

بسم الله الرحمن الرحيم

إحترف بايثون الآن

تم تحميل هذا الكتاب من موقع كتب

www.kutub.info

للمزيد من الكتب في جميع مجالات التقنية ، تفضلوا بزيارتنا

خاضع لرخصة FDL

Copyright (c) 2003 under terms of FDL license

إهداء

اهدى هذا الكتاب لكل مبرمج على طريق الإحتراف

مقدمه

هذا الكتاب مقسم لحلقات ف الشرح وذلك لظنى أن هذا سيكون أفضل ف إستيعاب القارئ
هذا الكتاب خاضع لرخصة GNU FDL

أى نسخ أو تصوير أو إقتباس لا يعد مخالفا للقانون ولكن إذا إستخدم ف عمل منتج مشتق
فإنه يخضع لرخصة FDI

كما لايمكنك الإدعاء بأنك من قمت بالقيام بهذا العمل ويجب عليك الإشارة لمؤلفه الأصلي
لاحظ أنا غير مسؤول عن أى خطأ مادی يقع عليك أو على جهازك وطبعاً ولا عن أى خطأ معنوى
لاحظ واجهتني بعض المشقه في الكتابه باللغه العربيه فأعذرني إن كانت الألفاظ غير سليمة لغوياً فلو أنه
بالإنجليزيه لكان الأمر سهلاً ولكن فليعيننا الله

كلمة عامة

بايثون من اللغات القلائل التي حققت المعادله الصعبه حيث أنها جمعت بين القوة والسهولة والمرونه مما
يجعلها مناسبه لكل من المبتدئين والمحترفين
برمجه ممتع مع بايثون

كلمة عن الكتاب

هذا الكتاب لايشترط أى خبره برمجيه سابقه وإن كانت موجوده فلاضرر

بعد قراءتك لهذا الكتاب بحمد الله ستكون قد علمت عن بايثون مايسمح لك بوضع لقب مبرمج بايثون هذا الكتاب جزء من سلسلة إحتراف اللغات البرمجة وهى سلسله مجانيه تهدف لتثقيف الشباب العربى ويتعامل هذا الكتاب تدريجيا مع مستوى القارئ بفرض أنه لايعلم شئ عن البرمجه ملاحظة تمت الإستعانه ببعض الكتب والمقالات ف كتابه هذه الحلقات التى سوف نتناول فيها إن شاء الله مقدمه صغيره عن لغة بايثون ولكن اولا

ماهى لغة بايثون؟؟؟

هى لغة تفسيريه (اى تحتاج برنامجا مفسر للأكواد للتنفيذ) وذات واجهه تفاعليه وتدعم البرمجه الموجهه واقوى مايميزها هو سهولتها وشعبيتها فهى أسهل من بيرل (لغه من أسهل اللغات) وتكاد تكون تعمل على هذه اللغة ستتغير حياتك.وتستطيع جميع أنظمة التشغيل ويكمن ان تفعل بها ماتريده بمجرد ان تتذوق طعم في الأكواد في لغة C++/Cدمجها برنامج على قناة CNNسميت بايثون بهذا الإسم على إسم “Monty python's Flying Circus”

سهولة بايثون أى أنك عندما تقرأ كودا بهذه اللغة تكون كاءك تقرأ إنجليزى وقواعدها سهله للغاية وحره ممايتيح لك التعديل فيها كما تريد ولغة عالية المستوى تبعدك عن تعقيدات إدارة الذاكره وغيره

من الأنظمة التى تعمل عليها هذه اللغة :

Linux/FreeBsd/Windows/macintosh/Solaris/VMS/(OS/2)/Amiga/BeOS/Palm
OS/QNIX/Psion/Playstation/VxWorks/Sharp Zaurus/
Acorn Risc Os/Windows CE,/Pocket PC
وأوضح انها تعد اسهل اللغات وهى تعد الخيار الأول للمبتدئين ولكنها لغه قويه ليست لعبه

كاتب هذه اللغة: Guido van Rossum هولندى الجنسية

ومكتبتها القياسيه :اكثر من رائعه فتفيدك ف الحسابات المعقده والتطبيقات GUI وكل شئ والتعامل مع الملفات وغيرها لاحظ انها تعد مثل بيرل ف القوه من ناحية الويب لاحظ انك يمكنك التعديل ف كودها المصدري (سورس اللغه (المكتوب بلغة سي)) كما تريد ويناسب ذوقك من اهم المواقع التى تفيدك هى

www.python.org
www.techbooksforfree.com

هناك عشرات الالاف من المستخدمين حول العالم يمكنهم مساعدتك غير طبعا مؤسس اللغة
استمع لرأي ذلك الكاتب **Eric S.Raymond**
وهو يعد من أعظم الكتاب والمبرمجين
" ان بايثون اصبحت هي لغته المفضله "
Bruce Eckel يقول
وهو كاتب Thinking in C++ و Thinking in Java
" يقول ان بايثون ربما هي اللغة الوحيدة التي تجعل العمل أسهل من اجل المبرمج
والعديد "

كتب مهمة لمبرمجي بايثون

انصح بقراءة

- 1-Non –programmer Tutorial for python for josh cogliati
- 2-byte of python for swaroop
- 3-the official documentation of python
- 4-Dive in python
- 5-Thinking in python

دعم بايثون على جهازك

بايثون على ويندوز

هناك نوعان من بايثون على ويندوز

ActivePython>>>free one and recommended

تستطيع تنزله من هنا

www.activestate.com/Products/ActivePython/

لو بتستخدم 98/me

هتحتاج تنزل windows Installer 2.0

official python>>>free but for developers u may use it

بايثون على ماك

أمامك خيارين صارت بايثون تأتي مع النظام Mac OS X

وهي تنصيب بايثون أولا أرجح أنك ستنصبها وهذا جيد

على Mac OS X10.2

ستجد أن هناك نسخه منسبه على الجهاز ولكن تتعامل مع سطر الأوامر فإن كنت مرتاحا معه فلا مشكله

لك سوى ف XML parser بالنسبه
حيث أنه ليس من ضمنها فبالتالى ستحتاج لتنصيب الإصدار كاملا
لتشغيله ستفعل الخطوات الآتية

/Applications

/Utilities

D.click on Terminal

ثم إكتب python

للخروج Ctrl ^D

للحصول على إصدار 2.3

press

أفضل أن تستخدم نظاما مفتوح المصدر كأنظمة جنو أنا افضل لينكس
لاحظ إذا لم تكن تستخدم IDLE

إستخدم احد محررات النصوص مثل Kwrite, Gedit

هتلونك النص وكده وذلك ف حال إن لم تستخدم vi or emacs

لاحظ انا اكره notepad , word pad

لأنهم محررات ضعيفه

لاحظ Cool edit رائع

ولكن يمكنك إستخدامهم القاعده العامه لتشغيل الأكواد ان تحفظها بإمتداد py

وتعطى التصاريح chmod +xw filename.py

وبعد ذلك تستدعى المترجم وتكتب إسم الملف

python filename.py

لاتعطى تصاريح ف ويندوز لاحظ إنك هتنزل المترجم طبعا فى ويندوز اولا هيا بنا

```
#!usr/bin/python
```

```
print "Hello world \n"
```

```
print 'I am talking from the wonderful python \n'
```

الأول حددنا مسار مفسر الكود وهذا المسار خاص بأنظمة يونكس وتغيره حسب مسار المفسر ولو ف

ويندوز ولا أى مشكله تغير المسار للمكان الموجود فيه المفسر

تانى سطر هو تعليمة print=printf=system.out.println

بالسى والجافا وبراحتك بعدها " او " بس لازم تقفلها

وهى دالة إخراج ممكن تستخدم مكانها

دالة الإخراج الأساسيه

بس لاحظ هتضطر تضيف مكتبته ع العموم سنتعرض لذلك بالتفصيل قريبا إن شاء الله

```
sys.stdout.write()
```

```
#!/usr/bin/python
import sys
sys.stdout.write("Hello World! \n") #\n=newline
```

\n بتديك سطر جديد طبعا عارفها لو جربت سى او بيرل وغيرها
إيه رأيك بجد سهله صح؟

التعليقات بتستخدم **#** وتكتب التعليق تختلف عن سى **/* */**
لاحظ أول سطر ف برنامجك تعليق خاص بتحدد فيه مسار المفسر اللى ستقوم بإستخدامه ال **shell**

```
#!/usr/bin/python
#This a comment
#This is another comment and go on
print "This program is just comments nothing else \n"
```

المتغيرات والثوابت يمكن لك تعريفها عادى وليس مثل لغة سى

```
#!/usr/bin/python
a=5
b=2
print "a+b = ",a+b          #adding
print "a x b = ",a*b        #multiply
print "a / b= ",a / b       #dividing
print "a^b= ",a**b
```

ملحوظة لمبرمجى C/C++

لايوجد مايسمى ب **char** فى بايثون وأظن أنك لن تحتاجه

الحلقات والشرط

```
#!/usr/bin/python
x=[1,2,3,4,5]
for l in x :
    print l
    باستخدام for ده حلقه تكراريه صغيره هتطبع 1 2 3 4 5 رأسيا
```

```
#!/usr/bin/python
x=1
while x<100 :
    x=x+1
    print x
```

الناتج هيطبعك الأرقام من 1 ل 100 على صورة رئيسيه

```
#!/usr/bin/python
x=1
while x<100 :
    x=x+1
    print x,
```

الناتج هيطبعك الأرقام من 1 ل 100 على صورة افقيه

مثال صغير برنامج باسوورد وقاعدهIf

```
#!/usr/bin/python
x=raw_input("what's ur name : ")
if x=="ahmed":
    print 'hello ahmed'
elif x=="l1nUx3r":
    print 'welcome l1nUx3r'
else: print 'u r not allowed to access'
```

