

SEPTEMBER 15 – 21, 2019



المبادرة السعودية للمطورين

تعلم .. فكر .. حاول .. أبداع

المبادرة السعودية للمطورين

مسار Python

مشرفي المسار:

عبدالله عوده – انتصار النصار – رؤى كردي – لينا المصعبي



ملاحظات قبل بدء الدروس:

- على المتدربين نشر كل يوم الجزئية التي تم كتابتها من النص البرمجي في الـ **Github** تحت **Topic** بعنوان **saudidevorg** كما تم توضيحه في دروس الـ **Github** سابقاً

- على المتدربين نشر كل يوم مقدار التقدم وصورة لما تم تعلمه وتطبيقه على **Twitter** تحت الهاشتاقات:
#المبادرة_السعودية_للمطورين
_100#يوم_برمجة
#100DaysOfCode

تمنياتنا لك بالتوفيق
المبادرة السعودية للمطورين

اليوم السابع والعشرون

الجمل الشرطية في لغة البايثون

Conditional Statements

Python If ... Else

➤ Python Conditions and If statements

الشروط في لغة بايثون وجملته الاختبار إذا

Python supports the usual logical conditions from mathematics: تدعم البايثون المعاملات المنطقية لمقارنة القيم

Equals: **a == b**

Not Equals: **a != b**

Less than: **a < b**

Less than or equal to: **a <= b**

Greater than: **a > b**

Greater than or equal to: **a >= b**

يمكن استعمال هذه الشروط بعدة طرق والعدد الذي تريده منها وتستخدم غالبا في جمل الاختبار **if** وفي الحلقات التكرارية

These conditions can be used in several ways, most commonly in "if statements" and loops.

"if statement" is written by using the **if** keyword.

نكتب جملة الاختبار **if** / **إذا** باستخدام الكلمة المحجوزة **if**
جملة الاختبار **if** / **إذا** تُستخدم لتنفيذ والتحقق من شرط معين.

سيوضح لك من خلال الأمثلة ..

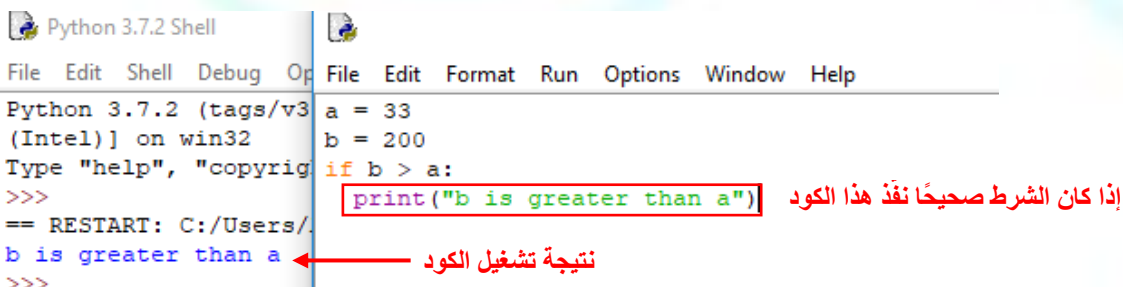
Example

If statement

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

في هذا المثال لدينا المتغير **a** ويحمل قيمة 33 والمتغير **b** ويحمل قيمة 200
فإذا كانت قيمة المتغير **b** أكبر من قيمة المتغير **a** سيتم طباعة الجملة

In this example we use two variables, **a** and **b**, which are used as part of the **if** statement to test whether **b** is greater than **a**. As **a** is 33, and **b** is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".



الذي حدث و بكل بساطة

هل قيمة **b** أكبر من قيمة **a** ؟ وجواب الشرط كان نعم **True** وتبعاً لذلك سيتم تنفيذ أمر الطباعة الموجود في الجملة الشرطية

➤ Indentation

المساحة البادئة / المسافات البيضاء

في لغة **البايثون** يتم استخدام المسافة البيضاء والإزاحات (في بداية السطر) بدلا من الأقواس في لغات أخرى للدلالة على أن هذه الأسطر البرمجية مرتبطة بالسطر السابق لها، ولكي يتم تحديد حجم الجمل البرمجية

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly brackets for this purpose.

تُستعمل لفصل الأكواد البرمجية التابعة لجزء معين من البرنامج أو ما تسمى بالـ **Blocks** تكون بكتابة الكود البرمجي بعد عدد معين من المسافات، تتكون من أربع مسافات غالبا

Example

إن لم تتبع قوانين الإزاحة (المسافة البيضاء) فلن يعمل البرنامج .. انظر المثال التالي

If statement, without indentation (will raise an error)

```
a = 33
b = 200
if b > a: ← colon
print("b is greater than a") # you will get an error
```



لاحظ هنا

لا توجد مسافة بادئة كما في المثال السابق

رمز الـ **colon** : هنا يُشير أن هذا الكود عبارة عن كتلة برمجية **code block**

في حين لغات برمجية أخرى تستخدم **{ }** للدلالة على الـ **block**

الأمر **print** ليس تابعا للبرنامج عامة، بل تابع للأمر أو جملة الاختبار **if**

IndentationError: expected an indented block

← نتيجة تشغيل الكود

قم بتجربة الأمثلة ولاحظ الفرق!

