Introduction to Python

Table of contents

More Will be added.

Introduction

From P.F

Not Implemented YET!

Chapter 2 : Introduction to Python

ماهى ال *Programming* ؟
انك تكتب برنامج معناه إنك تدى تعليمات لل *Computer* ينفذها .. بمعنى ابسط إنك تقدر

تخاطب ال *Computer* .

مامعنى ال *Debugging* ؟
ال *Debugging* هى عملية تصحيح للأخطاء الموجودة بال *Code* بتاعك .. لأنك بتخاطب ال *Computer* بطريقة خاطئة فمش هيفهمك تعليماتك .. او ممكن ينفذ بطريقة خاطئة .

ماهى Python ؟
هى Script language قام بكتابتها Guido van Rossum

ماهى مميزات Python ؟
اللغة إستخدامها والسورس بتاعها For free لأنها Open تعد منافس قوى فى عديد من المجالات سواء من ناحية ال Windows Applications او Web Applications وغالبا نشوف مقارنات بينها وبين لغات مثل Perl, Ruby, Java, ... etc
إضافة لقوتها وسرعتها النسبية ايضا نجد :

- **Very clear, readable syntax**
- **Strong introspection capabilities**
- **intuitive object orientation**
- **natural expression of procedural code**
- **full modularity, supporting hierarchical packages**
- **exception-based error handling**
- **very high level dynamic data types**
- **extensive standard libraries and third party modules for virtually every task**
- **extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)**
- **embeddable within applications as a scripting interface**

اللغة تتميز بأنها Portable مما يعنى قدرتها على العمل على انظمة كثيرة جدا منها :
Windows, Linux/Unix, OS/2, Mac, Amiga, among others. There are even versions that runs on .NET, the Java virtual machine, and Nokia Series 60 cell phones

Downloading and Installing Python

ادخل هنا وقم بالتحميل حسب نظام تشغيلك .
اللغة غالبا مدعمة عند مستخدمى Linux

-- ملاحظة لمستخدمى Windows
بعد ماتقوم بتستيب ال Python على جهازك يجب ان تدعمها فى ال PATH
إفتح ال Command Line
Start -> Run -> cmd

واكتب

set path=%path%;C:\Python25

حيث ان ال C:\Python25 هو دا المسار اللى إتستبت فيه ال Python على جهازك

1)Right Click on My Computer -> Properties
2)Advanced Tab
3)Environment Variables
4)in Variables for (UserName) : Click on PATH - > Edit
5)Add C:\Python25 ;
don't Forget the semi colon (;)

-- ملحوظة لمستخدمى *UNIX/UNIX-Like* :
بعد ماتكتب البرنامج بتاعك ديما تديله تصريح التنفيذ *Execute* عن طريق *chmod*

```
chmod +x Program.py
```

اول سطر فى برنامجك دائما عبارة عن تحديد مسار Python على الجهاز
دى مسارها على نظام لينكس على جهازى وغالبا نفس المسار لديك `#!usr/bin/python`
المسار على نظام ويندوز على جهازى ايضا قد يختلف لديك `#!C:\Python25\python.exe`

ن ال Terminal او ال CMD إكتب Python هيشغلك ال Intrepeter فى ال Ineratctive Mode بمعنى إنه مود تفاعلى ..
يعنى النواتج هتكون لحظية ؛ بتستخدم المود دا فى اختبار اجزاء صغيرة فقط!

Hello, World!

كل الكتب والتوتريلز بتبدأ بالمثال دا ديما فى اى لغة برمجة فهنبدأ بيه ^^

فى ال IDE اللى بتوفرهالك Python

```
>>> print "Hello, World!"
Hello, World!  # Output
```

جملة print هى جملة الطباعة فى python كل مابعد print وداخل علامتى التنصيص " " هو اللى هيتم طباعته
مابين علامتى التنصيص إسمه String
تقدر تستخدمها كالتالى

```
>>> print "Hello, " + "World"
Hello, World
```

فهتعمل دمج لل 2 String وتخليهم String واحدة بإسم Hello, World

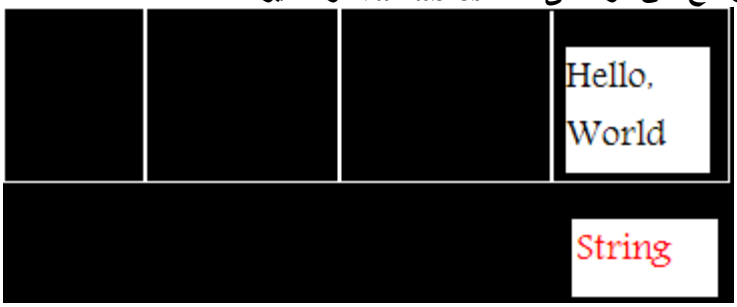
ينفع تدمجهم بإستخدام ال Comma بدل من ال Plus

```
>>> print "Hello, ", "World"
```

Hello, World

```
>>> string = "Hello, World!"
>>> print string
Hello, World!
```

عشان نشرح المثال دا لازم نوضح شئ هو معنى كلمة Variables او متغيرات



لاحظ ال string اللى إسمها Hello, World متخزنه فى مكان معين فى الميمورى .. عشان نستدعيها داخل برنامجنا لازم نعمل زى Alias لمكانها عشان نوصل للبيانات المتخزنة فيه .. فإحنا قولنا إن عشان نستخدم Hello, World دى هندعوها String كل لما نستخدم كلمة String هنستدعى Hello, World !!
دا فى حال لو كلمة String دى مش بين علامتى تنصيص معناها هيكون Hello, World ولكن لو داخل علامتى تنصيص هيكون معناها هو String قيمته "String" لاحظ المثال التالى

```
>>> print "string"
string
>>> print string
Hello, World!
```

نفس النظام للأرقام لاحظ المثال التالى فيه عمليات الحساب الأساسية من جمع وطرح وضرب وقسمة

```
>>> x = 1
>>> y = 2
>>> print x+y
3
>>> print y-x
1
>>> print x*y
2
>>> print y/x
2

>>> #Hello, World
>>>
```

مش حصل حاجة فى السطر السابق ليه ؟ ومش إدى إيرور معناه إنه صح !

هو صح .. السطر دا بيطلق عليه Comment او تعليق ممكن تستخدمه داخل برنامج ولكن لايؤثر عليه بأى شكل من الأشكال لأن ال Interpeter بيتجاهله فى عملية التنفيذ لاحظ التالى :

```
>>> print "Hello, World!"
Hello, World!
>>> print "Hello, World!" #This line prints Hello, World!
Hello, World!
```

تم كتابة Hello, World! فى كلا الحالتين دون ادنى تأثير ولكن التعليق بيكون للمبرمج مش لل Interpeter
عشان توضح انت إستخدمت الكود دا ليه وهكذا ..

يلا نكتب اللى إتعلمناه لحد الوقتى فى Text Editor وليكن Vim او اى شئ مش هتفرق كتير !

```
#!bin/python

############################
# Written by : StrikerX
# Purpose : Simple program
# Date : I don't know :S
############################



print "Hello, World!"  #prints "Hello, World" at the output

string = "Hello, World" #Declaring a string and assign "Hello, World" to it!

print string #prints the value of string which is "Hello, World" to the output

a = 1
b = 2
#You may assign the values  like this a=1; b=2 or a, b = 1, 2

print a, b #prints 1 2

print a+b #prints 3

print b-a #prints 1
```

البرنامج اكيد إنت فهم كل كلمة فيه لكن ماعدا اول سطر

#!bin/python

السطر دا بحدد بيه مسار ال Interpeter عندى على الجهاز .. تقدر تقول نوع خاص للتعليق ولكن لنظام التشغيل عشان يعرف هيشغل السكربت بإيه بالظبط !