لاحظ يمكنك استخدام `input=raw_input`
إذا كان المدخل عددي استخدم `int(input("number:"))`

بس إيه رأيك بجد؟ لغة سهلة لاحظ حاول ان تبتعد عن سى او سى ++ ف البرمجه طالما لاتحتاج للتحكم ف الجهاز بتلك الدرجه فاستخدم لغة تفسيريه تهدر موارد النظام ك بايثون وبيرل

شروط للمتغيرات

لا تكون كلمه محجوزه ف اللغه مثل **class**
يجب ان يكون الحرف الأول حرف ابدى سواء كبير أو صغير أو يبدأ ب
underscores(_)
يمكن أن يحتوى على أرقام
تختلف المتغيرات باختلاف الإسم حيث **var1#!Var1**
لا يحتوى على مسافات

نسق الكتابة

هناك عدة أساليب ف الكتابة مثل إنك تكتب أول حرف كبير
مثال

myname
Myname
myName

الكلمات المحجوزه Reserved words

هى كلمات طبعا لايحوز لك إستخدامها بتسمية المتغيرات

#	and	elif	global	or	#
#	assert	else	if	pass	#
#	break	except	import	print	#
#	class	exec	in	raise	#
#	continue	finally	is	return	#
#	def	for	lambda	try	#
del	from	not	while	#	

ملخص الحلقة

تعرفنا ف هذه الحلقة على لغة بايثون وكاتبها وآراء العديدين حولها
تعرفنا على اللغات التفسيريه وعلى التعليقات وكيفية تعريف المتغيرات
والثوابت وشرطية if

الكلمات المحجوزة
وظيفة الإخراج `print,sys.stdout.write`

يجب أن تعطى التصاريح حتى يعلم النظام أن هذا النص قابل للتنفيذ

الحلقة الثانية سنتناول فيها إن شاء الله بعض أهم الدوال ف بايثون وطريقه تعريفها

مقدمه عن الدوال

معنى الدوال : هى عبارة عن جزء من الكود تمت كتابته مره واحده ويتم إستخدامه كثيرا جدا فتم وضع ذلك الكود جاهزا بإسم داله

ملحوظه افضل إستخدام IDLE اثناء التجربه حتى تكون الإجاباه لحظيه

يمكنك إستخدام محرر نصوص كما قلنا ولكنك ستضيف السطر التالى
#!/usr/bin/python

دالة len

بتعطيك عدد ما وبالمثال عدد حروف الكلمه

```
>>>a='word'
>>>len(a)
4
```

هناك إستخدامات متقدمه سنتحدث عنها قريبا

دالة Range

```
>>>range(10)
[0,1,2,3,4,5,6,7,8,9]
>>>range(0,10,3)
[0,3,6,9]
>>>a=['ahmed','went','there']
      for x in range(len(a)):
          print x ,a[x]

0 ahmed
1 went
2 there
```

هذه الداله بتكون مصفوفه من عدد العناصر التى تحددھا مثل المثال الأول

3 وفيها الأعداد تتزايد بمقدار 10 إلى 0 وف المثال الثانى تكونت مصفوفه من
وف المثال الثالث إستخدمنا جملة دواره for
وذلك لترتيب المصفوفه كل عنصر مع ترتيبه ف المصفوفه

داله pass

من غير ضحك الداله ده مش بتعمل حاجه خالص سنتناولها ف تعريف الدوال

دالة chr

هذه الداله تقوم بتحويل القيمه المدخله من جدول الآسكى إلى الرموز الأبجديه والأرقام

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

دالة ord

هذه الداله عكس سابقتها تقوم بتحويل المدخلات إلى القيم المناظره من جدول الآسكى

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

دالة help

تقوم بها بالإستعلام عن أى شئ تريده مثلا
هذه المخرجات من على جهازى

```
>>>help('help')
```

Welcome to Python 2.3! This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at <http://www.python.org/doc/tut/>.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and

return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules", "keywords", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose summaries contain a given word such as "spam", type "modules spam".

داله dir

داله تستخدم ف عرض محتويات المكتبيه اللتى تريدها لاحظ المثال

```
>>>import sys,fibo
>>>dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__name__', '__stderr__',
 '__stdin__', '__stdout__', '_getframe', 'api_version', 'argv',
 'builtin_module_names', 'byteorder', 'callstats', 'copyright',
 'displayhook', 'exc_clear', 'exc_info', 'exc_type', 'excepthook',
 'exec_prefix', 'executable', 'exit', 'getdefaultencoding', 'getdlopenflags',
 'getrecursionlimit', 'getrefcount', 'hexversion', 'maxint', 'maxunicode',
 'meta_path', 'modules', 'path', 'path_hooks', 'path_importer_cache',
 'platform', 'prefix', 'ps1', 'ps2', 'setcheckinterval', 'setdlopenflags',
 'setprofile', 'setrecursionlimit', 'settrace', 'stderr', 'stdin', 'stdout',
 'version', 'version_info', 'warnoptions']
```

إستخدمنا import

حتى نستطيع إلحاق مكتبة sys,fibo

سنتعرض لهما إنشاء الله ف شرح بعض المكتبات

دور dir

قامت بعرض كل الدوال التى تحتويه المكتبيه

حسنا لقد تعرضنا لبعض الدوال لاحظ انه توجد ف المكتبيات دوال مهمه
للغايه سنتعرض لبعضها حين نشرح بعض هذه المكتبات

تعريف الدوال

لاحظ الأمثله التاليه

```
>>>def hi():          #defining function called hi
    """this function to print hi"""    #show what does it do
    print 'hi'          #this what the function do just print hi
>>>hi()               #we used the function
hi
```

ماهى def؟
 هى كلمه مفتاحيه نخبر بها بايثون بأننا نريد تعريف داله
 ماهى hi():
 هى الداله التى نريد إنشاءها ويجب ان تعقب إسمها ب ()
 ومتنساش النقطتين:
 بعد كده وضحنا ماتقوم به الداله "" "" ذلك سيظهر على شكل تعليق ف حال إستخدام
 IDLE
 وبعد كده اضفنا خصائص الداله وقد إختارنا ان نطبع الداله كلمه Hi
 وبعد كده إستخدمنا الداله مباشره hi()
 وإطبعت كلمه hi
 لاحظ المثال التالى على داله متقدمه شويه صغيرين

```
>>> def fib(n):  # write Fibonacci series up to n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while b < n:
...         print b,
...         a, b = b, a+b
...
>>> # Now call the function we just defined:
... fib(2000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

بنجمع فيها الرقمين والناتج نجمعه ع التالت وكده
 نشرح الكود بالتفصيل

عرفنا داله وداخلها متغير fib(n) والثانى =0 وحددنا وظيفتها "" "" وبعد كده عرفنا جواها متغيرين الأول =
 جملة while الشرطيه إستخدمناها لتحديد الشرط هو انا طالما المتغير الثانى اكبر من متغير الداله
 n
 يطبع العدد السطر قبل الأخير b ويأخذ العدد a قيمه العدد b الذى اصبح يساوى مجموع العددين السابقين

1000 لإستخدامنا الداه وجعلنا قيمه المتغير داخلها =
وطبع الجهاز الناتج كما تراه

مثال على دالة pass

```
>>>def hi():  
    pass  
>>>hi()  
>>>
```

قمنا بعمل داله لاتفعل شئ

بعض الردود عن أسئلته حول الحلقة السابقة
لماذا يجب ان اتعلم البايثون؟
اخى لأن هذه لغه سهله كما اوضحت ف الحلقة السابقة ولأنك لن تتعب نفسك ف تتبع الأخطاء بتلك الطريقه
كالتى مع C/C++

مالها لغات C#, Java, Perl, Pascal, Basic, C/C++

لن اتحدث عن لغة C/C++, Java, Pascal
فهى لغات عظيمه ولكن هناك شئ واحد انا هذه اللغات ان بحبها بجد ولكن من ناحية تتبع الخطأ تكون العمليه
مرهقه ومتعبه للغاية وإسئلوا عن ذلك مبرمجى سى وجافا وباسكال

C#

لغه جميله ولكن من عيوبها انها لن تعمل على بيئة تشغيل غير ويندوز إلا ببرنامج مونو على لينكس
وبعض الأنظمة الأخرى كـماك وهو ليس بكفاءه دوت نت ولكن جارى التطوير

Basic

هذه اللغه بدأت فيها تعلم البرمجه وبحمد الله لم اكن افهم شئ ومالأخر انا باكره هذه اللغه اللتى تذهب العقل
وأضف على ذلك انها لغه ركيكه حاولوا جعلها سهلها ولكنهم فشلوا بل محوها تماما وأرجو ألا يسألنى احد
حيث انا بها عيوب تملأ كتابا ولكن ليس لى حق النقد إذا اردت ان تستخدمها انت حر ولكنها تعلم عادات
برمجييه سيئه وايضا بالمقارنه بسى ليس للبيزيك اى لازمه معذره أخوانى مبرمجى بيبيك
وقد تكون فيجوال بيبيك دوت نت حلوه لكنى لن انتازل عن بايثون
صحيح اخى إن كنت تسأل عن الفيجوال بيبيك كنت ارجو ان تسألنى على دلفى بيئة التطور للغه العريقه
باسكال

لن اقوم بالرد عن اسئلته مقارنة بين لغات برمجييه

Pick ur programming language as u like

Perl

هي لغة جميلة وتتميز ف السلاسل النصيه وإداره الأنظمه والويب وإن لم تعجبك بايثون إستخدمها ولكن كود بيرل اقرب للسى ولكنه سهل ويشرف بايثون ان تكون من اللغات القلائل التى تنافس بيرل بإذن الله سيلي كتاب خواطر حول بايثون كتاب حول بيرل

اصحاب باقى الأسئلة قد أرسلت لكم الردود على المنتدى معذرة لم اجد المجهود لإضافة باقى الأسئلة

www.python.org
www.GNU.org
www.freetechbooks.com
www.linuxjournal.com
www.linux4arab.com

ملخص

تكلما ف هذه الحلقة عن بعض أهم الدوال الجاهزه
تناولنا تعريف الدوال
تحدثنا عن بعض لغات البرمجه

المصفوفات ف بايثون

المصفوفه (هي عباره عن مجموعه متغيرات) يعنى ممكن يكون العنصر الأول منها سلسله حرفيه وممكن يكون رقم اى شئ

اولا المصفوفات ف بايثون تتعامل مع اول عنصر فيها على انه رقم صفر والتالى مثله رقم واحد ومن الناحيه الأخرى ف المصفوفه يبدأ العد بالسالب فاخر عنصر يساوى 1- لاحظ المصفوفه هى مجموعه من العناصر ولا يشترط ان تكون ارقام قم بمراجعة الحلقة السابقه ف داله range