➤ Elif

الجملة **و إلا إذا**

The **elif** keyword is **python's** way of saying

"if the previous conditions were not true, then try this condition".

الأمر **elif** هو اختصار لـ **else if** ويعني أنه إذا لم يتحقق الشرط في الأمر **if** ، يتم التحقق من الأمر **elif**

Example

انظر للمثال التالي

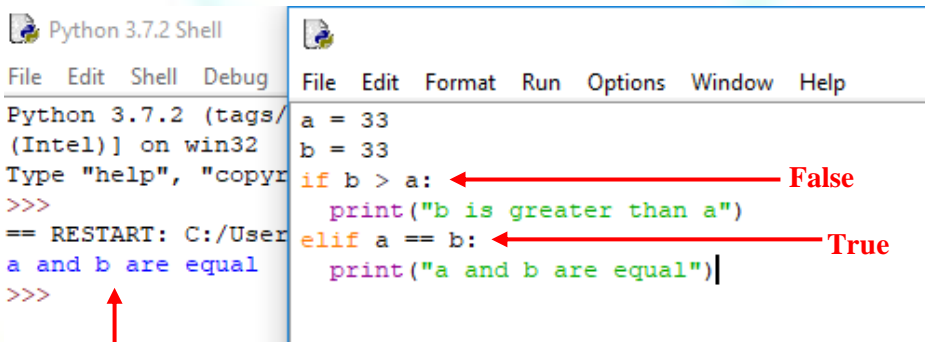
In this example **a** is equal to **b**, so the first condition is not true, but the **elif** condition is true, so we print to screen that "**a and b are equal**".

```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

إذا كان ناتج التعبير المنطقي (**b أكبر من a**) بـ **False**

فسينتقل للشرط الذي يليه وهنا هو **elif** (هل قيمة **a** تساوي قيمة **b**) وجواب الشرط هنا بـ **True**

فسيقوم بتنفيذ أمر الطباعة الموجود في الجملة الشرطية **elif**



```
Python 3.7.2 Shell
File Edit Shell Debug
Python 3.7.2 (tags/
(Intel)] on win32
Type "help", "copyr
>>>
== RESTART: C:/User
a and b are equal
>>>
```

```
File Edit Format Run Options Window Help
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

نتيجة تشغيل الكود

الجملة **elif** غالبا توضع في الوسط بين جملتي **if** و **else**

ستتضح لك الفكرة أكثر في المثال القادم

➤ Else

جملة وإلا

The **else** keyword catches anything which isn't caught by the preceding conditions.

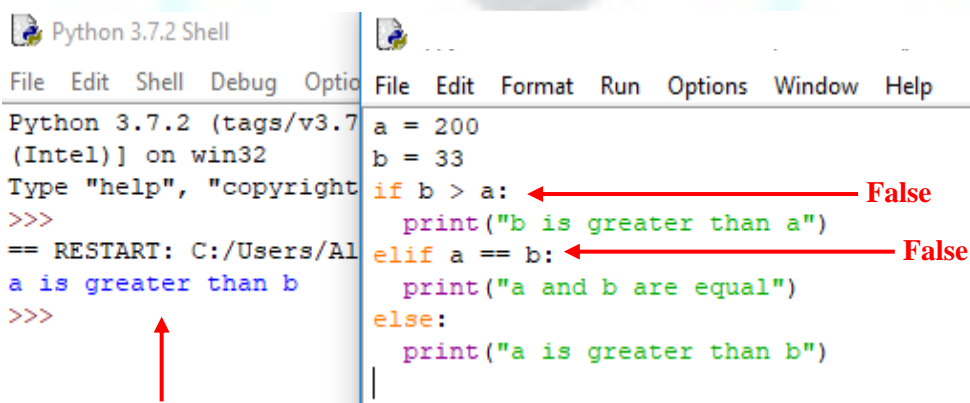
الأمر **else** يوضع دائما في الأخير، يقوم بتشغيل كل ما بداخله من كود إذا لم تتحقق جميع الشروط التي تسبقه ولم يتم تنفيذها

Example

In this example **a** is greater than **b**, so the first condition is not true, also the **elif** condition is not true, so we go to the **else** condition and print to screen that "**a is greater than b**".

