            os._exit(1)

        si = open('/dev/null', 'r')
        so = open(out_log, 'a+', 0)
        se = open(err_log, 'a+', 0)
        os.dup2(si.fileno(), sys.stdin.fileno())
        os.dup2(so.fileno(), sys.stdout.fileno())
        os.dup2(se.fileno(), sys.stderr.fileno())
        # Set custom file descriptors so that they get proper buffering.
        sys.stdout, sys.stderr = so, se
    else:
        def become_daemon(our_home_dir='.', out_log=None, err_log=None, umask=022):
            """
            If we're not running under a POSIX system, just simulate the daemon
            mode by doing redirections and directory changing.
            """
```

```
os.chdir(our_home_dir)
os.umask(umask)
sys.stdin.close()
sys.stdout.close()
sys.stderr.close()
if err_log:
    sys.stderr = open(err_log, 'a', 0)
else:
    sys.stderr = NullDevice()
if out_log:
    sys.stdout = open(out_log, 'a', 0)
else:
    sys.stdout = NullDevice()

class NullDevice:
    "A writeable object that writes to nowhere -- like /dev/null."
    def write(self, s):
        pass
```

إذا لم تفهم الكود السابق من خبرة سابقة في برمجة لينكس لاتقلق يكفيك استخدام الدالة مباشرة

استخدم telnet localhost 58000

```
striky@striky-desktop:~/workspace/pytut/src/nettut/quoter$ telnet localhost 58000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Life is just a chance to grow a soul. - A. Powell
Connection closed by foreign host.
```


Chapter 13 (Python on the WEB)

Grok

هو اطار عمل مبني على مكتبات Zope

التثبيت

بكل سهولة

```
easy_install grokproject
```

وبعد كذا

```
striky@striky-desktop:~/workspace/pytut/src/nettut$ mkdir groktut  
striky@striky-desktop:~/workspace/pytut/src/nettut$ cd groktut/
```

وقم بتنفيذ grokproject والحقه باسم المشروع

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut$ grokproject FirstProject
```

اسم المدير وليكن grok

```
Enter user (Name of an initial administrator user): grok
```

كلمة السر (لن تظهر تم استخدام getpass لإلغاء ظهورها) اكتبها grok

```
Enter passwd (Password for the initial administrator user):  
Downloading info about versions...  
Creating directory ./FirstProject  
Invoking zc.buildout...  
Develop: '/home/striky/workspace/pytut/src/nettut/groktut/FirstProject/'  
Installing eggbasket.  
Getting distribution for 'grok==0.14.1'.  
eggbasket: Distributions are not installed. A tarball will be downloaded.  
eggbasket: Distributions are not installed. A tarball will be downloaded.  
eggbasket: Downloading http://grok.zope.org/releaseinfo/grok-eggs-0.14.1.tgz ...  
eggbasket: Downloading http://grok.zope.org/releaseinfo/grok-eggs-0.14.1.tgz ...  
eggbasket: Finished downloading.  
eggbasket: Finished downloading.  
eggbasket: Extracting tarball contents...  
eggbasket: Extracting tarball contents...
```

```
eggbasket: Installing eggs to /home/striky/.buildout/eggs which will take a while...
eggbasket: Installing eggs to /home/striky/.buildout/eggs which will take a while...
Getting distribution for 'grok==0.14.1'.
Got grok 0.14.1.
```

الآن انت جاهز

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut$ cd FirstProject/
```

اعرض ال tree

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut/FirstProject$ tree
```

```
.
|-- bin
|   |-- buildout
|   |-- i18nexttract
|   |-- i18nmergeall
|   |-- i18nstats
|   |-- test
|   `-- zopectl
|-- bootstrap.py
|-- buildout.cfg
|-- develop-eggs
|   `-- FirstProject.egg-link
|-- parts
|   |-- app
|   |   |-- debugzope
|   |   |-- runzope
|   |   `-- site.zcml
|   |-- data
|   |-- i18n
|   |   `-- configure.zcml
|   |-- test
|   `-- zopectl
|       |-- zdaemon.conf
|       `-- zope.conf
|-- setup.py
|-- src
|   |-- FirstProject.egg-info
|   |   |-- PKG-INFO
|   |   |-- SOURCES.txt
|   |   |-- dependency_links.txt
|   |   |-- entry_points.txt
|   |   |-- not-zip-safe
|   |   |-- requires.txt
|   |   `-- top_level.txt
```

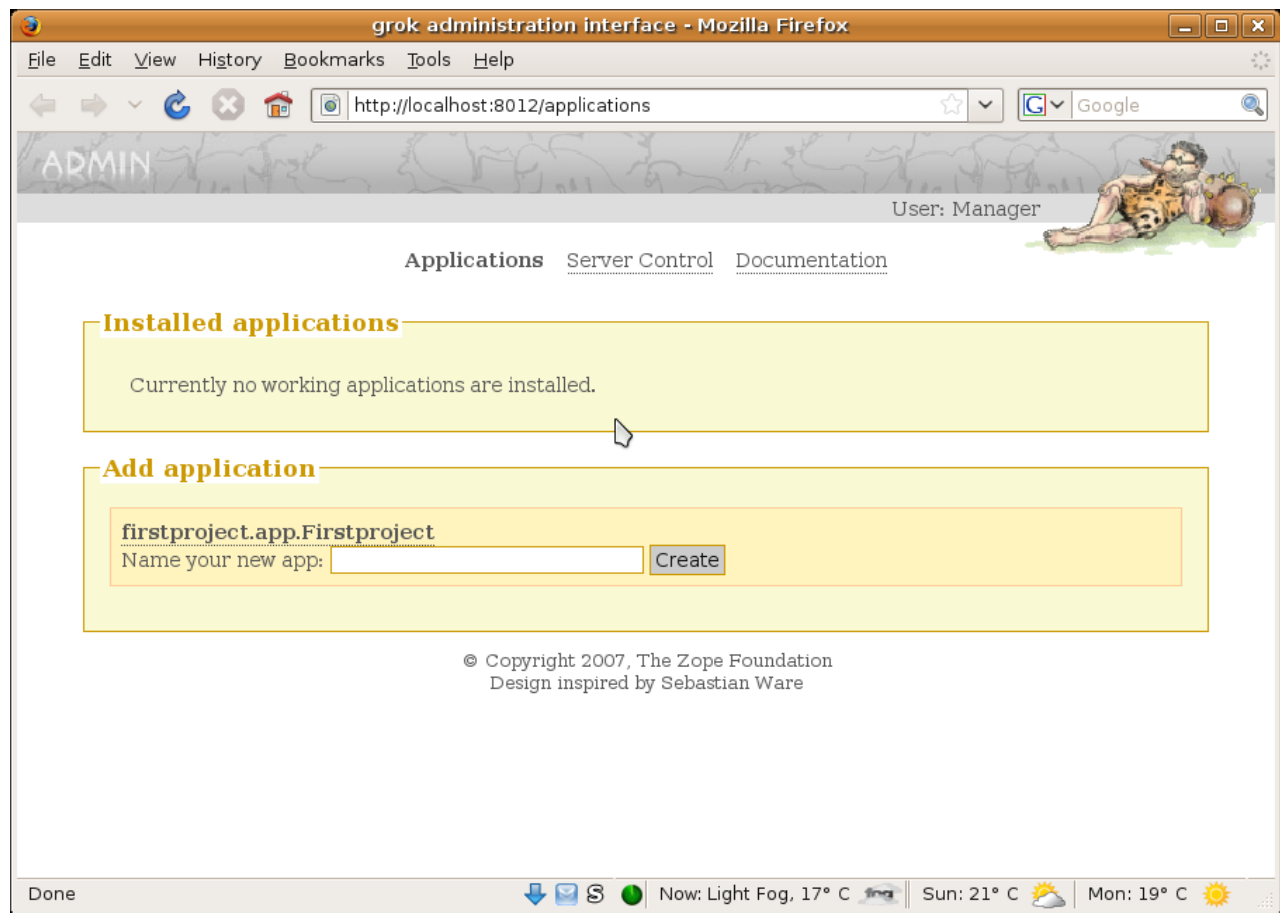
```

|-- firstproject
|   |-- __init__.py
|   |-- app.py
|   |-- app.txt
|   |-- app_templates
|   |   |-- index.pt
|   |   |-- configure.zcml
|   |   |-- ftesting.zcml
|   |   |-- static
|   |   |-- README.txt
|   |-- tests.py
|-- versions.cfg

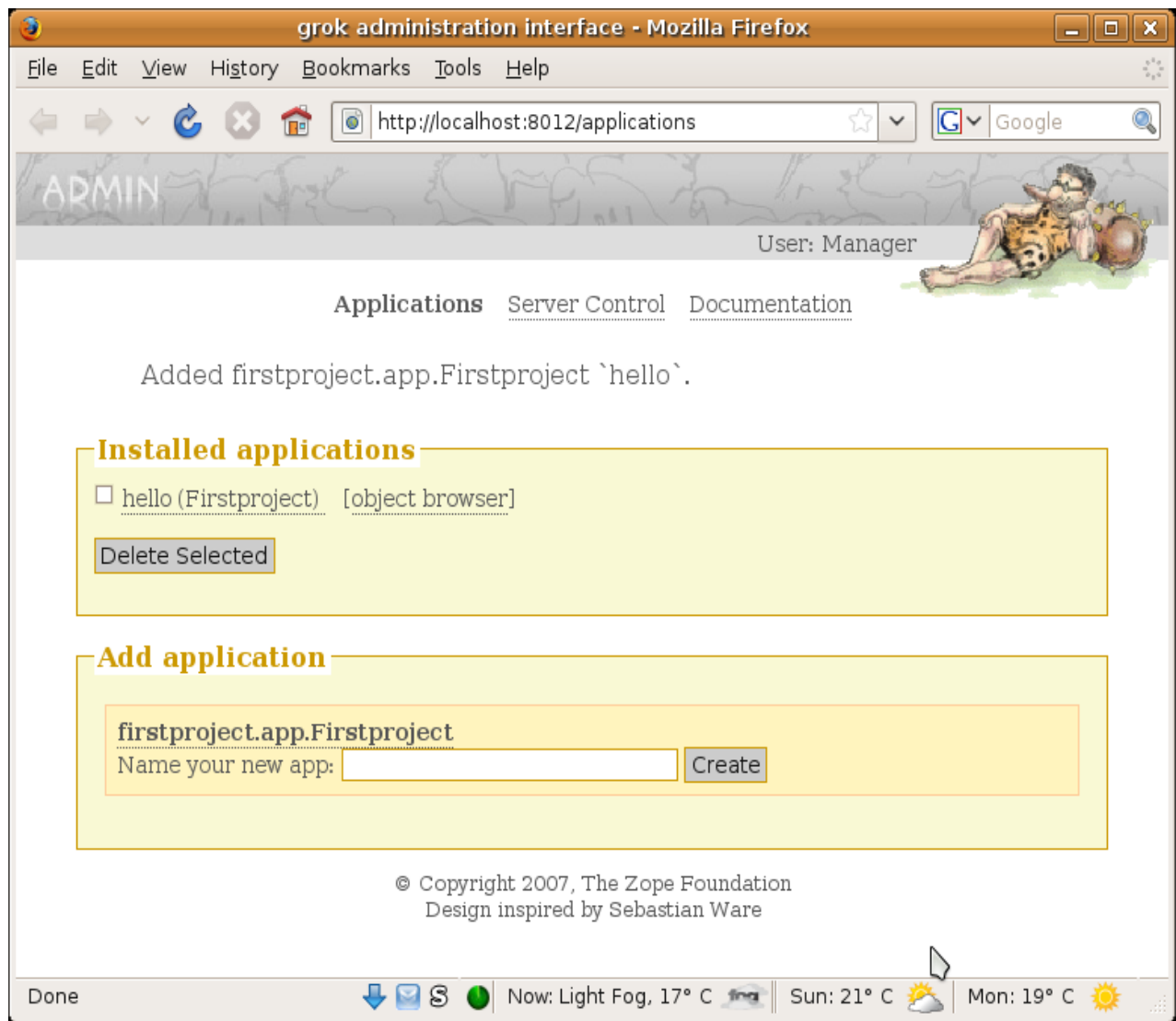
```

13 directories, 32 files

الهيكليّة واضحة وملفات تنفيذية و templates وملفات static واعدادات الخ
قم بتشغيل zope



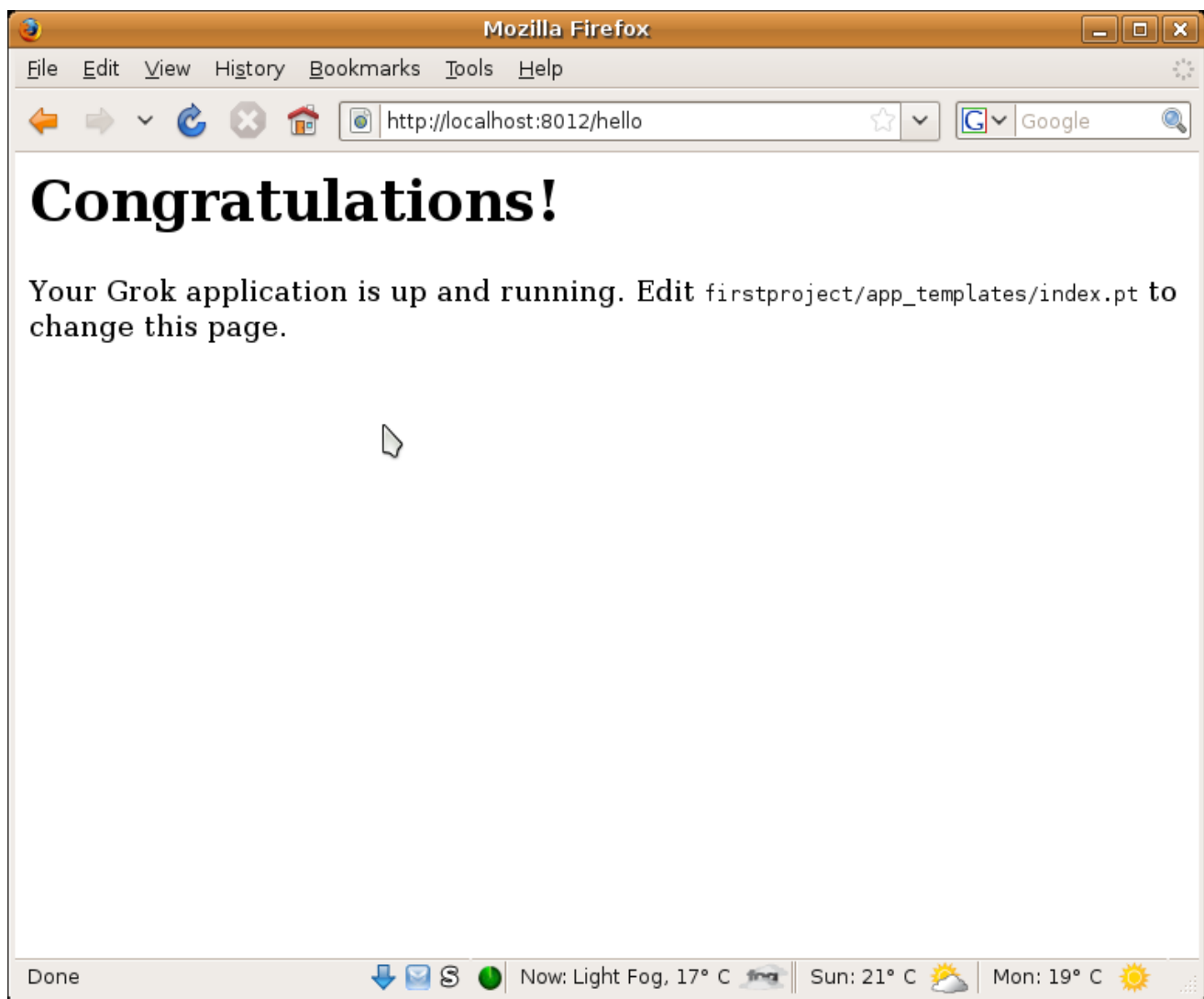
قم بتسمية التطبيق مثلا hello



اكتب فى المتصفح ذلك العنوان

localhost:8012/hello

ستجد امامك نافذة مشابهه لهذه تخبرك بمسار ال templates لتعديلها وهى فى المسار firstproject/app_templates/index.pt



الهيكليّة كالتالى

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut/FirstProject/src/firstproject$ tree
```

```
.
|-- __init__.py
|-- __init__.pyc
|-- app.py
|-- app.pyc
|-- app.txt
|-- app_templates
|   |-- index.pt
|-- configure.zcml
|-- ftesting.zcml
|-- static
|   |-- README.txt
|-- tests.py
```

2 directories, 10 files

قم بفتح ذلك الملف index.pt فى ال app_templates ستجده مشابه للتالى

```
<html>
<head>
</head>
<body>
  <h1>Congratulations!</h1>

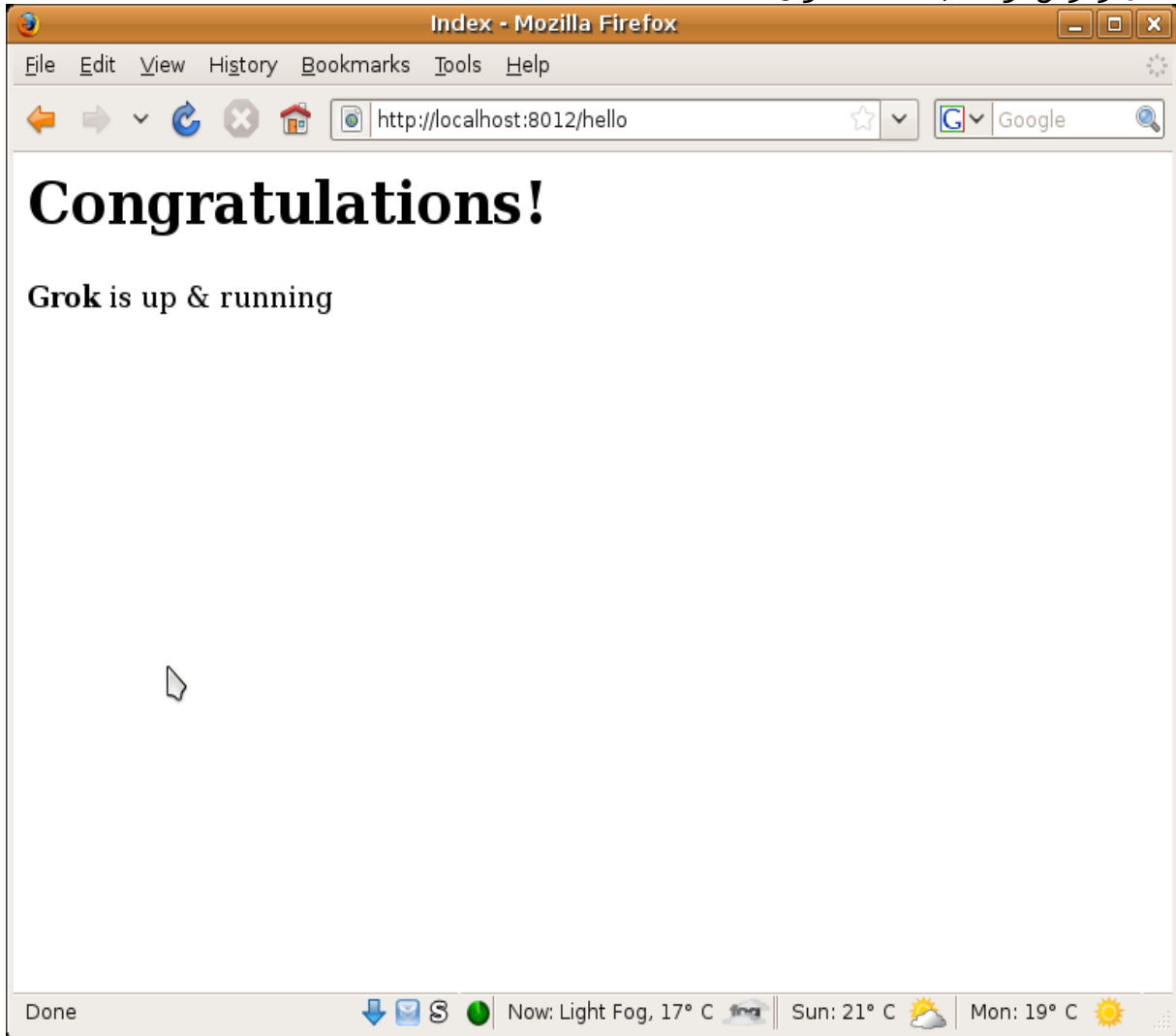
  <p>Your Grok application is up and running.
  Edit <code>firstproject/app_templates/index.pt</code> to change
  this page.</p>
</body>
</html>
```

قم بتعديله للتالى مثلاً

```
<html>
  <head>
    <title>Index</title>
  </head>
  <body>
    <h1>Congratulations!</h1>

    <p>
      <b>Grok</b> is up & running
    </p>
  </body>
</html>
```

وقم بعمل رفرش او اذهب لذلك العنوان localhost:8012/hello



الآن التطبيق لديه view واحدة وهى index التى يتم عند استدعائها عرض ال template فى الملف index.pt
افتح الملف app.py ستجده مشابه للتالى

```
import grok

class Firstproject(grok.Application, grok.Container):
    pass

class Index(grok.View):
    pass # see app_templates/index.pt
```

قم بإضافة 2 views وهما Hi, Bye

```
class Hi(grok.View):
    pass #renders app_templates/hi.pt

class Bye(grok.View):
    pass #renders app_templates/bye.pt
```

قم بإنشاء الملفات hi.pt و bye.pt ليتم عرضها عند استدعاء تلك ال views
الملف hi.pt

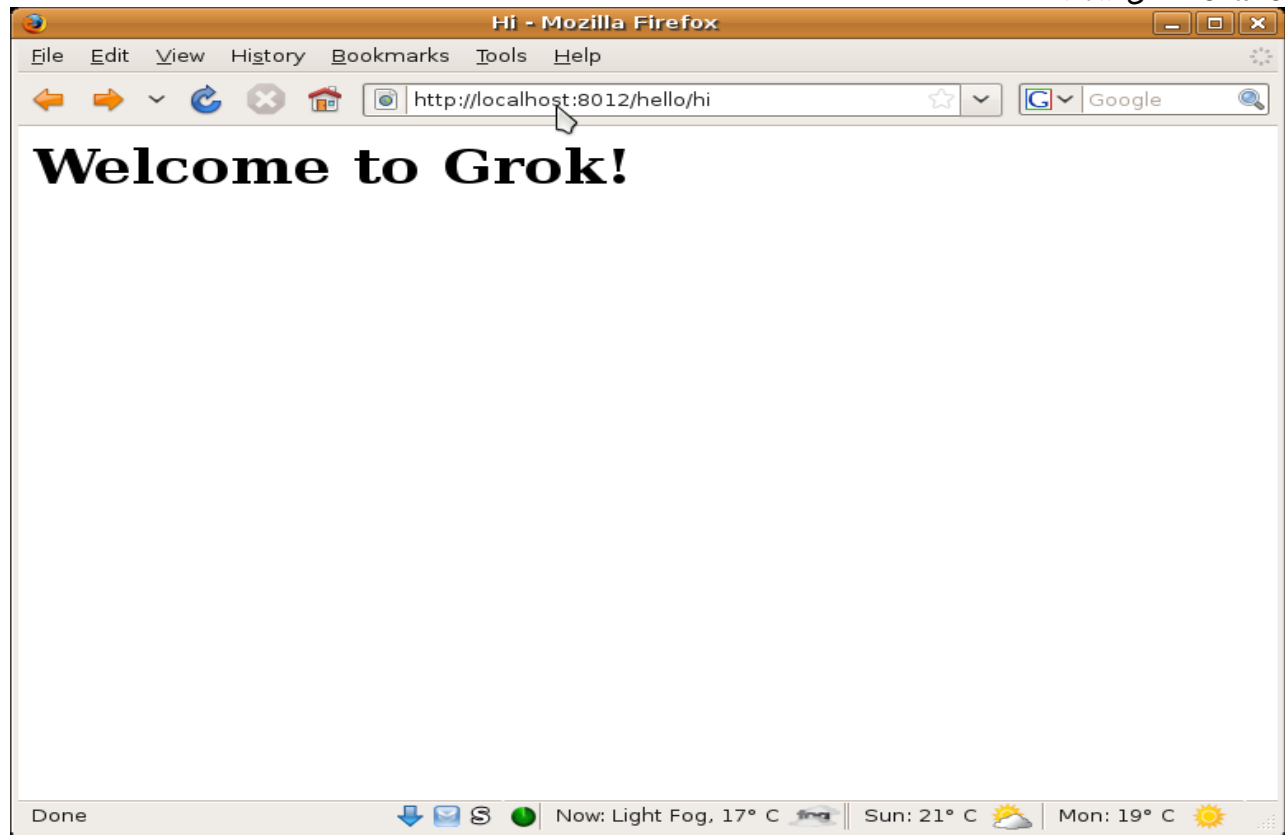
```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <h1>Welcome to Grok!</h1>
  </body>
</html>
```

الملف bye.pt

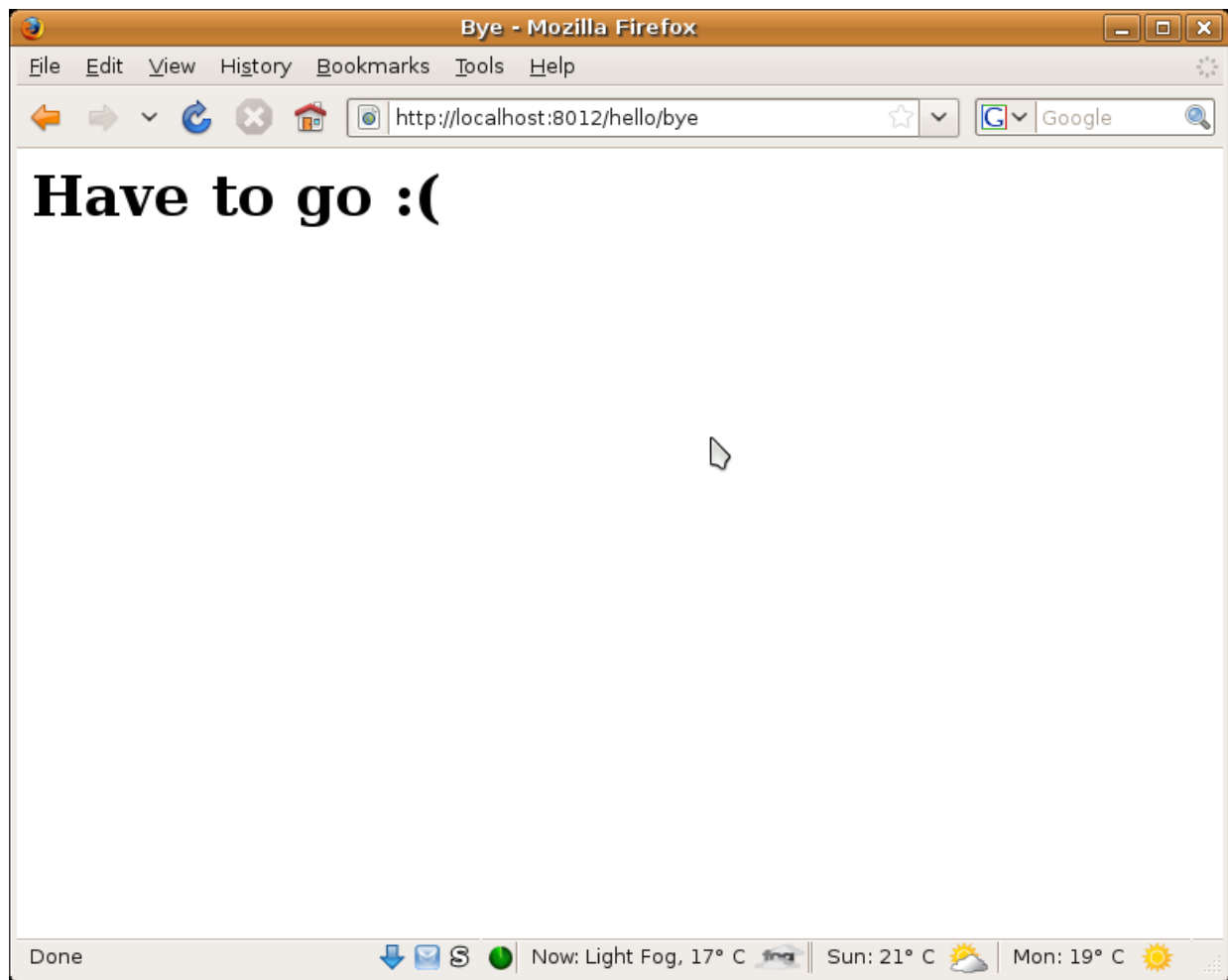
```
<html>
  <head>
    <title>Bye</title>
  </head>
  <body>
    <h1>Have to go :(</h1>
```

```
</body>  
</html>
```

عند استدعاء ال hi view



عند استدعاء ال bye.view



ملحوظة: قد تريد احيانا تغيير رقم البورت الافتراضى ل zoep بدل من 8080 (الذى قد تكون اسندته لخدمة اخرى او غيره)
كل ما عليك هو اعداد ملف buildout.cfg فى القسم [zopectl]
بإضافة ذلك السطر

```
address = localhost:8012
#controlling the listening port, re-run buildout script.
```

وقم بتشغيل ملف ال buildout

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut/FirstProject$ bin/buildout
Develop: '/home/striky/workspace/pytut/src/nettut/groktut/FirstProject/'
Uninstalling zopectl.
Updating eggbasket.
Updating app.
Updating data.
Installing zopectl.
Generated script '/home/striky/workspace/pytut/src/nettut/groktut/FirstProject/bin/zopectl'.
Updating i18n.
```

The recipe for i18n doesn't define an update method. Using its install method.
i18n: setting up i18n tools
Updating test.