```
>>>a=[1,2,3,4,5,6,7,8,9]
>>>a
[1,2,3,4,5,6,7,8,9]
>>>a[0]
1
>>>a[2]
3
>>>a[-1]
9
```

لاحظ إذا وضعت ف المصغوفه كلمات تضعها بين علامتي تنصيص 's'

ملحوظہ لمبرمجبی perl /php

مفیش فرق ف بايثون بين علامات التتصيص سواء احاديه أو ثنائيه او ثلاثيه

```
>>>b=['python','c','c++','java','lisp']
>>>b[0]
python
>>>b[2]
c++
```

مثال ب استخدام مصفوفه for

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

```
>>> for x in a: print x
```

```
1
2
3
4
5
6
```

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

```
>>> for x in a: print x, # comma is used to avoid printing a newline
1 2 3 4 5 6
```

عندما نريد **إضافة** عناصر جديدة للمصفوفه ماذا نفعل؟؟

الإجابة ببساطه نكمل على المثال السابق `a.append(element)` فيضاف العنصر الذى نريده
وللترتيب نستخدم `a.sort()`

```
>>>a.append(0)
```

```
>>>a
```

```
[1,2,3,4,5,6,0]
```

```
>>>a.sort()
```

```
>>>a
```

```
[0,1,2,3,4,5,6]
```

بعض العمليات ع المصفوفات

```
>>>a=[1]
```

```
>>>a
```

```
1
```

```
>>>a.append(0) #add 0 to a
```

```
>>>a.append(2) #add 2 to a
```

```
>>>a.sort() #sort a
```

```
>>>a #ask what is a now?
```

```
[0,1,2]
```

```
>>>a[0:2] = [3, 12]  #repalce the first and the second elements with  
0 and 1
```

```
>>>a
```

```
[3, 12, 2]
```

```
>>>a[:]=[]  #remove all elements in a
```

```
>>>a
```

```
[]
```

```
>>>len(a)  #counting how many elements in a
```

```
0
```

```
>>>s=[0,1,2,3,4,5,6]
```

```
>>>s.append(9)
```

```
>>>s.pop()  #last in first out
```

```
9
```

```
>>>s
```

```
[0,1,2,3,4,5,6]
```

لاحظ هذا إستخدام اخر من إستخدامات دالة len وذلك ف عدد عناصر المصفوفة
أرجو ان اكون اعطيتم فكره ضئيله عن المصفوفات
هذا جدول لبعض اهم الوظائف

<code>list[a:b]=list2</code>	تكوين مصفوفة جديدة من عنصرين
<code>list.append(n)</code>	إضافة للمصفوفة
<code>list.sort()</code>	للترتيب المصفوفة
<code>list.insert(n,list)</code>	إضافة مصفوفة داخل مصفوفة
<code>list.index(X)</code>	رقم العنصر الذى تريده = X
<code>List.pop()</code>	اخر عدد أضيف واول عدد يخرج من المصفوفة
<code>del list</code>	تمحو المصفوفة
<code>list.pop(n)</code>	تعطى الرقم حسب الترتيب للمتغير الداخلى ف الدالة n
<code>list.remove(X)</code>	تحدد قيمة العدد الذى تريد محوه من المصفوفة
<code>List.reverse</code>	تعطيك المصفوفة بترتيب مخالف من اليمين للشمال
	يمكن لك ان تضيف عدة وظائف اخرى فلنعد ذلك واجبا

مع العلم ان المصفوفات المركبه والقواميس لن اتحدث عنها الآن
طيب ماذا الآن سأقول لكم سنتحدث إنشاء الله عن السلاسل النصيه الحلقه القادمه ومكتبه

sys

حسنا وحتى ذلك الحين إن كنت قد إستفدت اى شئ أرجو منك كمجرد محاوله كتابه كود لبرنامج بسيط باسوورد

كلمات سر ويقولك مرحبا لو باسوورد تانى يقولك إنت مش مسموح لك 3بحيث يقبل البرنامج والإصدار الثانى برنامج باسوورد يطلب الإسم و الباسوورد البرنامج الثانى هو برنامج مقارنه بين عددين يخبرك اى منهما الأكبر البرنامج الثالث برنامج يعطيك المضاعف المشترك الأكبر لعددين اى مساعده ف الأكواد لا تتردد

وبعد الدرس القادم إنشاء الله ستشعر بجد بالتقدم مع بايثون

بعض الردود عن الأسئلة

لماذا لأستخدم C/C++

إستمع إلى

لأنى أجبت عن هذا السؤال من قبل البايثون لغه قويه للغاية وقد لمحت ف احد الردود انها توازى بيرل ف القوه ولكن اكثر مايميزها هو انها مناسبه جدا للمبتدأ وليس تعلمها ف صعوبة تعلم السى طب هقولك على حاجه ف لغه اسمها كوبول ده ممكن تكتب فيها برنامج 200 سطر ولكن سى تقوم بنفس الوظيفه ف 30 سطر والبايثون تقوم به ف 20 سطر كفرض طب ثوانى انا هغير قاعده وسأظهر لك بعض الأكواد من عدة لغات

C language

```
#include<stdio.h>
```

```
void main()  
{  
    char* msg="Hello World!";  
  
    puts(msg);  
}
```

C++ language

```
#include <string.h>  
#include <stdio.h>
```

```
class Message // define a message class  
{  
private:  
    char msg[50];
```

```

public:
    Message(char* s)
    {strcpy(msg,s);};
    void print()
    {puts(msg);};
};

void main()
{
    m = new Message("Hello World!");
    m->print();
    delete m;
}

```

Perl language

```

#!/usr/bin/perl
$msg = "Hello World";
print $msg;

```

python language

```

#!/usr/bin/python
print 'hello world'

```

```

#!/usr/bin/python
class Hello{
    def hi(self):
        print 'hello world'

```

```

x=Hello()
x.f(self)

```

Java language

```

class HelloWorld{                                //define Hello World class
static void main (String args[]){
System.out.println("Hello world")                //print Hello world
}
}

```

إيه رأيك اخی لقد جعلتني اخالف قاعده هامه ولكن لايهمك الآن قرر مارأيك إيهم أسهل من هذه اللغات وتعلمه والترتيب مره اخرى باللغات التي إذا اردت ان تتعلم البرمجه فهي تعد من أساسيات المبرمج وثقافته الأساسيه

Python,Perl,Java,C/C++,lisp

أرجو مره اخرى منكم ای حد عنده فكره عن الליسب او البرلوج یراسلنی للأهمیه لاحظ اخی ان من قوة البايثون تعد بها شركة ردهات التطبيقات بتاعتها وشركه ماندریفا تستخدم بيرل لاحظ إنی بعد كل ذلك سأطلب منكم تعلم السی ولو أنك لن تكتب بها ای كود بل لمجرد التعلم من هذه اللغه العریقه

www.gnu.org
www.python.org
www.perl.org

ملخص

ف هذه الحلقة قمنا بالتعرض للعديد من خصائص المصفوفات

السلاسل النصيه وبعض المكتبات

نكمل معكم الحلقة الرابعه إن شاء الله فيها سنتحدث بصورة بسيطة عن السلاسل النصيه تعريفها هي مجرد حروف مدموجه معا (كلمة) او مفردة تحاط بعلامات تنصيب احاديه او ثنائيه او ثلاثيه

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

مثال علیہم

```
'this is a string'
"this is a string also"
"""This is another one"""
```

یاتری مجمعین؟؟ یعنی ای کلام داخل علامات تنصیص هو سلسله نصیه

التعامل مع المتغيرات

```
>>>i=5
>>>print i
5
>>>i=i+1
>>>print i
6
>>>s="hello how are u?"
>>>print s
hello how are u
>>>print i;           #for C/C++ and perl programmers
6
```

ف المثال السابق عرفنا ثابتا هو |
وأعطيناها القيمة 5

على قيمته الأصلية 1 وكانت هي إضافة | وبعد ذلك وضعنا قاعدة لمتغير يسمى
hello how are u? أيضا أعلننا عن سلسلة نصية
أليسيت بايثون سهله ف التعامل مع السلاسل النصية والمتغيرات؟؟

ملاحظه لبرمجى بيرل وسى وجافا مجرد عاده عند سى وبيرل وجافا استخدام الفاصله
المنقطه ; او لا إنت وراحتك ولكنها مجرد عاده عند سى وبيرل حتى لاتغير قواعدك الثابته
عندك

ندخل على مكتبة sys حالا فهى مكتبة جميله للغاية

ناخد مثال؟؟

ارجو ان تستخدم IDLE ولاضرر ف محرر النصوص