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

إذا كان ناتج التعبير المنطقي للجملة **if** (b أكبر من a) بـ **False** يتم تشغيل الأمر الذي يليه
وهنا الأمر هو **elif** (هل a تساوي b) والنتيجة ستكون بـ **False** أيضا، سيتم تشغيل الأمر الذي يليه
وهو أمر الطباعة الموجود في جملة **else** وذلك لعدم تحقق الشروط التي تسبقه



نتيجة تشغيل الكود

الخلاصة

إذا نفذ البرنامج جملة **if** أو **elif** فإنه سيتم تجاهل الجملة **else**
وإذا لم ينفذ البرنامج أي جملة من **if** أو **elif** فإنه سينفذ الجملة **else**

Syntax

إذا طريقة وضع الشروط كالتالي

if condition :

سيتم تنفيذ الكود الذي هنا إذا كان شرط الـ **if** صحيحا

elif condition :

سيتم تنفيذ الكود الذي هنا إذا كان شرط الـ **elif** صحيحا

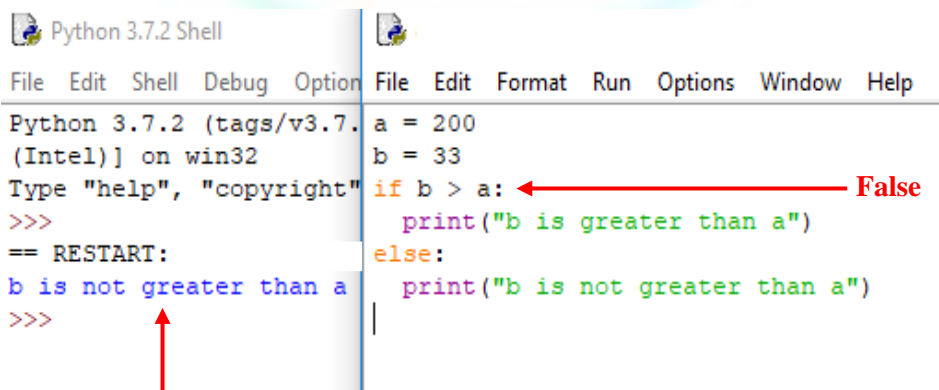
else :

سيتم تنفيذ الكود الذي هنا إذا لم يتحقق أي شرط

يمكنك استخدام الجملة **else** بدون استخدام الجملة **elif** You can also have an **else** without the **elif**

Example

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
else:
    print("b is not greater than a")
```



```
Python 3.7.2 Shell
File Edit Shell Debug Option File Edit Format Run Options Window Help
Python 3.7.2 (tags/v3.7.2:1a7ff65, Aug 14 2019, [AMD64]) on win32
Type "help", "copyright", "credits() or "license()" for more
>>> a = 200
>>> b = 33
>>> if b > a:
>>>     print("b is greater than a")
>>> else:
>>>     print("b is not greater than a")
>>>
b is not greater than a
>>>
```

نتيجة تشغيل الكود

➤ Short Hand If

اختصار للأمر if

يمكنك كتابة الأمر **if** بسطر واحد، إذا كان لديك جملة واحدة تحتاج للتنفيذ (يتم كتابتها على نفس السطر)

If you have only one statement to execute, you can put it on the same line as the **if** statement.

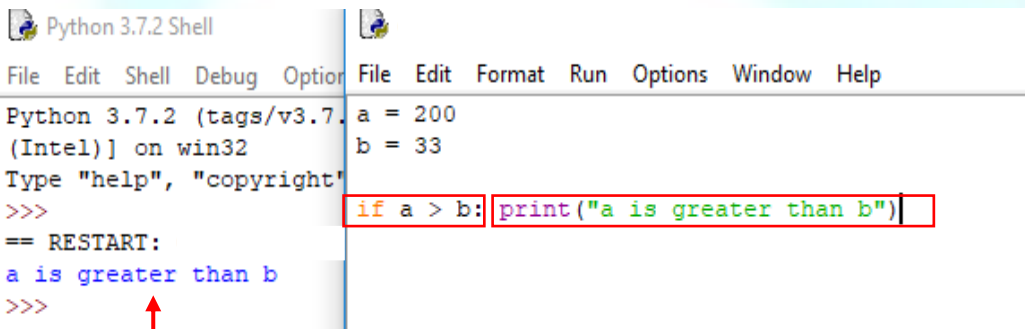
Example

One line **if** statement