الى اين الآن ؟
تستطيع الذهاب الى <http://grok.zope.org> وتقوم بقراءة الوثائق الخاصة وتنشئ بعض التطبيقات الحقيقية

Webpy

فريمورك بسيطة وممتازة وغير معقدة لاحتاج منك الكثير من المفاهيم ومبنية من اجل البساطة (ربما اذا اكملت الكتاب للآن تستطيع ان تفهم كودها المصدري)

للتستيب

easyinstall web.py

او قم بتحميلها من الموقع الرسمي <http://webpy.org>
وقم بتشغيل سكريبت ال setup

```
python setup.py install
```

ابسط تطبيق

```
import web

urls = (
    '(/.*)', 'index'
)

class index:
    def GET(self, name):
        return 'Hello, World!'

app = web.application(urls, globals())

if __name__ == "__main__":
    app.run()
```


1- استدعاء webpy

```
import web
```

2- انشاء ال urls على صورة tuple

```
'/(.*)', 'index'
```

على سبيل المثال هنا هيتم توجيه اى طلب الى ال view المسماة index اللتى قمنا بتعريفها كالتالى

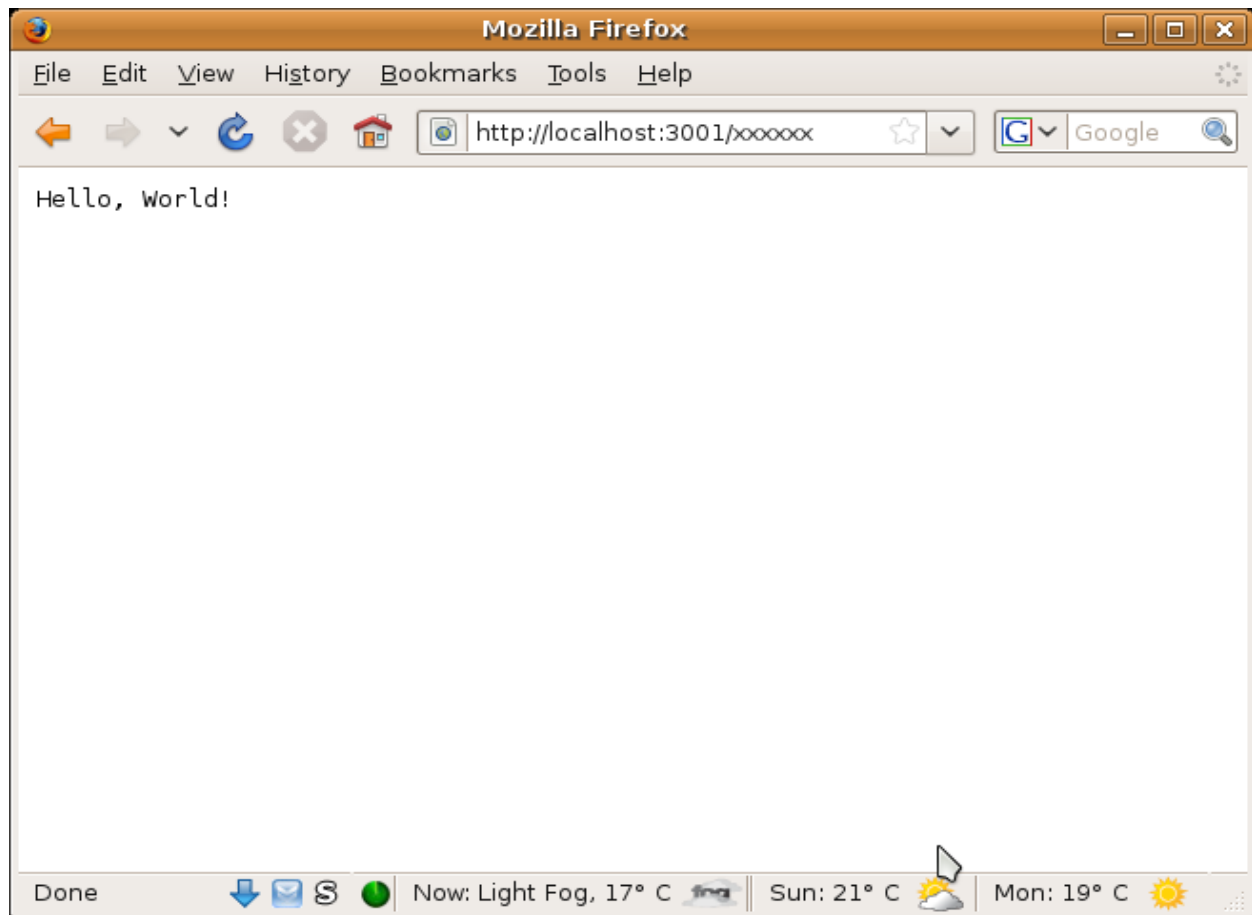
```
class index:
    def GET(self, name):
        return 'Hello, World!'
```

اخيرا انشاء عنصر التطبيق

```
app = web.application(urls, globals())
```

وتشغيله باستخدام الطريقة run

```
app.run()
```



لاحظ هنا تشغيلنا للسيرفر على البورت 3001 وذلك بإضافة ذلك المعامل الى السكريبت عند التشغيل والاسيكون البورت 8080

```
striky@striky-desktop:~/workspace/pytut/src/nettut/webpytut$ python hello.py 3001
```

تجد العديد من الأمثلة هنا
<http://webpy.org/src>

مثال على انشاء بلوج بسيط بإستخدام webpy
تجده هنا

<http://k4ml.com/wiki/python/webpy/simpleblog>

The Big Three

هناك 3 اطرارات تتصدر عالم بايثون فى الويب وهم Django و TurboGears و Pylons

Pylons

Pylons هى اطار حديث نسبيا ويجمع افضل ما فى العوالم python, ruby, perl واستفاد كثيرا من تجارب الإطارات السابقة

<http://pylonshq.com/>

Hello World: Pylons

```
striky@striky-desktop:~/workspace/pytut/src/nettut$ paster create -t pylons helloworld
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-i686.egg/dispatch/__init__.py:98:
Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/CherryPy-2.3.0-py2.5.egg/cherrypy/lib/profiler.py:54: UserWarning: Your
installation of Python doesn't have a profile module. If you're on Debian, you can apt-get python2.4-profiler
from non-free in a separate step. See http://www.cherrypy.org/wiki/ProfilingOnDebian for details.
  warnings.warn(msg)
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:239: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:263: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:281: Warning: 'as' will become a reserved keyword in Python 2.6
Selected and implied templates:
Pylons#pylons Pylons application template
```

Variables:

egg: helloworld
package: helloworld
project: helloworld

Enter template_engine (mako/genshi/jinja/etc: Template language) ['mako']:

Enter sqlalchemy (True/False: Include SQLAlchemy 0.4 configuration) [False]:

Enter google_app_engine (True/False: Setup default appropriate for Google App Engine) [False]:

Creating template pylons

Creating directory ./helloworld

Recurring into +package+

Creating ./helloworld/helloworld/

Copying templates/default_project/+package+/__init__.py_tmpl to ./helloworld/helloworld/__init__.py

Recurring into config

Creating ./helloworld/helloworld/config/

Copying templates/default_project/+package+/config/__init__.py_tmpl to

./helloworld/helloworld/config/__init__.py

Copying templates/default_project/+package+/config/deployment.ini_tmpl_tmpl to ./helloworld/helloworld/
config/deployment.ini_tmpl

Copying templates/default_project/+package+/config/environment.py_tmpl to

./helloworld/helloworld/config/environment.py

Copying templates/default_project/+package+/config/middleware.py_tmpl to

./helloworld/helloworld/config/middleware.py

Copying templates/default_project/+package+/config/routing.py_tmpl to

./helloworld/helloworld/config/routing.py

Recurring into controllers

Creating ./helloworld/helloworld/controllers/

Copying templates/default_project/+package+/controllers/__init__.py_tmpl to

./helloworld/helloworld/controllers/__init__.py

Copying templates/default_project/+package+/controllers/error.py_tmpl to

./helloworld/helloworld/controllers/error.py

Recurring into lib

Creating ./helloworld/helloworld/lib/

Copying templates/default_project/+package+/lib/__init__.py_tmpl to

./helloworld/helloworld/lib/__init__.py

Copying templates/default_project/+package+/lib/app_globals.py_tmpl to

./helloworld/helloworld/lib/app_globals.py

Copying templates/default_project/+package+/lib/base.py_tmpl to ./helloworld/helloworld/lib/base.py

Copying templates/default_project/+package+/lib/helpers.py_tmpl to ./helloworld/helloworld/lib/helpers.py

Recurring into model

Creating ./helloworld/helloworld/model/

Copying templates/default_project/+package+/model/__init__.py_tmpl to

./helloworld/helloworld/model/__init__.py

Recurring into public

Creating ./helloworld/helloworld/public/

Copying templates/default_project/+package+/public/bg.png to ./helloworld/helloworld/public/bg.png

Copying templates/default_project/+package+/public/index.html_tmpl to

./helloworld/helloworld/public/index.html

Copying templates/default_project/+package+/public/pylons-logo.gif to

./helloworld/helloworld/public/pylons-logo.gif

```

Recurring into templates
  Creating ./helloworld/helloworld/templates/
Recurring into tests
  Creating ./helloworld/helloworld/tests/
  Copying templates/default_project+/package+/tests/__init__.py_tmpl to
./helloworld/helloworld/tests/__init__.py
  Recurring into functional
  Creating ./helloworld/helloworld/tests/functional/
  Copying templates/default_project+/package+/tests/functional/__init__.py_tmpl to ./helloworld/helloworld/
tests/functional/__init__.py
  Copying templates/default_project+/package+/tests/test_models.py_tmpl to
./helloworld/helloworld/tests/test_models.py
  Copying templates/default_project+/package+/websetup.py_tmpl to ./helloworld/helloworld/websetup.py
  Copying templates/default_project/MANIFEST.in_tmpl to ./helloworld/MANIFEST.in
  Copying templates/default_project/README.txt_tmpl to ./helloworld/README.txt
  Copying templates/default_project/development.ini_tmpl to ./helloworld/development.ini
Recurring into docs
  Creating ./helloworld/docs/
  Copying templates/default_project/docs/index.txt_tmpl to ./helloworld/docs/index.txt
  Copying templates/default_project/ez_setup.py to ./helloworld/ez_setup.py
  Copying templates/default_project/setup.cfg_tmpl to ./helloworld/setup.cfg
  Copying templates/default_project/setup.py_tmpl to ./helloworld/setup.py
  Copying templates/default_project/test.ini_tmpl to ./helloworld/test.ini
Running /usr/bin/python setup.py egg_info

```

سيطلب منك 3 اجابات

1- ال template engine والإفتراضى هو mako

2- دعم sqlalchemy فى التطبيق والإفتراضى لا

3- تجهيز خيارات ل Google AppEngine

الهيكليّة

```

|-- MANIFEST.in
|-- README.txt
|-- development.ini
|-- docs
| `-- index.txt
|-- ez_setup.py
|-- helloworld
| |-- __init__.py
| |-- config
| | |-- __init__.py
| | |-- deployment.ini_tmpl
| | |-- environment.py
| | |-- middleware.py
| | `-- routing.py
| |-- controllers
| | |-- __init__.py
| | `-- error.py

```

```

| |-- lib
| | |-- __init__.py
| | |-- app_globals.py
| | |-- base.py
| | `-- helpers.py
| |-- model
| | `-- __init__.py
| |-- public
| | |-- bg.png
| | |-- index.html
| | `-- pylons-logo.gif
| |-- templates
| |-- tests
| | |-- __init__.py
| | |-- functional
| | | `-- __init__.py
| | `-- test_models.py
| `-- websetup.py
|-- helloworld.egg-info
|-- PKG-INFO
|-- SOURCES.txt
|-- dependency_links.txt
|-- entry_points.txt
|-- not-zip-safe
|-- paster_plugins.txt
|-- requires.txt
| `-- top_level.txt
|-- setup.cfg
|-- setup.py
`-- test.ini

```

ننشئ controller (وهو ما يحوى ال actions مثل ال views اللتى فى grok او webpy)

```

striky@striky-desktop:~/workspace/pytut/src/nettut/helloworld$ paster controller hello
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-i686.egg/dispatch/__init__.py:98:
Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/CherryPy-2.3.0-py2.5.egg/cherrypy/lib/profiler.py:54: UserWarning: Your
installation of Python doesn't have a profile module. If you're on Debian, you can apt-get python2.4-profiler
from non-free in a separate step. See http://www.cherrypy.org/wiki/ProfilingOnDebian for details.
  warnings.warn(msg)
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:239: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:263: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:281: Warning: 'as' will become a reserved keyword in Python 2.6
Creating /home/striky/workspace/pytut/src/nettut/helloworld/helloworld/controllers/hello.py
Creating /home/striky/workspace/pytut/src/nettut/helloworld/helloworld/tests/functional/test_hello.py
striky@striky-desktop:~/workspace/pytut/src/nettut/helloworld$

```

تمام افتح الملف فى المسار helloworld/controllers/hello ستجده مشابه للتالى

```
import logging

from pylons import request, response, session, tmpl_context as c
from pylons.controllers.util import abort, redirect_to

from helloworld.lib.base import BaseController, render
#from helloworld import model

log = logging.getLogger(__name__)

class HelloController(BaseController):

    def index(self):
        # Return a rendered template
        # return render('/template.mako')
        # or, Return a response
        return 'Hello World'
```

يتم التعامل مع العناوين كالتالى

mysite.com/controller/view

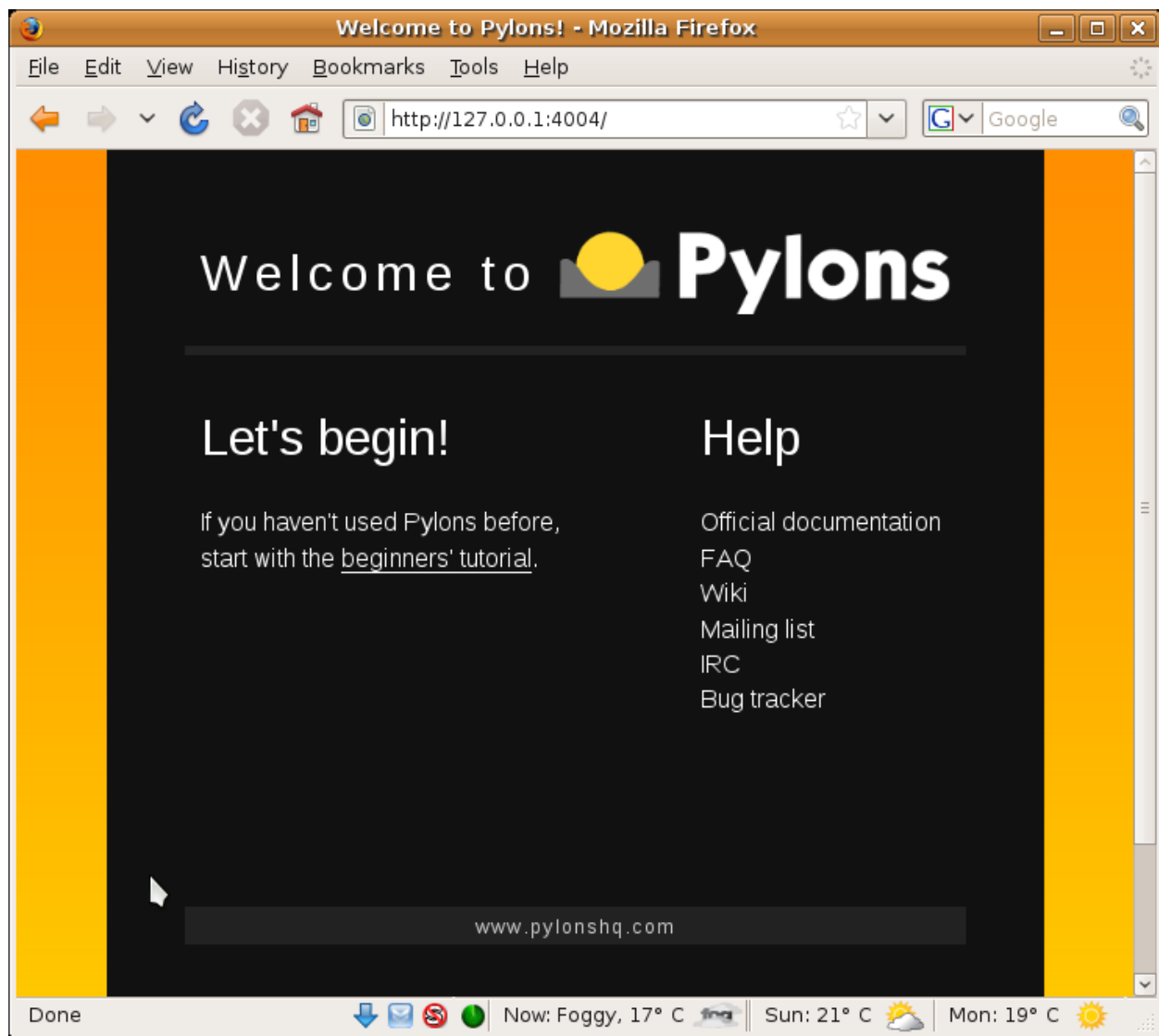
مثلا عن ارسال

mysite.com/hello/index

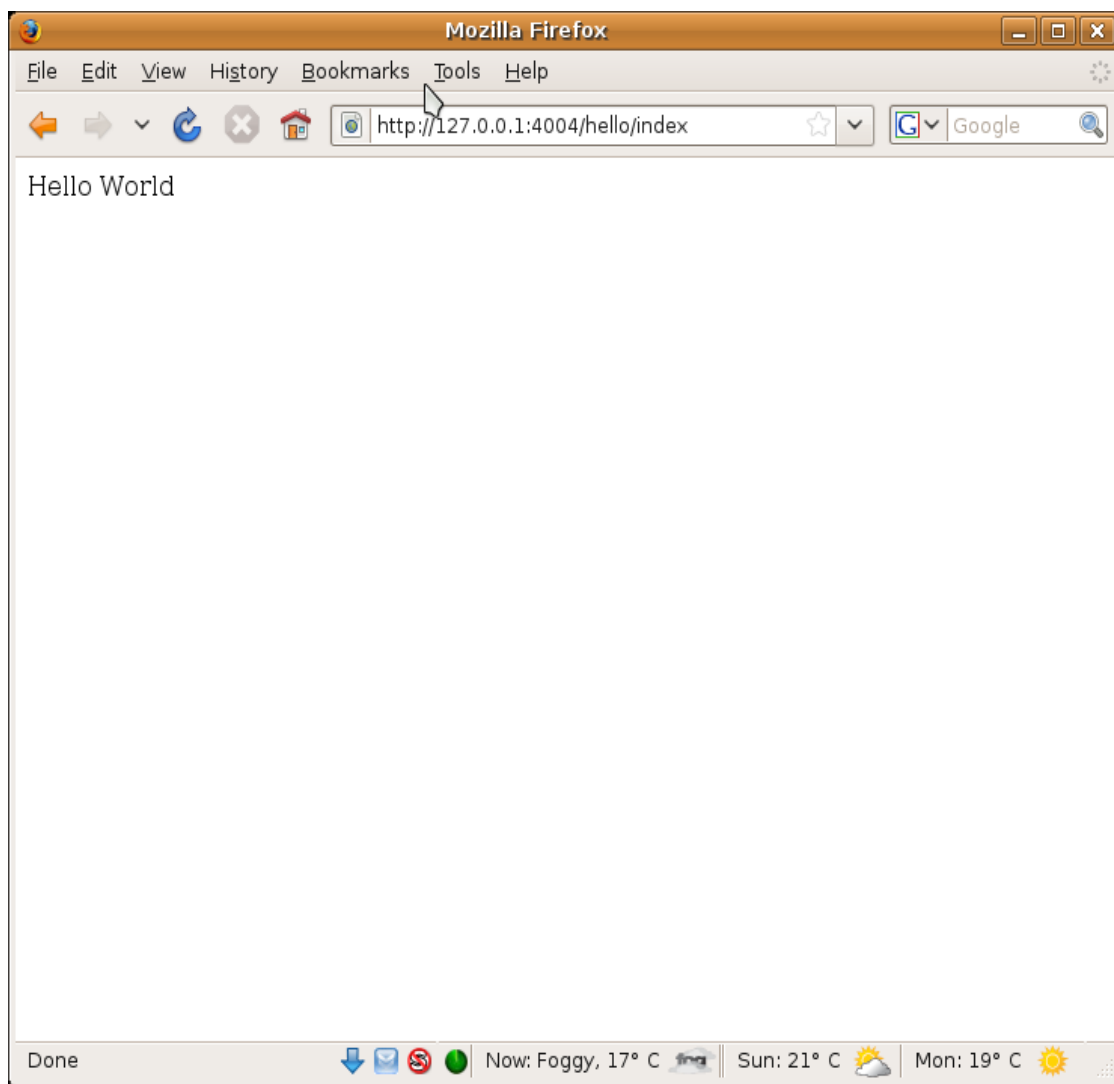
فيتم استدعاء الكنترولر (المقسم) hello ومنه يتم اختيار ال action المناسبة
قم بتشغيل السرفر

```
striky@striky-desktop:~/workspace/pytut/src/nettut/helloworld$ paster serve --reload development.ini
```

ملف ال development.ini يحوى معلومات عن البيئة كالهوست والبورت ومتغيرات التطبيق الخ الخ



الآن اكتب hello/index فى العنوان



TurboGears

تربوجيرز هي اطار عمل رائع يقوم على ربط التقنيات الحالية فى عالم بايثون للخروج بأفضل نتيجة فلتتعامل مع قواعد البيانات يتم استخدام SQLAlchemy او SQLAlchemy و للتعامل مع ال templates يتم استخدام kid وهكذا

لتستيب تربوجيرز قم اولا بتحميل الحزمة من الموقع [/http://turbogears.org](http://turbogears.org)
راجع صفحة التستيب <http://docs.turbogears.org/1.0/Install>

قم بتحميل سكربت tgsetup.py وتشغيله
<http://www.turbogears.org/download/tgsetup.py>

نفذ سكربت التستيب

```
striky@striky-desktop:~/Desktop$ sudo python tgsetup.py

[sudo] password for striky:
Sorry, try again.
[sudo] password for striky:
TurboGears Installer
Beginning setuptools/EasyInstall installation and TurboGears download

Searching for TurboGears==1.0.8
Reading http://www.turbogears.org/download/
Reading http://pypi.python.org/simple/TurboGears/
Reading http://www.turbogears.org
Reading http://www.turbogears.org/
Reading http://www.turbogears.org/download/filelist.html
Best match: TurboGears 1.0.8
Downloading http://files.turbogears.org/eggs/TurboGears-1.0.8-py2.5.egg
Processing TurboGears-1.0.8-py2.5.egg
removing '/usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg' (and everything under it)
creating /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg
Extracting TurboGears-1.0.8-py2.5.egg to /usr/lib/python2.5/site-packages
Removing TurboGears 1.0.7 from easy-install.pth file
Adding TurboGears 1.0.8 to easy-install.pth file
Installing tg-admin script to /usr/bin

Installed /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg
Reading http://files.turbogears.org/eggs/
Processing dependencies for TurboGears==1.0.8
Searching for Extremes>=1.1
Reading http://pypi.python.org/simple/Extremes/
Best match: Extremes 1.1
Downloading http://pypi.python.org/packages/2.5/E/Extremes/Extremes-1.1-py2.5.egg#md5=4015e2546295858558cca16faca5f34f
Processing Extremes-1.1-py2.5.egg
```

Moving Extremes-1.1-py2.5.egg to /usr/lib/python2.5/site-packages
Adding Extremes 1.1 to easy-install.pth file

Installed /usr/lib/python2.5/site-packages/Extremes-1.1-py2.5.egg
Searching for PyProtocols>=1.0a0dev-r2302
Reading <http://pypi.python.org/simple/PyProtocols/>
Reading <http://peak.telecommunity.com/PyProtocols.html>
Reading <http://peak.telecommunity.com/dist/>
Best match: PyProtocols 1.0a0dev-r2302
Downloading http://files.turbogears.org/eggs/PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg
Processing PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg
Moving PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg to /usr/lib/python2.5/site-packages
Adding PyProtocols 1.0a0dev-r2302 to easy-install.pth file

Installed /usr/lib/python2.5/site-packages/PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg
Finished processing dependencies for TurboGears==1.0.8