```
>>>import sys
>>>n=input("enter 0 to quit and 1 to advance")
>>>if n==0:
```

```
    sys.exit()
```

```
>>>elif n==1: print "hiiii"
```

لقد استخدمنا وظيفه sys.exit() الموجوده ف sys وهى نستغلها ف الخروج

من البرنامج

ملحوظه مبرمجى سى وجافا

لست مضطرا ف بايثون لإضافة الشرط بين اقواس

```
>>>sys.platform
```

هيديك ناتج مختلف من شخص لأخر حسب نظام تشغيله بس هو هيديك نوع النظام
ممکن طريقه اخرى

```
>>>from sys import * #it equals to import sys
```

```
>>>exit() #we used the function directly with out sys
```

الطريقه ده عشان مش تقعد تكتب إسم المكتبه وبعد كده الوظيفه لكن تختار الوظيفه على طول لكن افضل الأولى لكن لك الإختيار فهنا لا إجبار

```
>>>sys.copyright
```

هيديك رقم الإصدار اللي معاك

```
>>>sys.getwindowsversion()
```

ده لمستخدمى ويندوز فقط هيديك الإصدار بتاعك إيه

```
>>>sys.version()
```

intel32 هيديك نوع الجهاز ف الغالب

كده تمام؟؟

هناك وظائف اخرى لم اذكرها من اراد كل الوظائف ينظر إلى وثائق بايثون الرسميه ف [موقعهم](#) نصيحه إذا اردت الدعم هناك فى رومات البرمجه وانظمة التشغيل خاصه بتاعت الياا هو ده طبعاً بعد ماتحاول لأن بجد سيرد عليك ردا لن تتقبله لأن هناك فلسفه تعنى عدم المحاوله ف شئ لم يحاول فيه مريده ولكن إن لم تعرف فليس هناالك مشكله لكن المحاوله اولاً والدعم ثانياً

عوده للسلاسل النصيه

```
>>>name='ahmed'      #string
```

```
>>>if name.startswith('ahm'):
```

```
    print 'yes,the string starts with "ahm"
```

```
>>>if a in name:
```

```
    print 'yes, it contains an a
```

```
>>>L= ' _**_ '
```

```
>>>list=['python','is','extremly','powerfull']
```

```
>>>print L.join(list)
```

```
python_**_is_**_extremly_**_powerful
```

هناك بعض الوظائف الأخرى ستضاف بمشيئه الله ف الحلقات القادمه

لاحظ اخى إن لم تعمل معك اى من الأمثله انظر اولاً ف التصاريح وإجعلها 775 وبعد ذلك راسلنى إن لم تعمل

برمجه التطبيقات هي مرحلة متقدمه ولكن تعتمد بطريقه كبيره على البرمجه الكائنيه

برنامج صغير القاعده اخى التى تريدها كتبتها لك ف الأسفل *

برنامج يحول درجات الحراره عن طريق الدوال
ومره اخرى عن الطريقه المعتاده وقارن بينهما

$(Celsius = (f - 32) * (5/9)$

ملخص

لقد تعرفنا على مكتبيه sys
وبعض وظائف السلاسل النصيه,

الأخطاء الشائعه ومكتبيه string

ف هذه الحلقة سنتحدث بمشيئة الله عن بعض ف بعض الأكواد وسنتناول فيها إن

شاء الله مكتبية string

```
>>>x=0
```

```
>>>while x<0 print x
```

لن يعمل ذلك الكود لأنك قد نسيت :

```
>>>while x<0: print x
```

```
>>>x=0
```

```
>>>if x==0 print 'wrong'
```

لن يعمل أيضا وذلك لأنك قد نسيت :

```
>>>if x==0: print 'wrong'
```

```
>>>a=[0,1,2,3]
```

```
>>>for x in a print a
```

لن يعمل أيضا وذلك لأنك قد نسيت:

```
>>>for x in a: print a
```

```
>>>Print 'hello'
```

لن يعمل ذلك لأن لغة بايثون حساسه جدا للحروف

```
>>>print 'hello'
```

```
>>>5*(2/0)
```

لن يعمل ذلك الكود لأنه لايمكن القسمة على 0

```
>>>'1'+1
```

لن يعمل ذلك الكود لأنه لايمكن جمع سلسله نصيه مع عدد
إلا ف حال إستخدام جملةeval()

```
>>>eval('3+4')
```

```
7
```

```
>>>eval('3'+4')
```

```
34
```

```
>>>3+p*5
```

لن يعمل ذلك الكود وذلك لأنك لم تعرف المتغير او الثابتpفليست له قيمه تدخل ف العمليه الحسابيه

```
>>>def hi()
```

لن تعمل هذه الداله مع اننا قمنا بتعريفها ف حلقه سابقه وذلك لأننا نسينا :

```
>>>def hi():
```

هذه بعض الأخطاء الشائعة ولكن هناك جملة `try_except` سنقوم بشرحها هي وملحقاتها إنشاء الله ف حلقات قريبيه قادمه عوده السلاسل النصيه
ملخص لبعض الوظائف لهذه المكتبه

<code>String.upper(string)</code>	تحويل الحروف الصغيره لكبيره
<code>String.lower(string)</code>	تحويل الحروف الكبيره لصغيره
<code>String.capitalize(string)</code>	تجعل اول حرف كبير
<code>String.swapcase(string)</code>	تجعل الحروف الكبيره صغيره والصغيره كبيره
	هناك العديد من الوظائف مطلوب منكم إكمال الجدول

```
>>>import string
>>>string.upper(ahmed)
AHMED
>>>string.lower(AHMED')
ahmed
>>>string.swapcase('AhMeD')
aHmEd
```

هذه كانت بعض الأمثله على مكتبيه `string` مع أنها امثله سهله ولكنها مفيده لاحظ انا لم اتحدث إلا عن اهم الوظائف لأن البحث واجب عليك فلن تعرض هذه الحلقات كل شئ عن بايثون ولاحظ انها مجرد مقدمه وحينما تتأقلم مع بايثون سوف تستطيع ان تجد المساعده والدعم كما قلنا

ملخص

تعرفنا على بعض الأخطاء الشائعه
تعرفنا على بعض وظائف مكتبيه `string`

القواميس والمصفوفات ذات التركيب المرتب

هذه الحلقة السادسة ف بايثون

سنتناول فيها إن شاء الله

اولا ماهى القواميس؟؟

هى نوع من المصفوفات ولكن له مفتاح سنرى حالا

افضل التعامل مع IDLE

حتى تكون النواتج لحظيه ولك الحريه ف غير ذلك بإستخدامك لمحررى نصوص

```
>>>dict1={"name":"ahmed","age":"83"}
```

```
>>>dict1
```

```
{'name': 'ahmed', 'age': '83'}
```

```
>>>dict["name"]
```

```
ahmed
```

```
>>>dict["age"]
```

```
83
```

شرح الكود

مع ان الكود واضح إلا اننا سنشرحه
الأول عرفنا قاموس {}

وضعنا داخله عناصر كل منها له مفتاحا
المفتاح هو اللي ع الشمال بحيث إنك لما تحب تعرف إيه اللي بيشير إليه المفتاح على طول بتكتب الكلمتين
إسم القاموس وبداخله المفتاح
جرب كده تكتب العكس إسم القاموس ثم الشئ الدال عليه المفتاح ستظهر لك رساله خطأ
تقدر تضيف زى ما إنت عايز مفاتيح وقيم وتغيرهم كمان

```
>>>dict["language"]="python"
>>>dict
{'name':"ahmed", "age":'83', "language":'python'}
>>>dict["age"]="30"
>>>dict
{'name':"ahmed", "age":'30' "language":'python'}
```

الجدول التالي به بعض الخصائص المتعلقة بالقواميس

Function	العمل
Dict.items()	يديك كل العناصر على صوره ازواج مرتبه من المفتاح والقيمه
Dict.clear()	محو كل العناصر
Dict.keys()	يديك كل المفاتيح
Dict.has_key(name)	تتحقق منطقيا من وجود المفتاح ام لا
Len(Dict)	يديك عدد عناصر القاموس

انا بيتهيا إلى إن كده تمام القواميس صح؟؟
يارب تكون تمام

tuple

سنسميها توبل بالعربى عشان إسمها كبير مصفوفه ذات تركيب مرتب
اولا ماهى التوبل؟؟؟؟؟؟
هى مصفوفه بس ثابتة مش ليها اى خصائص غير التأكد موجود ولا لا
ليه بنستخدمها بقى؟؟؟

عشان هى اسرع م المصفوفات ولكن عادى ولايهمك هنا ليك الحريه البرنامج ممكن تكتبه ب10 طرق ع
العموم إنت وراحتك
المهم يلا نشوف التوبل ده مالها

```
>>>a=(1,2,"a","b")
>>>a[1]
2
```

```
>>>a[1:2]
(2,"a")
>>> 1 in a
True
```

جرب كده اخى بعض خصائص المصفوفات على التوبل ولاحظ ماذا سيحدث
سوف نطلق إنشاء الله حلقة للغة سى ستجدونها بالمنتدى قريبا حتى يكون عندك فكره عما تقوم به هذه اللغة
ولن اطلب منك شيئا سوى ان تستطيع قراءه الكود البرمجى فقط المكتوب ولك مطلق الحريه ف إستخدام اى
لغه تشاء

ملخص

تحدثنا ف هذه الحلقة عن القواميس المصفوفات المرتبه

lambda

هذه هى الحلقة السابعة ف بايثون وسنتناول فيها إنشاء الله عدة أشياء مثل

lambda

بعد الطلب الكبير على عدة مميزات ف بعض لغات البرمجه وايضا ف ليسب تمت إضافة العديد
منها لبايثون مثل هذه الكلمه
اظن انك لن تحتاجها ع العموم سنشرحها
لن تناول عليها سوى مثالين حتى يتضح لك دورها والباقى لإبداعاتكم