```
a = 200
b = 33

if a > b: print("a is greater than b")
```

إذا كان التعبير المنطقي صحيحاً، سيتم طباعة الجملة النصية



نتيجة تشغيل الكود

تابع تقدمك

➤ Short Hand If ... Else

اختصار للأمر **if else**

إذا كان لديك جملة واحدة للتنفيذ، واحدة للأمر **if** والأخرى للأمر **else** يمكنك كتابتهما بسطر واحد (على نفس السطر)

If you have only one statement to execute, one for **if**, and one for **else**, you can put it all on the same line

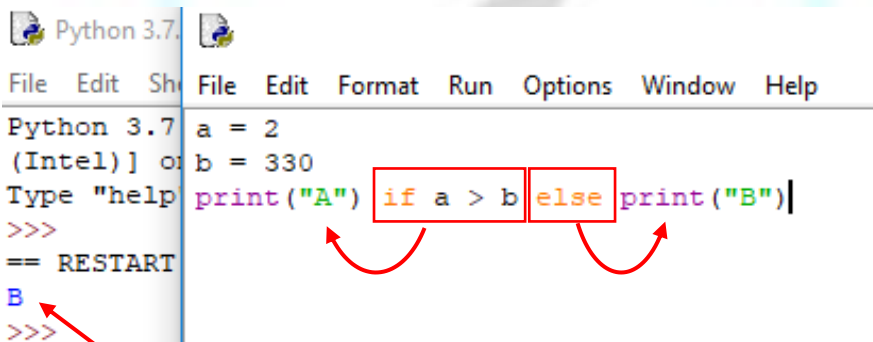
Example

One line **if else** statement

إذا كان التعبير المنطقي $(a > b)$ صحيحا، سيتم طباعة الجملة النصية "A"

غير ذلك سيتم طباعة الجملة النصية "B"

```
a = 2
b = 330
print("A") if a > b else print("B")
```



```
Python 3.7.4
File Edit Shell Format Run Options Window Help
Python 3.7.4 (Intel) on
Type "help()" for more
>>> print("A") if a > b else print("B")
B
>>>
```

نتيجة تشغيل الكود

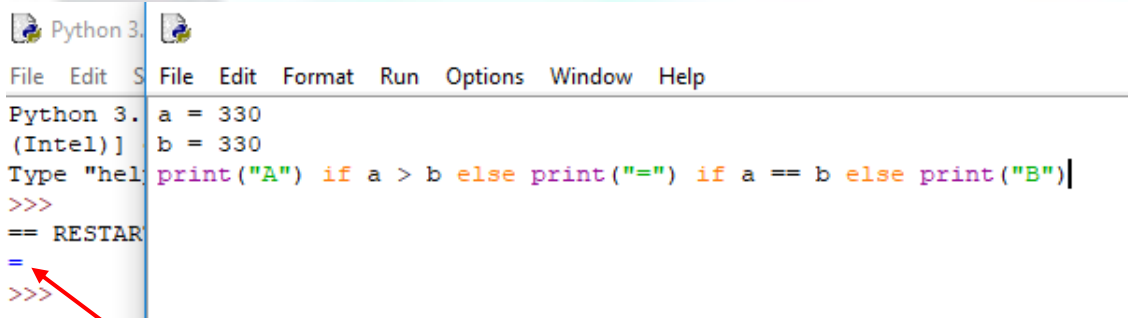
You can also have multiple **else** statements on the same line

يمكنك استخدام عدة جمل **else** على سطر واحد

Example

One line **if else** statement, with 3 conditions

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```



نتيجة تشغيل الكود

أفضل طريقة للفهم هي التطبيق

➤ And

باستخدام العامل **and** يمكنك أن تضع أكثر من شرط بداخل جملة الشرط

The **and** keyword is a logical operator, and is used to combine conditional statements

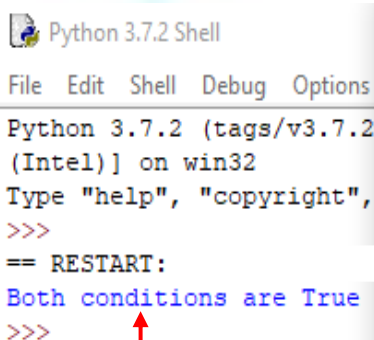
Example

Test if **a** is greater than **b**, **AND** if **c** is greater than **a**

```
a = 200
b = 33
c = 500
if a > b and c > a:
    print("Both conditions are True")
```

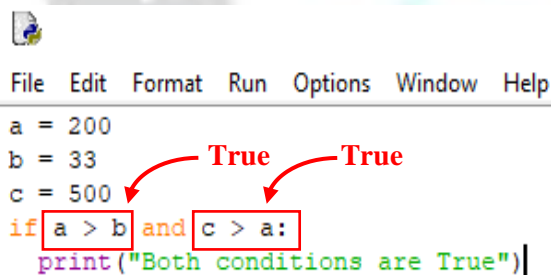
يُستخدم **AND** لتنفيذ الكود بأكثر من شرط

إذا كانت نتيجة جميع الشروط هي **True**



```
Python 3.7.2 Shell
File Edit Shell Debug Options
Python 3.7.2 (tags/v3.7.2
(Intel)] on win32
Type "help", "copyright",
>>>
== RESTART:
Both conditions are True
>>>
```

نتيجة تشغيل الكود



