مبادئ البرمجة بلغة بايثون

احمد سالم الصاعدي

3 رمضان 1437

المحتويات

باب 2

اساسيات لغة بايثون

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

1.2 أهداف الباب

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

1. بايثون لغة تفرق بين كون احرف التركيب اللغوي مكتوبة باحرف صغيرة ام كبيرة ونتعامل معها بشكل مختلف
2. بايثون تهتم بعدد المسافات المتروكة قبل بداية كل سطر برمجي (indentation)
3. كيفية طباعة نص على شاشة الكمبيوتر
4. طريقة تدوين الملاحظات على الكود البرمجي (comments)
5. اجراء العمليات الحسابية الاساسية
6. طريقة كتابة المتغيرات (variables) لتخزين البيانات من اجل تحليلها ومعالجتها
7. أنواع البيانات في بايثون (data types)
8.

2.2 الاحرف الصغيرة ≠ الاحرف الكبيرة:

لغتنا العربية الجميلة لاتحوي على مفهوم الحروف الصغيرة والكبيرة بعكس ما هو موجود في اللغة لانجليزية. وبما ان لغة بايثون مكتوبة باللغة الانجليزية فان هذه اللغة تهتم بما اذا كان الكود البرمجي او جزء منه مكتوب بالاحرف الصغيرة او الكبيرة. فالامر print مثلا يكتب بالاحرف الصغيرة كما في المثال التالي:

```
>>> print "welcome to python"
welcome to python
```

ولكن عندما نحاول كتابته احد حروف هذا الامر بحرف كبير فان مفسر لغة بايثون يعطينا خطأ مخبرا ان التركيب اللغوي غير صحيح كما في المثال التالي:

```
>>> prinT "welcome to python"
  File "<stdin>", line 1
    prinT "welcome to python"
                ^
SyntaxError: invalid syntax
```

ربما قد تبادر الى اذهان البعض الان ان تذكر وضع حالة الاحرف ما اذا كانت صغيرة ام كبيرة يعد امرا شاقا. لكن علينا ان نتادرك الموقف بسرعة ونبين لهم ان هناك ضوابط وضعت من قبل مطويرين لغة بايثون تجعل من تذكر وضع حالة الاحرف غاية في السهولة.

الضابط الاول: ان جميع اوامر لغة بايثون دائما تكون مكتوبة بالاحرف الصغيرة.
الضابط الثاني: ان الاحرف الكبيرة تستخدم فقط عند كتابة الثوابت.

قد لا يكون هذا الامرواضحا بما فية الكفاية الان ولكن أعدك ان أوضح هذه المسألة بعد ان أتطرق الى المفاهيم الاساسية في لغة بايثون. فكل ما عليك فهمه الآن هو ان لغة بايثون تفرق بين حالة الاحرف الصغيرة والكبيرة في تراكيبها اللغوية.

3.2 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: بينما لغة بايثون تعمد الى ترك مسافة عند بداية كتابة الجزء الداخلى للكود لتجديد مداه وعلاقتة بالاجزاء الاخرى كما في المثال التالي: 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. يمكنك التأكد من هذا بترك مسافة قبل السطر البرمجي كما في المثال التالي:

```
>>> print "hello"

File "<pyshell# 2>", line 1
print "hello"
^
IndentationError: unexpected indent
>>>
```

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.

4.2 طباعة نص على شاشة الكمبيوتر:

كما هو المعتاد في تعلم اي لغة برمجة جديدة فان البدء دايما ما يكون بتعلم كيفية طباعة نص على شاشة الكمبيوتر. وسوف نسير على هذا العرف نحن هنا ايضا. فامر الطباعة

على الشاشة في بايثون يكون باسيتخدام الامر print متبوعا بما يراد طباعتة. وبما اننا في بداية المشوار ولم نتطرق لمفوهم المتغيرات فسوف نبدأ بطباعة النصوص اولا. فالسطر البرمجي التالي يقوم بطباعة python to welcome على الشاشة:

```
python" to "welcome print
```

لاحظ ان هناك قواعد يجب معرفتها عند طباعة النصوص ومن بين هذه القواعد مايلي:

1. يمكن استخدام علامات التنصيص الاحادية (') او الثنائية (") او الثلاثية ("') لتحيط بالنص المراد طباعته على الشاشة. مع ملاحظة انه يجب ان تسخدم نفس علامة التنصيص في بداية النص واخره وعدم الخلط بينها. كما هو موضح في المثال التالي:

2. علامة التنصيص الاحادية والثنائية تستخدم لكاتبة نصوص من سطر واحد بينما علامة التنصيص الثلاثية تسمح بطباعة اكثر من سطر. كما في المثال التالي:

3. يمكن استخدام رمز التجاهل "لتمكين علامة التنصيص الاحادية والثنائية من طباعة اكثر من سطر على الشاشة. كما في المثال التالي:

4. يمكن استخدام علامة تنصيص او اكثر داخل علامتي تنصيص 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. كما ماهو موضح في الامثلة التالية:

5.2 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 الهاشتاق (#) كما في المثال التالي:

مثال رقم 2.5.1

```
1
2 T=45      # T means temperature
```

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

مثال رقم 2.5.2

```
1 ''' this is a list of temperature data
2 for the first week of year 2010 in
3 sauid arabia'''
4
5 Temperatures=[34,45,43,41,44,35,40]
```

6.2 انواع البيانات في بايثون

تنقسم انواع البيانات في بايثون لثلاثة اقسام. بيانات رقمية وبيانات منطقية وبيانات نصية. البيانات الرقمية تش

1. البيانات المنطقية Boolean : وهي البيانات التي تحتوي على قيمتين فقط صح True و خطأ False
2. القيم العددية : تشمل الاعداد الاعداد الطبيعية "....." والاعداد ذات الفاصلة والاعداد الثنائية التى تتكون قيمة صفر وواحد وبيانات ست عشرية والتي تاخذ القيم من صفر لتسعة بالاضافة الى الخمسة الحروف الاولى من اللغة الانجليزية.
3. القوائم
4. المجموعات

5. المترافقات
6. القواميس (Dictionaries) : هي عبارة عن مجموعة من البيانات الثنائية توضع بين قوسين متعرجين . جزءها الاول يسمى المفتاح (key) والآخر يسمى القيمة (value) يتم الفصل بين هذين الجزئين بنقطتين فوق بعض (:) ويتم الفصل بين كل بيان ثنائي وآخر بفاصلة كما في المثال التالي:

```
>>> my_dic={"Ali":90,"Ahmad":93,"Hassan":85}
```

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

```
>>> my_dic["Hassan"]
85
```

7.2 اجراء العمليات الحسابية الاساسية

كما هو المعتاد مع لغات البرمجة الاخرى فان العمليات الحسابية الاساسية يمكن القيام بها باستخدام الرموز الاتية:

1. "+" للجمع
2. "-" للطرح
3. "*" للضرب
4. "/" للقسمة
5. "**" للأس
6. "%" للباقي
7. "//" لناتج القسمة بعد اهمال الباقي

والامثلة التالية توضح استخدام هذه العمليات:

```
>>> 3+4
7
>>> 7-5
2
>>> 4*6
24
>>> 8/2
4
>>> 3**4
81
>>> 5%4
1
>>> 7//2
3
```

بالنسبة لعملية القسمة فان هناك اختلاف بسيط بين اصدارة بايثون 2 و 3 يجب التنبه له. 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 فان ناتج القسمة يكون 1 وليس 1.5 هذه المشكله غير موجوده في اصدارة بايثون 3.

```
>>> 3/2
1
```

ولتصحيح هذه المشكلة يمكن استخدام الاعداد الصحيحة في عملية القسمة سواء في البسط او المقام او كليهما كما في المثال التالي:

```
>>> 3.0/2
1.5
>>> 3/2.0
1.5
>>> 3.0/2.0
1.5
```

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:

```
>>> 3+4*2
11
>>> 10/2-1
4
>>> 10**2/5+4
24
>>> (3+4)*2
14
>>> 15/(6-3)
5
```

8.2 المتغيرات (variables)

المتغيرات هي اسماء تستخدم لدلالة على قيم بيانات موجود في ذاكرة الكمبيوتر. واستخدام المتغيرات في كتابة الاكواد البرمجية ذو اهمية قصوى بحيث لا يكاد يخلو برنامج من جود متغيرات وذلك لانها تسهل على المبرمج تذكر البيانات باسماء يسهل حفظها بدلا من استخدام قيم البيانات ذاتها. لاسناد قيمة الى متغير فان بايثون يستخدم علامة اليساوي للقيام بذلك كما في الامثلة التالية:

```
>>> x=5
>>> _car="blue"
>>> password="a435"
>>> user_name="omar"
>>> student1=80
>>> t2m=40
```

لاحظ من المثال السابق ان هناك قواعد يجب اتباعها عند كتابة أسماء المتغيرات ونتلخص هذه القواعد فيما يلي:

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

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

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

4. الكلمات المتسخدمة في المتغيرات يجب ان تكون مختلفة عن الكلمات المستخدمة في التركيب اللغوي لبايثون والجدول التالي يبين الكلمات المحجوزة من قبل لغة بايثون:

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

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

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

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 لايحتاج 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:

مثال رقم 2.8.1

```
>>> x="hello"
>>> type(x)
<type 'str'>
>>> t=12
>>> type(t)
<type 'int'>
>>> f=3.4
>>> type(f)
<type 'float'>
>>> b=True
>>> type(b)
<type 'bool'>
>>>
```

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.