مبادئ البرمجة بلغة بايثون

احمد سالم الصاعدي

## مقدمة

الحمد لله الذي تتم بفضله الصالحات واصلي وأسلم على خير البشر نبينا محمد صلى الله عليه وسلم وعلى آله وصحبه الطيبين الطاهرين. اما بعد:   
فلقد لاحظت خلال فترة تعلمي للغة بايثون افتقار المكتبة العربية الى كتاب مكتمل يشرح مبادي هذه اللغة بشكل منظم وبسيط. فعقدت العزم على تأليف هذه الكتاب وذلك لما رايته من الأهمية بمكان ان يتعلم القارئ العربي هذه اللغة والتي أصبحت اللغة البرمجية المحبوبة لدي الكثير من العلماء والباحثين والمهندسين. فمعظم جامعات العالم اليوم أصبحت تدرسها لطلابها لانها لغة برمجية سهلة و قوية في نفس الوقت ويمكن استخدامها في مجالات عدة. لذلك اردت ان يكون هذا الكتاب لبنة أولى للمساهمة في تعليم هذه اللغة وحافزا الى تدريسها في مدارسنا الحكومية في مراحل مبكرة كالمتوسطة والثانوية مثلا وذلك لان الأجيال الحالية لديها شغف غير مسبوق على التعرف على كل ماهو جديد في عالم التقنية وخصوصا الكمبيوترية منها. فكل ما يحتاجة هذا الجيل هو استخدام طريقة سهلة وشيقة تعرفهم بهذه اللغة وتعزز شغفهم بالتقنية فيصبحو قادرين على تعلمها والاستفادة منها دون مشقة او عناء. وبما ان هذا العمل بشري المصدر فانه لا يصل الى درجة الكمال لذلك ارجو ممن سنحت له الفرصة لقراءة هذا الكتاب ان يساهم في تحسين محتواه بارسال ملاحظاته الى ايميل المؤلف ahmad.alsaadi@kaust.edu.sa الذي يعدكم على اخذها في عين الاعتبار متى ما سنحت الفرصة لاصدار طبعة جديدة لهذا الكتاب.

# المحتويات

لفصل الاول: [نبذة عن لغة بايثون](http://localhost:8888/notebooks/Documents/git-repo/Arabic-python-notebook/Introduction.ipynb#chapter1)  
الفصل الثاني: [المبادئ الاساسية للغة بايثون](http://localhost:8888/notebooks/Documents/git-repo/Arabic-python-notebook/Introduction.ipynb#chapter2)  
الفصل الثالث: [التعامل مع القوائم](http://localhost:8888/notebooks/Documents/git-repo/Arabic-python-notebook/Introduction.ipynb#chapter3)  
الفصل الرابع: [حلقات التكرار](http://localhost:8888/notebooks/Documents/git-repo/Arabic-python-notebook/Introduction.ipynb#chapter4)  
الفصل الخامس: [اتخاذ القرارات](http://localhost:8888/notebooks/Documents/git-repo/Arabic-python-notebook/Introduction.ipynb#chapter5)

# الفصل الاول

# نبذة عن لغة بايثون

تعريف لغة بايثون ¶

بايثون (python) لغة برمجية مفتوحة المصدر سهلة التعلم يمكن الاعتماد عليها في كتابة الكثير من التطبيقات البرمجية القوية. وأكبر دليل على ذلك هو استخدام وكالة الارصاد الامريكية ناسا وشركتا قوقل وياهو وغيرها من الشركات الكبرى لهذه اللغة في بناء برامجهم المختلفة.

## نشئة لغة بايثون ¶

كانت بدايات نشئة هذه اللغة في هولندا على يد شخص يدعي جويدو فان روزم (Guido van Rossum) في نهاية الثمانيات الميلادية من القرن العشرين. حيث تم الاعلان عنها في عام ١٩٩١م. كما يعتبرفتح مصدرهذه اللغة من اهم الاسباب التي ادت الى زيادة شهرتها من خلال تكوين مجتمع برمجي نشط حولها اسهم في انشاء مكتبات كثيرة سهلت على المطورين الاخرين بناء تطبيقاتهم بسرعة و سهوله فائقة مقارنة باللغات البرمجية الاخري.

## مزايا لغة بايثون ¶

للغة بايثون مزايا عدة جعلت منها اللغة المفضلة الاولى لدى كثير من المبرمجين ومن بين اهم هذه المزايا نذكر:

سهولة تراكيبها اللغوية: فاكوادها البرمجية تكتب بطريقة قريبة جدا من اللغة الانجليزية. لذلك نجدها لاتشكل اي عائق امام أي مبرمج ان يفهم الأكواد المكتبوبة من قبل مبرمجين اخرين عندما يستدعي الامر صيانة تلك الاكواد اوتحديثها.

المرونة: يمكن تشغيل وتطوير البرامج المكتوبة بلغة بايثون على معظم انظمة التشغيل المعروفة. فالأكواد التي تم تطويرها على نظام ويندوز يمكن تشغيلها على نظام ماك ولينكس والعكس صحيح دون الحاجة الى اعادة بناء الأكواد (compiling).

كثرة المكتبات: يعتبر توفر المكتبات من اهم المزايا التي تقدمها اللغة للمبرمجين لتزيد من فعاليتهم في بناء التطبيقات. لذلك عند تنصيب اصدارة بايثون نجد انها تحتوي على مكتبات قياسية كثيرة بعضها يعتبر جزء لا يتجزء من تراكيب اللغة كمكتبة الارقام والقوائم وبعضها الاخر يعمل على تسهيل التعامل مع انظمة التشغيل اما الجزء الاكبر من هذه المكتبات فهو اختياري يتم استيراده متى ما دعت الحاجة لذلك. كما ان هناك مكتبات اخري تحتاج الى تنصيب قبل ان يتمكن المبرمج من استيرادها واستخدامها في برنامجه. وهذه المكتبات مجانية ويمكن تحميلها وتنصيبها اما من الموقع الخاص بالمطورين لهذه المكتبة او من موقع [http://pypi.python.org](http://pypi.python.org/) والذي يحتوي حتى وقت كتابة هذا الكتاب على 69478 مكتبة مجانية جاهزة يمكن استخدامها في بناء التطبيقات المختلقة.