```
File Edit Format Run Options Window Help
a = 200
b = 33
c = 500
if a > b and c > a:
    print("Both conditions are True")
```

➤ Or

أيضا باستخدام العامل **or** يمكنك أن تضع أكثر من شرط بداخل جملة الشرط

The **or** keyword is a logical operator, and is used to combine conditional statements

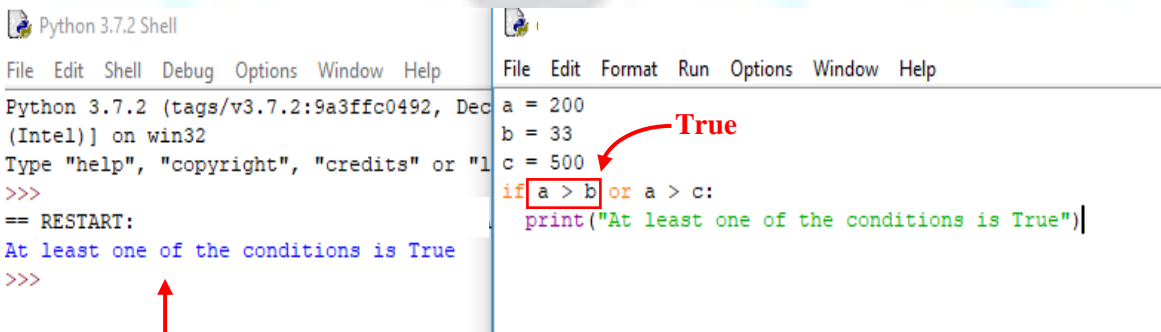
Example

Test if **a** is greater than **b**, **OR** if **a** is greater than **c**

```
a = 200
b = 33
c = 500
if a > b or a > c:
    print("At least one of the conditions is True")
```

يُستخدم **OR** لتنفيذ الكود بأكثر من شرط

إذا كانت نتيجة شرط واحد على الأقل هي **True**



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 15 2019, [AMD64] on win32)
Type "help", "copyright", "credits" or "license()"
>>>
== RESTART:
At least one of the conditions is True
>>>
```

نتيجة تشغيل الكود

➤ Nested If

الشروط المتداخلة

يمكننا وضع عدة شروط بداخل بعضها داخل الجمل الشرطية **if**

You can have **if** statements inside **if** statements, this is called nested if statements.

Example

```
x = 41

if x > 10:
    print("Above ten,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

Python 3.7.2 Shell

File Edit Shell Debug
Python 3.7.2 (tags/...
(Intel)] on win32
Type "help", "copyr...
>>>
== RESTART: C:/User...
Above ten,
and also above 20!
>>>



File Edit Format Run Options Window Help

```
x = 41

if x > 10:
    print("Above ten,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

نتيجة تشغيل الكود

أتممت درسك بنجاح!

روابط قد تهتمك

Useful links

- [If statements](#)
- [Python tutorial If Statement single : العبارة الشرطية](#)
- [Python tutorial Else If Statement العبارة الشرطية](#)
- [Python tutorial Else If Statement مثال على الـ](#)
- [Python tutorial Nested If Statement](#)
- [python - 5 الجمل الشرطية / دورة برمجة بايثون](#)
- [19- Python|| conditional "IF" العبارات الشرطية](#)
- [20- Python|| conditional "IF- Else" العبارات الشرطية](#)
- [21- Python|| conditional Nested IF العبارات الشرطية](#)
- [10 - Python - Beginners Tutorial - IF statement](#)
- [Conditionals in Python - Coderbyte](#)
- [Python Programming #8 - Conditional Statements](#)
- [How To Write Conditional Statements in Python 3](#)
- [Python If Else, If Elif Else, Nested If for Decision Making](#)
- [#19 Python Tutorial for Beginners | If Elif Else Statement in Python](#)
- [الجمل الشرطية - 8 if elif else - تعلم البرمجة بلغة بايثون](#)
- [Learn Python in Arabic #23 if statement and else Python في الجمل الشرطية بايثون - 23](#)
- [Learn Python in Arabic #24 else if statement and else Python في الجمل الشرطية - 24](#)

طبق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم الثامن والعشرون

الحلقات التكرارية في لغة البايثون

Python Loops

While Loop

الحلقات التكرارية في بايثون

➤ Python Loops

Python has two primitive loop commands.

الحلقات التكرارية/الدورانية الموجودة في بايثون هي نوعين

- while loops.
- for loops.

- الحلقة التكرارية عندما / ما دام
- الحلقة التكرارية من أجل

➤ The while Loop

With the **while** loop we can execute a set of statements as long as a condition is true.