```
>>>def inc(n):
    return lambda x: x+n
```

```
>>>f=inc(1)
>>>f(2)
3
```

شرح الكود

اولا عرفنا داله وبداخلها متغير ما
داخل متن الداله وضعنا القاعده وهى ان يتم تعريف متغير X
حيث قيمته هى مجموع متغير الداله الذى اصبحت ثابتا والمتغير الجديد الذى نحدده

شرح الكود مره اخرى

لاحظ اننا قد عرفنا داله ووضعنا فيها متغير n
بعد ذلك وضعنا قواعد هذه الداله وهى ان تعيد مجموع المتغير السابق الذى يمكن ان نعه اصبحت

ثابتا ف الداله و المتغير X
مثال اخر

```
>>>def inc(n,l):  
    return lambda x=x+n+l  
>>>f=inc(4,3):  
>>>f(5)  
12
```

عرفنا داله اخرى ووضعنا بداخلها متغيرين
وجعلنا قاعدة الداله ان تجمع المتغيرين وأيضا العدد X
لاحظ يتم ف ثالث سطر وضع قيم المتغيرين
وف الرابع يتم وضع قيمة المتغير X
ننظر الآن لجزئيه ربما تكون مهمه للبعض وهى معرفة وثيقة الداله حيث حينما تعرضنا لتعريف
الدوال
نعطى مثال للتذكير هيا بنا

```
>>>def function():  
    """doesn't do anything ok  
    really it doesn't"""  
    pass  
>>>print function.__doc__  
"""doesn't do anything ok  
really it doesn't"""
```

معنى ذلك هو ان كل مايكتب بين "" ""
هو وثيقه مبسطه للداله يوجد بها ماذا تفعل هذه الداله وكيفيه إستخدامها

مثال واقعي

```
>>>import sys  
>>>print sys.exit.__doc__  
exit([status])
```

Exit the interpreter by raising SystemExit(status).
If the status is omitted or None, it defaults to zero (i.e., success).
If the status is numeric, it will be used as the system exit status.

If it is another kind of object, it will be printed and the system exit status will be one (i.e., failure).

هذا المثال على دالة sys.exit()
فيعطيك كل ماتفعله الدالة

عودة للمصفوفات والدوال الجاهزه

هذه هي الحلقة الثامنه ف بايثون

سنعود ف هذه الحلقة إلى المصفوفات وسنتعامل إن شاء الله مع بعض الدوال الجاهزه الخاصه بها

وهذه الدوال هي

map(),reduce(),filter(),zip()

اولا filter()

تستخدم لإعطاء مصفوفه مستثناه من عناصر تحدد انت طريقه الإستثناء عن طريق داله تقوم بإنشاءها
مثال على هذه الداله

```
>>>def f(x): return x % 2 !=0 and x %3 !=0
>>>filter(f, range(2,25))
[5, 7, 11, 13, 17, 19, 23]
```

مالذي حدث؟؟؟

3 ولا على 2 اولا قمنا بتعريف داله تعيد القيم التي لاتقبل القسمه على

إستخدمنا filter()

25 إلى 2 بأننا وضعنا قاعده الداله السابقه وحددنا عناصر الصفوفه من

داخل مصفوفه 3 ولا على 2 والتي لاتقبل القسمه على 25 إلى 2 قامت الداله بإعاده كل القيم من داله مفيده ف رأيي

```
>>>def f(x): return x % 5 !=0 and x % 6 !=0
>>>filter(f,range(2,25))
[2, 3, 4, 7, 8, 9, 11, 13, 14, 16, 17, 19, 21, 22, 23]
```

ولا 5ذلك مثال اخر حيث إستثنينا م المصفوفه المطلوبه كل الأعداد التي لاتقبل القسمه على 6على

ثانياً map()

تعيد لك مصفوفة على هيئة متتابعه بمجرد ان تعرف داله وتضع القاعده
كلام كاللوجاريتمات صح؟؟
فلنرى بعض الأمثلة

```
>>>def square(x): return x**2 #u may use x*x
>>>map(square,range(1,11))
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

كوننا مصفوفه من مربعات الأعداد المحصوره بين 1 و 11
حيث حددنا ف الداله التي عرفناها اننا نريد مربعات هذه الأعداد
ف المثال التالي حددنا ف الداله المعرفه اننا نريد مكعبات هذه الأعداد

```
>>>def cube(x): return x**3 #u may use x*x*x
>>>map(cube,range(1,11))
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

مثال اخر

```
>>>seq=range(9)
>>>def square(x):
    return x*x
>>>map(None,seq,map(square,seq))
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64)]
```

ومربعاتها على هيئة أزواج مرتبه مكونه من العدد 1 إلى 1 هذا المثال اعاد لك الأعداد من
ومربعه

بمعنى اخر اننا حولنا زوج من المصفوفات إلى مصفوفه مكونه من أزواج
حيث المصفوفه الأولى range(9)
والثانيه هي مصفوفه الأعداد المربعه وأستخدمنا كلمه None
للقايه مثل النون ف اللغه العربيه ف بعض الأحيان

ثالثاً reduce()

هذه الدالة سهلة للغاية ف فكره عملها
حيث انك تعرف داله اولا ثم تستدعي تلك الداله من خلال دالة `reduce`
لنوضح ذلك بأمثله

```
>>>def add(x,y): return x+y  
>>>reduce(add,range(1,11))  
55
```

مالذي حدث ???
الذي حدث اننا عرفنا داله تقوم بجمع عدد م المتغيرات
ثم إستدعينا دالتنا `reduce`
ومررنا لها اننا سنستخدم الداله الأولى كقاعده وان المتغيرات التي ستستخدمها الداله الأولى هي
11 إلى 1 الأعداد من
وكان حاصل جمعهم 55

رابعاً zip() مثال

```
#!/usr/bin/python  
#this is on zip(y,x) that matches y and x  
  
questions=['name','age','favorite color']  
answers=['ahmed','17','red']  
for q,a in zip(questions,answers):  
    print 'What is your %s? it is %s' %(q,a)
```

ما هذا ؟

هذه الدالة فعلا خطيره كل اللى عملناه هنا هو إن إحنا حددنا مصفوفتين وقامت هذه الدالة بتشكيل
خليط منهم لاحظ الناتج

www.python.org
www.c4arab.com
www.perl.org
www.perl.com

ملخص

عوده للدوال الجاهزه مثل

map(),reduce(),filter(),zip()

وتمت مناقشتهم تفصيلاً

لغة سي

هذه الحلقة التاسعة ف بايثون

وسنتحدث فيها عن لغة سي إجمالاً

ستساعدك ف وضع قدمك على الطريق ف لغة سي لاحظ هناك اجزاء سأجنبها

بالكامل مثل المؤشرات وقد قام الأخ جيمي بشرحها على اكمل وجه

ومن يريد الأستفاضه ف لغة سي يمكنه مراسلتى بهذا الخصوص وسأدله على العديد مما يستطيع

قراءته والله ولى التوفيق

اولا ماهى لغة سي؟

هى لغة قياسيةه وصممت ف الأساس لكتابة أنظمة التشغيل والبرامج الكبيره فتعطى كفاءه عاليه

وتحكم كبير للمبرمج ف الجهاز فممكن للبرامج المكتوبه بـسي بأن تكون ف سرعه البرامج

المكتوبه

بالأسمبلى وذلك ف حال إستخدامك لكومبايلر جنو gcc

هناك ايضاً مترجمين جيدين مثل الذى من شركه بورلاند

لكن انصحك بألا تستخدمها إلا ف الحاجه فقط وغير ذلك فمن يبحث الآن عن البيتس والأن

المساحات بالجيجا والذاكره صارت كبيره فإذا اردت إستخدم لغه تهدر لك موارد النظام مثل

بايثون او بيرل

وإستخدم سي حينما تريد ذلك الكم الكبير من التحكم

لاحظ سنستخدم كومبايلر جنو ف الشرح وانت حر مع اى مترجم اخر لكنى افضله فهو مجانى

وهو الأفضل

اولا يجب ان تختار محرر نصوص يلونك السكريبت اللى بتكتبه مثل

vi,emacs,Gedit,kwrite,pico,nano

تكتب السكريبت وتحفظه ف مكان ما بإمتداد

filename.c

وتعطى التصاريح للقراءه والتنفيذ عن طريق

```
chmod +775 filename.c
```

```
chmod +xw filename.c
```

ولتشغيل البرنامج إكتب الأمر التالى

```
gcc filename.c -o filename
```


سنتحدث ف هذه الحلقة عن برامج بسيطة للغاية

```
#include<stdio.h>
#include<stdlib.h>
main()
{
puts ("hello world talking from c \n");
return 0;
}
```

اولا تعليمه `include`

وتسبق ب#

تعنى إضافة المكتبيه التى تليها فأضفنا مكتبتى `stdio.h,stdlib.h`

ف السطر الثانى قمنا بوضع الداله الرئيسيه `main`

واتبعناها ببلك البدايه {

ف السطر الثالث إستخدمنا تعليمه `puts`

ويمكن إستخدام `printf`

أضفنا الفاصله المنقوطه لاحظ انا سى ليست ف سهوله بايثون

ولكن يجب ان تتعلم ولو ان تقرأ الكود الذى امامك فقط

بعد ذلك اعطينا الأمر بالتنفيذ يمكن الاتطلب منك بعض الكومبيلات إضافة `return`

على غير القاعده العامه وبعد ذلك اغلقنا البلك 1 والفشل =0التنفيذ=}

لاحظ الفاصله لازمه بعد كل امر

تعريف المتغيرات ف المثال التالى