التكامل مع لغات برمجية اخرى: يمكن استخدام بايثون كلغة مساندة تمكن المستخدم لبرنامج مكتوب بلغة سي (C) او سي بلس بلس (C++) مثلا من زيادة او تعديل خصائص ذلك البرنامج ليتناسب مع احتياج المستخدم. ومن أقرب الامثلة على ذلك هو استخدام لغة بايثون في برنامج فري كاد (FreeCAD) كلغة برمجة نصية لتحكم بكافة خصائص البرنامج ووظائفه.

## اصدارات لغة بايثون ¶

هناك اصدارتان لبايثون. الإصدارة الأولى تعرف ببايثون 2 وهي الاقدم والاصدارة الاخرى تدعى بايثون 3 وهي الاحدث.

## توزيعات بايثون ¶

بالاضافة الى التوزيعة الرسمية التي يمكن تحميلها من [www.python.org](http://www.python.org/) هناك توزيعات اخري تأتي محملة بمكتبات ومدير ادارة مكتبات تهدف الى اراحة المستخدم من عناء تحميل المكتبات واضافتها للاصدارة الرسمية. معظم هذه التوزيعات تأتي على شكل اصدارة مجانية واصدارة تجارية.ومن بين هذه التوزيعات مايلي:

توزيعة Anaconda: يمكن تحميل هذه التوزيعة من [www.continuum.io](http://www.continuum.io/) ويمكن تنصيب هذه التوزيعة على ويندوز وماك ولينكس. تحتوي التوزيعة على اكثر من 100 مكتبة و مدير ادارة مكتبات يدعى conda. كما يوجد لهذه التوزيعة اصدارة مصغرة تدعى miniconda واصدارات تجارية اخرى.

توزيعة Enthought Canopy: يمكن تحميل هذه التوزيعة من [www.enthought.com/products/canopy](http://www.enthought.com/porducts/canopy) . تأتي هذه التوزيعة باصدارة تجارية واصدارة مجانية لمدة سنه قابلة للتجديد للاكاديمين والطلاب و لكن بعد اجراء عملية التسجيل. تحتوي الاصدارة على مفسر بايثون 2 و اكثر من 450 مكتبة متخصصة للاغراض العلمية والتحليلية. كما ان عملية التسجيل تسمح للمستخدم بالاطلاع على الفيديوهات التعليمية المعمولة من قبل Enthought .

توزيعة ActivePython:

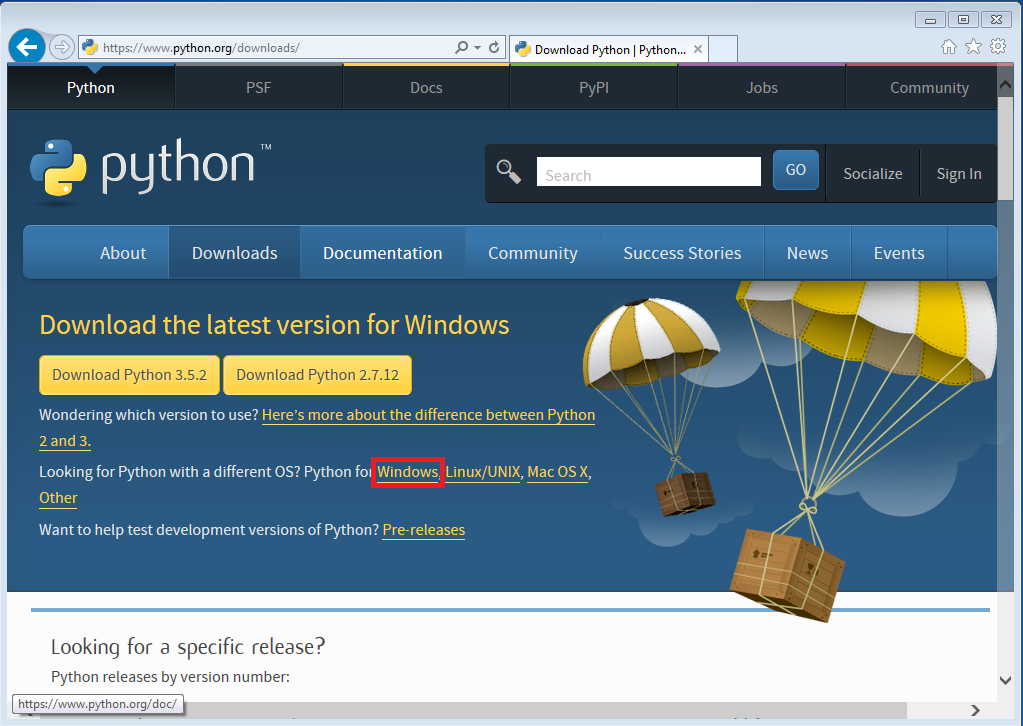
## تنصيب بايثون ¶

بما ان ميكروسوفت ويندوز مازال اكثر انطمة التشغيل شيوعا وخاصة في الوطن العربي فاننا سنبدأ بشرح طريقة تنصيب مفسر لغة بايثون على هذا النظام اولا.