تُستخدم الحلقة **while** لتنفيذ الكود عدة مرات طالما أن الشرط صحيح **True**، ولا تتوقف إلا عندما يكون الشرط خاطئاً **False**

Example

Print **i** as long as **i** is less than 6

في هذا المثال قمنا بتعريف حلقة **while** لطباعة الأرقام من 1 إلى 5

1/ **initialization** نضع قيمة مبدئية (متغير للعداد) و ننفذ مرة واحدة `i = 1`

2/ **condition** نقوم بكتابة الشرط الذي يحدد متى تتوقف الحلقة، ينفذ هذا الشرط في كل مرة `while i < 6:`

3/ **statement** تنفيذ أمر الطباعة، وينفذ في كل دورة `print(i)`

4/ **increment** نقوم بزيادة قيمة المتغير (العداد)، وينفذ في كل دورة `i += 1`

```
File Edit File Edit Format Run Options Window Help
Python 3 i = 1
(Intel)] while i < 6:
Type "he. print(i)
>>> i += 1
== RESTA |
1
2
3
4
5
>>>
```

نتيجة تشغيل الكود

تذكر أن تزيد قيمة العداد كما في الخطوة رقم 4 في المثال السابق، وإلا فإن الحلقة التكرارية لن تتوقف

Note: remember to increment **i**, or else the loop will continue forever.

نحتاج أن نقوم بتعريف متغير في البداية كعداد في الحلقة التكرارية ونعطيه قيمة، كما في الخطوة رقم 1 في المثال السابق

The **while** loop requires relevant variables to be ready, in this example we need to define an indexing variable, **i**, which we set to 1.

➤ The **break** Statement

جملة التحكم توقف

التعبير أو الجملة **break** تُستخدم للخروج من حلقة التكرار قبل إكمال تنفيذها، حتى لو مازال شرط الحلقة صحيحا

With the **break** statement we can stop the loop even if the while condition is true.

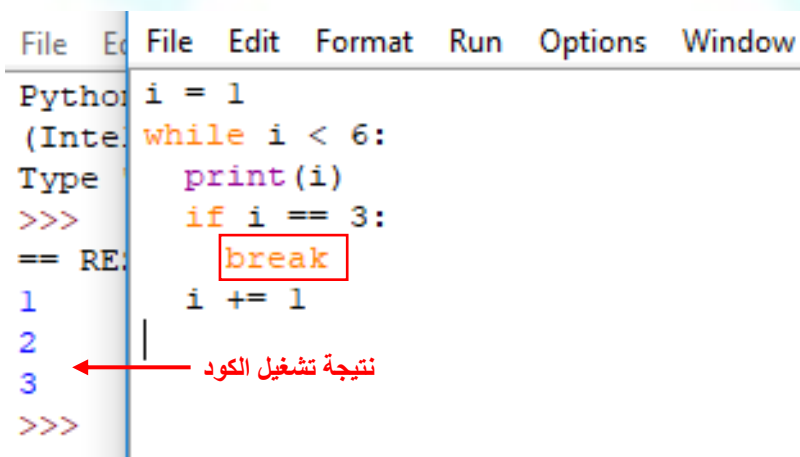
Example

Exit the loop when **i** is 3

هنا في هذا المثال

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

في حال تحقق شرط معين وهو (**i == 3**) يتم إيقاف الحلقة التكرارية والخروج منها للانتقال للكود التالي



```
File Edit File Edit Format Run Options Window
Python i = 1
(Intel while i < 6:
Type print(i)
>>> if i == 3:
== RE: break
1 i += 1
2
3
>>>
```

نتيجة تشغيل الكود

هذا المثال كان سيطبع بواسطة الحلقة التكرارية **while** الأرقام من 1 إلى 5
إلا أننا استخدمنا الأمر **break** للخروج من الحلقة عند تحقق شرط معين
وهو عندما تكون قيمة العداد تساوي 3

جملة التحكم استمر

➤ The **continue** Statement

التعبير أو الجملة **continue** تسمح لنا بتخطي جزء/دورة من الحلقة التكرارية عند شرط معين، أثناء تنفيذها والانتقال إلى الدورة التالية

With the **continue** statement we can stop the current iteration and continue with the next.

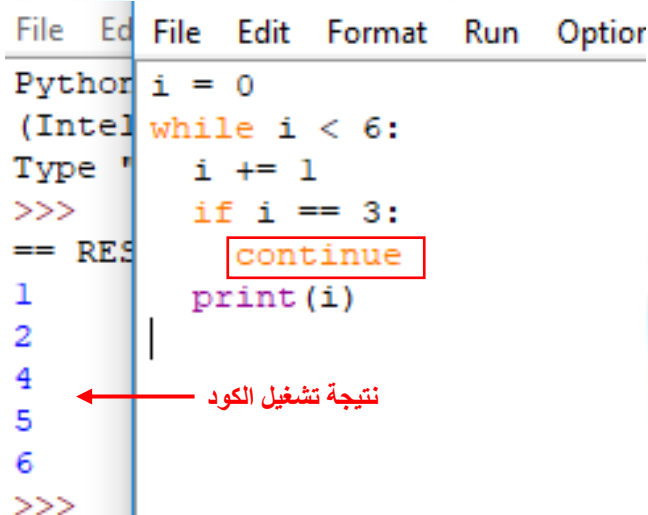
Example

Continue to the next iteration if **i** is 3

هنا في هذا المثال

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

في حال تحقق شرط معين وهو (**i == 3**) ستتجاهل الحلقة التكرارية تنفيذ هذه الدورة وستنتقل للدورة التي بعدها لتنفيذها