Hello World: TG

لإنشاء تطبيق سريع كل ما عليك هو تنفيذ tg-admin quickstart

```
striky@striky-desktop:~/workspace/pytut/src/nettut$ tg-admin quickstart
/usr/lib/python2.5/site-packages/CherryPy-2.3.0-py2.5.egg/cherrypy/lib/profiler.py:54: UserWarning: Your
installation of Python doesn't have a profile module. If you're on Debian, you can apt-get python2.4-profiler
from non-free in a separate step. See http://www.cherrypy.org/wiki/ProfilingOnDebian for details.
  warnings.warn(msg)
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-i686.egg/dispatch/__init__.py:98:
Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:239: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:263: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:281: Warning: 'as' will become a reserved keyword in Python 2.6
Enter project name: hello
Enter package name [hello]: hello
Do you need Identity (usernames/passwords) in this project? [no]
Selected and implied templates:
  TurboGears#tgbase      tg base template
  TurboGears#turbogears  web framework

Variables:
egg:      hello
elixir:   False
identity: none
package:  hello
```

```
project:      hello
sqlalchemy:   False
sqlobject:    True
sqlobjectversion: SQLAlchemy>=0.10.1
Creating template tgbase
Creating directory ./hello
  Recursing into +ename+.egg-info
    Creating ./hello/hello.egg-info/
    Copying PKG-INFO to ./hello/hello.egg-info/PKG-INFO
    Copying paster_plugins.txt to ./hello/hello.egg-info/paster_plugins.txt
    Copying sqlobject.txt_tmpl to ./hello/hello.egg-info/sqlobject.txt
  Recursing into +package+
    Creating ./hello/hello/
    Copying __init__.py to ./hello/hello/__init__.py
    Copying release.py_tmpl to ./hello/hello/release.py
  Recursing into static
    Creating ./hello/hello/static/
    Recursing into css
      Creating ./hello/hello/static/css/
Skipping file /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg/turbogears/qstemplates/qsbases/
+package+/static/css/empty_tmpl
  Recursing into images
    Creating ./hello/hello/static/images/
    Copying favicon.ico to ./hello/hello/static/images/favicon.ico
    Copying tg_under_the_hood.png to ./hello/hello/static/images/tg_under_the_hood.png
    Copying under_the_hood_blue.png to ./hello/hello/static/images/under_the_hood_blue.png
  Recursing into javascript
    Creating ./hello/hello/static/javascript/
Skipping file /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg/turbogears/qstemplates/qsbases/
+package+/static/javascript/empty_tmpl
  Recursing into templates
    Creating ./hello/hello/templates/
    Copying __init__.py to ./hello/hello/templates/__init__.py
Creating template turbogears
  Recursing into +package+
    Copying commands.py_tmpl to ./hello/hello/commands.py
  Recursing into config
    Creating ./hello/hello/config/
    Copying __init__.py to ./hello/hello/config/__init__.py
    Copying app.cfg_tmpl to ./hello/hello/config/app.cfg
    Copying log.cfg_tmpl to ./hello/hello/config/log.cfg
  Copying controllers.py_tmpl to ./hello/hello/controllers.py
  Copying json.py_tmpl to ./hello/hello/json.py
  Copying model.py_tmpl to ./hello/hello/model.py
  Recursing into static
    Recursing into css
      Copying style.css to ./hello/hello/static/css/style.css
    Recursing into images
      Copying header_inner.png to ./hello/hello/static/images/header_inner.png
      Copying info.png to ./hello/hello/static/images/info.png
```

```
    Copying ok.png to ./hello/hello/static/images/ok.png
    Recursing into templates
    Copying login.kid to ./hello/hello/templates/login.kid
    Copying master.kid to ./hello/hello/templates/master.kid
    Copying welcome.kid to ./hello/hello/templates/welcome.kid
    Recursing into tests
    Creating ./hello/hello/tests/
    Copying __init__.py to ./hello/hello/tests/__init__.py
    Copying test_controllers.py_tmpl to ./hello/hello/tests/test_controllers.py
    Copying test_model.py_tmpl to ./hello/hello/tests/test_model.py
    Copying README.txt_tmpl to ./hello/README.txt
    Copying dev.cfg_tmpl to ./hello/dev.cfg
    Copying sample-prod.cfg_tmpl to ./hello/sample-prod.cfg
    Copying setup.py_tmpl to ./hello/setup.py
    Copying start-+package+.py_tmpl to ./hello/start-hello.py
    Copying test.cfg_tmpl to ./hello/test.cfg
    Running /usr/bin/python setup.py egg_info
    Manually creating paster_plugins.txt (deprecated! pass a paster_plugins keyword to setup() instead)
    Adding TurboGears to paster_plugins.txt
    running egg_info
    paster_plugins not set in setup(), but hello.egg-info/paster_plugins.txt exists
    writing requirements to hello.egg-info/requires.txt
    writing hello.egg-info/PKG-INFO
    writing top-level names to hello.egg-info/top_level.txt
    writing dependency_links to hello.egg-info/dependency_links.txt
    writing entry points to hello.egg-info/entry_points.txt
    reading manifest file 'hello.egg-info/SOURCES.txt'
    writing manifest file 'hello.egg-info/SOURCES.txt'
striky@striky-desktop:~/workspace/pytut/src/nettut$
```

لإختيار البورت الذى تريد الإنصات عليه قم بتحرير ملف dev.cfg حيث يشمل اعدادات التطبيق

```
server.socket_port=40003
```

مسار قاعدة بيانات sqlite

```
sqlobject.dburi="sqlite://%(current_dir_uri)s/devdata.sqlite"
```

إذا قمت بعمل أى جداول .. الخ
قم بتشغيل التطبيق بإستخدام ال start script وهنا ستجد اسمه start-hello.py
افتح متصفحك وحدد العنوان localhost:40003 او غيره اذا قمت بإعداد البورت ستشاهد صفحة مثل هذه



بنفس فلسفة pylons ستجد المتحكمات (controllers) فى ملف controllers.py

```
#controllers.py
```

```

import turbogears as tg
from turbogears import controllers, expose, flash
# from hello import model
# import logging
# log = logging.getLogger("hello.controllers")

class Root(controllers.RootController):
    @expose(template="hello.templates.welcome")
    def index(self):
        import time
        # log.debug("Happy TurboGears Controller Responding For Duty")
        flash("Your application is now running")
        return dict(now=time.ctime())

```

وهذا هو ال controller الرئيسى وتم كشفه لل template فى المسار hello/templates/welcome.kid
ملحوظة لكشف أى action ل template ما استخدم @expose وظيف ليها معامل template وقيمتها = مسار ال template المطلوب

ملف welcome.kid

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:py="http://purl.org/kid/ns#"
py:extends="master.kid">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" py:replace=""/>
<title>Welcome to TurboGears</title>
</head>
<body>

<div id="sidebar">
<h2>Learn more</h2>
Learn more about TurboGears and take part in its
development
<ul class="links">
<li><a href="http://www.turbogears.org">Official website</a></li>
<li><a href="http://docs.turbogears.org">Documentation</a></li>
<li><a href="http://trac.turbogears.org/turbogears/">Trac
(bugs/suggestions)</a></li>
<li><a href="http://groups.google.com/group/turbogears"> Mailing list</a> </li>
</ul>
<span py:replace="now">now</span>
</div>
<div id="getting_started">
<ol id="getting_started_steps">
<li class="getting_started">
<h3>Model</h3>
<p><a href="http://docs.turbogears.org/1.0/GettingStarted/DefineDatabase">Design models</a> in the
<span class="code">model.py</span></p></li>

```

```

    Edit <dev.cfg> to <a href="http://docs.turbogears.org/1.0/GettingStarted/UseDatabase">use a different backend</a>, or start with a pre-configured SQLite database. <br>
    Use script <tg-admin sql create> to create the database tables.</p>
</li>
<li class="getting_started">
<h3>View</h3>
<p> Edit <a href="http://docs.turbogears.org/1.0/GettingStarted/Kid">html-like templates</a> in the <templates> folder;<br>
    Put all <a href="http://docs.turbogears.org/1.0/StaticFiles">static contents</a> in the <static> folder. </p>
</li>
<li class="getting_started">
<h3>Controller</h3>
<p> Edit <controllers.py> and <a href="http://docs.turbogears.org/1.0/GettingStarted/CherryPy">build your website structure</a> with the simplicity of Python objects. <br>
    TurboGears will automatically reload itself when you modify your project. </p>
</li>
</ol>
<div class="notice"> If you create something cool, please <a href="http://groups.google.com/group/turbogears">let people know</a>, and consider contributing something back to the <a href="http://groups.google.com/group/turbogears">community</a>.</div>
<!-- End of getting_started -->
</body>
</html>

```

تعالى نجرب اضافة action جديد وليكن greet
بكل بساطة ضيفه فى ال controllers.py كطريقة لل RootController

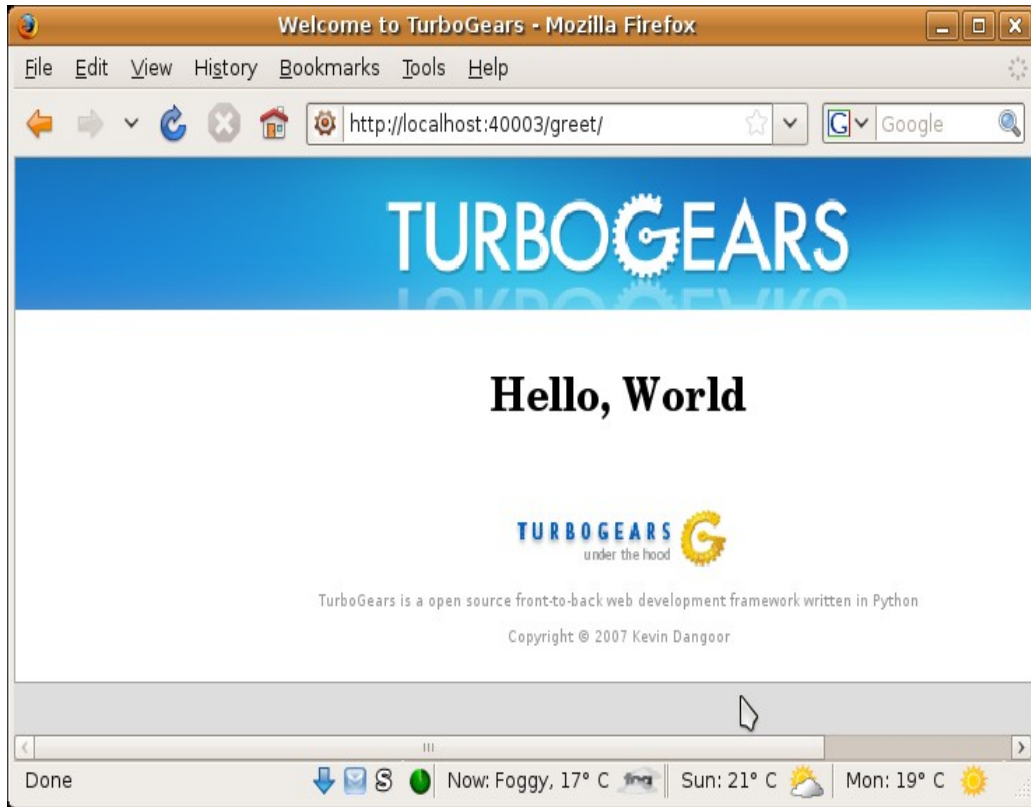
```

@expose(template='hello.templates.greet')
def greet(self, who='World'):
    return dict(g='Hello, '+who)

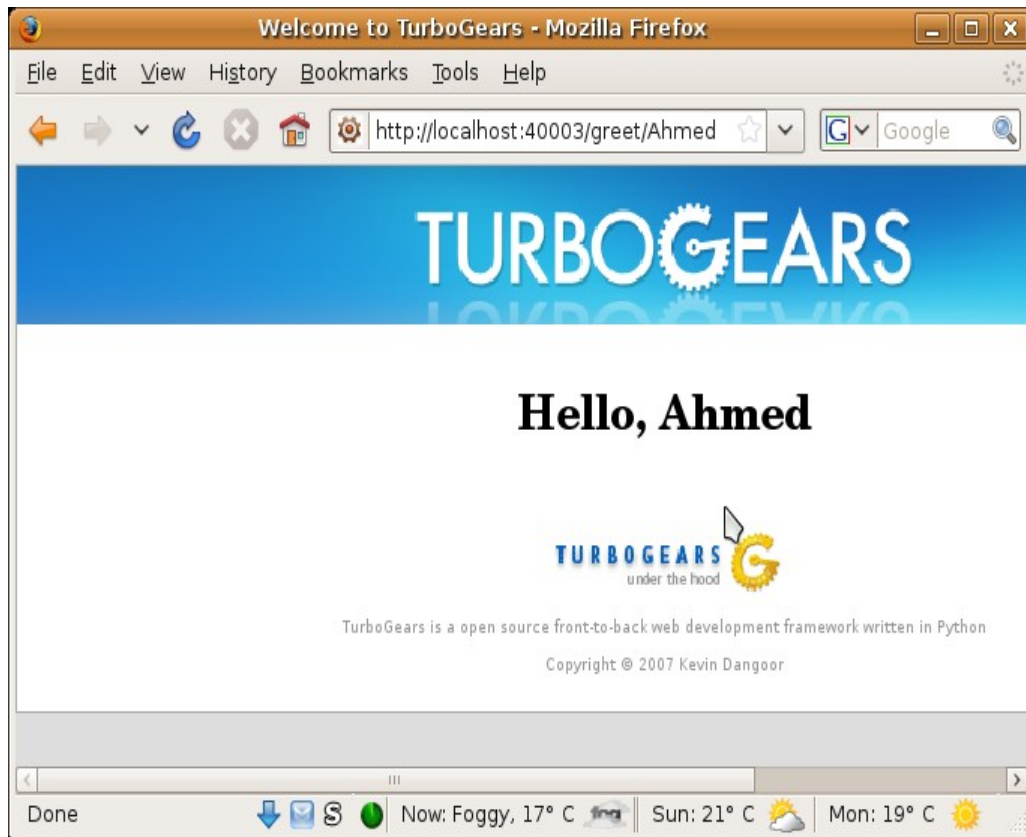
```

ماهذا ؟ ايه معنى who ؟
بكل بساطة هى معامل يتم استدعائه بعد اسم ال action فى العنوان مثلا
localhost:40003/hello/Ahmed
فتصبح قيمة who هى Ahmed وفى حال عدم تحديدها تكون قيمتها World
نعيد dict من تلك ال action يشمل المتغيرات التى ستصبح مكشوفة فى ال template ليتم استخدامها فيه

عند الإستدعاء بدون معاملات



باستخدام معاملات



يوجد اطرارات عمل اكثر من رائعة مثل [Django](#) التى لم نتعرض لها فى الكتاب وايضا [Web2Py](#) وغيرها
تجد قائمة بأهم اطر العمل هنا
<http://wiki.python.org/moin/WebFrameworks>

Chapter 14 (Extending Python)

Pure C

ماهى ال Extensions ؟
هى امتدادات لل Python مكتوبة بال C او C-Like مثل ال ++C + على سبيل المثال

سهل كتابة extensions لل Python ولكن ماله هدف ؟
الهدف إنك تضيف built-in modules لل Python مكتوبة بال C بهدف السرعة مثلا او إضافة built-in types او عمل encapsulation لل C lib functions او System Calls او حتى إخفاء ال source code الخاص بيك :

المتطلبات: خبرة جيدة بال C و Python
اول شئ بكل تأكيد هو إننا هنستخدم Python API/C وذلك بضم python.h لل project

ملحوظة: قم بضم python.h قبل اى header اخر. جميل نبدأ ب Hello, World (:
1- انشئ ملف helloMod.c
2- اكتب التالى

```
#include <Python.h>

static PyObject* hola(PyObject* self, PyObject* args)
{

    if (!PyArg_ParseTuple(args, "", NULL))
        return NULL;

    printf("Hola!");

    Py_RETURN_NONE;
}

static PyMethodDef HolaMethods[] =
{
    {"hola", hola, METH_VARARGS, "prints Hola\n"},
    {NULL, NULL, 0, NULL}
};

PyMODINIT_FUNC

inithola(void)
{
    (void) Py_InitModule("hola", HolaMethods);
}
```

نبدأ ب module بسيطة وهى hola هنتشرح سطر سطر
اولا نعمل include ل python api header كالتالى

```
#include <Python.h>
```

اى Object فى Python تقدر تعرفه ك Pointer ل ب PyObject
احنا الاول نريد ان نعرف function بسيطة تطبع كلمة Hola!

```
static PyObject* hola(PyObject* self, PyObject* args)
```

لاحظ self هو Pointer فى حال لو ال PyObject Class ودى هتكون ال method تبعه. لكن لو Function بيقة self
هيكون NULL. نريد ان يتم استخدام ال Function كالتالى

```
>>> hola.hola()  
Hola!
```

لاحظ شئ إن ال Function مش هتأخذ اى argument و مش ليها return او ال Return ب NONE
فالأول نختبر هل فى arguments تم تمريرها لل Function او لا

```
if (!PyArg_ParseTuple(args, "", NULL))  
    return NULL;
```

PyArg_ParseTuple هى Function بتستخدم فى عمل Parse او تحليل لل Arguments وفيها بيتم تحويل ال Python
Values ل C Values زى ما هنتشوف فى مثال قادم.
args هى ال arguments اللى هتتمرر لل Function
هى بتعبر عن ال Data Type الخاص بال Argument مثلا s او i وهكذا

s: String
i: Integer
.. etc

NULL هنا بيعبر عن ال متغيرات اللى هتأخذ القيم اللى تم تمريرها لل args -هنتطلع عليها اكثر فى مثال قادم-

```
return NULL;
```

للخروج مباشرة من تنفيذ ال Function
بعد كذا نيجى لل ال Function هتعمله وهو طباعة كلمة Hola باستخدام printf

```
printf("Hola!");
```

واخيرا زى ماقلنا اننا مش نريد ان اى return من ال Function فهنعمل Py_RETURN_NONE

```
Py_RETURN_NONE;
```

هنحتاج نعرف ال methodTable وهو عبارة عن array بتشمل معلومات عن ال Function زى ال name, address وال Documentation الخاصة بيها وهكذا

```
static PyMethodDef HolaMethods[] =
{
    {"hola", hola, METH_VARARGS, "prints Hola"},
    {NULL, NULL, 0, NULL}
};
```

اول field هو ال name الخاص بال function
التانى هو ال function نفسها
الثالث بيعبر عن ان ال Arguments اللتى هتتمرر هى Python-Level Arguments وغالبا بنستخدم METH_VARARGS
الرابع هو الوصف الخاص بال function ونحجز المكان التانى فى ال Array ب

```
{NULL, NULL, 0, NULL}
```

نعمل Initialize لل Module بتاعتنا كالتالى

```
PyMODINIT_FUNC
inithola(void)
{
    (void) Py_InitModule("hola", HolaMethods);
}
```

PyMODINIT_FUNC هى إختصار ل Python Module Initializer Function و فيها بيتم تجهيز ال Module بإستدعاء Py_InitModule وهى Function وهى اللتى بتقوم بالتجهيز بالفعل وبتاخذ 2Arguments
1- اسم ال Module
2- ال Methods Table

جميل جدا.. كدا كتبنا اول Module خاصة بيها!
هنحتاج نضم ال Extension لل Python ولكن إزاي ؟

بكل بساطة افتح ال Editor المفضل عندك وهنعمل setup script بال Python بإستخدام Distutils

```
from distutils.core import setup, Extension

modExt = Extension('hola', sources = ['hola.c'])

setup (name = 'HolaPackage',
       version = '1.0',
       description = 'Simple demo',
       ext_modules = [modExt])
```

الخطوات سهلة وسلسلة كالتالي:
 1- إستدعينا setup, Extension من Disutils.core
 2- عملنا Extension Object كالتالي

```
modExt = Extension('hola', sources = ['hola.c'])
```

وفيه بنحدد ال source واسم ال extension
 3- تجهيز ال setup script بأننا نسجل فيه إسم ال Package و الإصدار والوصف و ال extensions كالتالي

```
setup (name = 'HolaPackage',
       version = '1.0',
       description = 'Simple demo',
       ext_modules = [modExt])
```

كل ما عليك هو

Python setup.py build

وبعد كذا تعمل Install لل Package كالتالي

Python setup.py install

نحرب ال Hola Module كالتالي
 1- اعمل اى Test Script وليكن HolaTest.py
 2- اعمل import ل hola Module كالتالي

```
import hola
```

3- استدعى ال hola function كالتالي

```
hola.hola()
#output:
Hola!
```

4- لنوضح ال return الخاص بال Function اكتب

```
print hola.hola()
#Output:
Hola!
None
```

بعد ماطلعنا على الأساسيات نجرب نكتب module فيها function بتقبل argument ك name و age وتطبعهم و Function اخرى لحساب القيمة المطلقة لرقم وواحدة تقسم عددين وواحدة تعمل ب return ب Tuple, Dictionary الفكرة باختصار:

- 1- نعرف ال Functions
- 2- نضمهم لل Methods Table
- 3- نعمل Initialize لل Module
- 4- نجهز ال setup script
- 5- نعمل Build و Install لل Module
- 6- نستخدم ال Module عن طريق TestScript مثلا كالتالى

1- تعريف ال Functions

ال Hola Function

```
static PyObject* hola(PyObject* self, PyObject* args)
{
    const char* name;
    int age;

    if (!PyArg_ParseTuple(args, "si", &name, &age))
        return NULL;

    printf("Name: %s", name);
    printf("Age: %i", age);

    Py_RETURN_NONE;
}

if (!PyArg_ParseTuple(args, "si", &name, &age))
```

لاحظ إننا هنا توقعنا إن هيتمرر لل Function التالى s وهى string و i وهى integer وقمنا بإسناد هذه القيم ل name و age

ملحوظة: انت لن تقوم بالتعديل على name فافضل تعريف إنه يكون const فيكون تعريفه كالتالى

```
const char* name;
```

MyABS Function

```
static PyObject* myabs(PyObject* self, PyObject* args)
{
    int number;

    if (!PyArg_ParseTuple(args, "i", &number))
        return NULL;

    if (number<0){
        number=-number;
    }
    return Py_BuildValue("i", number);
}
```

لاحظ اننا توقعنا ان هيتمرر لل Function التالى i وهو integer وبيعبّر عن الرقم اسندنا القيمة الى number (التحويل من Python Value إلى C Value)

ال Return لازم يكون عبارة عن Python Value (او PyObject) وهو ال Return Type الخاص بال myabs Function كما لاحظت، فبالتالى هحتاج نحول من ال C Value إلى Python Value ودا هيتم عن طريق إستخدام Py_BuildValue وهنا تم تحديد ان هيتم عمل return ل i وهو Integer وقيمتة مساوية ل number

holaDict Function

```
static PyObject* holaDict(PyObject* self, PyObject* args)
{
    const char* key;
    int value;

    if (!PyArg_ParseTuple(args, "si", &key, &value))
        return NULL;

    return Py_BuildValue("{s:i}", key, value); //Returns a Dict.
}
```

لاحظ ان بيتم إعادة Dictionary Object واحنا حددنا كدا بالجزئية دي {s:i} إذا حبيت تعمل return ب List فكل ما عليك هو انك تعدل ال Format للتالى
return Py_BuildValue("[s,i]", key, value); //Returns a List object
وإذا حبيت تعمل return ب Tuple فكل ما عليك هو انك تعدل ال Format كالتالى (s,i)

hola Tuple Function


```
static PyObject* holaTuple(PyObject* self, PyObject* args)
{
    const char* name;
    int age;

    if (!PyArg_ParseTuple(args, "si", &name, &age))
        return NULL;

    return Py_BuildValue("(s,i)", name, age); //Returns a Tuple
}
```

divTwo Function

```
static PyObject* divTwo(PyObject* self, PyObject* args)
{
    int first;
    int second;
    int result;

    if (!PyArg_ParseTuple(args, "ii", &first, &second))
        return NULL;

    printf("First: %i\n", first);
    printf("Second: %i\n", second);

    if(second==0){ //DivByZeroError!
        PyErr_SetString(PyExc_ZeroDivisionError, "DivByZero");
        return NULL; //Get out!
    }
    result = first/second;

    return Py_BuildValue("i", result);
}
```

إذا كان المقسوم عليه يساوى 0 يقة فى Error! ونقدر نبليغ ال (المفسر (Interpreter)) بيه باستخدام
 PyErr_SetString
 نوع ال Error هو PyExc_ZeroDivisionError وال message هتكون DivByZero

2- الضم لل Methods Table كالتالى

```
static PyMethodDef SimpleModuleMethods[]=
{
    {"hola", hola, METH_VARARGS, "prints name and age"},
    {"myabs", myabs, METH_VARARGS, "returns the abs of a number"},
}
```

```

{"divTwo", divTwo, METH_VARARGS, "DIV 2 "},
{"holaTuple", holaTuple, METH_VARARGS, "returns a tuple"},
{"holaDict", holaDict, METH_VARARGS, "returns a dict"},

{NULL, NULL, 0, NULL}
};

```

لاحظ إن آخر عنصر في ال Array هو حاجز..
 3- عمل Initialize لل Module كالتالي بنستدعى فيها ال Py_InitModule Function كالتالي

```

PyMODINIT_FUNC

initsimplemodule(void)
{
    (void) Py_InitModule("simplemodule", SimpleModuleMethods);
}

```

4- ال Setup Script كالتالي

```

from distutils.core import setup, Extension

modExt = Extension('simplemodule', sources = ['simplemodule.c'])

setup (name = 'SimpleModPackage',
       version = '1.0',
       description = 'hola, myabs',
       ext_modules = [modExt])

```

بنوضح فيه اسم ال Package والإصدار والوصف وال extension اللذي يشمل اسم ال module وال source
 5- عمل Build و Install كالتالي

```

python setup.py build
python setup.py install

```

6- عمل Test Script واختبار ال Module كالتالي

```

#!/bin/python

import simplemodule as sm

sm.hola("Ahmed", 18)
print sm.myabs(-10) #10
print sm.myabs(7) #7
print sm.holaDict("python", 1)
print sm.holaTuple("ahmed", 999)
print sm.divTwo(2, 0)

```

```
#Output:
Name: Ahmed
Age: 18
10
7
{'python': 1}
('ahmed', 999)
First: 2
Second: 0
Traceback (most recent call last):
File "C:\Python25\Projects\exten\smTest.py", line 10, in <module>
print sm.divTwo(2, 0)
ZeroDivisionError: DivByZero
```

References:

- 1- [Extending Python](#)
- 2- [Programming Python 3rd Edition](#)

Related:

- 1- [Style Guide for C Code](#)
- 2- [SWIG](#)
- 3- [CXX](#)

Swig

TODO...

ctypes

سنطور في هذه الجزئية extension ل libmagic -وهي مكتبة نستطيع من خلالها تحديد نوع الملف من خلال الرقم السحري magic number- لنستطيع استخدامها في بايثون عن طريق ال foreign Function library او ال FFL للإختصار والمسماه ب ctypes وهي تقدم انواع بيانات متوافقة مع تلك الموجودة في ال C وتسمح لنا بإستدعاء الدوال بالمكتبات لنستطيع استغلالها كما نريد (في الغالب ستريد تقديم wrapper حول مكتبة ما لنستطيع استخدامها بسهولة في بايثون)

1- قم بتثبيت libmagic وملفات التطوير libmagic-dev

2- افتح الملف الرأسى magic.h

ستجده يشمل الثوابت المعرفة والدوال المكشوفة والهياكل structures

```
#ifndef _MAGIC_H
#define _MAGIC_H

#include <sys/types.h>

#define MAGIC_NONE 0x000000 /* No flags */
#define MAGIC_DEBUG 0x000001 /* Turn on debugging */
#define MAGIC_SYMLINK 0x000002 /* Follow symlinks */
#define MAGIC_COMPRESS 0x000004 /* Check inside compressed files */
#define MAGIC_DEVICES 0x000008 /* Look at the contents of devices */
#define MAGIC_MIME_TYPE 0x000010 /* Return only the MIME type */
.....
#define MAGIC_NO_CHECK_TROFF 0x000000 /* Don't check ascii/troff */

typedef struct magic_set *magic_t;
magic_t magic_open(int);
void magic_close(magic_t);
.....
const char *magic_file(magic_t, const char *);
const char *magic_descriptor(magic_t, int);
const char *magic_buffer(magic_t, const void *, size_t);
```

3- نستدعى ال FFL لساحة العمل

```
from ctypes import *
```

4- نقوم بتعريف الثوابت

```

MAGIC_NONE=0x000000      # No flags
MAGIC_DEBUG=0x000001     # Turn on debugging
MAGIC_SYMLINK=0x000002    # Follow symlinks
MAGIC_COMPRESS=0x000004   # Check inside compressed files
MAGIC_DEVICES=0x000008    # Look at the contents of devices
MAGIC_MIME_TYPE=0x000010  # Return only the MIME type
MAGIC_CONTINUE=0x000020  # Return all matches
MAGIC_MIME_ENCODING=0x000400 # Return only the MIME encoding
...
# Defined for backwards compatibility; do nothing
MAGIC_NO_CHECK_FORTTRAN=0x000000 # Don't check ascii/fortran
MAGIC_NO_CHECK_TROFF=0x000000    # Don't check ascii/troff

```

نقوم بتحميل المكتبة -loading-

```
libmagic=CDLL("libmagic.so") #or even better, check ctypes.util
```

سريعا حول انواع البيانات

ctypes type	C type
c_char	Char
c_wchar	wchar_t
c_byte	Char
c_ubyte	Unsigned char
c+short	Short
c_ushort	Unsigned short
c_int	Int
c_uint	Unsigned int
c_long	long
c_ulong	Unsigned long
c_longlong	Long long
c_ulonglong	Unsigned long long
c_float	Float
c_double	Double
c_longdouble	Long double

c_char_p	Char* (null terminated)
c_wchar_p	wchar_t* (null terminated)
c_void_p	Void*

5- نحدد ال types

```
magic_t=c_void_p #void pointer.
```

6- نحدد الدوال

لتحديد دالة بنحدد (القيمة العائدة ال return) والمعاملات
مثلا الدالة magic_open بتدي عائد من النوع magic_t ويتاخذ معامل int