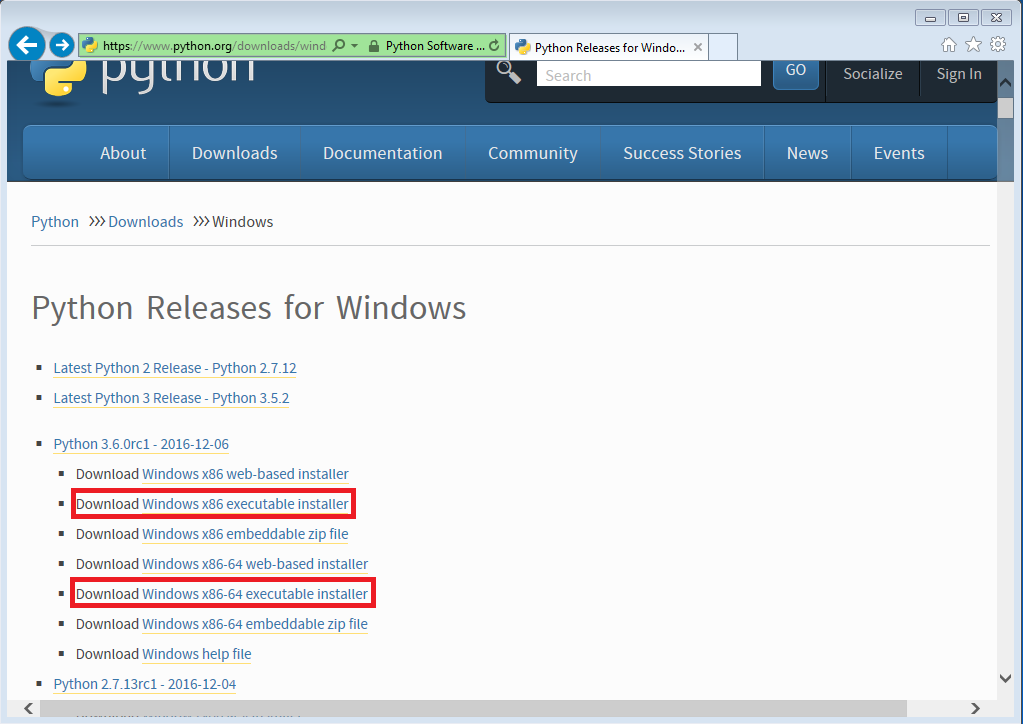
تنصيب بايثون على نظام ويندوز: ¶

كما ذكرنا سابقا فان بايثون ياتي بتوزيعة رسمية من موقع [www.python.org](http://www.python.org/) وتوزيعات اخرى تحتوي على بعض الاضافات مثل anaconda, canopy وغيرها من التوزيعات وفي هذا المقام فاننا سوف نقوم بشرح طريقة تنصيب مفسر بايثون 3 من الموقع الرسمي وكذلك توزيعة anaconda لشهرتها الواسعة بين العلماء والباحثين. لذلك اذا كنت تستخدم اي من انظمة ويندوز سوا القديم منها او الجديد فانك تحتاج الى تنصيب مفسر لغة بايثون على هذا النظام. واليك الخطوات التالية التي تساعدك على عمل ذلك:

1. اذهب الى الموقع التالي: [www.python.org/downloads](http://www.python.org/downloads)



2. اضغط على الرابط "windows" للانتقال الي خيرات التحميل الخاصة بنظام وندوز



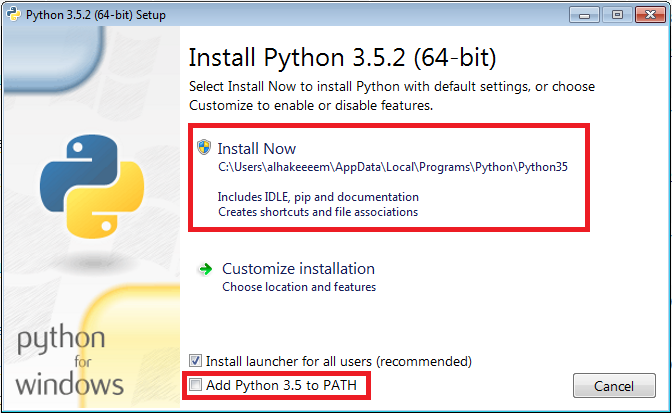
3. هناك ثلاث خيرات لتنصيب مفسر بايثون 3 على ويندوز. قم باختيار التنصيب عن طريق ملف تنفيدي كما هو معلم في الصورة السابقة. أذا كان نظام الوندوز لديك 32 بت اضعط على الرابط

"Windows x86 executable installer"

اما اذا كان نظام الويندوز 64 بت فقم بالضغط على الرابط

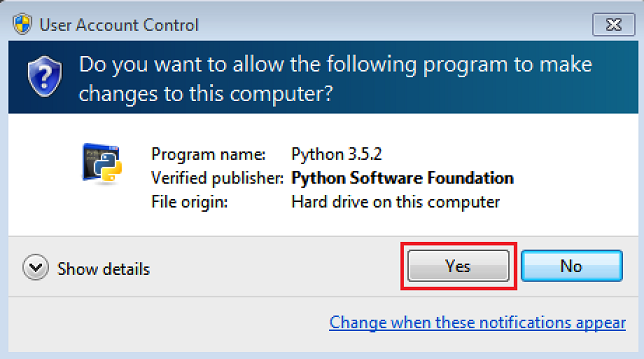
"Windows x86-64 executable installer"