هذا المثال كان سيطبع بواسطة الحلقة التكرارية **while** الأرقام من 1 إلى 6 كاملة إلا أننا استخدمنا الأمر **continue** لتخطي دورة معينة عند تحقق شرط معين وهو عندما تكون قيمة العداد تساوي 3 والانتقال إلى الدورة التالية

➤ The **else** Statement

استخدام الأمر **else** مع الحلقة التكرارية **while**

With the **else** statement we can run a block of code once when the condition no longer is true.

الجملة الشرطية **else** ستنفذ كتلة من الكود (مجموعة أوامر) بعدما تتوقف الحلقة التكرارية عن العمل

أي عندما تصبح نتيجة الشرط في الحلقة **while** هي **False**

عند استخدام الأمر **else** مع الحلقة التكرارية **while** فإنه يُنفذ دائماً

إلا إذا كانت الحلقات التكرارية تتكرر باستمرار إلى الأبد دون الخروج منها

Example

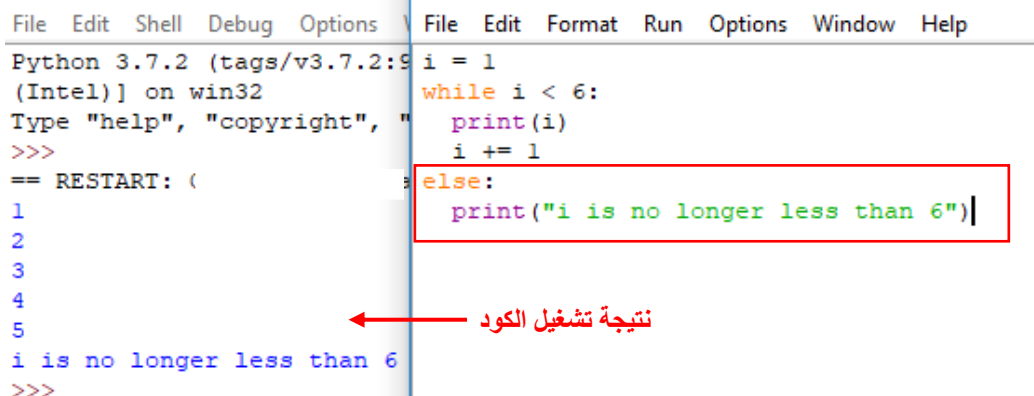
Print a message once the condition is false.

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")
```

في هذا المثال

تم كتابة الأمر **else** مباشرة بعد الحلقة التكرارية

وسيتم تنفيذه عند توقف الحلقة **while**



إذا أردت التأكد، قم بتغيير قيمة المتغير **i** بدل القيمة 1 بحيث أن القيمة التي تريد إدخالها لن يتم تنفيذها بشرط الحلقة **while** وستلاحظ أن أمر الطباعة في الجملة الشرطية **else** سيتم تنفيذه

ضع القيمة 10 مثلاً .. و جَرِّب!

أتممتَ درسك بنجاح!

تابع التقدم

روابط قد تساعدك

Check the links below

- [العبارات التكرارية While Loop Python|| 22-](#)
- [12 - Python - Beginners Tutorial - While loop statement](#)
- [Learn Python in Arabic #28 - Python in Arabic while loop else Python](#)
- [Learn Python in Arabic #29 - while loop list tuple dictio](#)
- [Python Programming #10 - While Loops and Control Statements](#)
- [Learn Python Programming - 21 - While Loops](#)
- [#20 Python Tutorial for Beginners | While Loop in Python](#)
- [Python tutorial Loop while العباره التكراريه](#)
- [شرح - 10 Break, Continue - تعلم البرمجة بلغة بايثون](#)
- [Python tutorial Nested While Loops](#)

طبق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم التاسع والعشرون

الحلقات التكرارية في لغة البايثون

Python Loops 2

For Loop

➤ The For Loops

A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). This is less like the **for** keyword in other programming languages and works more like an iterator method as found in other object-orientated programming languages.

With the **for** loop we can execute a set of statements, once for each item in a list, tuple, set etc.

الحلقة التكرارية **for** تستعمل للمرور على جميع عناصر السلسلة لـ القوائم أو الصفوف أو القواميس .. الخ وتستخدم لتنفيذ الكود عدة مرات محددة، فهي لا تحتاج إلى شرط معين لتحديد عدد مرات التكرار فهو معلوم قبل الدخول في الحلقة

Example

Print each fruit in a fruit list

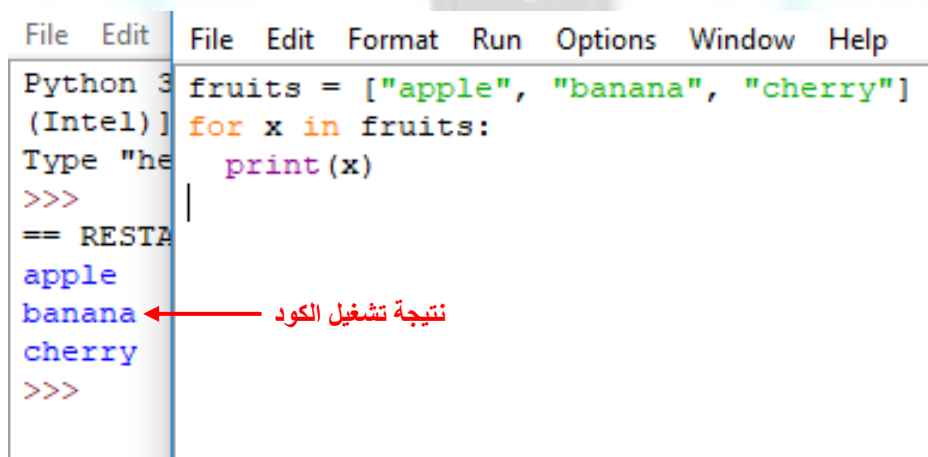
```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