إحفظ السكربت بأى إسم وليكن Hello.py لاحظ الإمتداد المستخدم py
قم بتشغيله عن طريق

%>python Hello.py

Naming Conventions

فى بعض الأشياء هتجدها اساسية فى معظم لغات البرمجة فإلتزم بالتالى
1- عدم بدأ إسم اى Data Type بأى رقم 0-9
2- إبعد عن إستخدام الرموز الخاصة مثل $ او % لأنها غالبا بيستخدمها ال Compiler فى عمليات داخلية إلا لو Perl مثلا
3- لاتستخدم مسافات داخل إسم ال Data Type كالتالى my Variable
اختار الصورة اللى بتريحك وإستخدمها

myVariable
MyVariable
My_Variable
my_Variable

افتح ال IDLE عشان هنكتب بعض الأكواد الخفيفة :

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

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

بنستخدم type فى إننا نعرف نوع ال Data Type وهنا كان نوع ال myString من النوع str وهى ال string فى python

```
>>> myString.capitalize()
'Hello, world!'
```

تقوم capitalize() بتحويل اول حرف لل UpperCase

```
>>> myString.lower()
'hello, world!'
```

تقوم lower() بتحويل جميع الحروف إلى LowerCase

```
>>> myString.upper()
'HELLO, WORLD!'
```

تقوم upper() بتحويل جميع الحروف إلى ال UpperCase

```
>>> myString.count("l")
3
```

تقوم count() بعد عدد مرات التكرار لحرف معين

```
>>> myString.endswith("!")
True
```

endswith() تعيد قيمة True او False حسب ماإذا كان ال Argument هو آخر مقطع فى ال string

```
>>> myString.startswith("H")
True
```

startswith()تعيد قيمة True او False حسب مإذا كان ال Argument هو اول مقطع فى ال string

```
>>> myString.swapcase()
'hELLO, wORLD!'
```

swapcase()تقوم بعكس حالة الأحرف

```
>>> myString.replace("Hello", "Goodbye")
'Goodbye, World!'
```

replace() تقوم بإستبدل مقطع من ال string بمقطع آخر

dir/help

dir() بنستخدمها لعرض محتويات ال Type او Module -سنتعرض لل Modules لاحقا-

```
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__ge__',
'__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__', '__hash__',
'__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__str__', 'capitalize', 'center',
'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'index', 'isalnum', 'isalpha', 'isdigit',
'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase',
'title', 'translate', 'upper', 'zfill']
```

اعادت لنا كل مابداخل ال Type str

بنستخدم help لعرض ال documentation الخاصة بال Argument فمثلا ال Argument هى الميثود capitalize ف help
عرضت لينا مثال للإستخدام وفايدة ال Method.

```
>>> help(str.capitalize)
Help on method_descriptor:

capitalize(...)
   S.capitalize() -> string

   Return a copy of the string S with only its first character
   capitalized.
```

عودة ل print وبالمناسبة هى statement وليست Function فى Python! حتى الإصدار 3 هتتحول ل Function
المهم .. تابع المثال التالى :

```
>>> name="StrikerX"
>>> age=18
>>> sex='M'
>>> print "Name is : %s, Age : %d, Sex : %c" %(name, age, sex)
Name is : StrikerX, Age : 18, Sex : m
```

%s معناها إن القيمة اللى هتحل مكانها هى string
%d معناها إن القيمة اللى هتحل مكانها هى decimal
%c معناها إن القيمة اللى هتحل مكانها char
وكل اللى إحنا عملناه إسمه Formating output يعنى نحدد الصورة النهائية اللى عايزينها للناتج.

Chapter 3 : Basics

حتى الآن إنت تعرف ماهى ال Variables وماهى ال Python بصورة عامة وكيفية إستخدام بعض الميثودز .. إلخ إلخ
تخيل معى إنك مطلوب منك برنامج فيه 4 متغيرات وتجمعهم مثلا قيمهم 5و 7 و 8 و 11

```
>>>var1=5
>>>var2=7
>>>var3=8
>>>var4=11
```

جميل ولكن إفرض كانو 100 متغير مثلا هل هتعرفهم بنفس الطريقة ؟ هتكون صعبة اكيد وهتطول الكود بدون داعى . تفدر تستخدم Data Type جميلة جدا إسمها list بتقدمهالك Python كالتالى

```
>>>myVars=[5, 7, 8, 11]
```

اول عنصر إسمه عنصر رقم 0
تانى عنصر إسمه عنصر رقم 1 وهكذا لأن ال List is Zero-Based index يعنى اول عنصر ترتيبه 0

```
>>> myVars[0] #اول عنصر
5
>>> myVars[1] #تانى عنصر
7
>>> myVars[2] # تالت عنصر
8
>>> myVars[3] # رابع عنصر
11
```

in

فى Keyword فى Python إسمها in المثال التالى يوضح كيفية إستخدامها

```
>>> 5 in myVars
True
```

بتعيد لينا قيمة True او False فى حال إستخدمناها مع Data Type بالطريقة السابقة

```
>>> 5 in myVars
```

تعنى هل ال 5 موجودة فى ال List اللى إسمها myVars ؟
الإجابة True او False

جميل جدا نترك المثال بتاع myVars ونتكلم عن ال List وال Methods اللى بتقدمهالنا

```
>>> myList=[1, 2, 3, 4, 5, 6, 7]
```

بنعرف List باسم myList

```
>>> myList.append(18)
```

نقوم بإضافة الرقم 18 داخل ال myList بإستخدام append()

```
>>> myList
[1, 2, 3, 4, 5, 6, 7, 18]
>>> myList.pop()
18
```

تقوم بحذف آخر عنصر بال myList وطبعه لينا

```
>>> myLista
[1, 2, 3, 4, 5, 6, 7]
>>> myList.insert(3, 11)
>>> myList
[1, 2, 3, 11, 4, 5, 6, 7]
```

بنستخدم insert() لما نيجى نضيف رقم معين فى ترتيب معين وهنا ضفنا الرقم 11 فى محل العنصر ال 4 ، بما ان ال Lists are zero-based index فهنكتب بدل ال 4 الترتيب 3

```
>>> myList.sort()
```

بنستخدم sort() فى ترتيب ال List

```
>>> myList
[1, 2, 3, 4, 5, 6, 7, 11]
>>> myList.reverse()
```

بنستخدم reverse() فى عكس ال List

```
>>>myList
[11, 7, 6, 5, 4, 3, 2, 1]

>>>myList.count(3)
```

بنستخدم count فى حساب كم مرة تكرر العنصر داخل ال List

```
>>>myList[0:4]
[11, 7, 6, 5]
```

تعيد لنا العناصر من العنصر ترتيبه 0 حتى العنصر الذى ترتيبه 4
وتقدر تكتبها بالصورة دى

```
>>>myList[:4]
[11, 7, 6, 5]


>>> myList[-1]
1
```

ألعنصر الأخير فى ال List بيكون ال index بتاعه -1 او تقدر تعد من الصفر لحد ماتوصله :D

```
>>> myList[-4:]
[4, 3, 2, 1]
```

تقوم بإعادة آخر 4 عناصر فى ال List

range()

ال Function دى بتستخدم بكثرة مع ال Lists

```
List=range(0, 10) #Populates the list with numbers from 0 to 10
>>> List
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

لو البداية من ال 0 يبقة لاداعى تضيفه تقدر تستخدمها كالتالى

```
List=range(10)
```

إذا عايز تحدد ال step تقدر تنفذها كالتالى

```
>>> List=range(0, 10, 2)
```

عمل List تبدأ من ال 0 وتنتهى ب 10 وبزيادة مقدارها 2 .. لاحظ إنك إذا لم تحدد مقدار الزيادة فهى 1

```
>>> List
[0, 2, 4, 6, 8]
```

Tuples

هى Data Type كتير مشابه لل List ولكنه غير قابل للتعديل إلا بإضافة Tuple ليها

```
>>>t=3, 4, 5, 6
>>> t
(3, 4, 5, 6)
```

تقدر تضيف اقواس كالتالى إذا حبيت

```
>>> t1=(3, 5, 8)
>>> t1
(3, 5, 8)

>>> t[0]
3
```

إعادة اول عنصر فى ال Tuple زيها زي ال List فى ال Zero-based index

```
>>> t[0:2]
(3, 4)
```

إستخراج العناصر 3و 4

```
>>>
```

تستطيع ان تجرى عملية واحدة وهى الإضافة ومش عن طريق عنصر ولكن عن طريق tuple كالتالى

```
>>>tup=1, 2, 3, 4
>>> tup += tup2 #or (5, 6, 7)
>>> tup
(1, 2, 3, 4, 5, 6, 7)
>>>
```

Dictionaries

ال Dictionaries او القواميس مسماة بكدا لأنها فعلا زى القاموس .. الكلمة فى القاموس ليها مدلول مثلا وليكن فى كلمة إسمها Name مدلولها هو StrikerX وهكذا

```
>>> DicData={}
>>> DicData['Name']='StrikerX'
>>> DicData['Age']=18
>>> DicData['Sex']='Male'
>>> DicData
{'Age': 18, 'Name': 'StrikerX', 'Sex': 'Male'}
>>> DicData['Name']
'StrikerX'
```

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

keys()بترجع ليك ال keys اللى حطينالها قيمة

```
>>> DicData.values()
[18, 'StrikerX', 'Male']
```

values() بترجع القيم اللى اسندناها لل Keys

```
>>> DicData.items()
[('Age', 18), ('Name', 'StrikerX'), ('Sex', 'Male')]
```

بترجع ليك كل Key و Value فى Tuples داخل List

has_key بتستخدم فى حال إختبار إذا فى Key معينة فى ال DicData.keys()
has_value بتستخدم فى حال إختبار إذا فى Value معينة فى ال DicData.values()

```
>>> DicData.update({'Name':'Squall'})
>>> DicData
{'Age': 18, 'Name': 'Squall', 'Sex': 'Male'}
```

update() بتستخدم فى تعديل على Value معينة فبتاخد بارميتر واحد من نوع dict بتحط فيه ال key ومعاها ال Value الجديدة

Conditions/Loops

تقدم لك Python كل ماتحتاجه للتحكم فى برامجك من إستخدامات لل Conditions و Loops ولكن بطريقة اكثر منطقية. حيث تخلت عن إستخدام Switch/For للقادمين من لغات اخرى.

if/elif/else

```
If Expression is True Then DoSomething
```

فى Python هتكون الطريقة مكافئة ل

```
if Expression == True : DoSomething
```

نستخدم مثال بسيط يوضح لينا الفكرة

```
>>> Number=3
>>> if Number==3 : print "Number is 3"

Number is 3
```

تقدر تكتب على عدة سطور ولكن لازم تاخد بالك إن لازم ال Block بتاع if يكون تحته بالصورة دى

```
>>> if Number==3:
        print “Number is 3”
```

مثال ولكن إكتبه فى اى Editor بعيدا عن ال IDLE

```
if Number==4:
        print "Number is 4"
elif Number==5:
        print "Number is 5"
elif Number == 7:
        print "Number is 7"
```

The output : Number is 7

فى اول سطر 4 == if Number هيتنفذ ال Block تبعه فى حال لو ال 4 == Number لو مش يساوى 4 يتنقل للإختبار اللى بعده elif .. وبيختبر إذا الرقم يساوى 5 لو بيساويها هيتنفذ ال Block تبعه لو لأ يتنقل للإختبار اللى بعده .