4. قم بتشغيل الملف الذي قمت بتحميله لبدأ عملية التنصيب

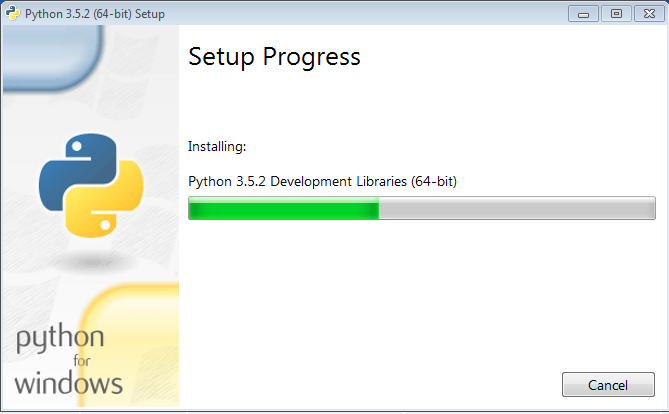


5. ضع علامة صح على الخيار "Add Python 3.5 to PATH" ثم اضغط على "Install Now"

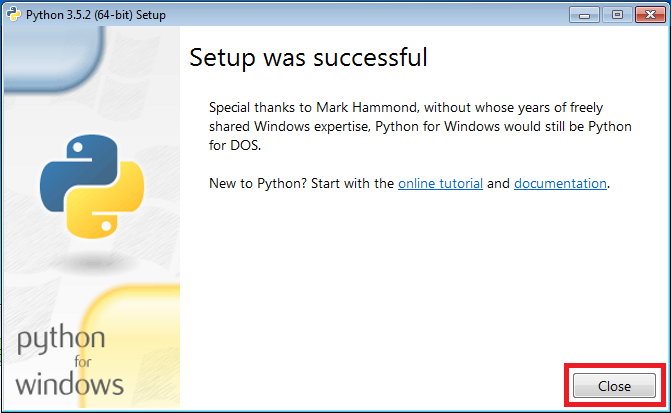
6. قد تظهر لك ملاحظة كما في الصورة التالية تسألك عن رغبتك في الاستمرار في عملية التنصيب فاضغط على "Yes"



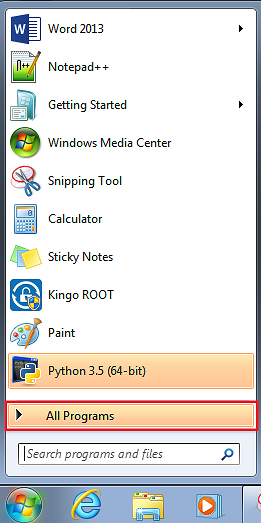
7. تستغرف عملية التنصيب قرابة الدقيقة.



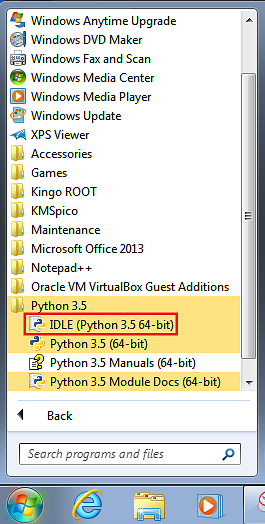
8. وعند اكتمال التنصيب سوف تظهر لك ملاحظة ان عملية التنصيب تمت بنجاح عندها اضغط على "Close"



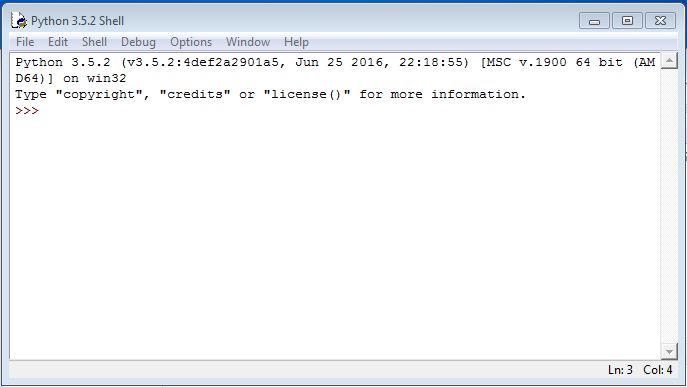
9. لتشغيل مفسر بايثون هناك طريقتان فبحسب نظام ويندوز 7 مثلا يمكنك الذهاب الى قائمة ابدأ ثم الضغط على كافة البرامج كما في الصورة التالية:



وسوف تجد ان عملية التنصيب اضافت اربعة ملفات داخل قائمة البرامج في مجلد بايثون. اول هذه الملفات يمكنك من تشغيل مفسر بايثون باستخدام "IDEL" وهو بيئة تطوير خاصة ببايثون كما هو موضح بالشكل التالي:



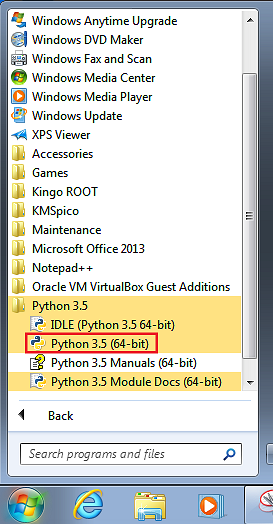
فعند الضغط على هذا الخيار تظهر لك نافذة "IDEL" التفاعلية كما هو موضح بالشكل التالي:



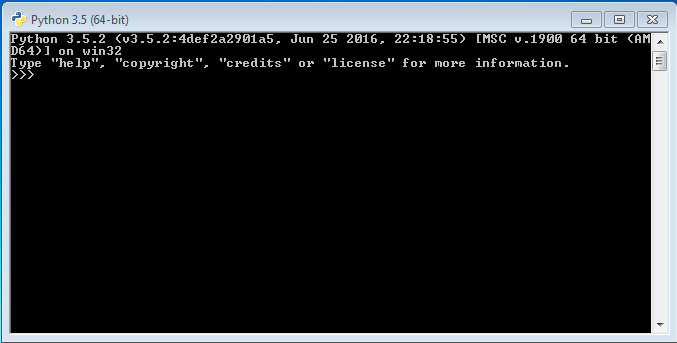
وعند كتابة اي امر بعد العلامات الثلاث "<<<" والضغط على زر الادخال تظهر لك النتائج مباشرة في نفس النافذة لذلك سميت بالنافذة التفاعلية.