```
magic_t magic_open(int);
```

```
magic_open=libmagic.magic_open
```

```
magic_open.restype=magic_t
```

تحديد القيمة العائدة

```
magic_open.argtypes=[c_int]
```

تحديد المعاملات

الدالة magic_file بتدي ريترن بمؤشر ل char ويتاخذ معاملات magic_t و مؤشر ل char

```
const char *magic_file(magic_t, const char *);
```

```
magic_file=libmagic.magic_file
magic_file.restype=c_char_
magic_file.argtypes=[magic_t, c_char_p]
```

بعد انتهاء اعلام بايثون عن الدوال اللى محتاجينها نقوم بتقديم الدوال البايثونية السلسلة فى الإستخدام

```
def guess(filepath, flags):  
    mc=magic_open(flags)  
    magic_load(mc, None)  
    res=magic_file(mc, filepath)  
    magic_close(mc)  
    return res
```

المشروع كامل تجده على

<http://bitbucket.org/xmonader/libmagicbindings/>

باسم pymagic وتجد فيها ايضا مكتبات اضافية لسى شارب وباسكال لإستخدام libmagic

لمثال اكبر ومتعمق فى ctypes تقدر تطلع على binding لمكتبة PCRE على

<http://bitbucket.org/xmonader/pypcr/>

Chapter 15 (Functional Python)

هذا الفصل اختياري.. هنتكلم عن وحدتي functools و itertools في الواقع بايثون ينقصها الكثير من التحسينات لتقديم [لغة وظيفية](#) حقيقية ولكنها تسمح لك بالبرمجة بهذا الأسلوب

functools

`reduce(function, iterable[, init])`

ملحوظة هي هي الدالة reduce
من اسمها "تقليل" الهدف منها هو تقليل sequence لقيمة واحدة

```
#reduce (func, seq[, init])
print reduce((lambda x, y: x+y), [1, 2, 3, 4, 5]) #15
#((((1+2)+3)+4)+5)
print reduce((lambda x, y: x*y), [1, 2, 3, 4, 5]) #120
```

`partial(func, *args, **kwargs)`

هدفها هو تغليف دالة ما ومعاملاتها واعادة كائن -دالة جزئية مازالت تنتظر باقى المعاملات- يحويها هذا المفهوم يسمى Currying

```
>>> def sayhi(to):
...     print "Hi, ", to
>>> f=lambda to="Ahmed":sayhi(to)
>>> f
<function <lambda> at 0xb7df08b4>
>>> f()
Hi, Ahmed
```

نقدر نسهل نفس المثال بإستخدام partial كالتالى

```
>>> part_f=partial(sayhi, "Ahmed")
>>> part_f()
Hi, Ahmed
```

تقدر تستخدمها بطريقة انك تغلف دالة ومجموعة معاملات قد تستكمل لاحقا

```
>>> def someparty(*guests):
...     print "Visitors: "
...     for g in guests:
...         print "\t", g
>>> someparty("Ahmed")
```

```
Visitors:
    Ahmed
>>> someparty("Ahmed", "Mido")
Visitors:
    Ahmed
    Mido
>>> p=partial(someparty, "Ahmed")
>>> p("Mido")
Visitors:
    Ahmed
    Mido
>>> p
<functools.partial object at 0xb79d8d74>
>>> p("Mido", "Youssef")
Visitors:
    Ahmed
    Mido
    Youssef
```

احد الإستخدامات العملية لها فى التعامل مع PyQt4 كالتالى مثلا -لأنك ستواجه صعوبة فى تمرير المتغيرات مع connect لل slot

```
self.connect(btn, SIGNAL("clicked()"),
             partial(self.onClicked, ARG))
```

طبعا تقدر تستخدم lambda بنفس الكيفية!

هنا مثلا نقوم بربط الإشارة clicked للزر button2 ب slot -تغلف self.anyButton التى تأخذ معامل واحد التى يتم تنفيذها عند محاولة استدعاء ال slot- هكذا قد تغلبنا على العملية اليس كذلك ؟

itertools

any(iterable)

عمرک کان عندک مثلاً list من مجموعة ارقام وعایز تعرف هل فیها رقم اکبر من 3 مثلاً ؟

```
print any(x>3 for x in [1, 3, 4, 5, 6, 7]) #True
```

all(iterable)

عمرک کان عندک مثلاً list من مجموعة ارقام وعایز تعرف هل کلها اکبر من 3 مثلاً ؟

```
print all(x>3 for x in [1, 3, 4, 5, 6, 7]) #False
```

الفكرة هى ان بتنشئ list من True/False ويتم حساب الناتج عن طريقها سواء على قيمة واحدة او كل القيم
لاحظ ان هنا ال iterable انشئناه باستخدام ال list comprehension ولو اتكلمنا عنها تانى هنتصرها ل
LC

repeat(obj, times=None)

بتقوم بعمل iterator يعيد نفس الكائن بعدد times او للأبد اذا كان None

```
>>> list(repeat(10, 3))  
[10, 10, 10]
```

هنلاحظ ان الناتج [10, 10, 10] تم تكرار الكائن 10 ل 3 مرات
وهكذا

takewhile(pred, seq)

بتأخذ عناصر من seq كلما يتم تأكيد الشرط المؤكد pred

```
>>> list(takewhile(lambda x: x<3, [0, 1, 2, 3, 4]))  
[0, 1, 2]
```

dropwhile(pred, seq)

العكس بقية بتترك عناصر من seq كلما يتم تأكيد ال pred

```
>>> list(dropwhile(lambda x: x<3, [0, 1, 2, 3, 4, 6, 7]))
[3, 4, 6, 7]
```

groupby(seq[, key=None])

على فرض عندنا سلسلة من العناصر وعايزين نجتمع كل العناصر من seq باستخدام key function
-lambda x: x افتراضيا ال key هو العنصر-

```
>>> list(groupby("111122233344"))
[('1', <itertools._grouper object at 0xb77ef36c>), ('2', <itertools._grouper object at 0xb77ef3ec>), ('3', <itertools._grouper object at 0xb77ef40c>), ('4', <itertools._grouper object at 0xb77ef42c>)]
print [list(i) for (x,i) in itertools.groupby("111122233344")]
```

الناتج

```
[[('1', '1', '1', '1'), ('2', '2', '2'), ('3', '3', '3'), ('4', '4')]
```

الفكرة ان groupby بتجمع العناصر وبتعيدها فى صورة العنصر و iterator ليه و هكذا) فعشان كدا عملنا
I list اذا عايز العناصر فقط مش list خد قيمة ال x بس

مثال اخر

```
things = [("animal", "bear"), ("animal", "duck"), ("plant", "cactus"), ("vehicle", "speed boat"),
("vehicle", "school bus")]
for key, group in groupby(things, lambda x: x[0]):
    for thing in group:
        print "A %s is a %s." % (thing[1], key)
    print "
```

الناتج

```
A bear is a animal.
```

A duck is a animal.
A cactus is a plant.
A speed boat is a vehicle.
A school bus is a vehicle.

مرجع للمثال <http://stackoverflow.com/questions/773/how-do-i-use-pythons-itertoolsgroupby>

`chain(*iters)`

ينشئ iterator ليعيد العناصر من اول عنصر يتم تمريره حتى ينتهى فينتقل للعنصر التالى وهكذا
او تقدر تقول بتنشئ سلسلة بين ال iters

```
>>> for el in chain("Hello", "World", range(3)):
...     print el
H
e
l
l
o
W
o
r
l
d
0
1
2
```

`ifilter(pred, seq)`

نفس فكرة filter.. تعيد iterator فى لكل عنصر يقع فى نطاق ال pred

```
>>> for x in ifilter(lambda x: x>3, [3,2, 1, 9, 8]): print x
...
9
8
```

```
>>> list(ifilter(lambda x: x>3, [3,2, 1, 9, 8]))
[9, 8]
```

filter=ifilter معادا فى كيفية اعادة الناتج

```
>>> filter(lambda x: x>3, [3, 2, 1, 9, 8])  
[9, 8]
```

حيث تعيد filter ناتج من list بينما تعيد ifilter ناتج من iterator

```
ifilterfalse(pred, seq)
```

حيث تعيد iterator لكل عنصر لايقع في نطاق pred

```
>>> list(ifilterfalse(lambda x: x>3, [3, 2, 1, 9, 8]))  
[3, 2, 1]
```

```
izip(*iters)
```

خمن؟ ايوه صح هي فعلا مشابهه ل zip ولكن بتعيد iterator

```
>>> list(izip("Hell", [1,2,3, 4]))  
[('H', 1), ('e', 2), ('l', 3), ('l', 4)]  
>>> zip("Hell", [1,2,3,4])  
[('H', 1), ('e', 2), ('l', 3), ('l', 4)]
```


Chapter 16 (PyInotify)

مقدمة

لينكس يقدم انترفيس inotify لمراقبة الأحداث الخاصة بالملفات مثل النقل، الحذف، الإنشاء وغيرها. على سبيل لدينا تطبيق مدير ملفات او برنامج للنسخ الاحتياطي او Real-Time indexer هل منطقي ان نجبر البرنامج كل شوية انه يقرأ المحتوى عشان نشوف الملفات الجديدة ؟ او اللي تم حذفها او غيره ؟ الكرنل بـ "يدفع" الأحداث والبرنامج يعالج الأحداث اللي تهمة

Watches

ال Watches .. بعد تجهيز inotify بيتم تجهيز ال watches وهى بتعبر عن مسار ما بـ mask "يحدد الأحداث المهتمين بمعالجتها" هنستخدم pyinotify فى الكلام.. يمكن اذا رينا ادانا العمر نتكلم عن inotify مباشرة

Events

تقدر توصل للأحداث هنا

http://seb.dbzteam.org/swp/pages/pyinotify...entsCodes_Class

اللى يهمنا منهم هو IN_CREATE و IN_DELETE و IN_MOVED_FROM و IN_MOVED_TO عند الإنشاء والحذف وبدأ النقل "من مسار ما" وعند انتهاء النقل "الى مسار ما"

*منطقي اننا عايزين نراقب الأحداث على المجلد وكل المجلدات بداخله فهنستخدم rec=True اختصار ل recursive = True

Going on

نبدأ المتطلبات

```
from pyinotify import *
```

لاحظ المفروض تستخدم pyinotify.WatchManger وهكذا بدل من استيراد الكل من pyinotify بننشئ ال WatchManager ليدبر كل العناصر المراقبة mask بنحدد فيه الأحداث اللي مهتمين فيها

```
wm=WatchManager()
mask=IN_DELETE | IN_CREATE | IN_MOVED_FROM | IN_MOVED_TO #to watch.
```

ثم بننشئ ال inotify (بمدير العناصر المراقبة و كائن من صف خاص بمعالجة الأحداث اللتى نريدها) ونضيف المسار المطلوب مراقبة الأحداث عليه ونحدد ال mask ونجعل rec=True لمراقبة الأحداث بما فى اسفله

```
p=PDir()
notifier=Notifier(wm, p)

#add a watch..
watch=wm.add_watch("/home/striky/Desktop", mask, rec=True)
```

ونبدأ حلقة المعالجة

```
notifier.loop()
```

PDir

لاحظ اننا مش عارفين ايه PDir دا اصلا؟
 باختصار هو صف عرفنا فيه معالجات خاصة بالأحداث دي بيورث pyinotify.ProcessEvent

```
class PDir(ProcessEvent):

    def my_init(self, **kwargs):
        self.mv_cookies={ } # {cookie: src}

    def process_IN_DELETE(self, event):
        print "Remove: %s"%event.pathname

    def process_IN_CREATE(self, event):
        print "Create: %s"%event.pathname

    def process_IN_MOVED_FROM(self, event):
        print "Moving FROM: %s"%event.pathname
        self.mv_cookies[event.cookie]=event.pathname

    def process_IN_MOVED_TO(self, event):
        print "Moving TO: %s"%event.pathname
        print " WAS MOVED FROM: ", self.mv_cookies[event.cookie]
```

Event

الصف مباشر جدا.. ملحوظة ال event فى اى من معالجات الحدث هو كائن من الصف Event وله صفات منها
 <= wd وهى رقم مميز (Watch Descriptor) للعنصر المراقب
 <= name وهى اسم العنصر الذى حدث عليه الحدث
 <= pathname مساره الكامل
 <= dir بتعبر عما اذا كان مجلد
 <= mask بتعبر عن ال bitmask للأحداث المراقبة فى ال wd

IN_MOVED_TO و IN_MOVED_FROM معا مثل cookie <= قيمة مميزة تستخدم لربط حدثين وهنا استخدمنا ال dict المسمى لأن كل منهم يمثل نصف عملية النقل وهما الإيتين يعبرو عن العملية. وهنا استخدمنا ال dict المسمى mv_cookies للربط بينهم

المثال كامل

```
import os
from pyinotify import *

wm=WatchManager()
mask=IN_DELETE | IN_CREATE | IN_MOVED_FROM | IN_MOVED_TO #to watch..

#no need.. use .pathname instead..
def fullname(event):
    return os.path.join(event.path, event.name)

class PDir(ProcessEvent):

    def my_init(self, **kwargs):
        self.mv_cookies={} # {cookie: src}

    def process_IN_DELETE(self, event):
        print "Remove: %s"%event.pathname

    def process_IN_CREATE(self, event):
        print "Create: %s"%event.pathname

    def process_IN_MOVED_FROM(self, event):
        print "Moving FROM: %s"%event.pathname
        self.mv_cookies[event.cookie]=event.pathname

    def process_IN_MOVED_TO(self, event):
        print "Moving TO: %s"%event.pathname
        print " WAS MOVED FROM: ", self.mv_cookies[event.cookie] #same unique number

#notifier
p=PDir()
notifier=Notifier(wm, p)

#add a watch..
watch=wm.add_watch("/home/striky/Desktop", mask, rec=True)

notifier.loop()
```

دا كان المختصر المفيد للمزيد حول pyinotify تقدر تراجع
<http://seb.dbzteam.org/swp/pages/pyinotify.html>

Chapter 17 (GUI)

PyGTK

هتتناول فى الجزئية دى مقدمة فى GTK



نريد ان نعمل window نافذة-مشابهه لى ونخليها متسنتره (فى منتصف الشاشة) اول ماتنشئ
1- استدعى ال gtk

```
import gtk
```

2- انشئ class يورث ال gtk

```
class Window(gtk.Window):  
  
    def __init__(self):  
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)  
        self.__init_comp()  
  
        self.show_all()
```

3 - هنا بنقول ان ال window دى TOPLEVEL مش POPUP
بنستدعى ال __init_comp -طريقة لإنشاء الواجهة-

.show_all

الطريقة show_all بتعرض كل ال components داخل ال Window وهنا مش فى غيرها بس اتعود تستخدمها
لأنك هتبقه تحط ويدجتس كثير جواها

.init_comp

```
def __init_comp(self):  
    self.set_title("Hello, World!")  
    self.set_position(gtk.WIN_POS_CENTER)
```

```
.set_title(new_title)
```

بتستخدم فى تغيير ال title على النافذة

```
.set_position(pos)
```

بتستخدم هنا لتحديد ال مكان الخاص بالنافذة
ولها عدة قيم زى

```
gtk.WIN_POS_CENTER
```

بتسنىر النافذة عند انشاءها

```
gtk.WIN_POS_CENTER_ALWAYS
```

هيم سنترتها عند اى تغيير فى ال size

```
gtk.WIN_POS_MOUSE
```

هيم اظهار النافذة عند مكان الماوس الحالى

```
.set_size_request(h,w)
```

لتحديد ارتفاع وعرض النافذة

```
if __name__=="__main__":  
    w=Window()  
    gtk.main()
```



هنا عندنا نافذة وفيها button واحد مكتوب عليه click me
طيب جميل

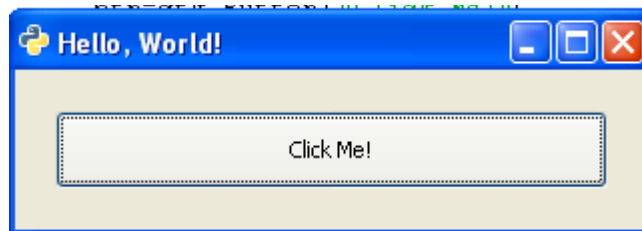
```
class Window(gtk.Window):  
  
    def __init__(self):  
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)  
        self.__init_comp()  
  
        self.show_all()  
  
    def __init_comp(self):  
        self.set_title("Hello, World!")  
        self.set_border_width(20)  
        self.set_position(gtk.WIN_POS_CENTER)  
        btn=gtk.Button("Click Me!")  
        self.add(btn)
```

التعريف العام لل button

```
button = gtk.Button(label=None, stock=None)
```

تقدر تتحكم فى ال border_width بإستخدام

```
.set_border_width(width)
```



نريد ان يظهر مسح لطيفة كلما نضغط على ال button دا

```
def __on_btn_clicked(self, widget, data):  
    md=gtk.MessageDialog(self, gtk.DIALOG_DESTROY_WITH_PARENT,  
gtk.MESSAGE_INFO, gtk.BUTTONS_OK, "Hi!")  
    print widget  
    print data  
    md.run()  
    md.destroy()
```

دى اسمها callback يعنى طريقة هيتم تنفيذها عند حدوث شئ معين زى الضغط على

MessageDialog

نيجى لل

اول معامل هو ال parent

تانى معامل فيه خصائص الديالوج

gtk.DIALOG_MODAL

لوا ه فهو اللذي هيصطاد اى ايفنت يحصل من الكيبورد (يمنع الوصول لل نافذة الأصلية إلا بعد انتهائه)

gtk.DIALOG_DESTROY_WITH_PARENT

هيتقفل فى حال قفل ال parent

gtk.DIALOG_NO_SEPARATOR

مش هيطهر خط فاصل بين الرسالة وال buttons بتوع الرسالة

تالت معامل هو نوع المسح

هل معلومة او تحذير او سؤال او خطأ

gtk.MESSAGE_INFO

معلومة

gtk.MESSAGE_WARNING

تحذير

gtk.MESSAGE_QUESTION

سؤال

gtk.MESSAGE_ERROR

خطأ

تمام كذا بس انا شغلت الكود ومش فى حاجة حصلت (:
فعلا لأننا لسه مش ربطنا ال callback بال signal
بكل بساطة اكتب التالى

```
btn.connect("clicked", self.__on_btn_clicked, None)
```

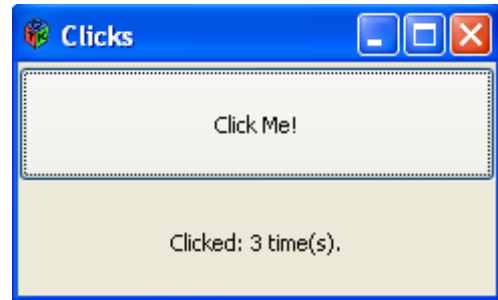


وبس كذا طبعا انت مدايق من None وياه المتغيرات اللتى تم تعريفها انا فى ال callback دى اصلا ؟ ايه
?? widget, data

طيب تمام جدا قيل ماتسألنى السؤال دا تقدر تعمل حاجة حلوة قوى بإنك تجرب print على widget وعلى
data فى ال callback

```
<gtk.Button object (GtkButton) at 0xb803f0>  
None
```

ال widget دا ال button فى مثالنا
ال data هى ال None تقدر طبعا تستبدلها بأى حاجة المهم انها تكون شئ هيفيدك..
هنتعرف عليها اكثر لاحقا...



لاحظ ان النافذة متقسمة لجزئين راسي اول جزء فيه button والثانى فيه label بيتكتب عليه عدد مرات الضغط على ال button
 احنا ممكن نكتبها بالطريقة المعتادة وممكن نكتبها بطريقة كثير افضل بإستخدام ال OOP
 تمام الأول عشان نخليهم متقسمين فى شكل معين افقى او رأسى بنستخدم Hbox (اختصار ل Horizontal Box)
 او Vbox (اختصار ل Vertical Box) تمام ؟ قشطة



ال vertical box بياخد ال widgets او بوكسز تانية فى صورة rows صفوف -صندوق رأسى-

ال horizontal box بياخد ال widgets او البوكسز التانى فى صورة Columns عواميد -صندوق افقى-



لإنشاء box بتبدأ اولاً ب homogeneous ودى معناها هل كل الأجزاء متساوية فى العرض او الطول ليها ترجمة بإسم متجانسة اعتقد مناسبة ؟ والمعامل التانى لتحديد عرض الفاصل

```
vbox=gtk.VBox(False, 2)
```

pack_start(child, expand, fill, padding)

لوضع ال widget من الشمال لليمين او من فوق لتحت "صورة فطرية!"

pack_end(child, expand, fill, padding)

لوضع ال widget من اليمين للشمال او من تحت لفوق وهى موجود لل Hbox وال Vbox

expand: هل عايزه يكبر مع اى زيادة فى حجم النافذة؟
 fill: فى حال التصغير هل يتم اخفاء جزء منه؟ وليس تصغيره
 padding: الهامش حوله

-استخدام الجداول

	0	1	2
0	+	+	+
1	+	+	+
2	+	+	+

لإنشاء جدول بنشئته كالتالى

```
gtk.Table( rows, columns, homogeneous )
```

عدد الصفوف وعدد الأعمدة وهل متجانسين او لا
 لإضافة child باستخدام ال attach method

```
.attach( child, left_attach, right_attach, top_attach, bottom_attach,  
xoptions, yoptions, xpadding, ypadding)
```

child: هو الويدجت هيتم اضافته فى الجدول
 left_attach : العمود على يسار المكان
 right_attach: العمود على يمين المكان
 top_attach : الصف فوق المكان
 bottom_attach: الصف تحت المكان
 مثال للتوضيح

	0	1	2
0	+	+	+
1	+	+	+
2	+	+	+

لو نريد ان نخط widget معين فى الكورنر اليمين من جدول زى مانت شايك 2X2
 يقع بين الخططين الرأسين 1 و 2 وهما دول ال left_attach, right_attach
 يقع بين الخططين الأفقيين 1 و 2 وهما دول ال top_attach, bottom_attach
 xoptions: الإختيارات ل x
 gtk.FILL: لو الجدول اكبر من الويدجت فالويدجت هيتمدد ليشغل المساحة
 gtk.EXPAND: هنا الجدول هيتمدد اذا كان فى مساحة فى ال window
 gtk.SHINK: اذا تم تصغير المساحة المتاحة للويدجت "مع تصغير الجدول مثلا" هيتم تصغيره

yoptions: الإختيارات ل y مشابه ل x

xpadding: الهامش من ناحية ال x

ypadding: الهامش من ناحية ال y

attach_defaults تستخدم لتقدير الخيارات الأساسية

```
attach_defaults( child, left_attach, right_attach, top_attach,  
bottom_attach )
```

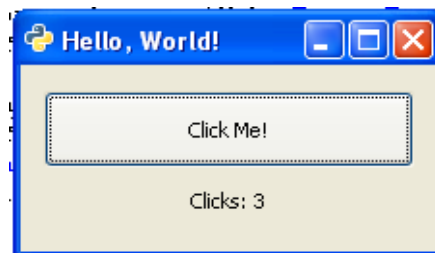
وإلى هتخليك تدخل ال left, right, top, bottom attach فقط والباقي هيكون بالإفتراضى لل x,y options

gtk.FILL | gtk.EXPAND

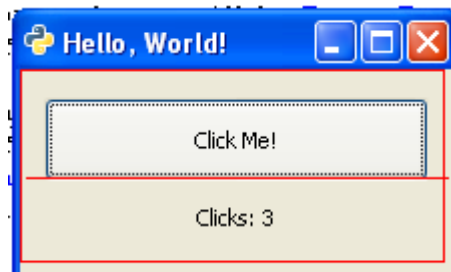
وال x,y padding هيكون 0

فين ال fixed ؟ موجود لعرض ال ويدجتس بتحديد ال مكان على الفورم ولكن "استخدام السابق افضل من حيث حماية طريقة وضعك لل ويدجتس من حيث التمدد والإنكماش وكذا"

تعالى نعمل مثال clicks بصورة واضحة



التصميم



المستطيل الأحمر الكبير عبارة عن Vbox
وجواه صفين الصف الأول فيه button والثانى فيه label
الأول عندنا متغير clicks__ يعبر عن عدد الضغوطات

```
class Window(gtk.Window):  
  
    def __init__(self):  
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)  
        self.__init_comp()  
        self.__clicks=0
```

```

        self.show_all()

    def getClicks(self):
        return self.__clicks

    def setClicks(self, value):
        self.__clicks = value

    def delClicks(self):
        del self.__clicks

    clicks = property(getClicks, setClicks, delClicks, "Clicks's Docstring")

```

ثانيا التصميم

```

def __init_comp(self):
    self.set_title("Hello, World!")
    self.set_position(gtk.WIN_POS_CENTER)
    self.set_border_width(12)

```

هنا نحدد خصائص النافذة title,position,border_width

```

mvbox=gtk.VBox(False, 0)

```

بننشئ Vertical Box عشان نضم فيه button, label

```

btnclicks=gtk.Button("Click Me!")

```

بننشئ ال Button مكتوب عليه Click Me!

```

lbl=gtk.Label("Clicks: ")

```

بننشئ Label مكتوب عليه Clicks:

```

mvbox.pack_start(btnclicks, True, True, 2)

```

```

mvbox.pack_start(lbl, True, True, 0)

```

بنضيف ال btnclicks, lbl لل mvbox

```

self.add(mvbox)

```

بنضيف ال mvbox لل window

```

btnclicks.connect("clicked", self.__on_btnclicks_clicked, lbl,

```

None)

بنربط ال clicked signal الخاصة ب btnclicks بطريقة بإسم

```

__on_btnclicks_clicked

```

صممناها كالتالى

```

def __on_btnclicks_clicked(self, widget, lblclicks, data):
    self.__clicks += 1

```

```
print widget, lblclicks, data
lblclicks.set_text("Clicks: "+str(self.__clicks))
```

المعامل الأول widget يعبر عن ال receiver لل signal
الثاني ال lblclicks يعبر عن ال label الذي نريد ان نغيره
الثالث ال data يعبر عن اى داتا اضافية

كل اللى هيحصل اتنا هنزود عدد ال clicks__ ونعدل ال تكست على ال lblclicks
باستخدام

```
.set_text(newtext)
```

الهيكليّة

```
GObject
|
+GtkObject
+GtkWidget
| +GtkMisc
| | +GtkLabel
| | | `GtkAccelLabel
| | +GtkArrow
| | | `GtkImage
| +GtkContainer
| | +GtkBin
| | | +GtkAlignment
| | | +GtkFrame
| | | | `GtkAspectFrame
| | | +GtkButton
| | | | +GtkToggleButton
| | | | | `GtkCheckButton
| | | | | `GtkRadioButton
| | | | `GtkOptionMenu
| | +GtkItem
| | | +GtkMenuItem
| | | +GtkCheckMenuItem
| | | | `GtkRadioMenuItem
| | | +GtkImageMenuItem
| | | +GtkSeparatorMenuItem
| | | | `GtkTearoffMenuItem
| | +GtkWindow
| | | +GtkDialog
| | | | +GtkColorSelectionDialog
| | | | +GtkFileSelection
| | | | +GtkFontSelectionDialog
| | | | +GtkInputDialog
| | | | | `GtkMessageDialog
| | | | `GtkPlug
| | +GtkEventBox
| | +GtkHandleBox
| | +GtkScrolledWindow
| | | `GtkViewport
| | +GtkBox
| | | +GtkButtonBox
| | | | +GtkHButtonBox
```

```

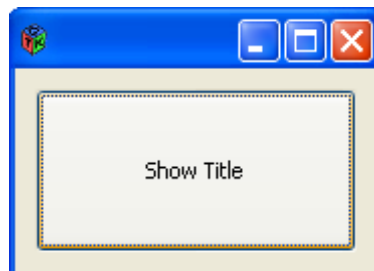
| | | | `GtkVButtonBox
| | | +GtkVBox
| | | | +GtkColorSelection
| | | | +GtkFontSelection
| | | | `GtkGammaCurve
| | | `GtkHBox
| | | +GtkCombo
| | | `GtkStatusbar
| +GtkFixed
| +GtkPaned
| | +GtkHPaned
| | `GtkVPaned
| +GtkLayout
| +GtkMenuShell
| | +GtkMenuBar
| | `GtkMenu
| +GtkNotebook
| +GtkSocket
| +GtkTable
| +GtkTextView
| +GtkToolbar
| | `GtkTreeView
+GtkCalendar
+GtkDrawingArea
| | `GtkCurve
+GtkEditable
| | +GtkEntry
| | `GtkSpinButton
+GtkRuler
| | +GtkHRuler
| | `GtkVRuler
+GtkRange
| | +GtkScale
| | | +GtkHScale
| | | `GtkVScale
| | `GtkScrollbar
| | +GtkHScrollbar
| | `GtkVScrollbar
+GtkSeparator
| | +GtkHSeparator
| | `GtkVSeparator
+GtkInvisible
+GtkPreview
| `GtkProgressBar
+GtkAdjustment
+GtkCellRenderer
| +GtkCellRendererPixbuf
| +GtkCellRendererText
| +GtkCellRendererToggle
+GtkItemFactory
+GtkTooltips
`GtkTreeViewColumn

```

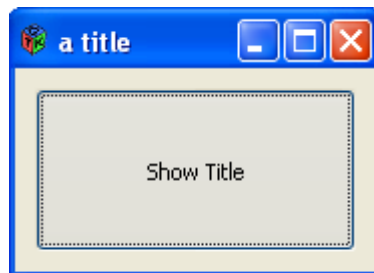
Toggle Button

```
toggle_button = gtk.ToggleButton(label=None)
```

مشابه لل check box ودا ليه حالتين True او False



هنا وهو مش متنشط يعني False



هنا هو متنشط يعني active
اول ما يضغط عليه بيرسل toggled signal

```
import gtk

class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)

        mvbox=gtk.VBox(False, 0)
        togbtn=gtk.ToggleButton("Show Title")
        togbtn.set_active(True)
```



```

togbtn.connect("toggled", self.__on_toggled)

mvbox.pack_start(togbtn, True, True, 2)

self.add(mvbox)

def __on_toggled(self, widget):
    if self.title.strip():
        self.set_title(" ")
    else:
        self.set_title("Hello, World!")