```
if Number==4:
        print "Number is 4"
elif Number==5:
        print "Number is 5"
else :
        print "I don't KNOW!"
```

else هنا هيتم تنفيذها فى حال فشل Number==4 و Number==5

for

صراحة ال Loop دى مشابهة ل Foreach اكثر من For

```
>>> List=[1, 2, 3, 4, 5]
>>> for Number in List:
        print Number

1
2
3
4
5
```

ال Loop دى بكل بساطة بتتنفذ على كل عنصر فى ال Data Type زى مانت شفت

```
>>> for char in "Python":
        print char

P
y
```

t
h
o
n

for (init; cond; inc;)

غير مدعمة فى ال Python وبنستخدم while مكانها

```
>>>start=1 #Init
>>> while start<=10: #Condition
        print start
        start += 1 #Inc
```

1
2
3
4
5
6
7
8
9
10

for(; ;)

بتسمى Forever loop لأنها غير منتهية بسبب عدم وجود Condition يخلينا نتحكم فى عدد مرات ال Loop دى

```
>>> while True :
        print "Hello, World!"
```

دى Loop غير منتهية هتستمر فى كتابة Hello, World! للخروج من ال Loop دوس C^

تطبيق على ال Loops
مثلا عايزين نطبع جدول فيه قيم كل حرف من جدول ال ASCII ،فلازم نعرف البداية والنهاية للجدول
الحروف هتبدأ من ال 'A' لحد ال 'Z' يعنى هنطبع حروف ال Uppercase فقط .. ال LowerCase دى واجبك انت :D

```
>>>ord('A')
65
```

بنستخدم ord() للحصول على قيمة الحرف من جدول ال ASCII

```
>>>ord('Z')
90
```

وللعكس نستخدم chr() حيث تعيد لنا الحرف الذى قيمته = ال Argument كالتالى

```
>>>chr(90)
'Z'

>>> charValue=65
>>> while charValue<=90:
        print "%c : %d" %(chr(charValue), charValue)
        charValue += 1


A : 65
B : 66
C : 67
D : 68
E : 69
F : 70
G : 71
H : 72
I : 73
J : 74
K : 75
L : 76
M : 77
N : 78
O : 79
P : 80
Q : 81
R : 82
S : 83
T : 84
U : 85
V : 86
W : 87
X : 88
Y : 89
Z : 90
```

ممكن نستخدمها كالتالى

```
for charValue in range(65, 90):
        print "%c : %d" %(chr(charValue), charValue)
```

Functions

ماهى ال Functions ؟
هى كود مكتوب سابقا للمبرمج ليستخدمه اكثر من مرة ... مثل ord, chr اللى تعرضنا ليهم وجميل ال Functions الخاصة بال Strings مثل ال lower, upper, .. etc
دى إسمها Functions جاهزة .. ناس كتبتها وانت بتستخدمها ، ولكنك تقدر تكتب ال Functions الخاصة بيك بالطبع!

```
>>> def Hi():
        '''This function prints Hi'''
        print "Hi"
```

اولا def هى Keyword فى Python بتستخدم لتعريف ال Functions ويليها إسم ال Function ولاتنسى ال () لأنك بتحدد فيها ال Parameters
تانى سطر إسمه ال Documentation لل Function انت بتوضح فيه وظيفة ال Function

```
>>> Hi.__doc__
'This function prints Hi'
```

او بتقدر تستخدم Help() لأنها اسهل كالتالى

```
>>> help(Hi)
Help on function Hi in module __main__:

Hi()
   This function prints Hi
```

لاحظ السطر التالى :

```
print "Hi”
```

هو ماستقوم ال Function بعمله كل مرة تستدعيها

```
>>> Hi()
Hi
>>> Hi()
Hi
>>> Hi()
Hi
```

مثال آخر :

```
>>> def Hi(YourName):
        '''This function says hi to you!'''
        print "Hi, %s" %(YourName)
```

YourName إسمه Parameter بيتباصى لل Function ، لاحظ الإستخدام

```
>>> Hi("StrikerX")
Hi, StrikerX

>>> Hi("Python")
Hi, Python


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

ال Function دى بتاخد 2 Parameter وترجع لينا مجموعهم دى فايدة D: return

```
>>> add(1, 3)
4

>>> def Data(Name, Age, Sex):
        print "Name : %s, Age : %d, Sex : %c" %(Name, Age, Sex)


>>> Data('StrikerX', 18, 'M')
Name : StrikerX, Age : 18, Sex : M
>>>
```

اتمنى تكون وضحت ليكم.

Modules

ماهى ال Modules ؟
هى عبارة عن مجموعة Functions و Classes والعديد من ال Data Types متجمعة فى ملف واحد وبيتاح ليك إستخدامهم بمجرد إستدعاء الملف دا كالتالى : فى Module إسمها math فى Python واضح جدا إنها موديل خاصة بالرياضيات فهى تشمل عديد من الثوابت مثل PI = 3.1416 وتشمل الدوال مثل دالة الصحيح والجذر وهكذا ..

نستدعى الموديل بالطريقة المعتادة بإستخدام the "import" Keyword

```
>>>import math
```

>>> math.floor(3.8) # دالة الصحيح ..
3.0

>>> math.sqrt(16) # دالة الجذر
4.0

>>> math.fabs(-19) # القيمة المطلقة
19.0

>>> math.exp(4) # e4**
54.598150033144236

>>> math.pi
3.1415926535897931
>>> math.e
2.7182818284590451

>>> math.log10(100) # 10 لوغاريتم 100 للأساس
2.0

>>>math.log(100,10) # المعامل الأول هو العدد والتانى هو الأساس
2.0

>>> math.pow(2, 3) # تقوم بإعادة 2 مرفوعة لأس 3
8.0

>>> math.hypot(3,4) # تقوم بإعادة جذر مجموع مربعات القيمتين 3 و 4 (الوتر من فيثاغورث)
5.0

>>> math.radians(30) # تقوم بإعادة الزاوية 30 بالتقدير الدائرى
0.52359877559829882

>>> math.degrees(0.52359877559829882) # إعادة الزاوية إلى التقدير الستينى
29.999999999999996

لاحظ لإستخدام كل Function داخل ال Module دى لازم تسبقها بإسم ال Module ودى طريقة آمنة ولكن مثلا إنت عايز فى برنامجك ال log10 function بكثرة ومش عايز تكتب إسم ال Module قبلها فهيكون الحل كالتالى
>>> from math import log10
وتستخدم ال Function بصورة مباشرة

>>>log10(100)
2.0

إذا عايز كل ال Functions اللى فى ال Module تستخدمهم بصورة مباشرة دون ذكر إسم Module قبلهم .. تقدر كالتالى
>>>from math import *
ال * معناها all

كيف تقوم بكتابة ال Modules الخاصة بك فى ال Python ؟

اكيد لاحظت اثناء شروحات ال Python إستدعاننا ل modules مثل math

لما بنيجى نكتب Module بنكتبها عشان نقدر نستخدم مرااات عديدة داخل برامجنا .. بمعنى ابسط
انت لما تيجى تكتب ميثود او فنكشن معينة داخل برنامجك .. بتكتبها عشان تقدر تستدعيها بعدد كبير من المرات وإختصارا للكتابة وتنظيم اكثر ..

ال Modules دى بقى عبارة عن عدد كبير من ال Classes, functions, variables, ...etc مثل الموجودة ب Math Module

كل ماعليك الآن هو إنك تفتح اى Editor وتعمل ملف بإسم rul3z.py مثلا ..