```
#include<stdio.h>
#include<stdlib.h>
main(){
int a;
puts("Enter a number:");
scanf("%d",&a);
printf("The number is %d",a);
return 0;
}
```

ماذا فعلنا ؟

مثل المثال الأول أضفنا المكتبيات

وعرفنا متغيرا سميناه `a`

اظهرنا رساله تطلب منك إدخال رقم

وبعد ذلك إستخدمنا تعليمه `scanf`

وهى تساوى ف بايثون `input`

ووضعنا فيها انها ستقبل عددا صحيحا وتم إعطاء ذلك العدد كقيمه للمتغير `a`

وبعد ذلك اظهرنا رساله مفادها ان الرقم الذى تم إدخاله هو

الذى سيظهر

%d (decimal)	تأخذ عددا صحيحا فقط
%f(float)	تأخذ عددا حقيقيا
%c(char)	تأخذ حرفا
%s(string)	سلسلة نصيه
%e(input numbers in scientific notation)	10 تأخذ رقما مضروباً ف أس

الدوال ف سي

```
#include<stdio.h>
#include<stdlib.h>
message(){
puts ("hi what's up")
}
main(){
message()
return 0;
}
```

المثال واضح للغايه

حيث قمنا بتعريف داله ما ووضعنا داخل متنها الخصائص المطلوبه وهى ف ذلك المثال ان تطبع كلاما ما بعد ذلك إستدعينا الداله من خلال الداله الرئيسيه main

سنتحدث عن جمل الشرط والدورات إجمالاً بعرض امثله والشرح وفقنا الله

if,else

```
#include<stdio.h>
main(){
int a;
printf("please enter number greater than 5");
scanf("%d",&a);
if (a < 5);
printf("the number u entered is less than five");
else
printf("the number u entered is greater than five");
```

عرفنا متغير ما داخل متن الداله الرئيسيه واطهرنا رساله تطالب بإدخال ذلك الرقم وإستخدمنا

دالة الإدخال scanf

ومررنا القيمه الداخله كقيمه للمتغير a

تظهر رساله بذلك وإن لم يكن تظهر رساله تخبرنا 5 ووضعنا الشرط اما إذا كان الرقم اقل من بغير ذلك يمكنك وضع عدد كبير م الشروط أليست جميله هذه اللغه؟؟

for

هذه الجملة قد إستخدمناها كثيرا ف بايثون لو لاحظت ع العموم فلنبدأ بمثال

```
#include<stdio.h>
main(){
int a;
for (a=1; a <=5;a++)
printf("Hello world from The C programming language");
return 0;
}
```

المثال واضح ف رأيى ولكن ماذا فعلنا؟؟؟

عرفنا داخل متن الداله متغيرا ما

وإستخدمنا جملة دواره for

ووضعنا بداخلها ان قيمه $a = 1$

وان a مرات للشرط وهو طباعه جملة ما 5 ولذلك فيجب التكرار 5 اقل من

حتى تصل إلى قيمه الخمسه وذلك بزياده مقدارها واحدا

هكذا تعرفنا على لغة سى إجمالا بحيث ان حينما اتحدث عن وظيفه اساسيه ف بايثون وأقارنها

بسى تستطيع التفاعل مع ذلك الكلام ونرجو من الله التوفيق

لاحظ تلك الحلقة ستعاد كتابتها بإذن الله عن طريق متخصص ف لغه سى ليس مثلى وسيكون

أسلوب الشرح افضل من ذلك وفقنا ووفقكم الله

Del & break

هذه هى الحلقة العاشره ف بايثون

بعد عوده من الكلام عن لغة سى ف الحلقة السابقه ولكن كان لابد منها حتى يكون

عندك

فكره عنها

كلمة **break** نستخدمها لإيقاف تنفيذ البرنامج ف حال تحقق شرط ما

```
>>>for x in range(0,10):
```

```
    print x
```

```
    if x==5:
```

```
        break
```

0

1

2

3
4
5

ذلك مثال إستخدامنا فيه جملة شرط إذا لاحظت for, if
10 إلى 0 قمنا بعمل دواره صغيره لعد الأرقام من
ثم وضعنا شرطاً داخل جملة for
وذلك بإستخدام شرطية if
وهي ان يتوقف البرنامج حينما x=5

جملة del

تستخدم على صورتها العاديه وهي للحذف سواء كان مصفوفه او متغير او ثابتاً و
صنف
او غيرهم

```
>>>i=5  
>>>i  
5  
>>>del i  
>>>i
```

السطر الأخير هيديك رسالة خطأ

```
>>>n=[1,2,3,4]  
>>>n  
[1,2,3,4]  
>>>del n  
>>>n
```

هيديك رسالة خطأ برده

ملخص

تعرفنا على

break,del

البرمجه الموجهه بالكائنات

الحلقه الحادية عشر ف بايثون

بعد العديد من الحلقات شاء الله ان يتم تقديم حلقات البرمجه الكائنيه وهى ستكون ف حدود حلقتين إلى ثلاثه

لقد أصبحت بحمد الله على درايه كبيره بالعديد من خصائص هذه اللغه وأنت معنا على مر هذه الحلقات ويمكنك ان تقوم ببرامج فعاله بدون البرمجه الكائنيه

سطر ألن يحتاج 100 ولكن هل لاحظت انك حينما تكتب برنامجا أكبر من للتنظيم والدقه وان تتبع الخطأ ناهيك ان هذه بايثون وليست سى فإن تتبع الخطأ أسهل

اولا ماهى البرمجه الكائنيه؟؟

هى تقسيم البرنامج لاصنوف وتضع الخصائص داخل كل صنف ومن ثم تستدعيه ف البرنامج فيكون التعامل مع البرنامج اسهل ومن ناحية التعديل أيضا ف أى جزئيه فتستطيع تطويره بسهولة وتطويع البرنامج لصالحك مثلا كلمات 10 حينما تعدل كلمه واحده اليس أسهل من ان تعدل لاحظ إن بايثون تنافس أقوى اللغات من ناحية البرمجه الكائنيه فهى تعد منافسى جافا الذين يستحقوا التقدير

جافا

هى لغة برمجه كائنيه صافيه حتى أنك ف أسهل الأكواد يجب أن تستخدم الكائنات وكودها أسرع ف تصنيفه من البايثون وهناك مترجم ف لينكس يصنفها للغة الآله مباشرة مما يجعلها مقاربه للسى عامة الجافا هى ثانى أفضل خيار ف تعلم البرمجه بعد البايثون فستكون بالنسبه لك

تجربه ممتعه

لاحظ الفيچوال البيزيك ليست لغه برمجه كائنيه حتى مع الذى يقوله الغير فينقصها العديد من خصائص البرمجه الكائنيه ممكن نقول عنها لغة برمجه ثلث كائنيه

ع العموم لن نتحدث عنها الآن نبدأ سنرى أولا بعض الأكواد

```
>>>class Hello:
    i=12
    def hi(self):
        print "Hi there"
>>>x=Hello()
>>>x.i
12
>>>x.f()
Hi there
>>>Hello().i
12
>>>Hello().f()
Hi there
```

اولا عرفنا الصنف الذى نريده وسميناه `hello` ووضعنا فيه متغيرا صنفيا أى لايمكن إستدعاؤه إلا من خلال الصنف الذى

عملناه

i=12

وبعد ذلك عرفنا داله وجعلنا بداخلها self

لاحظ self=This in java

لاحظ self

ليس له أى تأثير ولكن كل داله تعرف داخل صنف يجب ان تحتوى عليه

الداله وظيفتها ان تطبع كلمه هاللى

لاحظ إستخدمنا حرف عادى X

وجعلناه يساوى الصنف ويمكن لك أن تتركه عادى مش مشكله

وبعد كده لو حبيت تستدعى اى شئ موجود داخل الصنف عادى بتكتب إسم

الصنف والشئ الذى تريده ولكن يجب أن يكون معرفا داخله

هذه هى الحلقة الثانية عشره ف بايثون

وثانى حلقات البرمجه الكائنيه

لقد توقفنا ف الحلقة السابقه عند صنفنا إذا تذكرتموه

والآن تعديل بسيط عليه

```
>>>class Hello:
```

```
    i=12
```

```
    def __init__(self,name):
```

```
        self.name=name
```

```
    def hi(self):
```

```
        print "Hi",self.name
```

```
>>>x=Hello('ahmed')
```

```
>>>x.Hi()
```

```
Hi ahmed
```

```
def __init__(self,name)
```

الداله ده بإختصار شديد هى دالة بناء أى ان ماسيوضع ف الصنف بتاعنا هو إسم وقلنا ان self

تتجاهله

لاحظ ف إثنين قبل وبعد

بعد كده أعطينا قيمه للأسم بتاعنا عشان نقدر نستخدمه ف الدوال الأخرى داخل الصنف

بمعنى اخر self هو حلقة الوصل بين الشئ الحقيقي والصنف والدوال التي داخله

ممکن تغیرہ بر احوک `self.name=name`

بعد كده الداله بتاعتنا واخبرنا فيها ان سنستخدم متغير self له إمتداد ما اللي هو الاسم

بعد کده ساوينا بين صنفنا وداخله اسم و حرف ما

```
x=Hello('ahmed')
```

X ہی حرف ما

Hello('ahmed') هو الصنف بداخله الاسم الذي اخبرنا الصنف عنه

بعد كده إستخدمنا الداله الداخليه اللي داخل الصنف

```
Hello('ahmed').hi()
```

و ممکن

x.hi()

أمثله بدون شرح

```
>>>class message:
```

```
def __init__(self, string):
```

```
self.string=string
```

```
def printIt(self):
```

```
print self.string
```

```
>>>a=message('hi')
```

```
>>>a.printIt()
```

hi

```
>>>b=message('ahmed')
```

```
>>>b.printIt()
```

ahmed

```
>>>n=['a','b']
```

```
>>>for x in n:
```

```
print x.printIt()
```

hi

ahmed

[illegible]

عرفنا صنف `class message`

عرفنا دالة البناء بأننا سنضع داخل الصنف أى بين القوسين سلسلة حرفيه

عشان نقدر نستخدمه ف باقى الموجودات ف الصنف كالدوال `self.string=string`

عرفنا داله عادى خالص وخلصنا الصنف

و عرفنا متغيرين وساويناهم بالصنف وداخله سلسلة نصيه