تمرين : اذا كنت تعمل على نظام ويندز فقم باتباع الخطوات السابقة لتنصيب مفسر بايثون 3 ثم قم بتشغيل مفسر بايثون من خلال "IDEL" جرب كتابة الامر التالي 2+4 ولاحظ النتيجة بعد الضغط على زر الادخال؟

اما الطريقة الثانية لتشغيل مفسر بايثون فتتم بالضغط على الملف الثاني من مجلد بايثون في قائمة البرامج كما يظهر في الشكل التالي:



فعند الضغط على هذا الخيار يظهر لك مفسر بايثون 3 في محرر الاوامر الخاص بنظام ويندوز كما في الشكل التالي:



تمرين: قم بتشغيل مفسر بايثون 3 داخل محرر اوامر ويندوز و جرب كتابة الامر 4+2 لترى ناتج الجمع بعد الضغط على زر الادخال؟

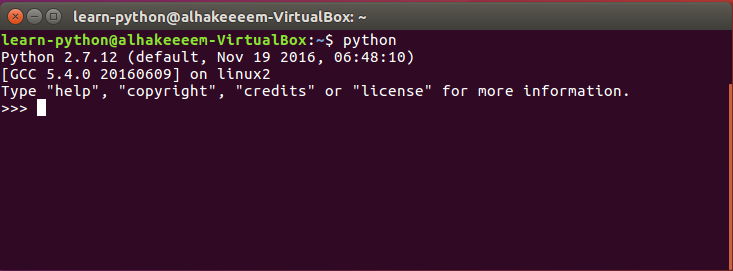
تنصيب بايثون على نظام ماك ولينكس: ¶

يأتي نظاما تشغيل ماك ولينكس وقد نصب عليهما مفسر بايثون ذو الاصدرة رقم 2. ويمكنك التأكد من ذلك باتباع الخطوات التالية:

1. قم بتشغيل محرر الاوامر في الماك او لينكس والذي يدعى "Terminal"

2. اكتب في محرر الاوامر الامر python بعد رمز الدولار "$" واضغط على زر الادخال:

سوف تجد ان مفسر بايثون قد ظهر في محرر الاوامر ومن خلال البيانات التي كتبت عن اصدارة المفسر سوف تجدها الاصدارة الثانية كما في الشكل التالي:



لتنصيب الاصدارة 3 على ماك اتبع الخطوات التالية:

1. اذهب الى الموقع التالي: [www.python.org/downloads](http://www.python.org/downloads)

2. اضغط على الرابط "Mac OSX" للانتقال الي خيرات التحميل الخاصة بنظام ماك

# الفصل الثاني

# المبادئ الأساسية للغة بايثون

بعد ان تأكدنا من ان نظام التشغيل الذي نعمل عليه يحتوي على احدى اصدارات لغة بايثون 3 يمكننا الان ان نبدأ رحلة التعلم والتي اتمنى ان تكون حافلة بالمتعة والفائدة.

## أهداف الباب

عند اتمام هذا الباب يجب ان يكون لديك المام بعدة مبادئ اساسية عن لغة بايثون والتي من اهمها:

طريقة كتابة المتغيرات

طريقة تدوين الملاحظات على الكود البرمجي

كيفية طباعة نص على شاشة الكمبيوتر

الفرق بين استخدام الاحرف الصغيرة والكبيرة في كتابة الكود البرمجي.

اجراء العمليات الحسابية الاساسية

انواع البيانات الاساسية في لغة بايثون.

القاعدة التي تحكم عدد المسافات الفارغة المتروكة قبل بداية كل سطر برمجي

## المتغيرات

المتغيرات هي اسماء تستخدم للدلالة على قيم بيانات موجودة في ذاكرة الكمبيوتر. واستخدام المتغيرات في كتابة الاكواد البرمجية ذو اهمية قصوى بحيث لا يكاد يخلو برنامج من وجود متغير واحد او اكثر وذلك لانها تسهل على المبرمج تذكر البيانات باسماء يسهل حفظها بدلا من استخدام قيم البيانات ذاتها. لاسناد قيمة الى متغير فاننا نختار اسما مناسب للمتغير ومن ثم نضع علامة اليساوي واخيرا نضع القيمة المراد اسنادها للمتغير كما في المثال التالي :

x=5

في المثال السابق علامة اليساوي تمسى معامل الاسناد. لذلك يصبح الان بمقدورنا استخدام x عوضا عن القيمة 5 في اي عمليات حسابية. كما يجب التنويه الى ان قيمة x قابلة للتغيير. فعند تكرار المثال السابق باستخدام القيمة 10 فان x اﻵن تصبح تشير للقيمة 10 عوضا عن القيمة 5 كما في المثال التالي:

x=5

x=10

print(x)

ان لغة بايثون تتطلب التقيد بقواعد اساسية عند كتابة أسماء المتغيرات وتتلخص هذه القواعد فيما يلي: 

المتغيرات يجب ان تبدأ بحرف او شرطة سفلية. عدا ذلك فان مفسر بايثون يعطي رسالة بوجود خطأ

أسماء المتغيرات يمكن ان تكون حرف او كلمة او مجموعة كلمات مربوطة بشرطة سفلية

يمكن استخدام الارقام في كتابة أسماء المتغيرات ولكن لايمكن استخدامها في بداية الاسم

لايمكن استخدام اي رمز في كتابة المتغيرات عدا الاحرف والارقام والشرطة السفلية.

المتغيرات المكتوبة بالاحرف الكبيرة يتعامل معها مفسر بايثون على انها مختلفة عن المتغيرات المكتوبة بالاحرف الصغيرة.