بصفة عامة هيكون ال Header هتعمله بطريقة مشابهه للتالية

رمز:

```
#-########################
# Module : rul3z
# Written by : StrikerX
# Date : 8 May 2007
#-########################

def Hello():
    print ("Welcome to Modules World ! ");


pi = 3.14
```

احفظها الملف دا بأى مسار بشرط إنه يكون فى ال Path اللى على الجهاز .. جرب تحفظه على ال Desktop :ي

تقدر تضيف مكان لل Path بعدة طرق

```
>>> from sys import *
>>> path.append(Path)
```

او بتعديل مثلا لو على Windows

My Computer -> right Click .
Properties .
Advanced tab -> Environment variables
Path -> edit
Add the path and a semi colon >>; << after

وإذا على ال Linux
قم بتعديله ملف bash_profile.

يفضل إنك تشتغل على ال IDLE عشان تكون النتيجة فورية

اكتب

```
>>>import rul3z
>>>rul3z.Hello()
Welcome to Modules World !
```

هل لاحظت ؟؟ إنك قدرت تستخدم The Hello Function from rul3z module

```
>>>rul3z.pi
3.1400000000000001
```

إستخدمنا pi من ال Module بتاعتنا ...

تقدر تكتب كدا

```
>>>from rul3z import *
>>>Hello()
Hello to the Modules World !
>>>
>>>pi
3.1400000000000001
```

File Handling

بداية هبدأ بتعبير قاله سامى N.H.2004 بخصوص التعامل مع الملفات ، لو إنت عندك كتاب إيه التفكير المنطقى عشان توصل لصفحة معينة داخله ؟؟
هتقوم من مكانك وتروح عند المكتبه بتاعتك ..... الخطوة الأولى .. إفتح ال Python او إعمل Script للكود بتاعك

هتفتح الكتاب ، .... ودى الخطوة التانية إنك تخلى Python تفتح الملف اللى إنت عايزه
تشوف الجزئية اللى عايز تقراها ... ودى الخطوة التالته .. إنك تتعامل مع الملف
اخيرا غلق الكتاب .... ودى الخطوة النهائية فى الموضوع كله

نبدأ التطبيق .. انا التعامل حاليا على Win32 ولكن ليس هناك اختلاف كبير سوى مسار الملف ليس اكثر .

اعمل ملف txt بأى Editor واكتب جواه

احفظ الملف بإسم rul3z.txt مثلا

شغل ال Python

Start -> Programs -> Python -> Python

او من ال Terminal على ال Linux إكتب Python

الخطوة الأولى .. نفتح الملف rul3z.txt ونجعله بوضع القراءة read يعنى هنستقبل بيانات من الملف مش هنضيف إليها

```
myFile = open("C:\rul3z.txt","r")
```

احنا عملنا إيه الوقتى ؟؟
عملنا حاجة إسمها File Handler زى متغير بيعبر عن الملف rul3z.txt وهو مفتوح .. وهو دا اللى هنتعامل بيه طول البرنامج

إستخدمنا ال Open ... Function وهى مختصة بفتح الملفات وصيغتها العامة هى التالى
open(path, permission)

path هو مسار الملف
Permission هى ماذا سنقوم بعمله مع الملف ؟ هل قراءة ، ام كتابه ام اضافة للملف

R => read للقراءة
w => write للكتابة

يوجد function اخرى تقوم بنفس الوظيفة وهى file وبتأخد نفس ال parameters زى open بالظبط

طب الخطوة التالية إيه ؟؟ احنا فتحنا الملف الوقتى .. عايزين نتعامل معاه ؟
نبدأ بعرض كل الأسطر اللى بالملف

```
For line in myFile.readlines() : print line
```

نوضح بشئ من التفصيل شوية
myFile اللى زى ماقلنا إنها ال File handler دى عبارة عن Class ... بيشمل العديد من ال methods .. نقدر نقول functions ولكن طالما بنقول Class تبقة Methods اصح فى التعبير

ودى بتقرا كل مافى الملف من سطور readlines() بإسم method دى methods من ضمن ال

تم قرائته إطبعيه readlines() لكل سطر داخل الميثود Python فالأمر اللى نفذناه كان عبارة عن إننا قولنا ل

هتلقى ال Output كالتالى

Live Free || Die

كدا إحنا خلصنا اللى عايزينه من الملف صح ؟
نقفل الملف ودى الخطوة الأخيرة

```
myFile.close()
```

كدا إحنا الأمر بقة سهل

تحبوا نصمم Tool زى Cat مثلا بال Python ؟؟

```
#!bin/python

Path = raw_input("Enter the File Path : ")
myFile = open(Path, "r")

For line in myFile.readlines() : print line

myFile.close()
```

احفظ ال Script وقم بتشغيله ستجده يقوم بوظيفة مشابهه ل Cat

ندخل على التعامل بالكتابة داخل ملف

فى حالة عدم وجودة ... وان كان موجودا سيتم محو كل مابداخله وفتحه للكتابه hi.txt إنشاء ملف بإسم

```
myFile = open("C:\hi.txt","w")


myFile.write("Hello,World " )
```

file handler ال ... class بتبقة تحت ال write إسمها method الكتابة داخل الملف بإستخدام

```
myFile.write("\n " )
```

لوضع سطر جديد

```
myFile.write("Hello,World " + "\n")
```

انت فهمت اكيد ... كتابة الجملة ووضع سطر جديد بعدها

! لاتنسى ابدا غلق الملف بعد الإنتهاء من إعداده

```
 myFile.close()
```

: تطبيق عملى

سكربت يقوم بعمل copy من ملف لآخر

احنا هنعمل copy من الملف rul3z إلى الملف hi
فهنفتح ملف rul3z للقراءة
وهنفتح ملف hi للكتابة

```
myFile1 = open("C:\rul3z.txt","r")
myFile2 = open("C:\hi.txt", "w")


for line in myFile1 : myFile2.write(line + "\n")

print "Done ! "
```

نقوم بغلق الملفات .. الصراحة لأنى مكسل هكتبها كدا

```
List = [myFile1, myFile2]
for x in List : x.close()
```

تقدر تقفلها بطريقة عادية جدا

```
myFile1.close()
myFile2.close()
```

OOP

ال OOP او Object Oriented Programming تكنيك جديد إستخدم واصبح شائع جدا للكفائة والمرونة والبساطة .. كانت فى البداية بيستخدمو ال Procedures وال Functions لتقسيم البرنامج ولكن اصبح ال Classes وال Interfaces هى الإسلوب الأمثل حاليا

Python مش بتجبرك إنك تستخدم ال Objects/Classes بالعكس تقدر تكتب برامجك Procedural style ولكن بما إنك بتستخدم Python فلازم تحاول تستفيد من ال OOP

ماهو ال Object ؟
هو أى شئ إنت شايفه قدامك .. العربية Object والطيارة Object والعمارة Object .. الإنسان Object الكمبيوتر Object

ماهو ال Class ؟
ال Class هو ال blueprint لل Object بمعنى مثلا هنتكلم عن الإنسان

الإنسان ال blueprint ليها يكون إيه ؟ إيدين ورجلين ودماغ وقلب ... etc .. الكلام دا على الورق
لكن لما يتحول الكلام اللى على الورق دا لواقع فدا يبقة إسمه Object

ماهى ال Interface ؟
لن اتعرض لمفهومها لأنها بتستخدم اكثر فى لغات زى ال C#/Java اللتى تدعم ال Multiple inheritance

ماهى ال Attributes او ال Data Fields ؟
ال Attributes هى صفات ال Class مثلا الإنسان صفاته إيه ؟ مثلا لون عيونه ولون بشرته وطوله ووزنه

ماهى ال Methods ؟
ال Methods هى ال behaviors الخاصة بال Class الشئ اللى يقدر ينفذه مثلا الإنسان يقدر ياكل ويشرب وينام ويلعب ويشتغل ..

مثلا دا ال Blueprint لإنسان

```
class Human:

    def __init__(self, name, sex, color):
        #Attributes
        self.name=name
        self.sex=sex
        self.color=color
        self.hands=2
    #Methods
    def move(self):
        print self.name, "is moving."
    def eat(self):
        print self.name, "is eating."
    def sleep(self):
        print self.name, "is sleeping."
```

The Constructor

ال Constructor هو Method زيه زى اى ميثود ولكن فى Python بيكون إسم ال Method دى هو __init__
اول parameter فيها لازم يكون self ودا بيعنى reference to the current object بيشير لل Object الحالى
وبعد كدا الداتا اللى عايز تديها لل Object مثلا ال name و sex وال color وتقدر تضيف حاجة زى hands = عدد الإيدى اللى عند الإنسان الطبيعى وهى 2

ال Fields او ال Attributes هى الصفات المميزة لل Human وهى ال name وال sex وال color .. etc
تقدر تضيف طبعا من عندك زيادة

ال Methods هى ال behaviors الخاصة بال Human زى إنه ينام وياكل ويتحرك لاحظ إذا عايز تأكسس ال Fields الخاصة بال Class عن طريق Methods لازم تضيف self فى البداية

```
>>> ahmed=Human('Ahmed', 'm', 'white')
```
عمل Object من ال Human Class بإسم ahmed واسندنا ليه ال name, sex, color
إستدعاء ال Attributes, methods عن طريق ال Object

```
>>> ahmed.name #gets the name field
'Ahmed'
>>> ahmed.sex #gets the sex
'm'
>>> ahmed.color #gets the color
'white'
>>> ahmed.move() #calling the move method
Ahmed is moving.
>>> ahmed.sleep() #calling the sleep method
```

```
Ahmed is sleeping.
>>> ahmed.eat() #calling the eat method
Ahmed is eating.
>>> ahmed.hands #gets the number of hands
2
```