```
a=message('hi')
```

```
b=message('ahmed')
```

وإستخدمنا الداله المعرفه داخل الصنف

لاحظ بنستخدم الداله كأنها إمتداد للصنف يعنى بنكتب الصنف الأول ونملاه باللى حددناه كالسلسله النصيه

```
a.printIt()
```

على فكره الجزئيه الأخيره

```
n=['a','b']
```

كانت مثال لمجرد الكسل البرمجى عادى لو مش عاجباك الطريقه بس عملنا مصفوفه من المتغيرين السابقين

وبإستخدام `for`

قمنا بإستدعاء الداله الداخليه ف الصنف `printIt`

عشان نطبع الرسالتين ف وقت واحد

لنرى مثال اخر

```
>>>class C:
```

```
    def __init__(self,val):
```

```
        self.val=val
```

```
    def printIt(self): print "Hi my value is ",self.val
```

```
>>>a=C(21)
```

```
>>>b=C(34)
```

```
>>>a.printIt()
```

```
Hi my val is 21
```

```
>>>b.printIt
```

```
Hi my val is 34
```

Simple class

```
#!/usr/bin/python
```

```
# this is SimpleClass.py
```

```
# OOP and classes
```

```
class Simple:
```

```
# the self arg is just like this (you can't delete it)
```

```
    def __init__(self, str):
```

```

    print "Inside the Simple constructor"
    self.s = str
# Two methods:
def show(self):
    print self.s
def showMsg(self, msg):
    print msg + ': ',
    self.show() # Calling another method
# if we are runned as a prog (else then this file is used as module)
if __name__ == "__main__":
    # Create an object:
    x = Simple("constructor argument")
    x.show()
    x.showMsg("A message" )

```

Area

```

#!/usr/bin/python

class Square:
    def __init__(self, side):
        self.side = side
    def calculateArea(self):
        return self.side**2

class Circle:
    def __init__(self, radius):
        self.radius = radius
    def calculateArea(self):
        import math
        return math.pi*(self.radius**2)

list = [Circle(5),Circle(7),Square(9),Circle(3),Square(12)]

for shape in list:
    print "The area is: ", shape.calculateArea()

```

هذه الحلقة الثالثة عشره ف البرمجه بإستخدام بايثون
سنستكمل ق هذه الحلقة بمشيئه الله الصنوف والبرمجه الكائنيه
لاحظ هذا الكتاب ليس إلا مجرد دوره سريعه للغاية ف بايثون

هذه الحلقة تعد من الحلقات النهائيه ف ذلك الكتاب
لكن أرجو ان يقرأ هذه الحلقة من لديه خبره عن البرمجه بإستخدام الجافا وإن لم تكن فلا ضرر
عالمه لن نخرج عن موضوع هذا الكتاب ألاوهو البرمجه بإستخدام بايثون ولن أظهر اكواد بلغة جافا لأنك
البرمجه ولكن الحديث سيكون عن This لايمكن ان تحدد هذه اللغة خلال سطورا فهي تعد من أعظم لغات
شيئ مميز لها وهو معامل
فإذا كنت من مبرمجي جافا فلاتغير قواعدك الراسخه في ذهنك فإذا اردت

self=this

مثال بإستخدام this

```
>>>class D:
    def __init__(this,realpart,imagpart)
        this.real=realpart
        this.imag=imagpart
    def printIt(this):
        print "The realpart equals:",this.real
        print "The imagpart equals:",this.imag
>>>x=input("the realpart :")
>>>y=input("the imagpart: ")
>>>n=D(x,y)
>>>n.printIt()
```

أظن أنك مذهول بالفعل هذا ماتراه عيناك
لك الحق في أن تكون فلك حريه الإختيار في إستخدام بعض قواعد الجافا مثل مثالنا
يمكن لك ان تقوم بتغيير this إلىself
الآن لتدرك هل قمت بإدراك مفهوم البرمجه الكائنيه أرجو منك تعديل المثال السابق
ليصبح بإمكانك ان تأتي بحاصل الضرب والقسمه والمجموع والفرق لأعداد مركبه
لاحظ أى إجراء يتم فصله الحقيقي مع الحقيقي والتخيلي مع التخيلي ماعدا الضرب
والقسمه

التعامل مع الملفات File handling

الحلقة الرابعة عشره ف بايثون

سنتناول فيها إن شاء الله التعامل مع الملفات

نبذة عن التعامل مع الملفات

المقصود هنا هو فتح الملف وإغلاقه وقراءته

لنبدأ بمثال بسيط ف محرر النصوص المفضل لديك إكتب فيه

this is a simple text about python
it's a really great programming language
whatever you do there's python for you

وإحفظ هذا النص ف أى مكان بأى إسم وليكن text.txt
لو أنت بتستخدم ويندوز إحفظه ف مكان المفسر غير كده ف أى مكان
بس إحفظ المسار

هذا البرنامج من أدواتك الخاصه فإن شئت طورها فهي مثل

cat >>>In UNIX, type>>>In Windows

افتح IDLE

أو أى محرر نصوص

وإكتب الآتى

```
#!/usr/bin/python
x=file("/home/ahmed/text/txt","r")
for line in x.readlines():
```

```
print line
x.close()
```

شرح الكود

إستخدمنا دالة `file()` وهى ومخصوصه لفتح الملفات
هذه الدالة بتأخذ مقطعين الأول مسار البرنامج والثانى ماستفعله سواء قراءه أو كتابه
هناك دالة أخرى لفتح الملفات `open()`

```
#!/usr/bin/python
x=open("/home/ahmed/text/txt","r")
for line in x.readlines():
print line
x.close()
```

لاحظ معنى `read"r"`
يعنى ذلك أن الملف المفتوح سيكون للقراءة فقط

ثم إستخدمنا دالة `for` لعرض السطور من خلال دالة `readlines()`
وإخبرنا بايثون أن تطبع لنا السطور الموجود ف الملف
ثم ستخدمنا دالة `colse()` لإغلاق

strip()

تستخدم هذه الدالة ف مسح المسافات بين الجمل يعنى لو سايب فاصل سطرين تقوم الدلة ده
تخليها سطر هتسأل طب وإيه الميزه؟؟
إنت هتلاحظها بنفسك

```
#!/usr/bin/python
x=open("/home/ahmed/text.txt","r")
for line in x.readlines():
print line.strip()
x.close()
```

كيف تكتب داخل ملف؟

سنستخدم دالة بطريقة بدائيه وسنستغلها ف الكتابة داخل ملف من ملف آخر كأنها دالة نسخ
إعمل ملف آخر فارغ وسميه `text2.txt`

```
#!/usr/bin/python
x=open("/home/ahmed/text.txt","r")
y=open("/home/ahmed/text.txt","w")
for line in x: y.write(line)
```

```
print "file is copied"
x.close()
y.close()
```

مثال سهل فعلا
لقد أحضرنا المخرجات من ملف القراءه ووضعناها منسوخه ف ملفنا الجديد
يالها من لغة

تلميحات

تستطيع تطوير هذه الأدوات البرمجيه بل وأن تحفظها ف مجلد البرامج عندي انا مثلا `/usr/bin`
ثم تستدعيها كأداة رئيسيه
تستطيع تعديل هذه الأدوات بأن تجعل المدخلات هي مدخلات سطر الأوامر
`commandline_arguments`
عن طريق مكتبيه `sys`
من خلال وظيفه `argv`
يجب أن تفتح الملفات قبل إستخدامها
ويجب أن تغلقها بعد الإستخدام
لاستطيع إستخدام خيار القراءه والكتابه معا
`\n` تعليق بسيط على متاعقه السطر الجديد
هناك أنظمة يختلف إستخدام هذه المتعاقبه من نظام لآخر
`<<<'\r\n'` ويندوز
`<<<'\n'` يونكس وشبيهاته وأنظمة جنو خصوصا لينكس
`<<<'r'` ماك
واليا `'\n'`
إرجع للوثيقه الرسميه بهذا الخصوص ونظام تشغيلك
`AIX,Irix,Solaris,Unix,Windows`

معاملات سطر الأوامر
هذه الجزئيه مع سهولتها الشديده ولكنى لن أقوم بشرحها لأنى أظن أنك ستستوعب الكود

echo

```
import sys
for x in sys.argv:
    print x, # the comma is for not making a new line
```

الحلقة الخامسة عشر ف بايثون

سنتناول فيها بمشيئة الله كيف تستطيع إنشاء مكتبيه ولكن أولا ماهى المكتبيه؟؟
المكتبيه ف أبسط كلمات هى مجموعه من الدوال الجاهزه والمتغيرات والثوابت التى قمت بكتابتها وتريد إستخدامها دون كتابتها مرات عديده أو تنسخها من برامج أخرى ولكن تستخدم لها أمر الأستدعاء `import sys` نعم مثل مكتبيه `sys`
ولكن كيف ننشئ هذه المكتبيه؟؟
أليس يجب أن تكون عبقرى للقيام بكتابة مكتبيه؟؟
الرد سهل ليست معظم المكتبيات كتبها صاحب اللغة ولكن كتبها بعض المطورين أيضا
سننشئ ف هذا الفصل مكتبيه بها عدة دوال جاهزه لنستطيع التحويل الحروف من صغيره لكبيره وأيضا بعض الوظائف المشابهه لوظائف `C/C++`

لتعرف أين يمكنك حفظ هذه المكتبه ثم إستدعائها إكتب الآتى