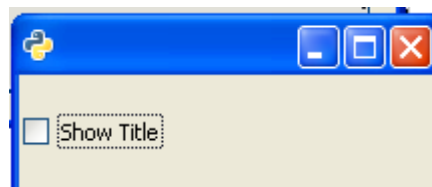
if __name__ == "__main__":
    w=Window()
    gtk.main()

```

CheckButton

الصورة العامة لإنشاءه

```
check_button = gtk.CheckButton(label=None)
```



نفس نظام السابق ولكن بشكل مختلف ليس اكثر

```

import gtk

class Window(gtk.Window):
    def __init__(self):

```

```

super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
self.__init_comp()

self.show_all()

def __init_comp(self):
    self.set_title("Hello, World!")
    self.set_position(gtk.WIN_POS_CENTER)

    mvbox=gtk.VBox(False, 0)
    chkbtn=gtk.CheckButton("Show Title")
    chkbtn.set_active(True)
    chkbtn.connect("toggled", self.__on_toggled)

    mvbox.pack_start(chkbtn, True, True, 2)

    self.add(mvbox)

def __on_toggled(self, widget):
    if self.title.strip():
        self.set_title(" ")
    else:
        self.set_title("Hello, World!")

if __name__=="__main__":
    w=Window()
    gtk.main()

```

RadioButton

الصورة العامة لإنشاءه

```
radio_button = gtk.RadioButton(group=None, label=None)
```

لازم تظبط ال group دى عشان يشتغل بصورة سليمة فبكل بساطة خليها None لأول radio button وبعد كذا خليها ال radio button اللذي انشئ اول واحد!

همنشئ حاجة مشابهه لى



التصميم



النافذة فيها 2 radio buttons وفيها Horizontal Separator و button

```
class Window(gtk.Window):  
  
    def __init__(self):  
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)  
        self.__init_comp()  
        self.gender="Male"  
        self.show_all()  
  
    def __init_comp(self):  
        self.set_title("Hello, World!")  
        self.set_position(gtk.WIN_POS_CENTER)  
        self.set_border_width(12)  
  
        mvbox=gtk.VBox(False, 0)  
        rd1=gtk.RadioButton(None, "Male")  
        rd1.set_active(True)  
  
        rd1.connect("toggled",self.__on_radio_toggled, "Male")  
  
        rd2=gtk.RadioButton(rd1, "Female")  
        rd2.connect("toggled",self.__on_radio_toggled, "Female")  
  
        mvbox.pack_start(rd1, False, False, 2)  
        mvbox.pack_start(rd2, False, False, 2)
```

```

mvbox.pack_start(gtk.HSeparator(), True, True, 0)
btninfo=gtk.Button("OK!")
btninfo.connect("clicked", self.__on_btninfo_clicked)

mvbox.pack_start(btninfo, False, False, 3)
self.add(mvbox)

def __on_radio_toggled(self, w, data):
    self.gender=data

def __on_btninfo_clicked(self, w):
    md=gtk.MessageDialog(self, gtk.DIALOG_DESTROY_WITH_PARENT,
gtk.MESSAGE_INFO, gtk.BUTTONS_OK, self.gender)
    md.run()
    md.destroy()

```

لاحظ هنا انشئنا Hseparator (فاصل افقى) وضمناه مباشرة

```
mvbox.pack_start(gtk.HSeparator(), True, True, 0)
```

ال signal المهمة هنا هى toggled لل radio buttons

Adjustment

هى مش ويدجت ولكن يستخدم فى تخزين ونقل معلومات محددة لإعدادات ويدجتس معينة زى السكرولبارز والسينرز والراينجز -الفترات- وغيرهم
لعمل adjustment object بننشئه كالتالى

```
Adjustment( value, lower, upper, step_increment, page_increment, page_size )
```

تقدر تعتبرها ك model لويدجت وهو يعرضها لك فى ال view بتاعته على كل حال هنشوف
ال value القيمة الأساسية
ال lower اقل قيمة
ال upper اعلى قيمة
ال step_increment مقدار الزيادة

Scale

فى منها افقى وفي رأسى لإنشاء الأوبجكتس منها -تقدر تطلق عليها منزلق-

```

VScale( adjustment )
VScale( min, max, step )
HScale( adjustment );

```

```
HScale min, max, step );
```

يا إما تمرر adjustment او تمرر اقل واكبر قيمة والزيادة
لإظهار القيمة مع ال scale او لأتقدر تستخدم draw_value او set_draw_value وتديهم قيمة true او false فى حالة
الإخفاء
-- هى true افتراضيا..

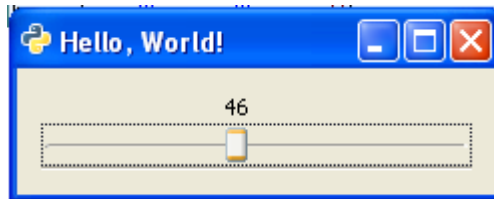
digits/set_digits

لتحديد عدد الأرقام بعد العلامة العشرية المرغوب فى ظهورها

set_value_pos(pos)

تقدر تستخدمهم لتحديد المكان الذى سيظهر عليه قيمة ال value ودى بتاخذ القيم

```
gtk.POS_LEFT  
gtk.POS_RIGHT  
gtk.POS_TOP  
gtk.POS_BOTTOM
```



مثلا لعمل المثال دا هنبشئه كالتالى

```
class Window(gtk.Window):  
  
    def __init__(self):  
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)  
        self.__init_comp()  
  
        self.show_all()  
  
    def __init_comp(self):  
        self.set_title("Hello, World!")  
        self.set_position(gtk.WIN_POS_CENTER)  
        self.set_border_width(12)  
  
        adj=gtk.Adjustment(5,1, 101)  
        hscale=gtk.HScale(adj)  
        hscale.set_digits(0)  
  
        mvbox=gtk.VBox(False, 0)  
        mvbox.pack_start(hscale, True, True, 0)  
        mvbox.pack_start(gtk.HSeparator(), True, True, 0)
```

```
self.add(mvbox)
```

update_policy

بتحدد متى تعديل ال value بال adjustment الخاصة بال range widget وترسل ال value_changed signal بتأخذ قيم مثل

gtk.UPDATE_CONTINUOUS

ترسل عند حدوث اقل تغيير ممكن فى ال range

gtk.UPDATE_DISCONTINUOUS

ترسل لما المستخدم يسحب الماوس ويكون ال range ثابت

gtk.UPDATE_DELAYED

ترسل بمجرد ان المستخدم يترك الماوس او يتوقف عن الحركة لفترة صغيرة ويتم التحكم فى ال policy من خلال

.set_update_policy(up_policy)

للحصول على ال adjustment الخاصة بال range استخدم

get_adjustment.

لتعديل ال adjustment استخدم

.set_adjustment(adj)



هنا عندنا 2 scales واحدة افقى والثانية رأسى والإثنين هندیهم adjustment واحدة عشان التغيير فيهم بيقة على التوازي

```

class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        adj=gtk.Adjustment(5,1, 101)
        hscale=gtk.HScale(adj)
        hscale.set_digits(0)
        vscale=gtk.VScale(adj)
        vscale.set_digits(0)

        mvbox=gtk.VBox(False, 0)
        mvbox.pack_start(hscale, True, True, 0)
        mvbox.pack_start(vscale, True, True, 2)

        self.add(mvbox)

```

`.set_digits(num)`

بتحدد عدد الأرقام المطلوبة بعد العلامة(خليناها 0)



نيجي لمثال تاني هنا عندنا Scale و SpinButton نريد ان نربطهم ان لما يتغير قيمة اى منهم يتعدل فى الثانية هنا هنستخدم ال adjustment object للإثنين بحيث ان يتعدل قيمة ال value فيها للإثنين (بما انها ال model الذى يعرضه كل من ال Scale, SpinButton)

كود المثال

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

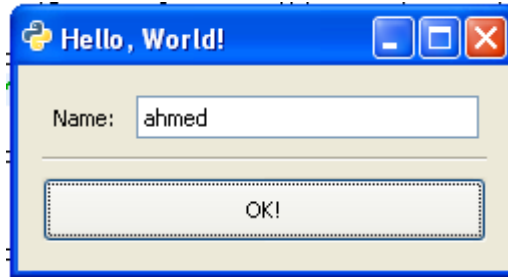
    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        adj=gtk.Adjustment(5,1, 101)
        hscale=gtk.HScale(adj)
        hscale.set_digits(0)
        spin=gtk.SpinButton(adj)
        spin.set_digits(0)

        mvbox=gtk.VBox(False, 0)
        mvbox.pack_start(hscale, True, True, 0)
        mvbox.pack_start(spin, True, True, 2)
        mvbox.pack_start(gtk.HSeparator(), True, True, 0)

        self.add(mvbox)
```


Label/Entry



ال Label يستخدم لعرض تكست ما بدون الحاجة لتغييره من المستخدم
ال Text Entry مشابه لل تكست فيلد بيدخل فيه المستخدم قيمة مطلوبة مثلا...



التصميم

الكود

```
class Window(gtk.Window):  
  
    def __init__(self):  
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)  
        self.__init_comp()  
  
        self.show_all()  
  
    def __init_comp(self):  
        self.set_title("Hello, World!")  
        self.set_position(gtk.WIN_POS_CENTER)  
        self.set_border_width(12)  
  
        mvbox=gtk.VBox(False, 0)  
  
        lblname=gtk.Label("Name:")  
        nameentry=gtk.Entry()  
  
        hbox=gtk.HBox(False, 0)  
        hbox.pack_start(lblname, True, True, 0)  
        hbox.pack_start(nameentry, True, True, 4)
```

```

mvbox.pack_start(hbox, True, True, 0)
mvbox.pack_start(gtk.HSeparator(), True, True, 2)

btnok=gtk.Button("OK!")
btnok.connect("clicked", self.__on_btnok_clicked , nameentry)

mvbox.pack_start(btnok, True, True, 2)
self.add(mvbox)

def __on_btnok_clicked(self, w,e, data=None):
    print e.get_text()

```

Entry

<code>.set_text(newtext)</code>	لتغيير ال text
<code>.insert_text(text, _from)</code>	بتضيف text من عند نقطة ال _from
<code>.select_region(_from,_to)</code>	بتظلل منطقة معينة تبدأ من _from وتنتهى ب _to
<code>.set_max_length(maxlen)</code>	بتحدد بيها اقصى عدد حروف لل entry
<code>.set_editable(bool)</code>	هل يقدر المستخدم يعدل فيها؟

Label:

<code>.set_text(text)</code>	تعديل ال text
<code>.get_text()</code>	اعادة ال text
<code>.set_justify(just)</code>	بتاخذ قيم كالتالى
JUSTIFY_LEFT يسار JUSTIFY_RIGHT يمين JUSTIFY_CENTER المنتصف	
<code>.set_line_wrap(bool)</code>	هل ي wrap السطور او لأ؟
<code>.set_markup(markup)</code>	تخزين markup

gtk.Arrow

يستخدم ليشير الى اتجاه ما لبرنامج (بوضع رأس السهم)



```
arrow = gtk.Arrow(arrow_type, shadow_type)
arrow.set(arrow_type, shadow_type)
```

arrow_type بتعبر عن نوع السهم اعلى ، اسفل ، يمين ، يسار

```
ARROW_UP
ARROW_DOWN
ARROW_LEFT
ARROW_RIGHT
```

و shadow_type بتحدد نوع ال shadow

```
SHADOW_IN
SHADOW_OUT # the default
SHADOW_ETCHED_IN
SHADOW_ETCHED_OUT
```

مثال

```
# Create an Arrow widget with the specified parameters
# and pack it into a button
def create_arrow_button(arrow_type, shadow_type):
    button = gtk.Button();
    arrow = gtk.Arrow(arrow_type, shadow_type);
    button.add(arrow)
    button.show()
    arrow.show()
    return button
```

```

class Arrows(object):
    def __init__(self):
        # Create a new window
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)

        window.set_title("Arrow Buttons")

        # It's a good idea to do this for all windows.
        window.connect("destroy", lambda x: gtk.main_quit())

        # Sets the border width of the window.
        window.set_border_width(10)

        # Create a box to hold the arrows/buttons
        box = gtk.HBox(False, 0)
        box.set_border_width(2)
        window.add(box)

        # Pack and show all our widgets
        box.show()

        button = create_arrow_button(gtk.ARROW_UP, gtk.SHADOW_IN)
        box.pack_start(button, False, False, 3)

        button = create_arrow_button(gtk.ARROW_DOWN, gtk.SHADOW_OUT)
        box.pack_start(button, False, False, 3)

        button = create_arrow_button(gtk.ARROW_LEFT, gtk.SHADOW_ETCHED_IN)
        box.pack_start(button, False, False, 3)

        button = create_arrow_button(gtk.ARROW_RIGHT,
gtk.SHADOW_ETCHED_OUT)
        box.pack_start(button, False, False, 3)

        window.show()

```

فى الدالة

```

# and pack it into a button
def create_arrow_button(arrow_type, shadow_type):
    button = gtk.Button();
    arrow = gtk.Arrow(arrow_type, shadow_type);
    button.add(arrow)
    button.show()
    arrow.show()
    return button

```

بنقوم بإنشاء button يحوى arrow قمنا بإنشاءه من خلال ال arrow_type،

add method ال باستخدام button لل shadow_type

gtk.Tooltips

بتستخدم لتحديد ال tooltip (نص مساعد على الويدجت)
تنشئ كالتالى

```
tooltips = gtk.Tooltips()
```

تقوم بتحديد نص التلميح باستخدام set_tip method

```
tooltips.set_tip(widget, tip_text)
```

widget هو الويدجت المطلوب تحديد ال tip له
tip_text النص

فقط قم بإضافة ال create_arrow_button

```
class Tooltips:
    def __init__(self):
        # Create a new window
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)

        window.set_title("Tooltips")

        # It's a good idea to do this for all windows.
        window.connect("destroy", lambda w: gtk.main_quit())

        # Sets the border width of the window.
        window.set_border_width(10)

        # Create a box to hold the arrows/buttons
        box = gtk.HBox(False, 0)
        box.set_border_width(2)
        window.add(box)

        # create a tooltips object
        self.tooltips = gtk.Tooltips()

        # Pack and show all our widgets
        box.show()

        button = create_arrow_button(gtk.ARROW_UP, gtk.SHADOW_IN)
        box.pack_start(button, False, False, 3)
        self.tooltips.set_tip(button, "SHADOW_IN")

        button = create_arrow_button(gtk.ARROW_DOWN, gtk.SHADOW_OUT)
        box.pack_start(button, False, False, 3)
```

```

self.tooltips.set_tip(button, "SHADOW_OUT")

button = create_arrow_button(gtk.ARROW_LEFT, gtk.SHADOW_ETCHED_IN)
box.pack_start(button, False, False, 3)
self.tooltips.set_tip(button, "SHADOW_ETCHED_IN")

button = create_arrow_button(gtk.ARROW_RIGHT,
gtk.SHADOW_ETCHED_OUT)
box.pack_start(button, False, False, 3)
self.tooltips.set_tip(button, "SHADOW_ETCHED_OUT")

window.show()

```

gtk.ProgressBar



هو ويدجت يستخدم لعرض تقرير عن الحالة

لإنشاءه

```
progressbar = gtk.ProgressBar(adjustment=None)
```

في حال عدم تحديد ال adjustment ه يتم انشاءها

`.set_fraction(fraction)`

لتحديد ال fraction وهي الكم المنتهى

`.set_orientation(orientation)`

لتحديد اتجاه ملء الزيادة

```
PROGRESS_LEFT_TO_RIGHT من اليسار لليمين  
PROGRESS_RIGHT_TO_LEFT من اليمين لليسا  
PROGRESS_BOTTOM_TO_TOP من أسفل لأعلى  
PROGRESS_TOP_TO_BOTTOM من أعلى لأسفل
```

`.get_text()`

للحصول على النص الظاهر على ال progressbar

`.set_text(to)`

تحديد النص الظاهر على ال progressbar إلى to

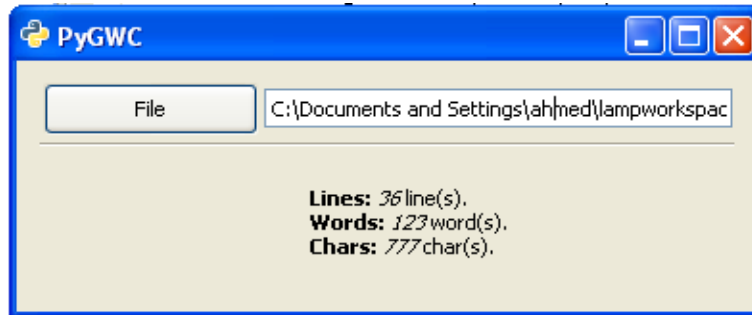
`.pulse()`

لتشير حدوث تغيير فى ال progressbar

```
def progress_timeout(pbj):  
    if pbj.activity_check.get_active():  
        pbj.pbar.pulse()  
    else:  
        # Calculate the value of the progress bar using the  
        # value range set in the adjustment object  
        new_val = pbj.pbar.get_fraction() + 0.01  
        if new_val > 1.0:  
            new_val = 0.0  
        # Set the new value  
        pbj.pbar.set_fraction(new_val)  
  
    # As this is a timeout function, return TRUE so that it  
    # continues to get called  
    return True
```

`.activity_check.get_active()`

هل progressobject نشط أولاً



```
class WordCounter(object):

    def __init__(self):
        pass

    def set_file(self, p):
        self.filepath=p
        fileobj=file(p, "r")
        self.txt=fileobj.read()

        fileobj.close()

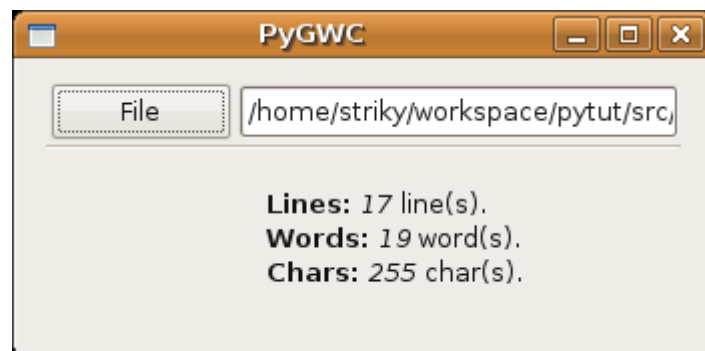
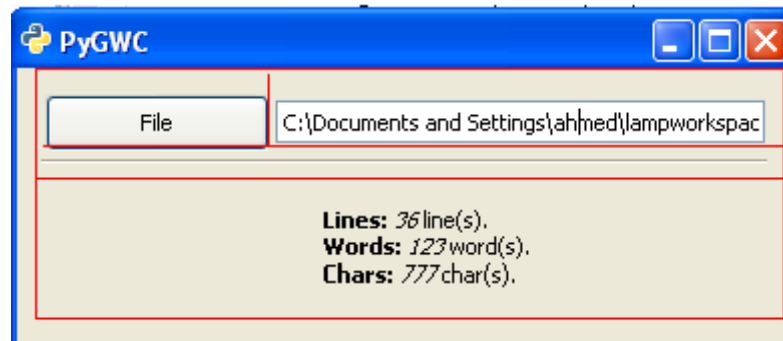
    def get_nwords(self):
        return self.txt.count(" ")+1

    def get_nlines(self):
        return self.txt.count("\n")

    def get_nchars(self, count_spaces=True):
        if count_spaces:
            return len(self.txt)
        else:
            return len(self.txt)-self.get_nwords()

    def get_info_as_markup(self, spaces=True):
        s="""
        <b>Lines:</b> <i>%s</i> line(s).
        <b>Words:</b> <i>%s</i> word(s).
        <b>Chars:</b> <i>%s</i> char(s).
        """%(self.get_nlines(), self.get_nwords(),
self.get_nchars(spaces))
        return s
```

التصميم



الكود

```
class Window(gtk.Window):  
  
    def __init__(self):  
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)  
        self.__init_comp()  
        self.wc=WordCounter()  
  
        self.show_all()  
  
    def __init_comp(self):  
        self.set_title("PyGWC")  
        self.set_position(gtk.WIN_POS_CENTER)  
        self.set_border_width(12)  
  
        mvbox=gtk.VBox(False, 0)
```

```

btnfile=gtk.Button("File")
btnfile.connect("clicked", self.select_file)
self.fileentry=gtk.Entry()
self.fileentry.set_editable(False)
hbox=gtk.HBox(False, 0)

hbox.pack_start(btnfile, True, True, 2)
hbox.pack_start(self.fileentry, True, True, 1)

mvbox.pack_start(hbox, True, True, 0)
mvbox.pack_start(gtk.HSeparator(), True, True, 2)

self.lblinfo=gtk.Label()
mvbox.pack_start(self.lblinfo, True, True, 0)

self.add(mvbox)

```

هنا نربط ال signal clicked الخاصة بال btnfile ب callback select_file المعرفة كالتالي

```

def select_file(self,w):

    sel = gtk.FileChooserDialog("Open..",
                                self,
                                gtk.FILE_CHOOSER_ACTION_OPEN,
                                (gtk.STOCK_CANCEL,
                                gtk.STOCK_OPEN, gtk.RESPONSE_OK)
                                )

    sel.set_default_response(gtk.RESPONSE_OK)

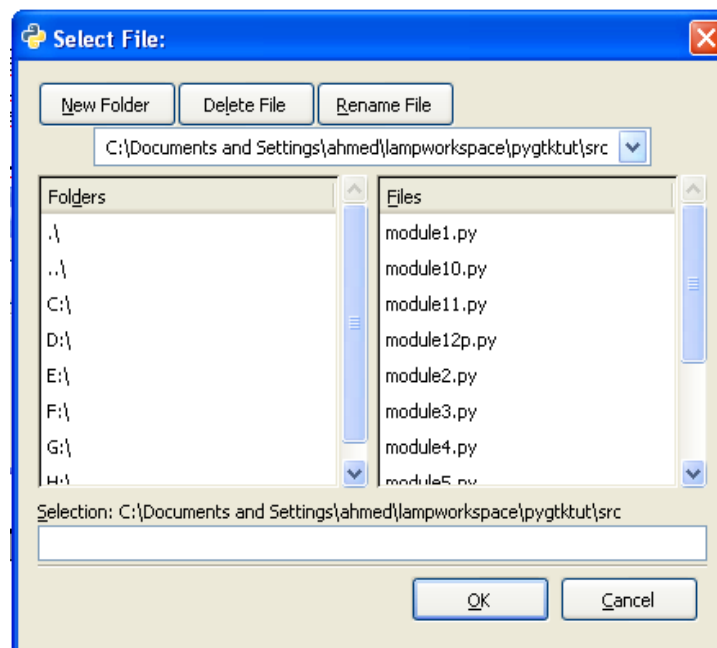
    res=sel.run()
    if res==gtk.RESPONSE_OK:
        self.wc.set_file(sel.get_filename())
        self.fileentry.set_text(sel.get_filename())
        self.lblinfo.set_markup(self.wc.get_info_as_markup(spaces=True))
    else:
        print "Dialog with RESPONSE_CANCEL!"

    sel.destroy()

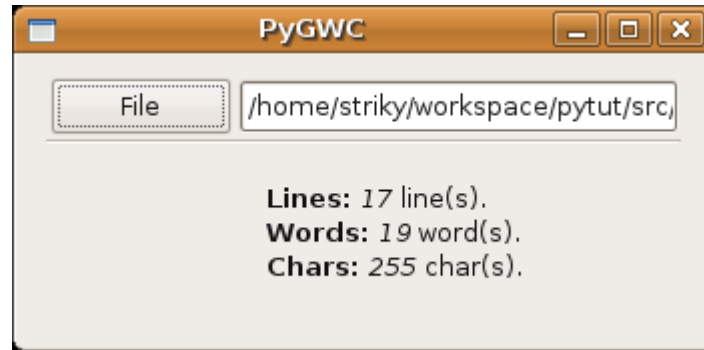
```

هنا بننشئ كائن من ال `FileChooserDialog` (لأختيار الملفات او الفولدرات) ونحدد العنوان بال constructor
واخبرناه بعنوان النافذة ، ونوع فعلها (`ACTION_OPEN` لفتح الملفات وليس لل حفظ) وانواع ردود
الفعل العائدة منها وهي `OK`, `CANCEL` وايضا ال `stocks`

```
sel = gtk.FileChooserDialog("Open",  
                             self,  
                             gtk.FILE_CHOOSER_ACTION_OPEN,  
                             (gtk.STOCK_CANCEL,  
gtk.RESPONSE_CANCEL,  
                             gtk.STOCK_OPEN, gtk.RESPONSE_OK)  
                             )
```



يتم تدمير ال sel ويتم تحديد اسم الفايل الخاص بال WordCounter object
نعدل التكبست على ال fileentry لإسم الفايل
اخيرا نضع ال markup الناتج من ال wc.get_info_as_markup على ال lblinfo بإستخدام ال set_markup method



ملحوظة: لاتقم بإستخدام FileSelection (تم تعديل المثال ليستخدم FileChooserDialog)



قائمة منسدلة بتشمل مجموعة من الإختيارات ، فى مثالنا هنا مجموعة من اسماء التوزيعات

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__()
        self.__init_comp() #prepare components

        self.show_all()

    def __init_comp(self):

        self.vbox=gtk.VBox(False, 2)

        entries = ["Slackware", "Ubuntu", "Mandriva", "Debian"]
        cbo=gtk.combo_box_new_text()
        map(cbo.append_text, entries)

        cbo.connect("changed", self._on_cboselection_changed)
        self.vbox.pack_start(cbo, False, False)

        self.add(self.vbox)

    def _on_cboselection_changed(self, widget, *args):
        print widget.get_active_text()
```

الطريقة السريعة هي انشاءها باستخدام `combo_box_new_text` (بتجهز الكومبو بتكست رندرر)
استدعى الطريقة `append_text` على كل المدخلات `entries` باستخدام `map`

اربط ال `changed signal` ب `on_cboselection_changed_`
وللحصول على العنصر النشط فى القائمة استخدام `get_active_text()`

Menus



هنبشئ فى المئال نافذة بشريط قوائم فيه قائمة واحدة File وبئشمل 3 عناصر open, save, quit