Inheritance

الوراثة .. هى إن بيقة فى Class يشمل كل خواص Class معين ويزيد عنها فى شئ او يكون معدل عنها فى شئ .. مثلا لاعب كرة .. هياخد كل مافى ال Human Class + بعض الإضافات الخاصة بلاعبى الكرة

```
class FootballPlayer(Human): #Inherits Human
        def __init__(self, name, sex, color, team):
                self.name=name
                self.sex=sex
                self.color=color
                Human.__init__(self, self.name, self.sex, self.color) #Calling the super Constructor
                self.team='Galaxy'
        def kick_ball(self):
                print self.name, "Hits the ball!"
```

لاحظ إنك فى ال Constructor الخاص ب FootballPlayer هنستدعى ال Constructor الخاص بال Human
عشان نجهز ال Attributes اللى جواه
عمل Object منه وليكن Beckham

```
>>> Beckham=FootballPlayer('David Beckham', 'm', 'white', 'Galaxy')
>>> Beckham.sleep() #لاحظ إنها من ال Human Class
David Beckham is sleeping.
>>> Beckham.kick_ball()
David Beckham Hits the ball!
```

Python بتدعم ال Multiple inheritance ولكن فى البداية هتفصل بين ال Classes اللى هتورثها لل Class

```
>>> class Helicopter:
```

```
        def __init__(self):
        pass
        def fly(self):
           print "FLYING!"

>>> class Boat:
        def __init__(self):
                pass
        def sail(self):
                print "SAILING!"

>>> class Car(Boat, Helicopter):
        def __init__(self):
                Boat.__init__(self)
                Helicopter.__init__(self)

>>> c=Car()
>>> c.fly()
FLYING!
>>> c.sail()
SAILING!
```

Properties in Python

ال Properties مفهومها موجود فى العديد من اللغات مثل ال C#.NET /VB.NET وكذلك ال python وفى لغات اخرى بتستخدم Getters/Setters methods مثل ال Java مثلا
ال Properties نفسها بتستخدم Getters/Setters ولكن بإسلوب الطف ، يعنى بدل ماتحفظ 2 method عشان تستخدمها هى Property واحدة بتستخدم ك get/set

لاحظ بدون إستخدام ال Properties او ال getters/setters انك تقدر تتعامل مباشرة مع ال Fields ودا شئ خطير ، لاحظ التالى رمز:

```
>>> class Person(object):

      def __init__(self, name, sex, age):
            self.name=name
            self.sex=sex
            self.age=age


>>> p1=Person('Ahmed', 'M', 18)

>>> p1.name
'Ahmed'
>>> p1.sex
'M'
>>> p1.age
18
```

```
>>> p1.name="RULES"
>>> p1.name
'RULES'
>>> p1.age=2
>>> p1.age
2
```

هنا نقدر نسند string ل age ودا شئ مش تمام لأنه المفروض مش ياخد غير int وتقدر تسند قيمة غير منطقية لل age مثلا 100000 سنة !

رمز:

```
>>> p1.age="WHATEVER"
>>> p1.age
'WHATEVER'
```

فعشان نتغلب على المشكلة دا لازم نعمل check وذلك بإستخدام getter/setter لكل field + نمنع الأكسس المباشر ليه فهنعمله private مش حد يقدر يوصله غير ال methods اللى داخل ال class نفسها فهنسبق إسم اى field ب 2 underscores

رمز:

```
class Person(object):

    def __init__(self, name, sex, age):
        self.__name=name
        self.__sex=sex
        self.__age=age
    #getter/setter for name
    def get_name(self):
        return self.__name
    def set_name(self, new_name):
        assert type(new_name) == str
        self.__name=new_name

    #getter/setter for sex
    def get_sex(self):
       return self.__sex
    def set_sex(self, new_sex):
       assert type(new_sex)==str
       self.__sex=new_sex

    #getter/setter for age
    def get_age(self):
       return self.__age

    def set_age(self, new_age):
       assert type(new_age)==int
       self.__age=new_age
```

لاحظ التالى:

رمز:

```
>>> p=Person('Ahmed', 'M', 19)
>>> p
<__main__.Person object at 0x00CEA530>
>>> p.name

Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    p.name
AttributeError: 'Person' object has no attribute 'name'
>>> p.get_name()
'Ahmed'
>>> p.set_name("RULES")
>>> p.get_name()
'RULES'
```

لاحظ عند محاولتنا إسناد str لل Field age هيدينا AssertionError وهيمنع الإستمرار !

رمز:

```
>>> p.set_age('19') #Assign age field to a string -> AssertionError!

Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    p.set_age('19') #Assign age field to a string -> AssertionError!
  File "<pyshell#2>", line 26, in set_age
    assert type(new_age)==int
AssertionError
>>> p.set_age(31)
>>> p.get_age()
31
```

لحد الآن هو دا افضل اسلوب ولكن هل هتضطر احفظ 2 method لكل field عشان استخدمه ؟؟

JAVA قالت آه إستخدم 2 method عشان كل field ، لكن لغات تانية زى C#.NET /Python قالو لأ إستخدم Properties وهى مشابهه لل Field ولكن بشروط ال getters/setters

وهنا نيجى للى إحنا محتاجينه

رمز:

```
class Person(object):
    #Constructor
    def __init__(self, name, sex, age):
        #Initializing fields
        self.__name=name
        self.__age=age
        self.__sex=sex

    #Private Getters
    def __get_name(self):
        return self.__name

    def __get_sex(self):
```

```
    return self.__sex

  def __get_age(self):
    return self.__age

  #Private Setters
  def __set_name(self, new_name):
    assert type(new_name)==str
    self.__name=new_name

  def __set_sex(self, new_sex):
    assert type(new_sex)==str
    self.__sex=new_sex

  def __set_age(self, new_age):
    assert type(new_age)==int
    self.__age=new_age
```

لحد الوقتى مش فى فرق عن ال مثال السابق غير إن ال setters/getters بقو private لأن سبقهم 2 underscores
زى مانتو شايفين فكدا مش هنقدر نتعامل معاهم بصورة مباشرة :S

بكل بساطة دا اللى إحنا عايزينه لأننا هنعمل Properties نستخدمها ولكن بالشروط اللى حددناها فى ال getters/setters
رمز:

```
  #Properties
  Name=property(fget=__get_name, fset=__set_name)
  Sex=property(fget=__get_sex, fset=__set_sex)
  Age=property(fget=__get_age, fset=__set_age)
```

property هى type بيشمل عدة اشياء اللى يهمنا منها fget, fset
وليكن إننا عملنا Object بإسم p1 كالتالى

رمز:

```
p1=Person('Ahmed', 'M', 18)
  print p1.Name
```

كأننا إستخدمنا

رمز:

```
p1.get_name()
```

رمز:

```
  p1.Name='YOUSSEF
```

'

كأننا إستخدمنا

رمز:

```
p1.set_name('YOUSSEF')
```

رمز:

```
  print p1.Name
```

```
print p1.Age
```

هنا حاولنا نعمل p1.set_age('Whatever') واحنا محددين ب assert ان لازم يكون ال type الخاص بال argument اللى هتتباصى لل set_age يكون من النوع int !

رمز:

```
p1.Age="WHATEVER" #AssertionError
print p1.age
```

ال Class بصورته النهائية

رمز:

```
class Person(object):
    #Constructor
    def __init__(self, name, sex, age):
        #Initializing fields
        self.__name=name
        self.__age=age
        self.__sex=sex

    #Private Getters
    def __get_name(self):
        return self.__name

    def __get_sex(self):
        return self.__sex

    def __get_age(self):
        return self.__age

    #Private Setters
    def __set_name(self, new_name):
        assert type(new_name)==str
        self.__name=new_name

    def __set_sex(self, new_sex):
        assert type(new_sex)==str
        self.__sex=new_sex

    def __set_age(self, new_age):
        assert type(new_age)==int
        self.__age=new_age

    #Properties
    Name=property(fget=__get_name, fset=__set_name)
    Sex=property(fget=__get_sex, fset=__set_sex)
    Age=property(fget=__get_age, fset=__set_age)
```

```
if __name__=="__main__":
    p1=Person('Ahmed', 'M', 18)
    print p1.Name
    p1.Name='YOUSSEF'
    print p1.Name

    print p1.Age
    p1.Age="WHATEVER" #AssertionError
    print p1.age
```

Operator Overloading in Python

4+1 دى إستخدام ال+Operator وهو إن يجمع عددين
2*1 إستخدام ال Operator * هنا إنه يضرب عددين
2-1 إستخدام ال – Operator هنا إنه يطرح عددين ولكن !