أسماء المتغيرات يجب ان تكون مختلفة عن الكلمات المستخدمة في التركيب اللغوي للغة بايثون والجدول التالي يبين الكلمات المحجوزة من قبل لغة بايثون:

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

المثال التالي يلخص القواعد السابقة:

\_car=2

password=4

user\_code=56

student1=80

t2m=40

تمرين: قم بتشغيل مفسر بايثون الثفاعلى واسند القيمة 100 للمتغير n. في سطر ثاني قم بتغير قيمة n الى القيمة 50. في سطر ثالث حاول ان تعرف القيمة النهائية للمتغير

n

n=100

n=50

print(n)

تدوين الملاحظات

تسمح لغة بايثون كغيرها من لغات البرمجة للمبرمج بان يكتب ملاحظاته داخل الكود البرمجي من اجل ان تساعدة على تذكر وظيفة الكود البرمجي او من اجل اعطاء شروحات وافيه للمبرمجين الاخرين الذين قد يعملون على صيانة وتطوير الكود البرمجي في المستقبل. ويمكن كتابة ملاحظة من سطر واحد في اي مكان من الكود البرمجي ولكن بعد ان يسبق الملاحظة علامات الهاش تاق (#). فعند كتابة علامة (#) في بداية السطر فان بايثون يعتبر جميع محتويات هذا السطر ملاحظات للمستخدم وليست كود برمجي كما في المثال التالي:

# ملاحظات وتعليقات على الكود تكتب هنا

x=10

كما يمكن للمبرمج ان يدون ملاحظاته باستخدام علامة (#) بعد ان كتب سطر من الكود البرمجي كما في المثال التالي

x=10 # ملاحظات وتعليقات على الكود البرمجي تكتب هنا

فمفسر بايثون هنا يتعامل مع الجزء الذي يسبق علامة (#) على انه كود برمجي يتحاج الى تفسير والجزء الذي يقع بعد علامة (#) يعتبره ملاحظات وتعليقات للمبرمج لاتحتاج الى تفسير فيتجاهلها.

كما يمكن كتابة ملاحظة متعددة السطور باستخادام علامة التنصيص الثلاثية كما في المثال التالي:

'''

تعليقات وملاحظات

تعليقات وملاحظات

تعليقات وملاحظات

'''

x=10

وسوف نتحدث بشىئ من التفصيل عن انواع علامات التنصيص واستخداماتها ولكن ليس الان بل عند بدء الحديث عن البيانات النصية

الطباعة على شاشة الكمبيوتر

ان من المعتاد عند تعلم اي لغة برمجة جديدة ان يتم البدء بتعلم كيفية الطباعة على شاشة الكمبيوتر و ذلك لانها تعتبر وظيفة اساسية في جميع لغات البرمجة.ونحن سوف نسير على هذا العرف هنا على الرغم من تأخيرنا له قليلا. فامر الطباعة على الشاشة في بايثون يكون باسيتخدام الدالة ()print متبوعا بما يراد طباعته داخل قوسي الدالة. وبما اننا في بداية المشوار ولم نتطرق لمفوهم المتغيرات فسوف نبدأ بطباعة قيم رقمية اولا. فالسطر البرمجي التالي يقوم بطباعة القيمة 5 على الشاشة:

In [27]:

print(5)

5

تمرين: قم باسناد القيمة 25 للمتغير m وفي سطر ثاني قم بطباعة قمية m؟

In [2]:

m=25

print(m)

25

الاحرف الكبيرة ≠ الأحرف الصغيرة:

لغتنا العربية الجميلة لاتحوي على مفهوم الحروف الصغيرة والكبيرة بعكس ما هو موجود في اللغة الانجليزية. وبما ان لغة بايثون مكتوبة باللغة الانجليزية فان هذه اللغة تهتم بما اذا كان الكود البرمجي او جزء منه مكتوب بالاحرف الصغيرة او الكبيرة. فالمتغير b يتعامل معه مفسر بايثون على انه مختلف عن B. فعندما نسند القيمة 5 مثلا للمتغير b فان اسناد القيمة 10 للمتغير B لا يغير من قيمة b لان مفسر بايثون يتعامل معهما على انهما متغيرين مختلفين كما في المثال التالي:

In [26]:

b=5

B=10

print(b)

print(B)

5

10

اجراء العمليات الحسابية

كما هو المعتاد مع لغات البرمجة الاخرى فان العمليات الحسابية الاساسية يمكن القيام بها باستخدام الرموز الاتية:   
(+) للجمع   
مثال:

In [10]:

print(3+4)

7

(-) للطرح   
مثال:

In [4]:

print(7-5)

2

(\*) للضرب   
مثال:

In [5]:

print(2\*3)

6

(/) للقسمة   
مثال:

In [6]:

print(8/2)

4.0

(\*\*) للأس   
مثال:

In [7]:

print(3\*\*2)

9

(%) لباقي   
مثال:

In [8]:

print(7%3)

1

(//) لناتج القسمة بعد اهمال الباقي   
مثال:

In [11]:

print(11//3)

3

بالنسبة لعملية القسمة فان هناك اختلاف بسيط بين اصدارة بايثون 2 و 3 يجب التنبه عليه. فعند اجراء عملية القسمة في اصدارة بايثون 2 على اعداد طبيعية فان ناتج القسمة يكون عدد طبيعي. بمعنى انه اذا اردنا قسمة العدد 3 على العدد 2 مثلا فان ناتج القسمة يكون 1 وليس 1.5 هذه المشكله غير موجوده في اصدارة بايثون 3. ولتصحيح هذه المشكلة لمن يستخدمون الاصدار 2 يمكنهم استخدام الاعداد الصحيحة في عملية القسمة سواء في البسط او المقام او كليهما. وكما هومتعارف عليه في علم الرياضيات فان عمليتي القسمة والضرب تسبق عملية الطرح والجمع وعملية الاس تسبق الضرب والجمع الا اذا استخدمت الاقواس لتحديد اسبقية العمليات الحسابية. وهذه امثلة اخرى توضح هذا المفهوم:

In [13]:

print( 3+4\*2 )

11

In [14]:

print( 10/2-1 )

4.0

In [15]:

print( 10\*\*2/5+4 )

24.0

In [16]:

print( (3+4)\*2 )

14

In [12]:

print( 15/(6-3) )

5.0

أنواع البيانات

البيانات العددية

لم نستخدم حتى الان الا نوع واحد من البيانات العددية وهي الاعداد الصحيحة (Integers) كالعدد 5 و 10 الخ. وهناك بيانات عددية اخرى كالاعداد العشرية (float) والبيانات الثنائية (binaries) والبيانات الست عشرية. فالاعداد العشرية هي الاعداد التي تحتوي على فاصلة عشرية سوا كانت موجبة او سالبة كما في المثال التالي:

In [17]:

x=5.34

y=-3.0

أما الاعداد الثنائية فهي الاعداد التي تحتوي على قيمتين فقط 0 و 1. وللتفرقة بين هذا النوع من البيانات والاعداد الصحيحة يتم وضع "0b" قبل العدد الثنائي لكي يميزه عن الاعداد الصحيحة. فالعدد 0b11 عندما يكتب في بايثون فان بايثون يتعرف عليه على انه عدد ثنائي. ولمعرفة قيمة 0b11 بالاعداد الصحيحة يمكن استخدام الدالة ()int لتحويل العدد الثنائي لعدد صحيح كما في المثال التالي:

In [3]:

x=0b11

int(x)

Out[3]:

3

وللتحويل من الاعداد الصحيحة الى الاعداد الثنائية يمكن استخدام الدالة ()bin كما في المثال التالي:

In [3]:

x=10

bin(10)

Out[3]:

'0b1010'

ولمعرفة الطريقة التفصيلية لطريقة التحويل من عدد ثنائي الى صحيح او العكس يمكن الرجوع الى الملحق أ للاستزادة.

الاعداد الست عشرية هي الاعداد التي تتكون من ستة عشر رقما وهي الاعداد 0 الي 9 بالاضافة الى الاحرف a,b,c,d,e,f. وللتمييز بين الاعداد الست عشرية والاعداد الصحيحة تستخدم "0x" قبل الرقم الست عشري. فعند كتابة 0x11 فان بايثون يتعامل مع هذا الرقم على انه ست عشري. ولتحويل هذا الرقم الى عدد صحيح فاننا نستخدم نفس الدالة السابقة ()int كما في المثال التالي:

In [2]:

x=0x11

int(x)

Out[2]:

17

ولتحويل الاعداد الصحيحة الى اعداد ست عشرية يمكن استخدام الدالة ()hex للقيام بذلك كما في المثال التالي:

In [6]:

x=10

hex(x)

Out[6]:

'0xa'

ولمعرفة الطريقة التفصيلية لطريقة التحويل من عدد ست عشري الى صحيح او العكس يمكن الرجوع الى الملحق أ للاستزادة.

البيانات المنطقية

البيانات المنطقية هي البيانات التي تحوى احدى القيمتين صح True أو خطأ False . لاحظ ان هاتين القيمتين تبتدأ بحرف كبير في لغة بايثون. وهناك عمليات مقارنة كثير تنتج عنها احدى هاتين القيمتين. فعملية مقارنة التساوي في بايثون هي علامتي يساوي متتابعتين "==" فعند مقارنة عددين مثلا فاننا نكتبهما بالشكل التالي:

In [8]:

5==4

Out[8]:

False

فعند عدم تساوي الرقمين مثلا فاننا نحصل على القيمة False اي خطأ. وعندما تكون القيمتين متساويتين فان الناتج يكون "True" اي صحيح كما في المثال التالي:

In [10]:

0b1111==15

Out[10]:

True

ويمكننا التأكد من النتيجة السابقة بتحويل القيمة 0b1111 الى عدد صحيح كما تعلمنا سابقا.

In [12]:

int(0b1111)

Out[12]:

15

البيانات النصية

كل ما يكتب بين علامتى تنصيص سواء كانت احادية ('z') او ثنائية ("z") او ثلاثية ('''z''') يطلق عليه بيان نصي string أو بشكل مختصر str. فالقيمة "123" يتعامل معها بايثون على انها سلسلة من الرموز النصية الرقمية مختلفة عن القيمة العددية 123. ويمكن التأكد من ذلك باستخدام الدالة ()type كما في المثال التالي:

In [6]:

x="123"

type(x)

Out[6]:

str

وهناك عدة قواعد يجب اتباعها عند كتابة البيانات النصيية والتي يمكن تلخيصها في النقاط التالية:

علامتي التنصيص لكل بيان نصي يجب ان تكون من نفس النوع. فعند البدء بعلامة تنصيص احادية يجب ان ينتهي البيان النصي بعلامة تنصيص احادية وكذلك الحال مع علامة التنصيص الثنائية والثلاثية كما في المثال التالي:

In [18]:

student\_name='Ahmad'

student\_name="Ali"

student\_name='''Hassan'''

لاحظ انه عندما نقوم باستخدام علامتي تنصيص مختلفة فان ذلك سوف يجعل مفسر لغة بايثون يظهر رسالة تفيد بوجود خطأ كما في المثال التالي:

In [19]:

print('welcome to python")

File "<ipython-input-19-314a6067dc16>", line 1

print('welcome to python")