```
import gtk

class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__()
        self.__init_comp() #prepare components

        self.show_all()

    def __init_comp(self):

        self.vbox=gtk.VBox(False, 2)

        mbar=gtk.MenuBar()
        self.vbox.pack_start(mbar, False, False, 2)

        self.connect("delete_event", lambda w,e: gtk.main_quit())

        file_item=gtk.MenuItem("File")
        fmenu=gtk.Menu()
        file_item.set_submenu(fmenu)

        open_item = gtk.MenuItem("Open")
        save_item = gtk.MenuItem("Save")
        quit_item = gtk.MenuItem("Quit")

        map(fmenu.append, [open_item,save_item, quit_item])

        open_item.connect("activate", self._on_item_activated, "Open")
        save_item.connect("activate", self._on_item_activated, "Save")
        quit_item.connect("activate", self._on_item_activated, "Quit")
```



```
mbar.append(file_item)

self.add(self.vbox)
```

خطوات متتالية لإنشاء القوائم
1- شريط القوائم من الصف MenuBar وإضافته للصندوق الرأسى

```
mbar=gtk.MenuBar()
self.vbox.pack_start(mbar, False, False, 2)
```

2- العنصر الذى سيحوى القائمة وهنا هو File وينشئ من الصف MenuItem

```
file_item=gtk.MenuItem("File")
```

3- القائمة من الصف Menu

```
fmenu=gtk.Menu()
```

4- ضم القائمة

```
file_item.set_submenu(fmenu)
```

5- إنشاء عناصر القائمة وإضافتهم باستخدام الطريقة append

```
open_item = gtk.MenuItem("Open")
save_item = gtk.MenuItem("Save")
quit_item = gtk.MenuItem("Quit")
```

```
map(fmenu.append, [open_item, save_item, gtk.SeparatorMenuItem(),
quit_item])
```

لاحظ اننا انشأنا فاصل افقى SeparatorMenuItem ليفصل بين عنصرى فتح وحفظ وعنصر اغلاق

6- ربط ال activate signal بال callback المناسبة

```
open_item.connect("activate", self._on_item_activated, "Open")
save_item.connect("activate", self._on_item_activated, "Save")
quit_item.connect("activate", self._on_item_activated, "Quit")
```

التي عرفناها كالتالى مثلا

```
def _on_item_activated(self, w, action):  
    print "Calling %s"%action
```

7- اخيرا ضم العنصر الذى يحوى القائمة الى شريط القوائم

```
mbar.append(file_item)
```

يوجد طريقة اسهل وهى بإستخدام ال UIManager توفر عليك الكثير من الكتابة

Gladizer

تقدر تصمم واجهات ممتازة بإستخدام glade وبدون الحاجة لكتابة الكثير من الأكواد فيما يتعلق بشكل الواجهة بالإستعانة ب gladizer ، ملف ال glade. الواجهة الممثلة بصورة XML لاتعلم بايثون عنها شئ سوا انها ملف XML فيجب ان تخبرها بأنها وصف لواجهة برنامج ما وانك تريد ان تستخدمها فى تطبيقك. تستطيع القيام بذلك يدويا، او بإستخدام gladizer او اى اداة مشابهة مثل glade2py, glade, الخ

1- افتح glade وانشئ نافذة وقم بحفظ الملف ب hello.glade
هيكون الكود مشابه للتالى

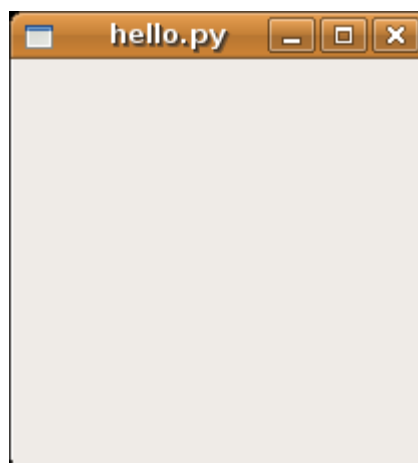
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE glade-interface SYSTEM "glade-2.0.dtd">
<!--Generated with glade3 3.4.5 on Tue Dec 2 06:07:51 2008 -->
<glade-interface>
  <widget class="GtkWindow" id="window1">
    <child>
      <placeholder/>
    </child>
  </widget>
</glade-interface>
```

استدعى gladizer ليقوم بالربط

```
striky@striky-desktop:~/Desktop$ gladizer.py -f hello.glade -p Python > hello.py
```

قم بالتنفيذ

```
striky@striky-desktop:~/Desktop$ python hello.py
```



تصفح الكود الناتج من جلاذيزر ستجده مشابه للتالى

```
#!/bin/python

##CODE GENERATED by gladizer 1.2

import pygtk
pygtk.require('2.0')

import gtk, gtk.glade

class MainWindow(object):

    def __init__(self):

        #Widget tree..
        self.wTree=gtk.glade.XML('hello.glade')

        #connect signals and handlers.
        self.wTree.signal_autoconnect(self)

        self._window1=self.wTree.get_widget('window1')
        self._window1.show()

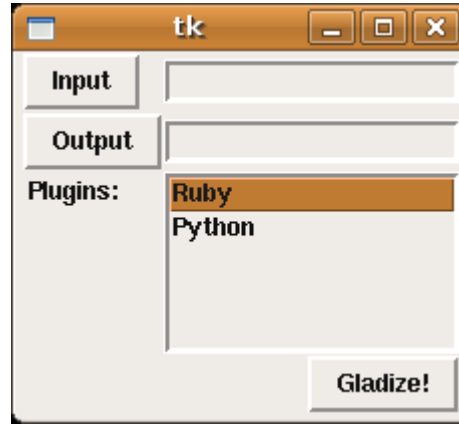
# run main loop

def main():
    mainwindow = MainWindow()
    #mainwindow.window1.show()
    gtk.main()

if __name__ == "__main__":
    main()
```

وبعد ذلك قم بإضافة ال callbacks كما تحب (:

تستطيع ايضا استخدام gladizer بواجهة رسومية من خلال gladizerguitk



صفحة gladizer

[/http://sourceforge.net/projects/gladizer](http://sourceforge.net/projects/gladizer)

ملحوظة هناك بعض الأمثلة محسنة من <http://pygtk.org/pygtk2tutorial/index.html> وهذا الفصل ليس إلا مقدمة لعالم GTK الرائع.

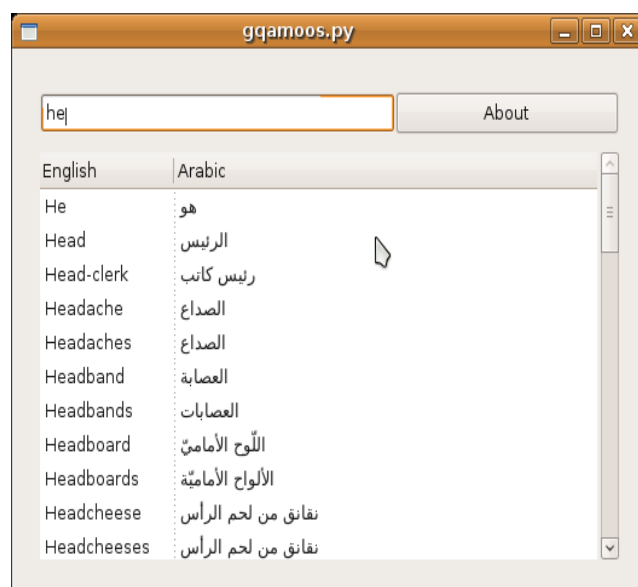
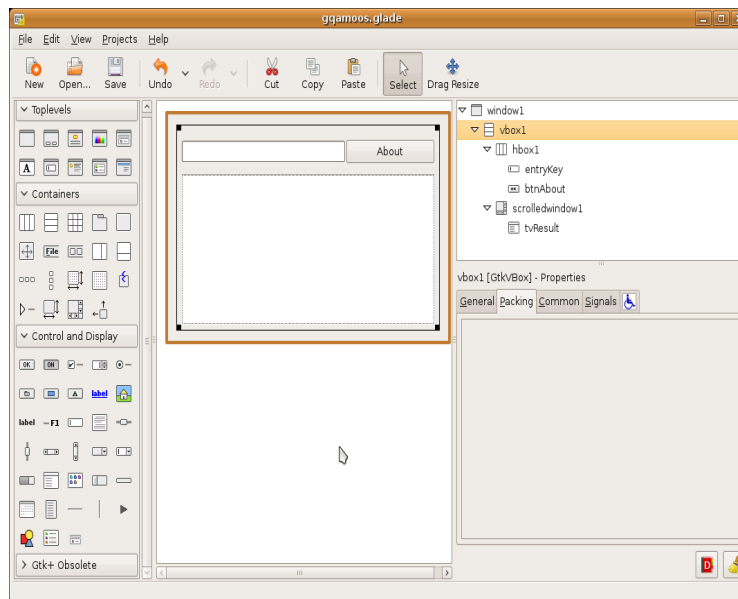
Rad with Glade (Gqamoos)

هنا شرح مثال على كيفية استخدام Glade مع PyGTK و Gladizer لإنشاء برنامج قاموس يعتمد على wordlist مع عرب ايز

1- حمل ال wordlist

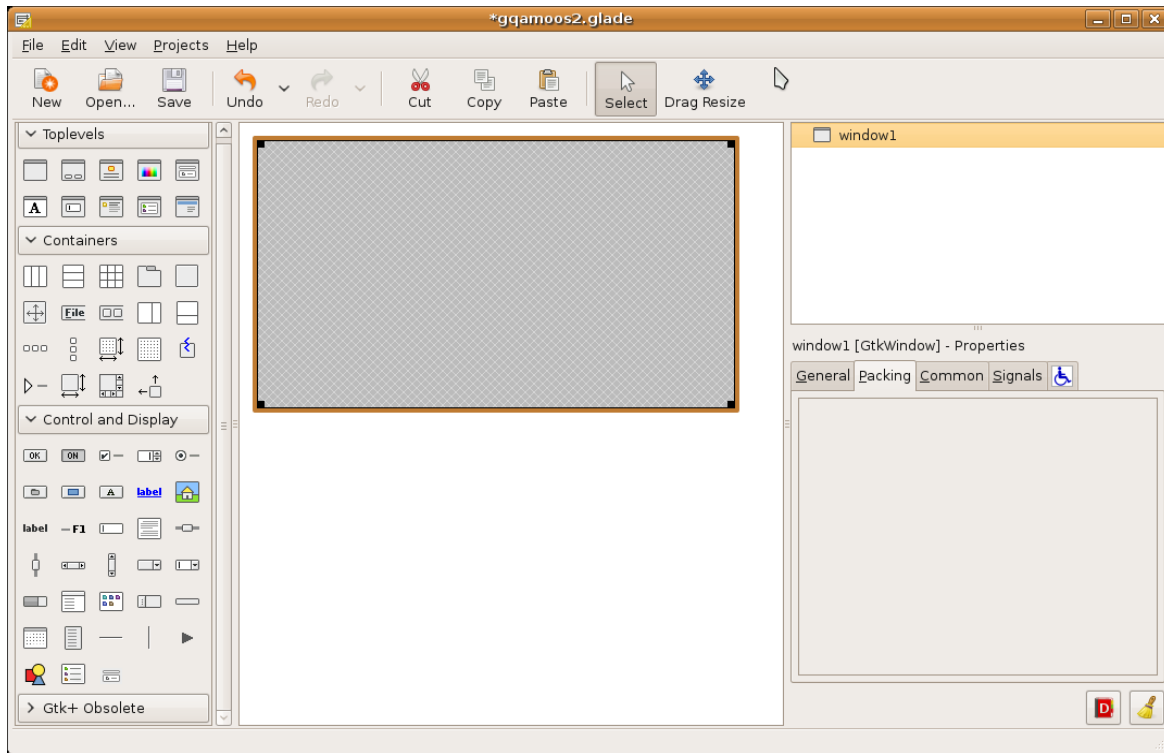
<http://www.arabeyes.org/project.php?proj=Wordlist>

2- افتح glade وانشئ ملف واحفظه ب gqamoos

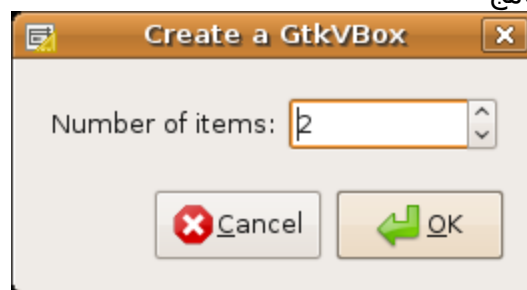


لاحظ الواجهة مقسمة للتالى

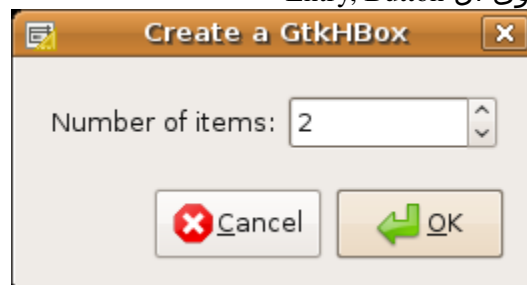
نافذة



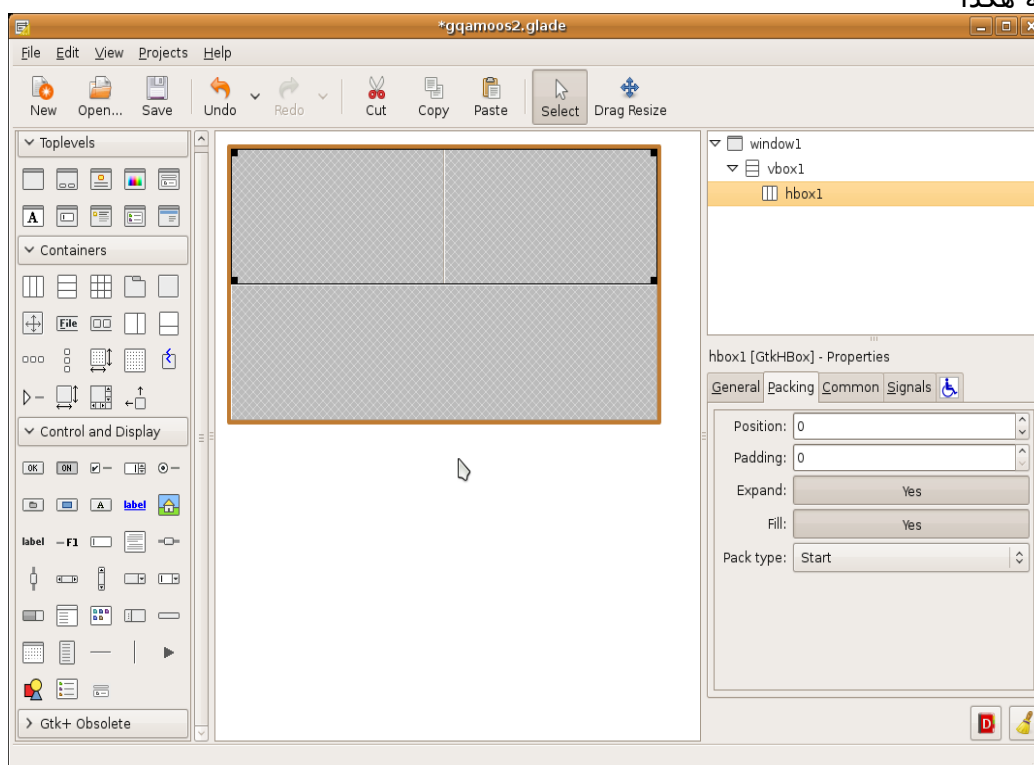
قم بإضافة VBox من اللوحة (لصفين) على النافذة ليشمل القسم الأعلى TextEntry, Button
ال TextEntry للكلمة المراد البحث عنها
ال Button لعرض رسالة حول البرنامج



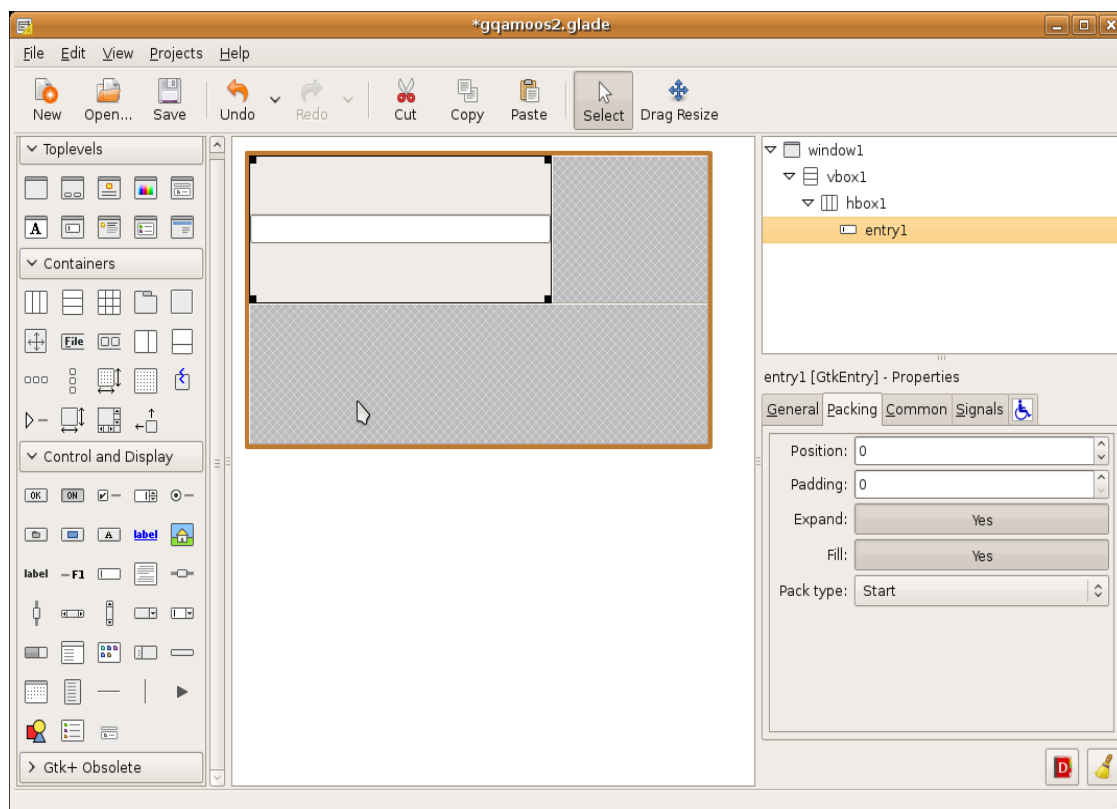
قم بإضافة صندوق افقي Hbox ليحوى ال Entry, Button



لتصبح الواجهة هكذا

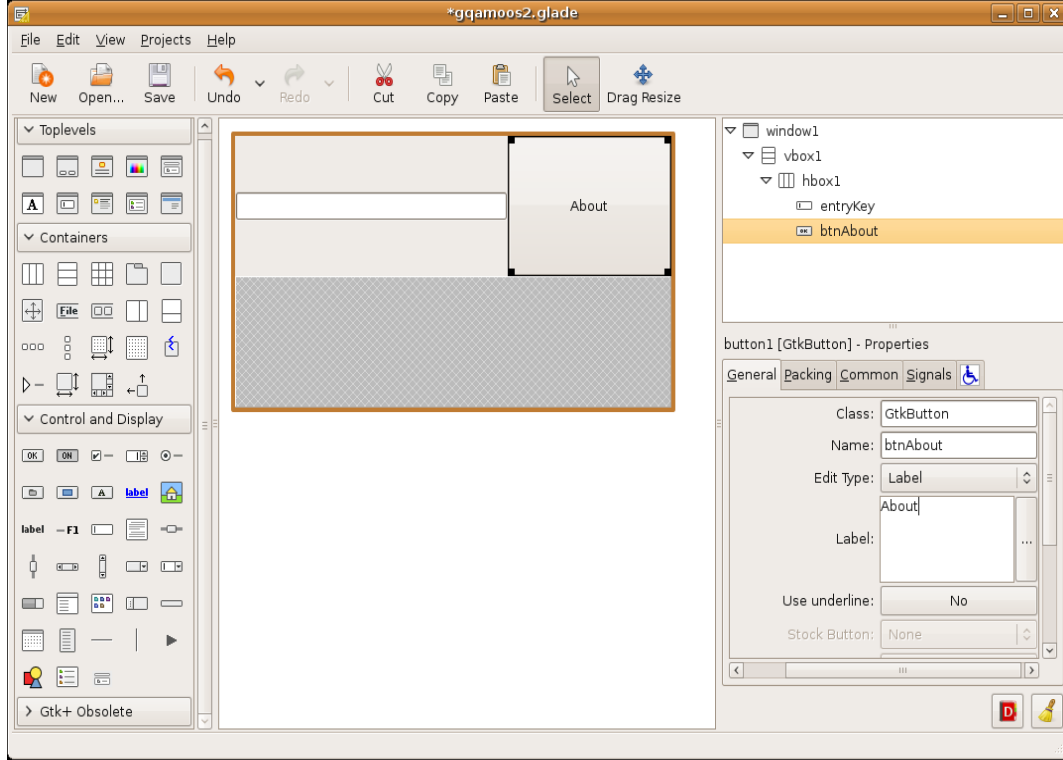


قم بإضافة ال Text Entry

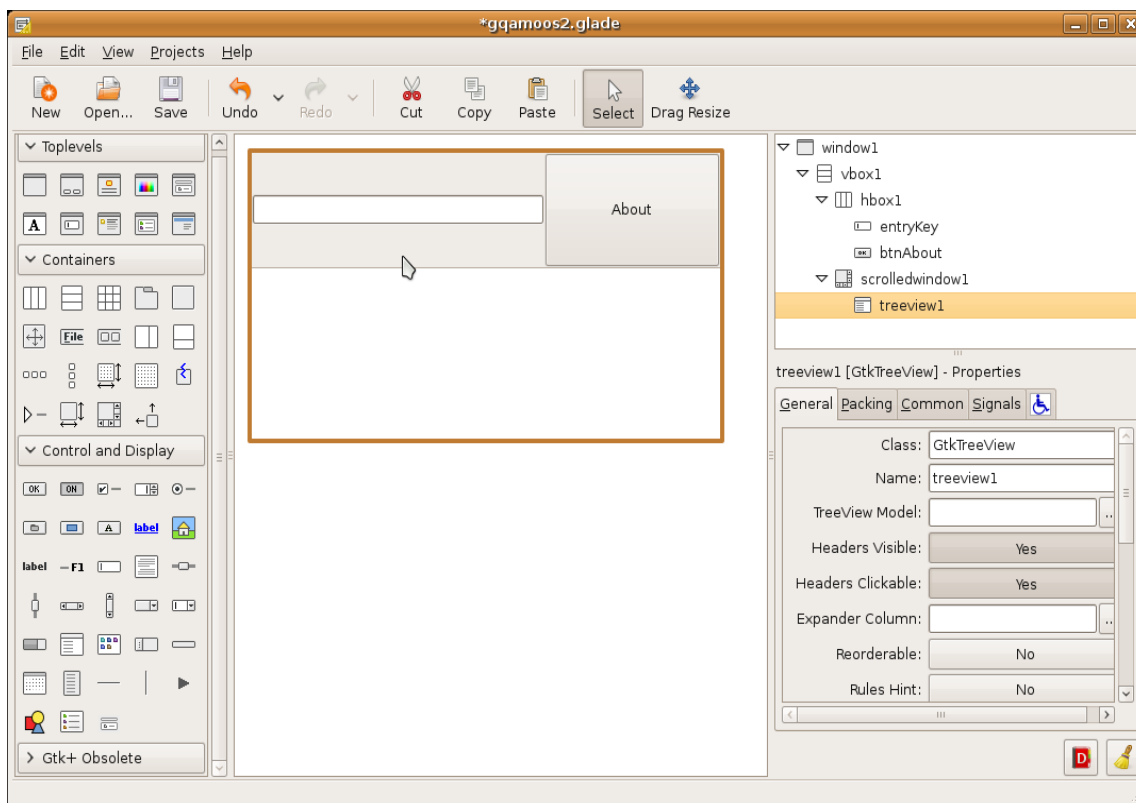


من قسم ال General بأسفل اليمين قم بتغيير اسم الويدجت ل entryKey

قم بإضافة زر Button من اللوحة على يمين ال Entry



فى القسم الأسفل اضع ScrolledWindow ليتيح لنا استخدام الـ "Scrollbars" على الـ TreeView
قم بسحب وإضافة الـ TreeView على الـ ScrolledWindow لتصبح الواجهة هكذا

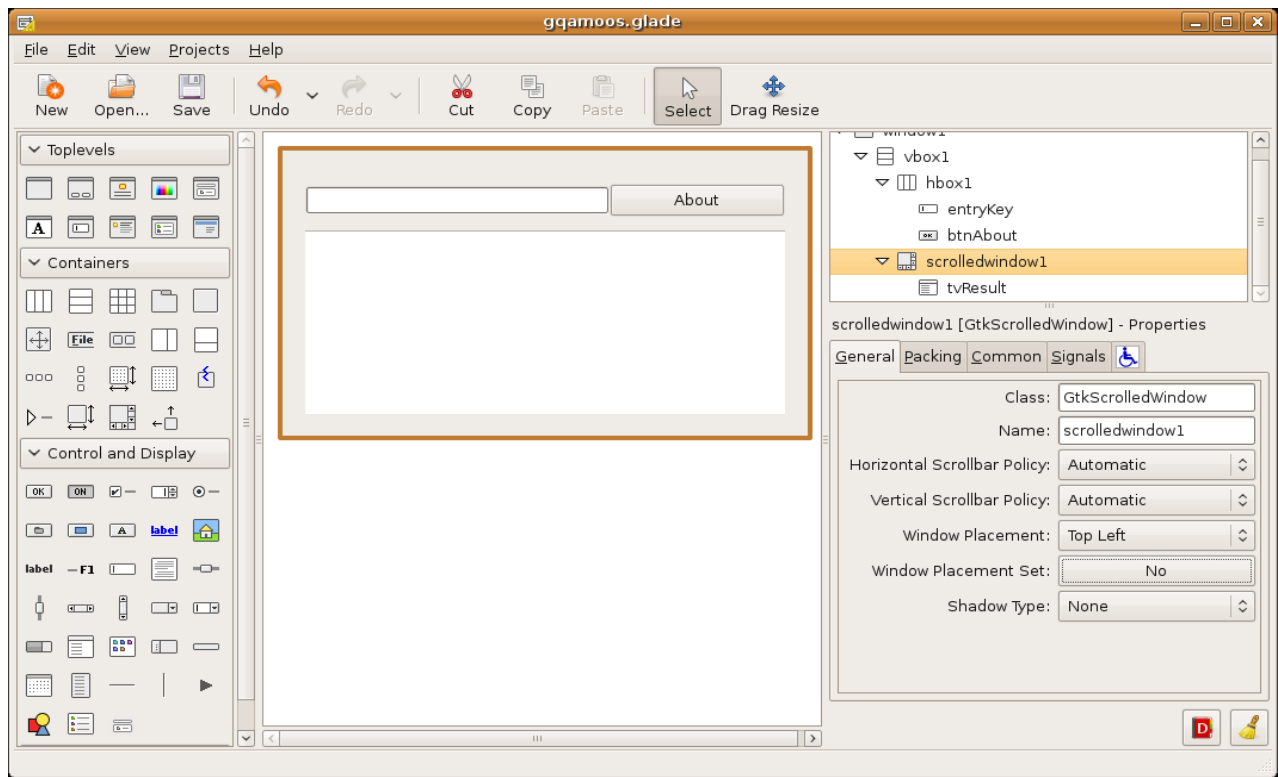


الآن سنقوم ببعض التعديلات على الخواص للويدجتس
قم بتعديل ال Label لل btnAbout إلى About
ومن قسم ال signals قم بإضافة on_btnAbout_clicked signal لل
قم بتغيير ال Expand, Fill لل Hbox الى القيم No, No