هل ينفع يكون ل Operator اكثر من إستخدام ؟
اها مثلا + Operator بيستخدم فى عمل Concatenation بين ال Strings

رمز:

```
>>> 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 اكثر 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

رمز:

```
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 Methods 3 كالتالى مثلا

رمز:

```
    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

رمز:

```
    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 الكلاس كالتالى

رمز:

```
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

رمز:

```
>>> 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

رمز:

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

```
Operators you can overload:

+ __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__
```

Inheriting Types

Python بيتميز بمرونة كبيرة من ناحية ال OOP ولكن كان هناك دائما الفرق بين ال Class وال Type وكان فى Limits فى التعامل ولكن الكلام دا اصبح من الماضى lol

تقدر تورث type ل Class بإسلوب بسيط للغاية كالتالى

رمز:

```
class eString(str):
    '''eString(str) -> Inherits str [type]'''
    def __init__(self, string=None):
        str.__init__(self)
```

دا Class كتبته توضيحى للعملية دى :

It supports
[*]All of str functions
[*]implode function as wrapper to (joiner.join(sequence))
[*]inserting
[*]setting slices
[*]shuffling
[*]strcmp (cmpTo)
[*]strncmp (nCmpTo)[*]md5
[*]sha1 #You may add more!
[*]substring method as wrapper to s[from:to]

[*]ROT13 Encoding
[*]ROT13 Decoding
[*]Reversing
[*]getType

```
#!bin/python
#############################################################
# Written by : Ahmed Youssef
# Purpose : Making strings-handling more easeier
# Date : 9-3-2007
# Site : Programming-Fr34ks.net
#############################################################

def implode(joiner, listOfWords):
   '''joiner.join(lisOfWords)'''
   return joiner.join(listOfWords)

class eString(str):
   '''eString(str) -> Inherits str [type]'''
   def __init__(self, string=None):
      self.string=string
      str.__init__(self)

   def getPercentage_slice(self, percentage):
         #  total -> x
         #  100   -> value
         value=int(percentage[:-1])
         needed=((len(self)*int(value))/100)
         return self[:needed]

   def splitV(seperator):
      L=[]
      for char in self:
         L += [char+seperator]

   def hasChar(self, char):
      if char in self.string:
         return True
      return False

   def getType(self):
      return type(self)

   def insert(self, value, index):
      '''An easy way to insert a char/string into a string'''
```

```
    _first=self.string[0:index]
    _last=self.string[index:]
    self.string=eString(_first+value+_last)
    return self.string

def reverse(self):
    '''String[::-1]'''
    return self.string[::-1]


def shuffled(self):
    '''Shuffling letters'''
    import random as rnd
    tmp=list(self.string)
    rnd.shuffle(tmp)
    return eString(''.join(tmp))

def shuffle(self):
    import random as rnd
    tmp=list(self.string)
    rnd.shuffle(tmp)
    self.string=''.join(tmp)

def toUpper(self):
    self.string=self.string.upper()

def toLower(self):
    self.string=self.string.lower()

def length(self):
    '''len(eString_Object)'''
    return len(self)

def md5(self):
    '''MD5 hash'''
    try:
        import md5
        tmp=md5.new(self.string).hexdigest()
        return tmp
    except:
      ImportError, "Error importing md5 module..."


def sha1(self):
    '''sha1 hash'''
    try:
        import hashlib
        tmp=hashlib.sha1(self.string).hexdigest()
        return tmp
```

```
        except:
            ImportError, "Error importing hashlib module..."

    def cmpTo(self, other):
        '''Compares eString with another object'''
        if self.string==eString(other):
            return 1
        else:
            return -1

    def nCmpTo(self, other, _from, _to):
        if self.string[_from:_to] == other[_from:_to]:
            return 1
        else:
            return -1
    def substring(self, _from, _to):
        '''Same as string[_from:_to]'''
        return self.string[_from:_to]

    def encodeRot13(self):
        '''Applies ROT13 algorithm on eString Object'''
        originalChars=list('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
,! 0123456789')
        convChars=list('NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm,!
0123456789')
        rotList=[]

        for char in self.string:
          if char in originalChars:
             rotList.append(convChars[originalChars.index(char)])

        rotString=''.join(rotList)
        return eString(rotString)



    def decodeRot13(self):
        '''Decode ROT13 string'''
        originalChars=list('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
,! 0123456789')
        convChars=list('NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm,!
0123456789')

        restoredList=[]
        for char in self.string:
          if char in convChars:
             restoredList.append(originalChars[convChars.index(char)])

        restoredString=''.join(restoredList)
```

```
        return eString(restoredString)

    def __eq__(other, self):
        other = eString(self)

    def setslice(self, _from, _to, value):
        '''Simple way to set a slice'''
        _first=self.string[:_from]
        _last=self.string[_to:]
        self.string=eString(_first+value+_last)
        return self.string

    #-You may add more functions to fit your needs.



if __name__=="__main__":
    s=eString("Hello, World!")
    print "The string is : [%s]"%s
    print "Suffled string : [%s]"%(s.shuffle())
    print "%s[%d:%d] is : [%s]"%(s, 0, 4, s.substring(0, 4))
    print "md5 of [%s] is : [%s] "%(s, s.md5())
    print "sha1 of [%s] is : [%s]"%(s, s.sha1())
    _ins=s.insert('CRUEL ', 7)
    print _ins
    c=s.setslice(0, 5, "CRUEL")
    print c
```

Python and MySQL

Python اثبتت كفاءتها فى العديد من المجالات منها التعامل مع الداتابيزس، هنختص ال MySQL بالذكر فى موضوعنا اليوم .

هنحتاج نحمل ال MySQLdb عشان نقدر نتعامل مع ال MySQL Server بإستخدام ال Module MySQLdb

اول شئ لازم نعمل import لل Module عن طريق

```
import MySQLdb
```

لعمل Connect على ال MySQL Server هتحتاج إلى

1- ال Host
2- Database
3- ال Username
4- ال Password

فهنعمل 3 متغيرات تعبر عن المتطلبات اللازمة لعمل ال Connection كالتالى

```
HostName='localhost'
DataBase='myDB' #او اى إسم لقاعدة بيانات عندك
UserName='Administrator' # غيره لليوزر بتاعك
```

غيرها للباسورد بتاعك# 'Password='123456

نعمل Connection على ال MySQL Server بكل بساطة عن طريق التالى :

```
MyConnection=MySQLdb.connect(host=HostName, user=UserName, passwd=Password,
db=DataBase)
```

بعد مانعمل ال Connection هنعمل Cursor زى MySQLCommand عشان ننفذ عن طريقة ال Transactions

```
Cursor=MyConnection.cursor()
```

لتنفيذ ال transaction هتحتاج تستخدم ال Method بإسم execute كالتالى

```
MySQLTransaction = " "
```

ضيف ال Transaction بين علامتى التنصيص ، وبكل بساطة

```
Cursor.execute(MySQLTransaction)
```

بعد ماتخلص ال Transactions لازم تقفل ال Cursor وال MySQL Connection كالتالى :

```
Cursor.close()
MyConnection.close()
```

برنامج PhoneBook بيستخدم MySQL

Configurations file :

```
#!bin/python

##################################
# Written by : StrikerX
# Purpose : PyPhoneBook Configuration
# Date : 8-21-07
##################################

import MySQLdb as mysql


theHost="localhost"
theUser="root"
thePassword="123456" #Your password
theDatabase="PythonDB"
theTable="PythonBookTable"



def Create():
   global theHost
   global theUser
   global thePassword
   global theDatabase
   mysqlConnection=mysql.Connection(host=theHost, user=theUser, passwd=thePassword)
   cursor=mysqlConnection.cursor()
```

```
    create_db_statement="create database %s" %(theDatabase)
    cursor.execute(create_db_statement)
    use_statement="use %s"%(theDatabase)
    cursor.execute(use_statement)
    #Create Table
    create_table_statement='create table %s (FirstName varchar(30), LastName varchar(10),
phone varchar(10))'%(theTable)
    cursor.execute(create_table_statement)
    mysqlConnection.close()


if __name__ == "__main__":
    Create()
```

You should run it at the first time.