```
>>>import sys
>>>sys.path
['/home/ahmed', '/usr/bin', '/usr/lib/python2.3.zip', '/usr/lib/python2.3', '/usr/lib/python2.3/plat-
linux2', '/usr/lib/python2.3/lib-tk', '/usr/lib/python2.3/lib-dynload', '/usr/lib/python2.3/site-
packages', '/usr/lib/python2.3/site-packages/Numeric', '/usr/lib/python2.3/site-packages/PIL',
.....]
```

شكل المكتبيه العام

```
#####module Da.py#####
pi=3.14
j=-1**.5
def hi(): print "Hi"
#####
```

على سبيل المثال بعد حفظ المكتبيه ف `home`

إكتب الآتى

```
>>>import Da
>>>Da.Pi
3.14
```

```
>>>Da.hi():
```

“Hi”

وهكذا
أرأيت يمكنك بعد أن تستخدم دوالك أن تنشئ مكتبيه مكونه منها
مثال على مكتبيه حقيقيه

```
##written by : l1nUx3r
```

```
def to_upper(string):  
    ##it converts a string to upper case  
    upper_case=""  
    for character in string:  
        if 'a' <= character <='z':  
            location=ord(character)-ord('a')  
            new_Ascii=location+ord('A')  
            character=chr(new_ascii)  
            upper_case=upper_case+character  
    print upper_case  
##..  
def printf(string): print string
```

مكتبيه جيده أليس كذلك بها وظيفة الإدخال الرئيسية ف سى
ودالة لتحويل الحروف الصغيرة لكبيرة

بعض السكريبتات الخفيفه

سكريبت لحساب قيمة محدد

```
#!/usr/bin/python  
#This program is under FDL lisnece
```

```
x="""  
|a b c|  
|d e f|  
|g h i|  
"""
```

```
print x
```

```
r1=['a','b','c']  
r2=['d','e','f']  
r3=['g','h','i']  
print "The rows are in lists []"  
r1=input("1st row in [] :")  
r2=input("2nd row :")
```



```
r3=input("3rd row :")
```

```
a=r1[0]
```

```
b=r1[1]
```

```
c=r1[2]
```

```
d=r2[0]
```

```
e=r2[1]
```

```
f=r2[2]
```

```
g=r3[0]
```

```
h=r3[1]
```

```
i=r3[2]
```

```
print "The value",(((a*e*i)+(b*f*g)+(c*d*h))-((b*d*i)+(a*f*h)+(c*e*g)))
```

تطبيق ع الكائنات والمتغيرات الداخليه (مهم)

```
#!/usr/bin/python
```

```
def options():
```

```
    print "1-circle \n""2-square \n""3-recetangle \n""4-info \n""5-quit\n"
```

```
class circle:
    def __init__(self,radius):
        self.radius=radius
    def area(self):
        import math
        return math.pi*(self.radius**2)
```

```
class square:
    def __init__(self,side):
        self.side=side
    def area(self):
        return self.side**2
```

```
class recetangle:
    def __init__(self,height,width):
        self.height=height
        self.width=width
    def area(self):
        return (self.height)*(self.width)
```

options()

```
x=input("Number :")
if x==1:
    r=input("The radius: ")
    n=circle(r)
    print "The area: ",n.area()
elif x==2:
    r=input("The side : ")
    n=square(r)
    print "The area: ",n.area()
elif x==3:
    h=input("The height: ")
    w=input("The width: ")
```

```
n=recetangle(h,w)
print "The area",n.area()
elif x==4:
    print "it's made by l1nUx3r "
elif x==5 :
    import sys
    sys.exit()
```

lwc (l1nUx3r word Counter)

عداد كلمات

```
#!/usr/bin/python
```

```
import sys
import string
```

```
def numwords(s:(
list=string.split(s(
```

```
return len(list(
```

```
n=sys.argv[0] #we won't use it cause it returns the ac program
x=sys.argv[1[
f=open(x,"r("
total=0
```

```
for line in f:
total=total+numwords(line(
```

```
print x," has %d words" % total
```

Cal viewer

```
#!/usr/bin/python

import calendar
print "choose the year"
year=input("the year??:")
calendar.prcal(year)
```

Cls

```
#!/usr/bin/python

print "\n"*1000
```

Cout (C++ printing function)

```
#!/usr/bin/python

def cout(string): print string

cout("using c++ function cout")
```

Factorial دالة المضروب

```
#!/usr/bin/python

n=input("the number?:")
def fac(n):
    if n<=1:
        print n,"! = 1"
    else: print n,"!=",n*fac(n-1)    #it should be "!= ", not = "!="=n*fac(n-1)
```

fibonnacci sequence

```
#!/usr/bin/python
```

#fibonacci sequence

```
a,b=0,1
count =0
max_count = 20
while count < max_count:
count=count+1
old_a=a
old_b=b
a=old_b
b=old_a+old_b
print old_a
```

Odd or Even

```
#!/usr/bin/python
#even
num=input("The number is: ")
if num % 2 ==0:
    print num,"is even"
elif num % 2==1:
    print num,"is odd"
else: print num,"is very strange"
```

Rate

```
#!/usr/bin/python

print "we are calcualting the time "
rate=input("The rate : ")
distance=input("The distance is : ")

print "The time is ",distance/rate
```

unstoppable loop

```
#!/usr/bin/python
#unstoppable loop
```

```
while 1==1:  
    print "hi i'm a crazy loop"
```

Size

```
#!/bin/python
```

```
from os.path import*
```

```
file1=input("put the path of the file : ")  
print "The size equals",getsize('file1')
```

NewLine

```
#!/usr/bin/python
```

```
def newline():  
    print
```

```
print "hi"
```

```
newline()
```

```
print "This was an empty line"
```

Password

```
#!/usr/bin/python
```

```
password="l1nUx3r "  
while password != "DaRkoOo":  
    password=raw_input("Password: ")  
print "welcome in"
```

Abs

القيمة المطلقة

```
#!/usr/bin/python
#abs means the absolute value
n=input("number? :")
if n < 0:
    print "the abs of ",n,"is",-n
else:
    print "the abs of ",n,"is",n
```

تمارين عامة

قم بتطوير هذه المكتبة لتحتوى على لوظائف الإدخال الرئيسية فC/C++ ,Pascal
طور الأدوات echo,cat
فتستطيع إستخدام معاملات سطر الأوامر ف مخرجاتهم
طور دوال وأضفها لمكتبيتك الخاصة للتعامل مع السلاسل النصية
إن كنت تهوى الجوانب الرياضيه قم بإضافة بعض الثوابت العلمية والعلاقات الرياضيه

ماذا الآن

بعد أن أنهيت قراءة الكتاب كل ما عليك هو أن تفتح محرر النصوص المفضل لديك وأن تكتب الكود التالى

```
#!/usr/bin/python
```

```
print "I am a really cool python hacker"
```

تعليقات بسيطة

حينما تكتب برنامجا الأفضل ان تضع تعليقا بسيطا يوضح كيفية عمله وألا تستخدم طرق صعبة ف فهم قارئ الكود

الآن يجب أن تحدد ماذا ستفعل؟؟

هناك عدة أشياء تستطيع فعلها مثل أن تتعلم عدة لغات برمجة أخرى وأشرح لك جافا أو بيرل كخطوة تالية أو أن تتعمق أكثر ف بايثون فتستطيع إقتحام المجالات الرسومية
هناك عدة أنواع من بايثون إذا كنت مازلت معجبا بهذه اللغة أظن أنك ستبحث عنها

أخي لا تحاول أن تمنع النفع عن إخوانك
فمن أكثر الناس عذابا كاتم العلم وإنما أهلكننا وفنينا بكتماننا العلم عن إخواننا
فالعلم ليس حكرا على أحد فلا تصغى لهراء حقوق الملكية الفكرية وذلك الكلام العقيم
فلا أقصد بذلك المحاولة لإختراق القانون
فإن كانت هناك برامج تحتاج أن تدفع لها المئات من الدولارات حتى يكون لك حق الإستخدام إعتراض ولا
تستخدمها لأنه يوجد بدل هذه البرامج المئات والمئات من البرامج الحرة والمجانية التي هي أفضل من هذه
البرامج الغالية التي ستدفع أموالا طائلة لإستخدامها وليس حتى تطويرها

تم بحمد الله كتــــــــــــــــاب إخترف بايثون

الكاتب

I1nUx3r

كلمة من الكاتب لاحظ أنك لست مطالب بقراءة هذه الجزئية فهي ليست ضمن مقرر الكتاب _
أشكر الله الذى وفقنى ف كتابة هذا الكتاب وتقديمه ف مثل هذا الشكل

أود أن أشكر العديدين ممن ساعدوني وألهموني ف كتابة هذا الكتاب وأن أتقدم لهم بالشكر بجانب والدى
ووالدتى
الكتاب إهداء خاص ل

لطقم من عرباوى

LaMoR,ABOHELAL,Dr_LeeDo,Rock,Zanger9210,Egyptian_Lady,Storm_3arabawy,Dr.
SaDa,Game MaSter

من SeCurity GuRuS

Storm,ACiDWareZ,rOCkMaStEr,Ne0,Safa7,Untrust,DarkLinux,MySQL,HackoBacko

هذا الكتـــــــــــــــــــــــــــــــــاب مكتوب على Open Office
وتمت كتابته هـــــــــــــــــــــــــــــــــه وتحويله لكتاب على منصة لينكس
وليكون الكلام دقيق

GNU/Linux Mandrake 10.1, GNU/Linux Mandrake 9.2, GNU/Linux Redhat 9

نرجو أن يكون هذا الكتاب قد نال رضاكم وإعجابكم
نسألکم الدعوة الصالحة
برمجــــــــــــــــة ممتعة لكم مع بايثون