^

SyntaxError: EOL while scanning string literal

علامة التنصيص الاحادية والثنائية تستخدم لكاتبة نصوص من سطر واحد بينما علامة التنصيص الثلاثية تسمح بطباعة اكثر من سطر. فعند استخدام علامة التنصيص الاحادية والثنائية لطباعة بيان نصي متعدد السطور فان مفسر بايثون يظهر لنا رسالة تفيد بوجود خطأ كما في المثالين التاليين:

In [33]:

print('welcome

to

python')

File "<ipython-input-33-5fc875d5d1be>", line 1

print('welcome

^

SyntaxError: EOL while scanning string literal

In [20]:

print("welcome

to

python")

File "<ipython-input-20-5fc875d5d1be>", line 1

print('welcome

^

SyntaxError: EOL while scanning string literal

اما عند استخدام علامة التنصيص الثلاثية فان مفسر بايثون يتم عملية الطباعة بدون اظهار اي رسالة خطأ كما في المثال التالي:

In [32]:

print('''welcome

to

python

''')

welcome

to

python

يمكن استخدام رمز التجاهل "\" لتمكين علامة التنصيص الاحادية والثنائية من طباعة اكثر من سطر على الشاشة. لكن هناك فرق بين الطريقتين فعلامة التنصيص الثلاثية تبقي البيان النصي كما كتب من ناحية تعدد السطور بينما استخدام رمز التجاهل يقوم بكتابة البيان النصي كسطر واحد كما في المثالين التاليين:

In [26]:

print ('welcome \

to \

python')

​

print("welcome \

to \

python")

welcome to python

welcome to python

In [27]:

print('''welcome

to

python''')

welcome

to

python

يمكن استخدام علامة تنصيص او اكثر داخل علامتي تنصيص اخرى ولكن بعد التأكد من ان علامة التنصيص الداخلية مختلفة عن علامة التنصيص المستخدمة في بداية ونهاية النص. كما يمكن استخدام رمز التجاهل المشار اليه سابقا لاداء نفس الوظيفة اذا كانت علامة التنصيص الداخلية مشابهة لعلامتي التنصيص الخارجية. كما ماهو موضح في الامثلة التالية:

In [30]:

print("They are 'like' python")

print('They are \'like\' python')

print("My brother's car is quite old")

print('My brother\'s car is quite old')

They are 'like' python

They are 'like' python

My brother's car is quite old

My brother's car is quite old

ترك مسافات

ترك مسافات في تركيب بايثون اللغوي في حين ان لغات البرمجة الاخرى تستخدم مصطلحات واقواس لتحديد بداية ونهاية الاجزاء الداخلية للكود البرمجي فان لغة بايثون تتبع نظام ترك المسافات عند بداية كتابة السطر البرمجي لاداء نفس المهمة. فلغة جافا تستخدم الاقواس لتحديد جزئية الكود الداخلي و علاقته ببقية الاجزاء كما في المثال التالي:

public class HelloWorld {

  public static void main(String[] args) {

    System.out.println("Hello, World");

  }

}

بينما لغة بايثون تعمد الى ترك مسافة عند بداية كتابة الجزء الداخلى للكود لتجديد مداه وعلاقتة بالاجزاء الاخرى كما في المثال التالي:

In [29]:

for i in [1,2,3,4]:

print (i)

1

2

3

4

ليس من المهم ان تفهم وظيفة الكود البرمجي السابق الان لاننا سوف نتطرق اليه في وقت لاحق ولكن المهم ان تعرف ان لغة بايثون تهتم بترك مسافات عند بداية كتابة الاسطر البرمجية لتحديد الاجزاء الداخلية من الكود.فعند كتابة برنامج من سطر واحد مثلا فان ترك اي مسافة قبل بداية السطر البرمجي يجعل مفسر بايثون يرفض التركيب اللغوي و يظهر رسالة تبين سبب المشكلة هو ترك مسافة عند بداية كتابة السطر البرمجي في موضع لايستدعي ترك اي مسافة. لكن ترك مسافة في اي موضع اخى من السطر ليس له اي تأثير على التركيب اللغوي. كما يجب الاشارة الى ان ترك اسطر فارغة بين اسطر الكود البرمجي ليس له اي تأثير يذكر ايضا. سوف نعاود الحديث عن ترك المسافات عندما نبدأ الحديث عن الحلقات التكرارية والدوال في بايثون حيث تستدعى الحاجة للحديث عن ترك مسافات عند كتابة هذه التراكيب اللغوية.

In [ ]:

​

​

In [38]:

fruits={'Apple':4,'Orange':5,'banana':10}

بتم استدعاء البيانات من القاموس بكتابة اسم القاموس ومن ثم وضع مفتاح البيان بين قوسين مربعين كما في المثال التالي:

In [39]:

fruits['Apple']

Out[39]:

4

في معظم لغات البرمجة المعروفه لايمكن استخدام المتغيرات الا بعد تعريفها مسبقا وذلك بتحديد نوع البيانات التي تشير اليه هذه المتغيرات. لكن الامر مختلف تماما في لغة بايثون. فالمبرمج لايجتاج لا يحتاج الى تعريف المتغيرات قبل استخدامها. لذلك يطلق على لغة بايثون بانها ديناميكية لان تقوم بتحديد نوع المتغيرات ذاتيا من خلال التعرف على نوع البيانات المستخدمة مع كل متغير. وهذه الخاصية تعطي المبرمج بلغة بايثون سهوله وسرعة غير مسبوقة في كتابة الاكواد البرمجية والمثال التالي يوضح هذه الخاصية: فعند اسناد قيمة نصية لبايثون يقوم بايثون بشكل تلقائي بالتعرف على نوع البيانات المستخدمة مع المتغير وتحديد نوعة بانه متغير نصي. فالامر type يمكن استخدامة للتعرف على انواع البيانات المخزنه في المتغيرات.