ولل entryKey قم بإضافة on_entryKey_changed signal "ستجدها تحت فرع ال GtkEditable"

وايضا قم بتعديل اسم ال TreeView الى tvResult

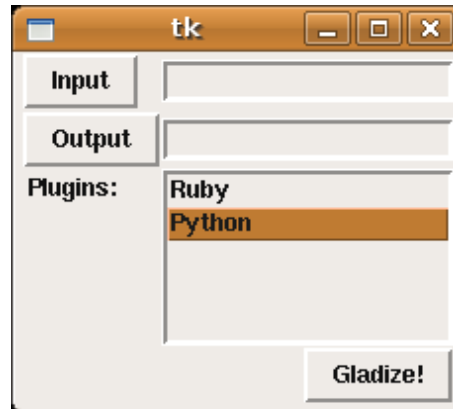
لتصبح الواجهة هكذا



تمام؟ كذا انهينا التصميم
قم بتشغيل gladizer على ملف gqamoos.glade ليكون الخرج هو gqamoos.py

```
striky@striky-desktop:~$ gladizer.py --file gqamoos.glade -p Python > gqamoos.py
```

او عن طريق gladizerguitk.py



اخيرا قم بتحرير الملف المشابه للتالى

```
#!/bin/python

##CODE GENERATED by gladizer 1.2

import pygtk
pygtk.require('2.0')

import gtk, gtk.glade

class MainWindow(object):

    def __init__(self):

        #Widget tree..
        self.wTree=gtk.glade.XML('gqamoos.glade')

        #connect signals and handlers.
        self.wTree.signal_autoconnect(self)

        self._window1=self.wTree.get_widget('window1')
        self._vbox1=self.wTree.get_widget('vbox1')
        self._hbox1=self.wTree.get_widget('hbox1')
        self._entryKey=self.wTree.get_widget('entryKey')
        self._btnAbout=self.wTree.get_widget('btnAbout')
```

```

        self._tvResult=self.wTree.get_widget('tvResult')
        self._window1.show()

    def on_entryKey_changed(self, widget, *args):
        pass

    def on_btnAbout_clicked(self, widget, *args):
        pass

# run main loop

def main():
    mainwindow = MainWindow()
    #mainwindow.window1.show()
    gtk.main()

if __name__ == "__main__":
    main()

```

وذلك بإضافة ال `looker service` للبحث في ال `wordlist`

```
import looker
```

وإضافة الوصول لأنواع البيانات الخاصة ب `gobject` مثل `TYPE_STRING` بإضافة ال `gobject`

```
import gobject
```

وإنشاء متغير داخلي من النوع `looker.WordListLooker`

```
self._wc=looker.WordsListLooker()
```

تجهيز ال `TreeView` باستخدام الطريقة `init_tv_`

```
self._init_tv()
```

الطريقة `init_tv_` معرفة كالتالي

```

def _init_tv(self):
    encell=gtk.CellRendererText()
    encol=gtk.TreeViewColumn("English", encell, text=0)
    arcell=gtk.CellRendererText()
    arcol=gtk.TreeViewColumn("Arabic", arcell, text=1)
    map(self._tvResult.append_column,[encol, arcol])

```

- سيتم ال TreeView باسماء الأعمدة ونوعهم وإضافتهم وذلك من خلال اضافة الأعمدة encol, arcol وهم من النوع gtk.TreeViewColumn وبأخذون معاملات كالتالى
- 1- الهيدر "النص الظاهر"
 - 2- ال CellRenderer وهو المسئول عن "رندرة" الخلية (مثل encell وهو "مرندر من النوع النصى" لعرض البيانات النصية)
 - 3- الترتيب فى العرض من ال store تابع الطريقة _get_model()

```
def _get_model(self):
    lstore=gtk.ListStore(gobject.TYPE_STRING, gobject.TYPE_STRING)
    self._wc.searchKey=self._entryKey.get_text()
    for row in self._wc.lookup():
        iter=lstore.append([row[0], row[1]])

    return lstore
```

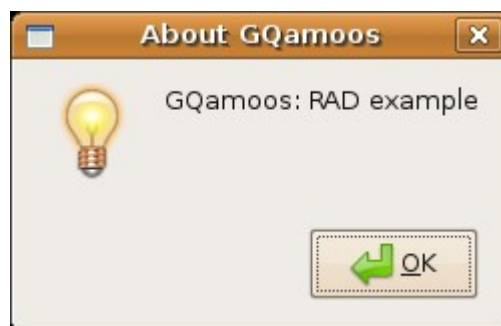
هنا نقوم بتحديد نوع البيانات المخزنة. تستطيع استخدام str بدلا من gobject.TYPE_STRING
 نحصل على النص الظاهر على ال entryKey من خلال الطريقة get_text ونقوم بإسناده لل searchKey property
 لل self._wc "الأوبجكت المسئول عن استغلال ال "looker service"
 نقوم بإضافة نتائج البحث من خلال lstore.append فى داخل list ونقوم بإعادة ذلك ال store
 نقوم بربط التغيير فى النص الظاهر على ال entryKey بتعديل ال model الخاص بال TreeView لتغيير نتائج البحث
 مع كل تغيير وذلك بتعديل on_entryKey_changed callback

```
def on_entryKey_changed(self, widget, *args):
    self._tvResult.set_model(self._get_model())
```

نقوم بتحديد ال model باستخدام الطريقة set_model الخاصة بال tvResult "ال TreeView"

تحديد رسالة حول البرنامج قم بتعديل ال on_btnAbout_clicked المسئولة عن معالجة ال clicked signal للزر btnAbout

```
def on_btnAbout_clicked(self, widget, *args):
    md=gtk.MessageDialog(self._window1,
    gtk.DIALOG_DESTROY_WITH_PARENT, gtk.MESSAGE_INFO, gtk.BUTTONS_OK,
    "GQamoos: RAD example")
    md.set_title("About GQamoos")
    md.run()
    md.destroy()
```



قم بتشغيل البرنامج



ستلاحظ اننا لم نحدد عنوان النافذة..تستطيع تعديل ذلك اما باستخدام الطريقة `set_title` لل `self._window1` او من خلال ملف `glade` بتعديل Title لل `window1` إلى `Gqamoos`



```

#!/bin/python

##CODE GENERATED by gladizer 1.2

import pygtk
pygtk.require('2.0')

import lookout

import GObject
import gtk, gtk.glade

class MainWindow(object):

    def __init__(self):

        #Widget tree..
        self.wTree=gtk.glade.XML('ggamoos.glade')

        #connect signals and handlers.
        self.wTree.signal_autoconnect(self)

        self._window1=self.wTree.get_widget('window1')
        self._vbox1=self.wTree.get_widget('vbox1')
        self._hbox1=self.wTree.get_widget('hbox1')
        self._entryKey=self.wTree.get_widget('entryKey')
        self._btnAbout=self.wTree.get_widget('btnAbout')
        self._tvResult=self.wTree.get_widget('tvResult')
        self._window1.show()
        self._wc=lookout.WordsListLooker()

        self._init_tv()

    def _init_tv(self):
        encell=gtk.CellRendererText()
        encol=gtk.TreeViewColumn("English", encell, text=0)
        arcell=gtk.CellRendererText()
        arcol=gtk.TreeViewColumn("Arabic", arcell, text=1)
        map(self._tvResult.append_column,[encol, arcol])

    def _get_model(self):
        lstore=gtk.ListStore(str, str) #GObject.TYPE_STRING,
        GObject.TYPE_STRING
        self._wc.searchKey=self._entryKey.get_text()
        for row in self._wc.lookup():
            iter=lstore.append([row[0], row[1]])

```



```

        return lstore

    def on_entryKey_changed(self, widget, *args):
        self._tvResult.set_model(self._get_model())

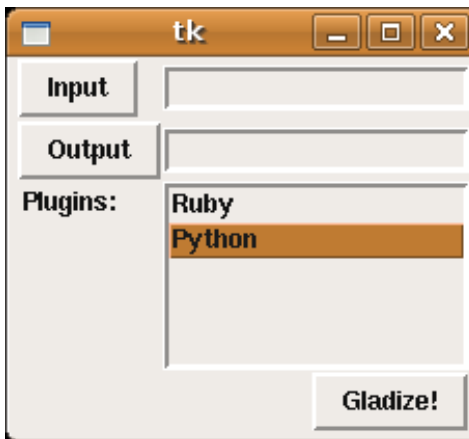
    def on_btnAbout_clicked(self, widget, *args):
        md=gtk.MessageDialog(self._window1,
gtk.DIALOG_DESTROY_WITH_PARENT, gtk.MESSAGE_INFO, gtk.BUTTONS_OK,
"GQamoos: RAD example")
        md.set_title("About GQamoos")
        md.run()
        md.destroy()
        #pass

# run main loop

def main():
    mainwindow = MainWindow()
    #mainwindow.window1.show()
    gtk.main()
if __name__ == "__main__":
    main()

```

HowTo GladizerguiTK ?



سكربت جلاذيزر تم التعرض ليه بالأعلى وتم فيه كتابة واجهة باستخدام tkInter
لاحظ الواجهة مقسمة لشبكة (صفوف واعمدة)
الصف الأول يشمل عمودين زر input و entry
الصف الثاني يشمل عمودين زر output و entry
الصف الثالث يشمل عمودين label و listbox تحوى عناصر ال plugins المتاحة
الصف الرابع زر Gladize

1- قم بإستدعاء الوحدات اللازمة

```
from Tkinter import *  
import tkFileDialog as tfd  
import tkMessageBox as tmd
```

كل مكونات Tkinter و وحدة tkFileDialog و tkMessageBox
ننشئ صف جديد يشتق ال Frame class

```
class GladizerWindow(Frame):  
  
    def __init__(self, master=None):  
  
        Frame.__init__(self, master)  
  
        self._inputfile=StringVar()  
        self._outputfile=StringVar()  
        self._usedplugin=None  
  
        self._init_comps()
```

لاحظ ال self._inputfile, self._outputfile هم objects من الصف StringVar () الذى سنقوم بربطه بال مسارات المطلوبة
استدعاءنا للطريقة __init_comps لإنشاء الواجهة التى ذكرناها بالأعلى
ويتم التعامل مع القيمة المخزنة باستخدام الطرق get, set

```
def _init_comps(self):

inp=Button(self, text="Input", command=self._getinputfile)
inp.grid(row=0, sticky=W+N)

self.inpentry=Entry(self, textvariable=self._inputfile)
self.inpentry.grid(row=0, column=1, sticky=E)

out=Button(self, text="Output", command=self._getoutputfile)
out.grid(row=1, sticky=W+N)

self.outentry=Entry(self, textvariable=self._outputfile)
self.outentry.grid(row=1, column=1, sticky=E)

Label(self, text="Plugins:").grid(row=2, column=0, sticky=N+W)
#plugins list.
self.lbplugins=Listbox(self, height=5, selectmode=BROWSE, relief=SUNKEN)
self.lbplugins.grid(row=2, column=1, columnspan=3, sticky=E+S)
if len(getplugins())>=1:
    for plugin in getplugins():
        self.lbplugins.insert(0, plugin)

self.lbplugins.select_set(0)

start=Button(self, text="Gladize!", command=self._gladize).grid(row=3, columnspan=2, sticky=E+S)
self.grid()
```

لإنشاء زر من الصف Button

```
inp=Button(self, text="Input", command=self._getinputfile)
```

يأخذ المشيد مجموعة من المعاملات
text هى النص الطاهر
ال command هى ال callback التى سيتم تنفيذها
الطريقة grid لتحديد مكانة وهى تأخذ عدة معاملات لتحديد الصف والعمود والإزاحة الخ مثل row و column,
columnspan
لإنشاء Entry

```
self.inpentry=Entry(self, textvariable=self._inputfile)
```

لاحظ ربطنا لل textvariable بالمتغير self._inputfile بحيث ان يكون التغير ديناميكي

لإنشاء ال Listbox

```
self.lbplugins=Listbox(self, height=5, selectmode=BROWSE, relief=SUNKEN)
```

نحددها على الشبكة (الصف والعمود. الخ)

```
self.lbplugins.grid(row=2, column=1, columnspan=3, sticky=E+S)
```

نضيف العناصر لها بإستخدام الطريقة insert وتأخذ اول معامل المركز الذى سيتم الإدخال عنده والثانى هو القيمة المدخلة

```
if len(getplugins())>=1:
    for plugin in getplugins():
        self.lbplugins.insert(0, plugin)
```

نحدد الإختيار الافتراضى بإستخدام الطريقة select_set التى تأخذ معامل قيمته المركز

```
self.lbplugins.select_set(0)
```

تأخذ عدة معاملات مثل الارتفاع ونوع الإختيار ولاحظ ايضا relief (التي جعلنا قيمتها SUNKEN) لجعلها غائصة

ال callbacks للزرين input, output

```
def _getinputfile(self, *args):

    self._inputfile.set(tfd.askopenfilename())
    #self.inpentry.delete(0, END)
    #self.inpentry.insert(0, self._inputfile)

def _getoutputfile(self, *args):

    self._outputfile.set(tfd.asksaveasfilename())
    #self.outentry.delete(0, END)
    #self.outentry.insert(0, self._outputfile)
```

تقدم لنا الوحدة tkinter الدوال اللازمة للتعامل مع الملفات (اختيار ملف مثلا) نقوم بإستخدام tfd.askopenfilename() للحصول على مسار الملف المطلوب فتحه ونخزن قيمته بإستخدام الطريقة set الخاصة بال StringVar object وبنفس الكيفية نحصل على مسار ملف الحفظ بإستخدام tfd.asksaveasfilename()

تقدم لنا الوحدة tkinter الدوال اللازمة للتعامل مع الرسائل مثل showerror, showinfo, ... etc

```
def showerror(self, msg):
    tmd.showerror("Error", msg)
```

لعرض رسالة خطأ showerror

```
tmd.showinfo("Gladized", "%s gladized successfully!"%self._outputfile.get())
```

لعرض رسالة معلومات showinfo
لاحظ الطريقة get الخاصة بال StringVar للحصول على القيمة المخزنة

```
def main():

    root=Tk()          #Master
    gw=GladizerWindow(root) #GladizerWindow
    root.mainloop()    #Entering the mainloop

if __name__=="__main__":
    main()
```

اخيرا نعرض النافذة وندخل التطبيق

```
root=Tk()          #Master
gw=GladizerWindow(root) #GladizerWindow
```

ونبدأ دائرة الأحداث بإستخدام الطريقة mainloop

```
root.mainloop()    #Entering the mainloop
```

*تطبيق: قم بإنشاء واجهة رسومية لسكرتير replacer

Qt

ماذا عن Qt ؟

تمت ترجمة الجزئية الخاصة بها من ZetCode وستجدها متوفرة للتصفح اونلين او تحميل ملف ال pdf سيتم توزيعها مع الكتاب وللمزيد من التعمق تستطيع جزئية تصميم الواجهات الرسومية بإستخدام Qt/Ruby (لن تحتاج منك الكثير من الجهد) وايضا قراءة الأمثلة المرفقة مع PyQt4

wxPython

تم الغاء ضمها لهذا الإصدار (ربما الإصدار القادم)

tKinter

ليست هناك نية للتعرض لها عبر مثال Howto GladizerGUItk (يفضل عدم استخدامها لأي مشاريع مستقبلية GTK او wxPython او Qt) هم افضل الاختيارات المتاحة لك

ماذا عن Py3K ؟
سيبقى الكتاب عن Python 2.5 وربما فى التحديث القادم يتم وضع اهم التغييرات
تستطيع الإطلاع عليها من هنا
<http://docs.python.org/3.0/whatsnew/3.0.html>

الخاتمة

لقد فعلتها!

مبروك انهاءك الكتاب!

الى اين ؟

تستطيع الإتجاه لمجالات عديدة فبايثون لغة عامة اي تستخدم فى كثير من المجالات وقد وفر لك الكتاب اساس جيد لذلك المشوار مثل التعامل مع بايثون و معالجة البيانات (XML, INI, Databases,.. etc) وايضا كيفية انشاء واجهات (استخدام GTK و Glade) وايضا تعلمت قدرا جيدا من اساسيات انشاء تطبيقات الشبكات تم توفير الكثير من المصادر حول بايثون وانشاء التطبيقات الرسومية منها

[Mono IronPython WinForms Tutorial](#)

[PyGTK Tutorial](#)

--للتعامل مع PyQt4 قم بفتح الملف المرفق مع الكتاب الذي يشمل ترجمة الدورة الشهيرة من ZetCode فى استخدام PyQt4 وايضا لاغنى عن الأمثلة المرفقة مع حزمة PyQt4 .. تستطيع الحصول عليها من هنا ايضا <http://ojuba.org/wiki/doku.php/docs/pyqt4>

اتجه الى صفحة الوثائق <http://python.org/doc>

مصادر اخرى

ارشح لك الكتب التالية لإستمرار المشوار

Learning Python 3rd

كتاب اقل مايقال عنه انه ممتاز يحقق لك الرسوخ فى اللغة بطريقة لاتتخيلها

Programming Python, 3rd Edition

الجزء الثانى (المتقدم) ل Learning Python 3rd

يتناول برمجة النظام الملفات والخيوط والعمليات وايضا الواجهات الرسومية بإستخدام tKInter وعدة تطبيقات مكتملة كمحرر نصوص ومستعرض صور ولعبة وغيره وجزئية متعمقة فى الشبكات من ناحية ال client side وال server side وبعض التعرض لهياكل البيانات ومعالجة النصوص والكثير

Core Python Programming, 2nd

باختصار هو الكتاب الأفضل فى رأى كمرحلة تانية لكتاب Learning Python او تكمله ل Programming Python 3rd يناقش الكثير والكثير من الموضوعات البايثونية بدرجة متعمقة

Foundations of Agile Python Development

كتاب سيجعلك مبرمج بايثون افضل للأبد

Foundations of Python Network Programming

اذا حبيت ابدا برمجة الشبكات فذلك الكتاب هو اختيارك الأول ليس خاص بالشبكات فقط ولكن ايضا ببعض الأجزاء عن الويب وتعدد المهام والويب سرفسر وايضا جزئيات متعلقة ب POP, IMAP, SMTP

Wrox Beginning Python

كتاب رائع ياخذك للناحية العملية وانشاء تطبيقات مفيدة واهتمام مكثف بال XML وال Networking ولكن لاتأخذه للإستفادة من الأساسيات فكتاب Learning Python افضل كثيرا فى هذه الجزئية

Twisted Network Programming Essentials

الكتاب الوحيد الذى يغطى twisted بتعمق (هو حلك الأفضل لكتابة تطبيقات شبكات امنة وعالية الجودة بإستخدام twisted)

Manning wxPython in Action

الكتاب الخاص ب wxPython ومقسم لمقدمة عن wxPython ومميزاتها والمخطط الأساسى واساسيات انشاء الواجهات واستخدام متقدم كالطباعة وبعض الويدجات المتقدمة

الكتاب والأمثلة التى تأتى مع الوثائق هم خيارك الأول لعالم wxPython الرائع

Rapid GUI Programming with Python and Qt

الكتاب رقم 1 للتعامل مع Python, Qt مقسم ايضا لعدة اجزاء الأساسيات كمقدمة عامة وبعض التطبيقات الخفيفة وايضا استخدام Qt Designer واساسيات استخدام المخططات وقواعد البيانات والطباعة والأحداث الخ الخ واستخدام متقدم كتعدد الخيوط

لاتنس قسم المقالات الخاص ب Programming-Fr34ks

<http://programming-fr34ks.net/smf/articles-12/>

شكرا!!

ايها القارئ لإختيارك الكتاب لمشوارك مع بايثون

اهداء

اهدى هذا الكتاب الذى اتمنى ان اكون قد وفقت فى عرضه الى والدى وإلى جميع اصدقائى لن اذكر اسماء حتى لانسى احدا

ملحق 1: استخدام Py2EXE

Py2EXE لايفضل استخدامه سوى فى حالة التحزيم وعدم معرفة المستخدم ببايثون وبكل المتطلبات للبرنامج الخ ولكن ليس من اجل اخفاء الكود، على كل حال

(1) قم بتحميل وتثبيت

<http://www.py2exe.org>

(2) فى محررك المفضل اكتب مايشابه التالى

```
#!/bin/python
print "Programming Fr34ks r0x !"
```

(3) احفظ الملف وليك pf.py وقم بتجربته حتى لا يكون هناك اخطاء

(4) اكتب سكريبت التثبيت

```
#!/bin/python
from distutils.core import setup
import py2exe
setup(console=['pf.py']) #pf.py is our script name
```

(5) احفظ ملف التثبيت وشغله

```
%>python setup.py py2exe
```

(6) انقل لمجلد ال dist

(7) شغله

```
%>pf.exe
Programming Fr34ks r0x !
```

ملحق 2: محررات/بيئات تطوير/مصمّمات نوافذ

VIM

محرر قوى جدا ومشهور على نظم UNIX-LIKE اكثر (لايكفى كتاب للكلام عنه)

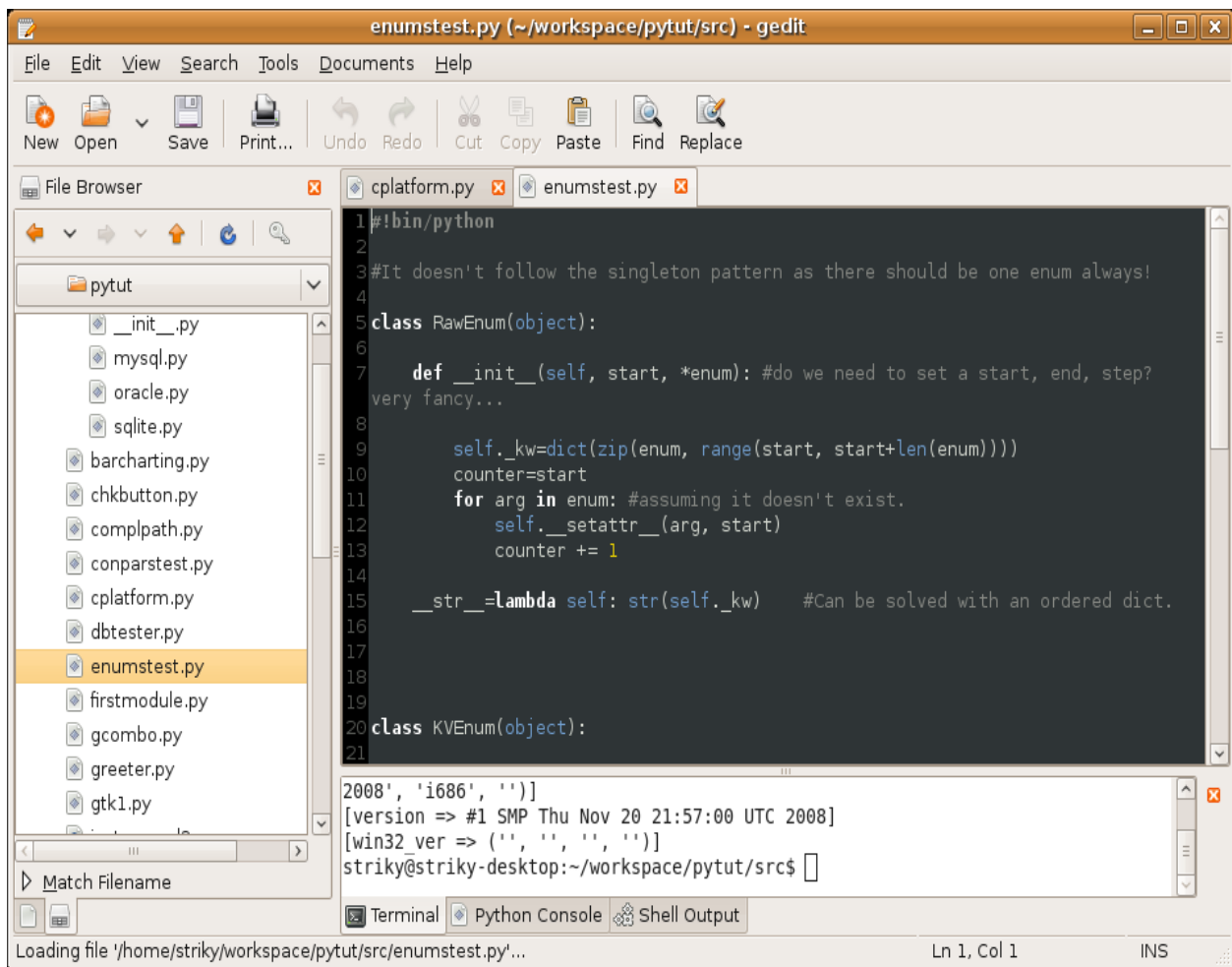
[http://en.wikipedia.org/wiki/Vim_\(text_editor\)](http://en.wikipedia.org/wiki/Vim_(text_editor))

افتح vimtutor لدروس تفاعليه مع vim ويوجد كتاب مجاني ومبسط عن استخدام vim

<http://www.swaroopch.com/notes/Vim>

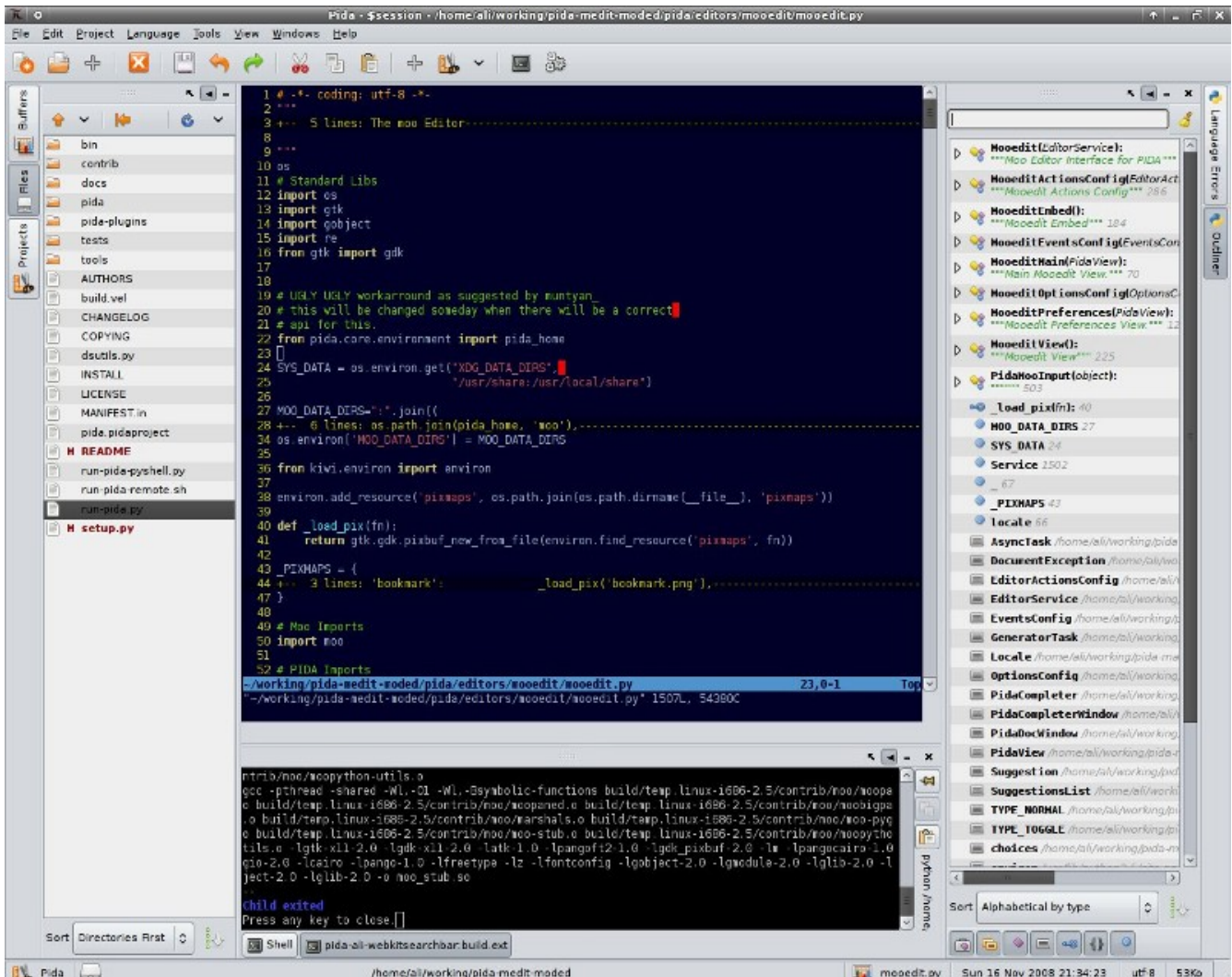
Gedit(Gnome Editor)