The phone-book

```
#!bin/python

#################################
# Written by : StrikerX
# Purpose : Phone-Book
# Date : 8-21-07
#################################

import MySQLdb as mysql
import pyPhoneConf


host_name=pyPhoneConf.theHost
user_name=pyPhoneConf.theUser
password=pyPhoneConf.thePassword
db_name=pyPhoneConf.theDatabase
table=pyPhoneConf.theTable

sqlConnection=mysql.connect(host=host_name, db=db_name, user=user_name,
passwd=password)
sqlCursor=sqlConnection.cursor()

def options():
    print '''
1-Add
2-Query
3-Update
4-Delete
5-View All
6-Quit
```

```
'''

def add_record():
   global sqlConnection
   global sqlCursor
   first_name=raw_input("Enter the first name : ")
   last_name=raw_input("Enter  the last name : ")
   phone_number=raw_input("Enter the phone number : ")
   insert_statement="insert into %s values ('%s', '%s', '%s')"%(table, first_name, last_name,
phone_number)
   sqlConnection.query(insert_statement)
   print "--Done--"

def query_record():
   global sqlCursor
   global sqlConnection
   queries=['FirstName', 'LastName', 'Phone']
   answer=int(raw_input('''Query by :
1- FirstName
2- LastName
3- Phone
# ?  '''))
   if answer==1:
         query_by='FirstName'
   elif answer==2:
         query_by='LastName'
   elif answer==3:
         query_by='Phone'
   else:
         exit()
   value=raw_input("Enter : ")
   statement="Select * from %s where %s='%s'"%(table, query_by, value)
   sqlCursor.execute(statement)
   for record in sqlCursor.fetchall():
      print record
   print "--Done--"


def update_record():
   global sqlCursor
   global sqlConnection
   phone_pre=raw_input("Enter phone number : ")
   first_name=raw_input("Enter the first name : ")
   last_name=raw_input("Enter  the last name : ")
   phone_number=raw_input("Enter the phone number : ")
   insert_statement="insert into %s values ('%s', '%s', '%s')"%(table, first_name, last_name,
phone_number)
   sqlConnection.query(insert_statement)
   print "--Done--"
```

```
def delete_record():
   global sqlCursor
   phone_number=raw_input("Enter the phone number : ")
   delete_statement="Delete from %s where Phone=%s"%(table, phone_number)
   sqlCursor.execute(delete_statement)
   print "--Done--"


def view_records():
   global sqlConnection
   global sqlCursor
   statement='select * from %s'%(table)
   sqlCursor.execute(statement)
   for record in sqlCursor.fetchall():
      print record
   print "--Done--"

options()

while True:
     option=int(raw_input("%> "))
     if option==1:
          add_record()
     elif option==2:
          query_record()
     elif option==3:
          update_record()
     elif option==4:
          delete_record()
     elif option==5:
          view_records()
     elif option==6:
          sqlConnection.close()
          exit()
     else:
          print "Unknown"
```

Command line arguments

ال cmdline arguments بيتمثل فى list فى python بإسم argv ودى موجودة فى ال sys module

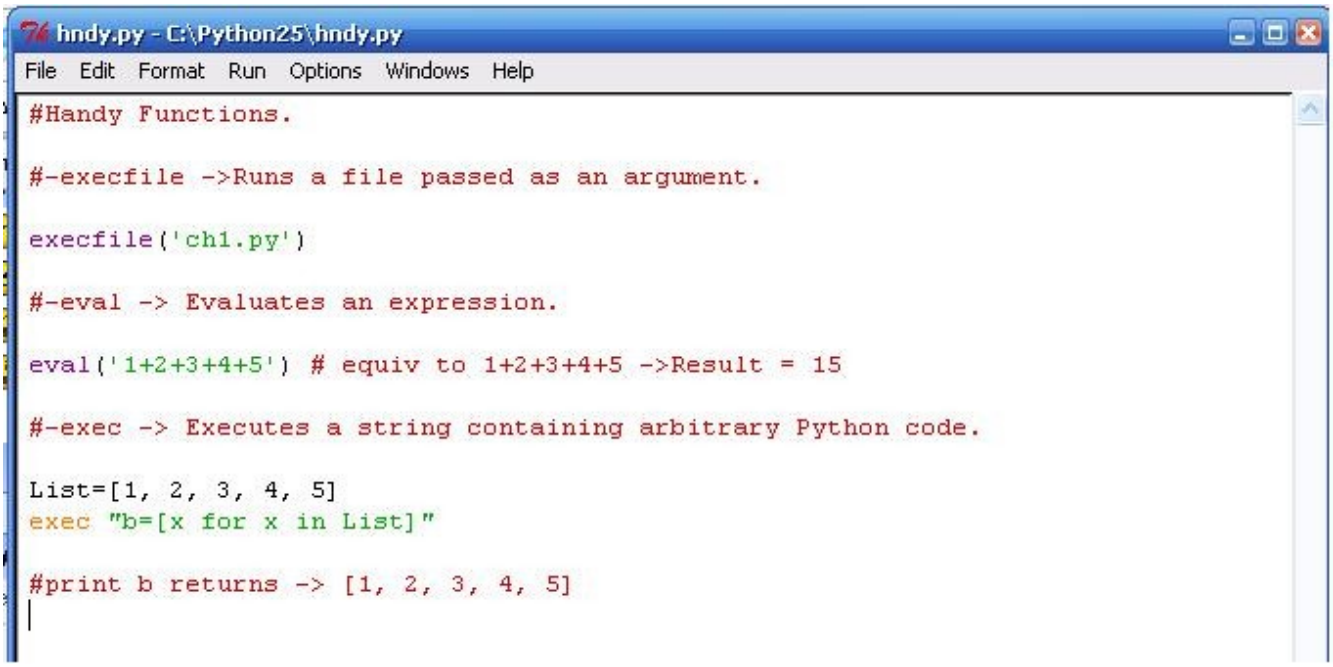
```
#!bin/python

# ./file.py  First Second Third Fourth
# ----------------------------------------
#      0     | 1   |   2     |  3   |  4
```

```
#----------------------------------------

#./file.py = argv[0]
#first = argv[1]
#second = argv[2]
#third = argv[3]
#fourth = argv[4]
```

Echo.py

```
#!bin/python

##############################################
# Written by : StrikerX
# Purpose : Echo-like tool written in python
##############################################

from sys import argv

def usage():
    "Usage : ./echo.py arg1 arg2 arg3 ..."

if len(argv) == 1 :
   usage()

for argument in argv[1:]: #To remove the script name.
   print argument, #comma to avoid the new line!
```

Execfile, eval and exec



Scripts

Password Generator :

```
#!bin/python

###################################
# Written by : StrikerX
# Purpose : Generating Passwords
# Date : 7-24-07
###################################

#-Imports
import random


def GenPassword():
        Contents=[ch r(x) for x in range(65, 123)] #All Letters. Note:Replace ch r with CHR in #lowercase :D
        Contents.extend(range(10))
        Length=8 #8 Chars!
        x=1
        password=""
        while x<=Length:
                password += str(random.choice(Contents)) #In case the choice was an integer!
                x += 1
    #import md5
```

```
        # Hashed=md5.new(password).hexdigest()
        # print Hashed

                print password #8 chars!

if __name__ == "__main__":
        GenPassword()
```

Touch.py

```
#!bin/python

############################
# Written by : StrikerX
# Purpose : touch-like tool
############################

from sys import argv

def usage():
        print "Usage : ./touch.py file1 file2 file3 ..."

if len(argv) == 1:
        usage()


for fileName in argv[1:]:
        f = open(fileName, 'w') #It's used to open a file to write or create it if it doesn't exist!
        f.close() #closing the file handler
```

Password Verifier.

```
#!bin/python

################################
# Written by StrikerX.
# Date : 8-9-07
# Purpose : Password Verifier.
################################


#- Only password that is mixed of (lowercase letters and digits is accepted!)
#- No uppercase letters.
#- Length 5 to 12
```

```
# Checking if it is mixed(lowercase/digits) or not.
def isMix(string):
        numberOfLowers=0
        numberOfDigits=0
        for char in string:
                if char.isupper() : #Uppercase letters aren't allowed!
                        return False
                if char.isdigit() :
                        numberOfDigits += 1
                if char.islower() :
                        numberOfLowers += 1
        if numberOfLowers > 1 and numberOfDigits > 1:
                return True
        return False


# Checking its length
def length(string):
        if len(string) > 5 and len(string) < 12:
                return True
        else:
                return False

if __name__ == "__main__":
        while True:
          password=raw_input("Enter a password [q to exit!]: ")
          if password == "q":
                        break
          if length(password) and isMix(password):
                   print "It's Fine!"
                else:
                        print "Rejected!"
```

Files Splitter/Combiner

```
#!bin/python

###################################
# Written by : StrikerX
# Date : 07-27-2007
# Purpose : Split & Combine
###################################

#-Imports
from os.path import getsize
```

```
#-Split function
def split(FILENAME, NUMBEROFCHUNKS):
	fileName=FILENAME
	numberOfChunks=NUMBEROFCHUNKS #To calculate the size
	#Renaming chunks
	listOfChunks=[str(fileName)+str(x) for x in range(1, numberOfChunks+1)] #Renaming
Chunks
	#List of objects!
	chunksObjects=[open(x, 'wb') for x in listOfChunks]
	sizeOfFile=getsize(fileName)/numberOfChunks
	#Creating file Object
	fileObj=open(fileName, 'rb') #Reading
	for chunkObj in chunksObjects: #Looping through each object and write data
		chunkObj.write(fileObj.read(sizeOfFile))
		chunkObj.close()


#-Join Function
def join(*files):
	#Create file
	fileName=files[0][:-1] #Strip the added number!
	fileObj=open(fileName, 'ab')
	#fileObjects
	fileObjects=[]
	for x in files:
		fileObjects.append(open(x, 'rb'))
	#Add it to fileObj
	for obj in fileObjects: #Loop through each object and write data
		fileObj.write(obj.read())
		obj.close()
	fileObj.close()

#-Menu
print "1 => Split File\n2 => Combine Chunks\n3 => About\n4 => Quit" #Simple menu

#-Choice
choice=int(raw_input("Enter your choice : "))


#-Testing the choice.
if choice == 1:
	FileName=raw_input("Enter filename : ")
	numberOfChunks=int(input("Enter number of chunks : "))
	split(FileName, numberOfChunks)
elif choice == 2:
	sFiles=raw_input("Enter files to combine with [, ] in between : ")
	listFiles=sFiles.split(", ")
	for File in listFiles:
		join(File)
```

```
elif choice == 3:
        print "Written by StrikerX"
else : exit()
```

Head.py

```
#!bin/python

#########################
# Written by : StrikerX
# Purpose : Head-Like
#########################


#--Imports--#

try:
        import sys
except ImportError :
        print "Error Importing the modules!"

fileName=sys.argv[1]

fileObject= open(fileName, 'r')
text=list(fileObject.readlines())
for Line in text[0:9]:
        print Line,

fileObject.close()
```

tail.py

```
#!bin/python

############################
# Written by : StrikerX
# Purpose : tail-Like Tool
############################

#-Imports
from sys import argv

def usage():
        print "%s <file>" %(argv[0])
        exit()

if len(argv) != 2:
        usage()
```

```
file_name=argv[1]
try:
	f=file(file_name, 'r')
	lines=f.readlines()
	print ''.join(lines[-11:-1]),
except:
	IOError, "ERROR!"
finally:
	f.close()

#End.
```

WC.py

```
#!bin/python

############################
# Written by : StrikerX
# Purpose : WC in python
############################
from sys import *
param = argv[1]
fileName = argv[2]

def usage():
	print "Usage : ./wc.py [param] [fileName]"

if len(argv) != 3:
	usage()
	exit(True)


def wChars(fileName):
	try:
		fH = open(fileName, 'r')
		txt = fH.read()
		fH.close()
	except IOError:
		print "Error!"
		raise SystemExit
	chars = len(txt)
	print chars

def wLines(fileName):
	try:
		fH = open(fileName, 'r')
		txt = fH.read()
		fH.close()
```

```
        except IOError:
                print "Error!"
                raise SystemExit
        lines = txt.count('\n') + 1 # The last line doesn't have '\n'
        print lines

if param == '-l':
        wLines(fileName)
elif param == '-c':
        wChars(fileName)
else:
         usage()
```

Cat.py

```
#!bin/python

#-################################
# Programmer : StrikerX
# Purpose : Cat Python PowerTool
###################################

#cat.py state fileName
#cat.py -h

import sys
import os


def read():
        fileName = sys.argv[2]
        f = open(fileName, "r")
        for line in f.readlines():
                print line
        f.close()


def append():
        arr = []
        fileName = sys.argv[2]
        if os.path.exists(fileName):
                f = open(fileName, "a")
                buf = "string"
                while buf != "":
                        buf = raw_input("")
                        arr.append(buf)
                for x in arr:
                        f.write(x + '\n')
```

```
                f.close()



def write():
                fileName = sys.argv[2]
                arr = []
                f = open(fileName, "w")
                buf = "string"
                while buf != "":
                        buf = raw_input("")
                        arr.append(buf)
                for x in arr:
                        f.write(x + '\n')
                f.close()

def h():
        print "\tParam -> State"
        print "\n\t > -> Write\n\t>> -> Append\n\t-r -> read\n\t-a -> About\n\t-h -> Help Menu"




param = sys.argv[1]

if param == "-h":
        fileName = ""
        h()
elif param == "-r":
        read()
elif param == "-w":
        write()
elif param == "-a":
        append()
else:
        print "an illegal parameter !!"
```

CmdCalc.py

```
#!bin/python

from sys import argv

if len(argv) < 1:
   print "Usage : ./cmdCalc.py"

while True :
```

```
    total = raw_input("")
    if total == "q":
       break
   else : print eval(total)
```

PyShell.py

```
#!bin/python

###################################
# Written by : StrikerX
# Date : May 10 07
# Purpose : simple shell in Python
###################################
from os import *  # to use getcwd(), listdir(), chdir(), system() .
from sys import * # to use exit().

print "Welcome to Python Shell ! "

command = ""

while (command != "exit"):
        command = raw_input(getcwd() + " %>#")
        command = command.strip()
        if command.strip()[:2] == 'cd':
                        chdir(command.strip()[3:]) #getting the Path to be changed 2 !
        elif command == "ls":
                for x in listdir(getcwd()):
                        print x
        else:
                system(command)

print "you are logging out of Python Shell ! "
exit() #EXIT
```

Encrypt/Decrypt files (XOR)

```
#!bin/python

#Encrypt/Decrypt files

from sys import argv
from operator import xor #Encrypt/Decrypt
from StringIO import StringIO
```

```
#For Streams, one to read and one to write !

program_name=argv[0]
option=argv[1].lower() # -e/-d
options_list=['-e', '-d']
file_name=argv[2]

def usage():
        print '''
-e [encrypt] : %s -e file
-d [decrypt] : %s -d file
'''%(program_name, program_name)
  exit()

if not option in options_list:
        usage()

def encrypt(file_name):
        password=raw_input("Enter the password : ")

        f1=open(file_name, 'rb')
        contents=f1.read()
        f1.close()
        f2=open(file_name+'.pyEnc', 'w')
        sReader=StringIO(contents)
        sWriter=StringIO(contents)
        #Set the position to 0
        sReader.seek(0)
        sWriter.seek(0)

        start=0

        for Byte in range(len(contents)):
                if start>=(len(password) - 1):
                        start=0
                passCharOrd=ord(password[start])
                start += 1
                #Encrypt each byte!
                ch=sReader.read(1)
                by=ord(ch)
                value=xor(by, passCharOrd)
                sWriter.seek(Byte)
                sWriter.write(c hr(value))
        sWriter.seek(0)
        f2.write(sWriter.read())
        f2.close()

        sReader.close()
        sWriter.close()
```

```
def decrypt(file_name):
        password=raw_input("Enter the password : ")

        f1=open(file_name, 'rb')
        contents=f1.read()
        f1.close()
        f2=open(file_name[:-6], 'w')
        sReader=StringIO(contents)
        sWriter=StringIO(contents)
        #Set the position to 0
        sReader.seek(0)
        sWriter.seek(0)

        start=0

        for Byte in range(len(contents)):
                if start>=(len(password) - 1):
                        start=0
                passCharOrd=ord(password[start])
                start += 1
                #Encrypt each byte!
                ch=sReader.read(1)
                by=ord(ch)
                value=xor(by, passCharOrd)
                sWriter.seek(Byte)
                sWriter.write(c hr(value))
        sWriter.seek(0)
        f2.write(sWriter.read())
        f2.close()

        sReader.close()
        sWriter.close()
        f2.close()


if option == '-e':
                encrypt(file_name)
elif option == '-d':
                decrypt(file_name)
```

Convert from Dec2Binary

```
#!bin/python

################################################
# Writter : StrikerX
# Purpose : Convert from decimal to bianry
################################################
```

```
Decimal = int(raw_input("Enter the number : "))

def convToBinary(Decimal):
   BinaryHolder=0
   BinaryResult=""
   while Decimal > 0 :
     BinaryHolder = Decimal % 2
   BinaryResult += str(BinaryHolder)
   Decimal=Decimal/2
   array = list(BinaryResult)
   myTrueResult = ''.join(reversed(array))
   return myTrueResult

#End.
```

Min/Max Functions

```
#!bin/python

def Min(array = []):  # Min function starts here .
        if (len(array) == 0): # To avoid passing an empty array .
                print "No elements in array"
        else:
                current = array[0]
                for i in array:
                        if i < current:
                                current = i
                return current
# The end of Min function .



def Max(array = []): # Max function starts here .
        if (len(array) == 0):
                print "No elements in array"
        else : # Our main work starts here .
                current = array[0]
                for i in array :
                        if i > current:
                                current = i
                return current

# The end of Max function .
```

Factorial.py

```
def factorial(n):
    if n == 0 :
       return 1
    elif n == 1 :
       return 1
    else:
      return n*fac(n-1)
```

md5 of string passed as cmd argument

```
#!bin/python

import sys, md5

def usage():
    print "Usage : ./program.py string"

if len(sys.argv) != 2:
   usage()
   exit()

string = sys.argv[1]
stringHashed = md5.new(string).hexdigest()

print stringHashed
```

Simple String Tokenizer

```
class strTokenizer:

        def __init__(self, string):
                        assert type(string)==str
                self.tokens=string.split()
                self.__current=0

                def has_next(self):
                        if self.__current < len(self.tokens):
                                return True
                        return False

                def get_current(self):
                return self.tokens[self.__current]

        def get_next(self):
                        if self.has_next():
                value=self.tokens[self.__current+1]
                                self.__current += 1
```

```
                return value
        else:
                raise Exception, "There's no next!"

def get_previous(self):
        value=self.tokens[self.__current-1]
        self.__current -= 1
        return value
```