محرر افتراضى لواجهة Gnome يدعم العديد من الإضافات كتلوين الكود المصدري والإكمال التلقائى والترقيم وغير عدد كبير من الإضافات



pida

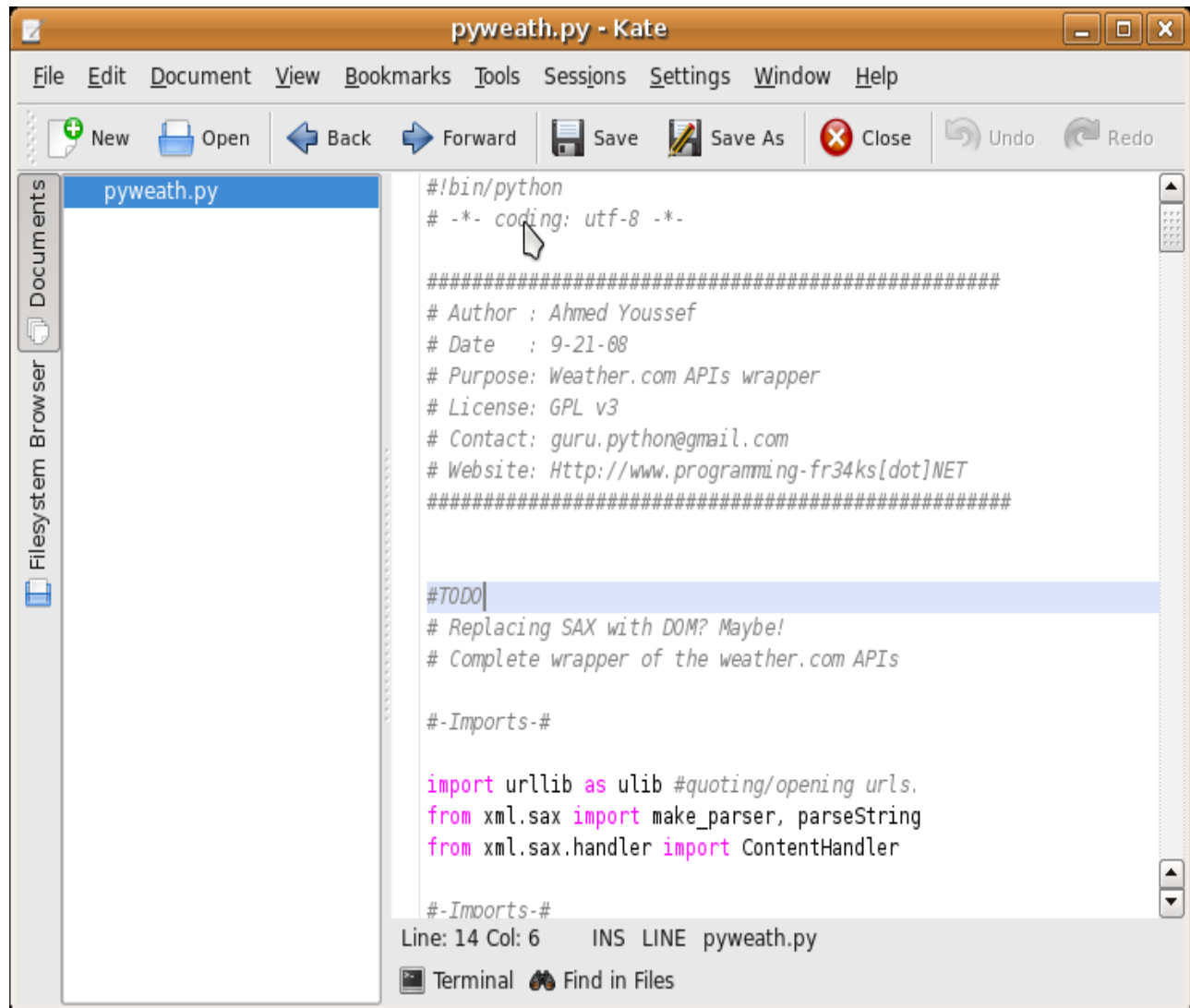
هى بيئة تطوير متقدمة مكتوبة ب pygtk ، تقدم تكامل مع vim و emacs ولها العديد من الإضافات مثل RegexToolkit التى تعرضنا لها



للمزيد من الصور: <http://pida.co.uk/screenshots>

Kate(KDE Advanced Editor)

محرر افتراضى لواجهة KDE مميزات مثل سابقه



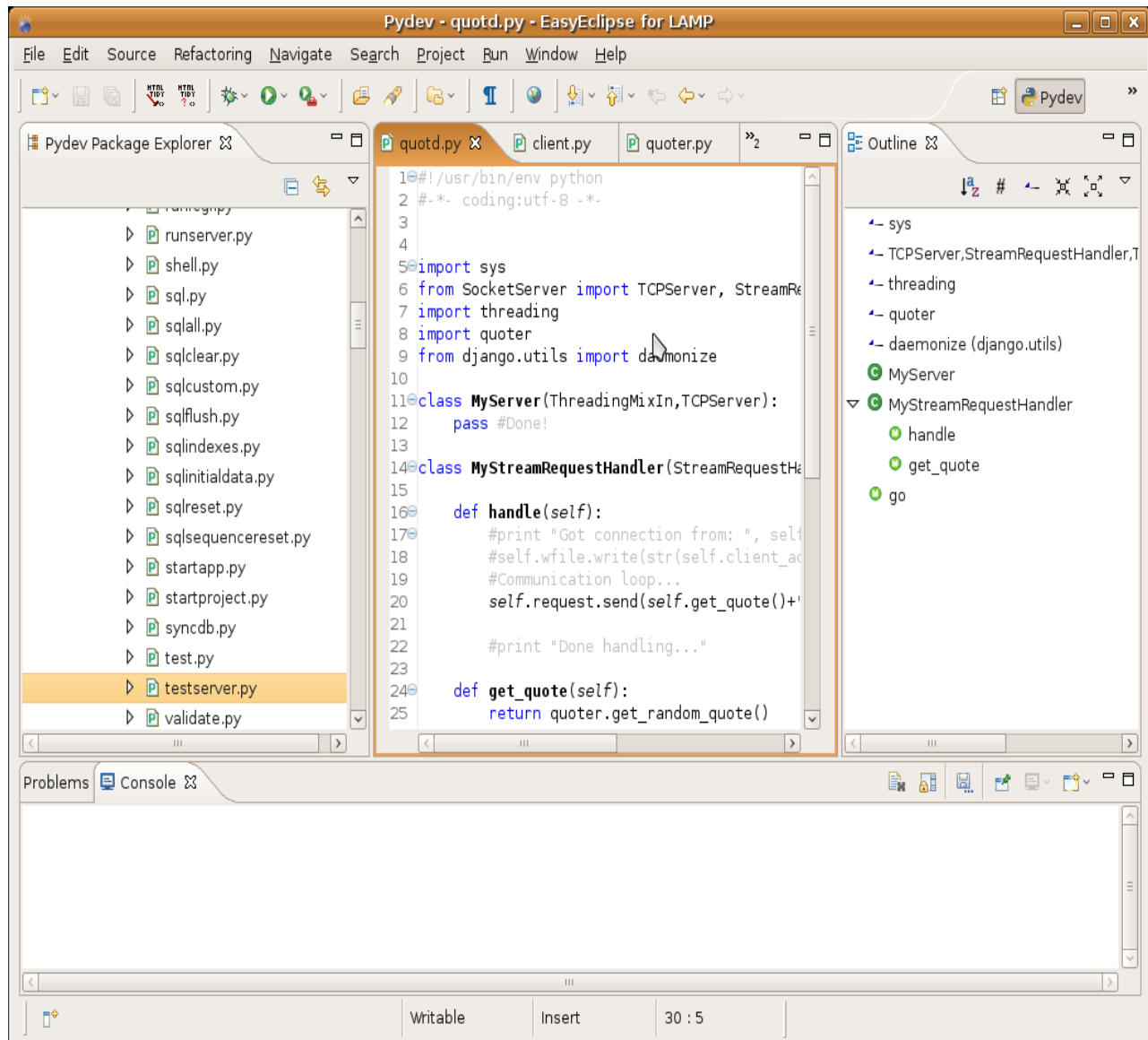
Komodo

IDE غير مجانية

OpenKomodo(KomodoEdit)

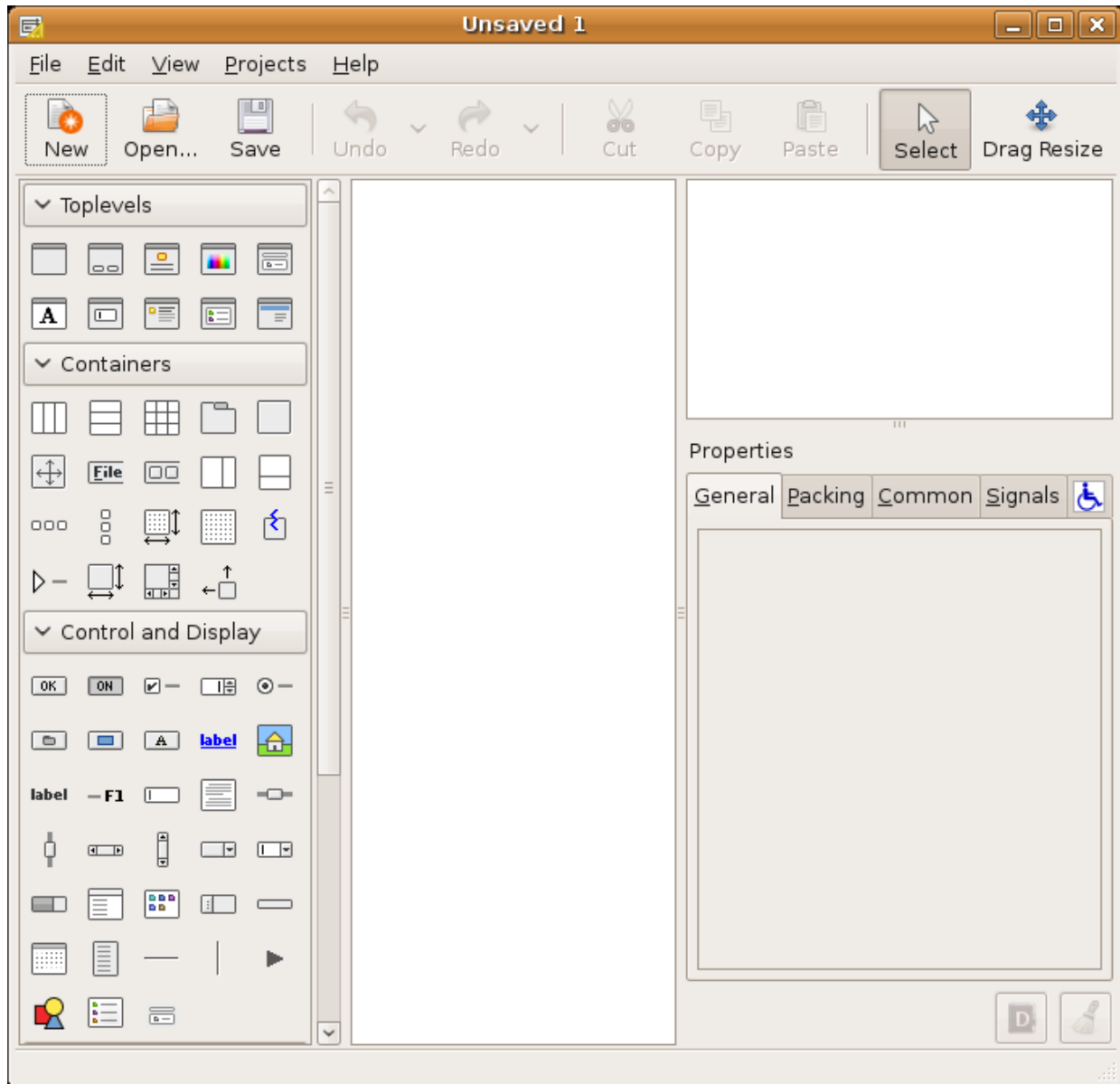
مبنية على المكونات المجانية المتاحة من Komodo

التعامل على حجم اكبر من احتياجات اكثر من ال refactoring و quick fixes و غيرها..

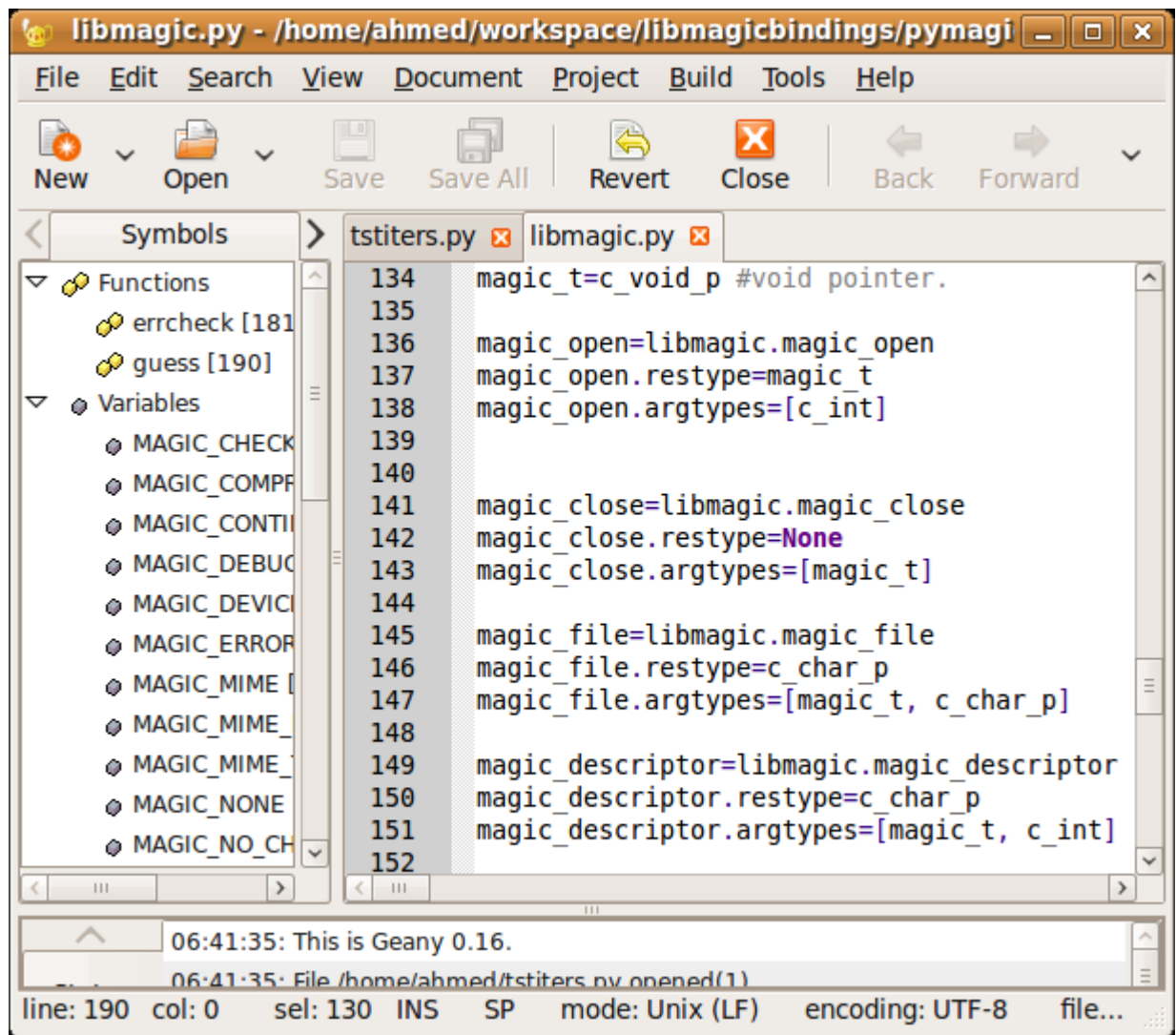


Glade

مصمم للواجهات مختص بـ GTK ذكرنا كيفية استخدامه في Gqamoos



Geany



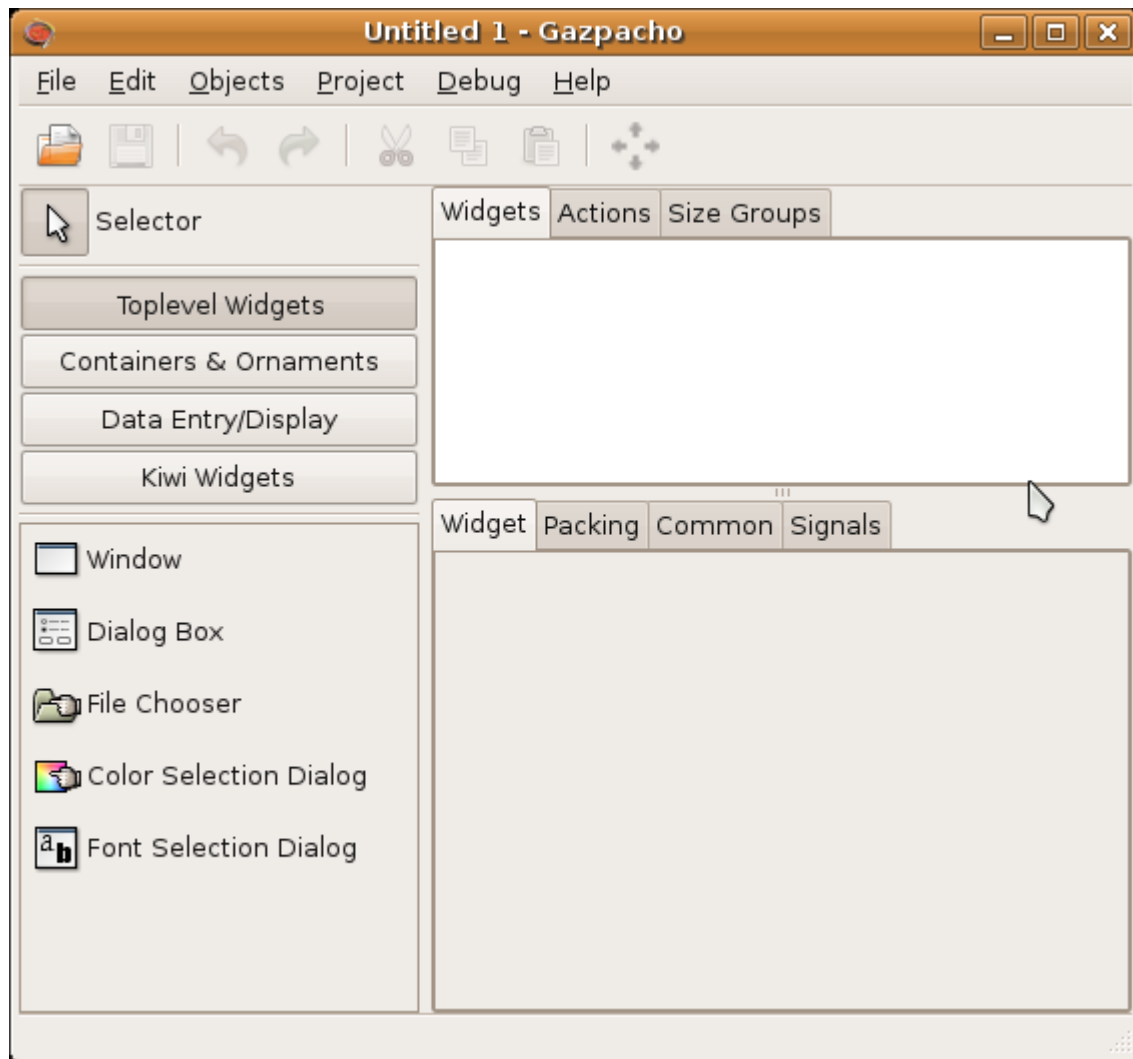
برنامج geany هو محرر نصوص متقدم مكتوب بـ GTK+ ويقدم بعض مزايا الـ IDE ليساعد في التطوير السريع للبرامج

للمزيد : <http://www.geany.org>

Gazpacho

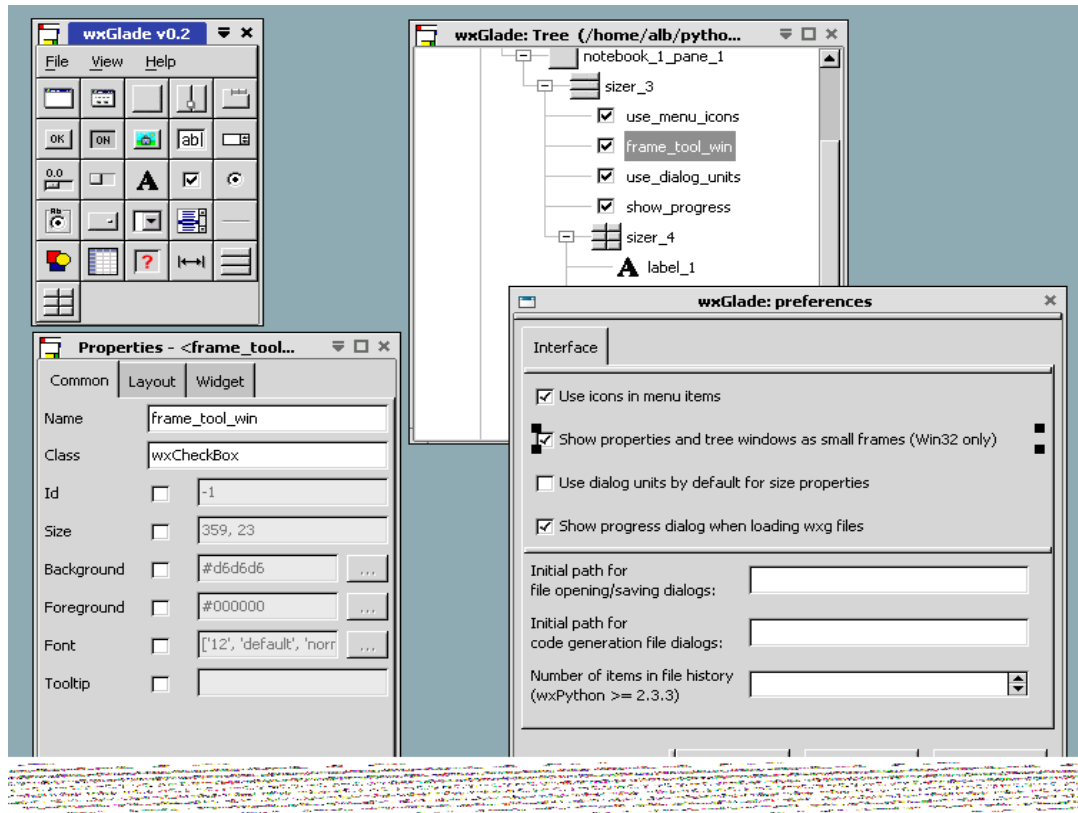
<http://gazpacho.sicem.biz/>

الجيل الجديد من glade مكتوب ببايثون و gtk



wxGlade

<http://wxglade.sourceforge.net/>



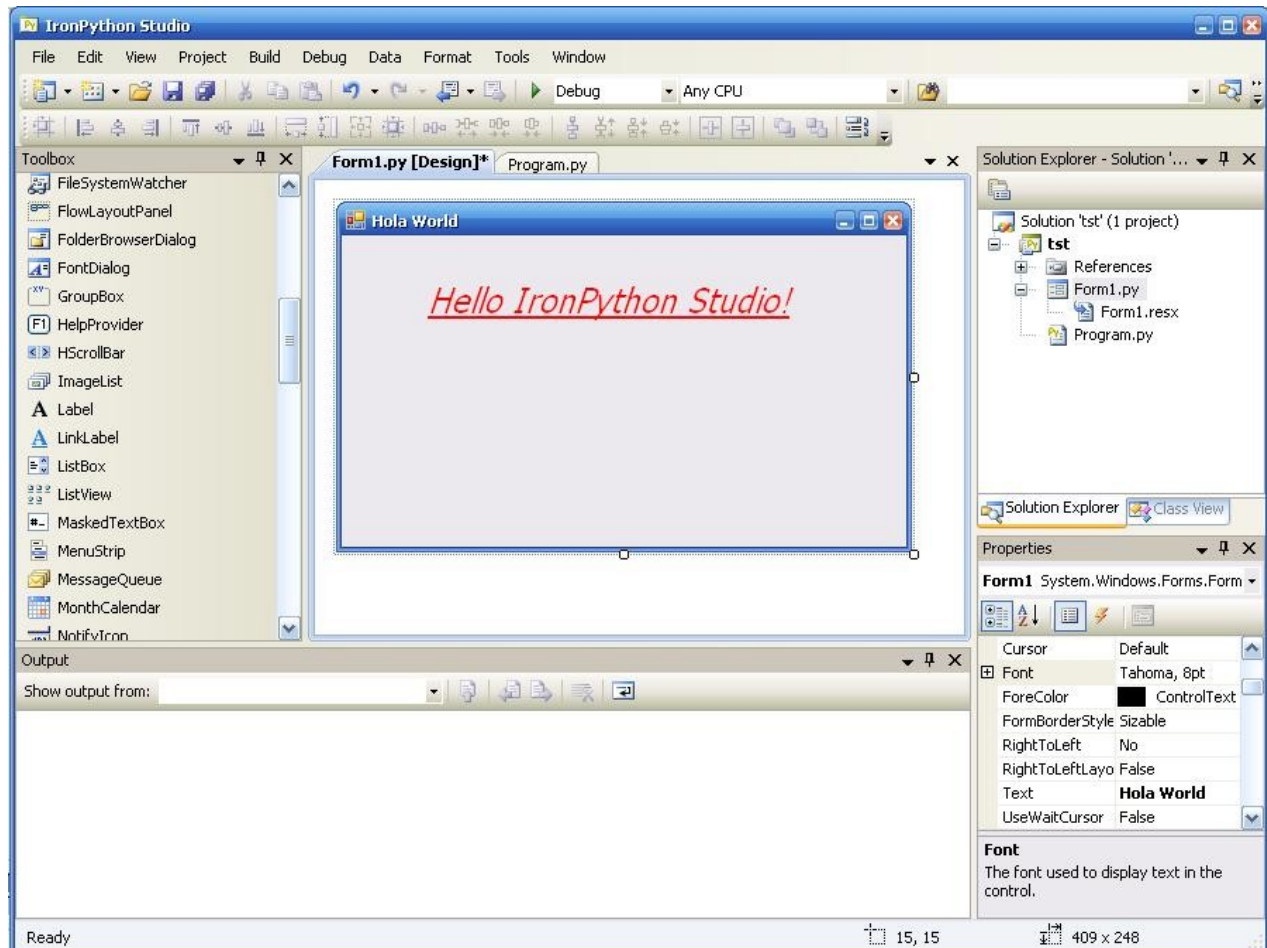
الصورة مأخوذة من الموقع الرسمي
مشابه ل Glade ولكن ل wx

BoaConstructor

<http://boa-constructor.sourceforge.net/>

IDE متكاملة محرر نصوص ومصمم واجهات (مشابه لدلفي) .. الخ الخ
للأسف المشروع شبة متوقف

IronPython Studio



IronPython Studio هي IDE متكاملة ومجانية للغة بايثون مبنية على Visual Studio 2008 Shell runtime

<http://www.codeplex.com/IronPythonStudio>

Quick installation guide:

1- قم بتحميل وتثبيت [Visual Studio 2008 Shell Isolated Mode Redistributable package](#)

2- اذهب الى المجلد "C:\VS 2008 Shell Redist\Isolated Mode"

3- شغل الملف "vs_shell_isolated.enu.exe"

4- حمل وستب [IronPython Studio](#)

Todo