١/ **element** متغير يتم تعريفه داخل الحلقة

٢/ **sequence** السلسلة التي نريد الوصول لعناصرها

٣/ **statements** جميع الأوامر التي ستنفذ في كل دورة

هنا حلقة **for** تقوم بالمرور على جميع عناصر القائمة بالترتيب من العنصر الأول إلى العنصر الأخير وفي كل دورة يتم تخزين قيمة العنصر في المتغير الذي قمنا بتعريفه وهو **x** ثم يتم عرض قيمته



```
File Edit Format Run Options Window Help
Python 3 fruits = ["apple", "banana", "cherry"]
(Intel) for x in fruits:
Type "he     print(x)
>>>
== RESTA
apple
banana
cherry
>>>
```

نتيجة تشغيل الكود

في حلقة **for** يتم تعريف المتغير للحد في بداية الحلقة وليس قبل البدء بكتابة الحلقة التكرارية كما في **while**

The **for** loop does not require an indexing variable to set beforehand.

➤ Looping Through a String

عرض أحرف النص

النص هو عبارة عن تكرار لسلسلة من الأحرف

Even strings are iterable objects, they contain a sequence of characters.

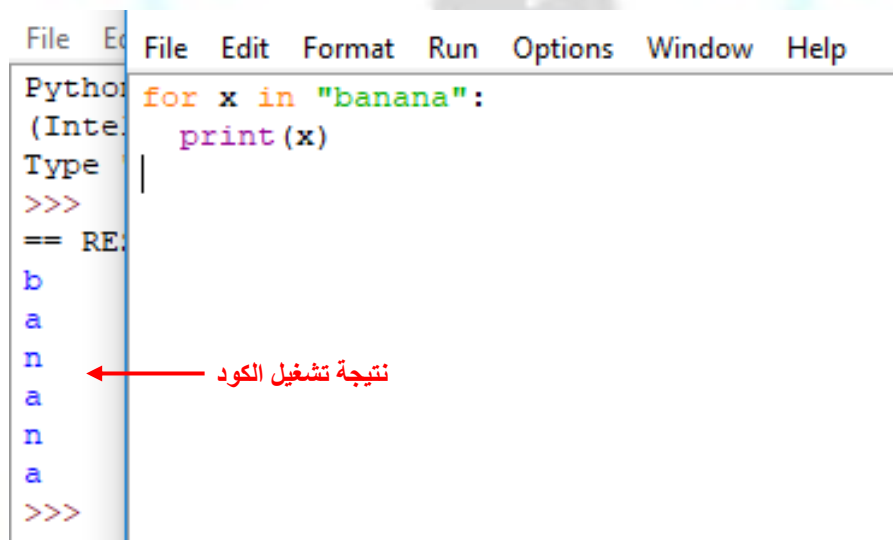
Example

Loop through the letters in the word "banana"

في هذا المثال

```
for x in "banana":  
    print(x)
```

ستقوم الحلقة for بالمرور على أحرف النص "banana" وطباعتهم حرفاً حرفاً



```
File Edit Format Run Options Window Help  
Python (Interpreter)  
Type |  
>>>  
== RE...  
b  
a  
n  
a  
n  
a  
>>>
```

نتيجة تشغيل الكود

➤ The **break** Statement

جملة التحكم **توقف**

التعبير أو الجملة **break** تُستخدم للخروج من حلقة التكرار قبل إكمال تنفيذها على كل العناصر، ستتوقف الحلقة عند شرط معين

With the **break** statement we can stop the loop before it has looped through all the items.

Example

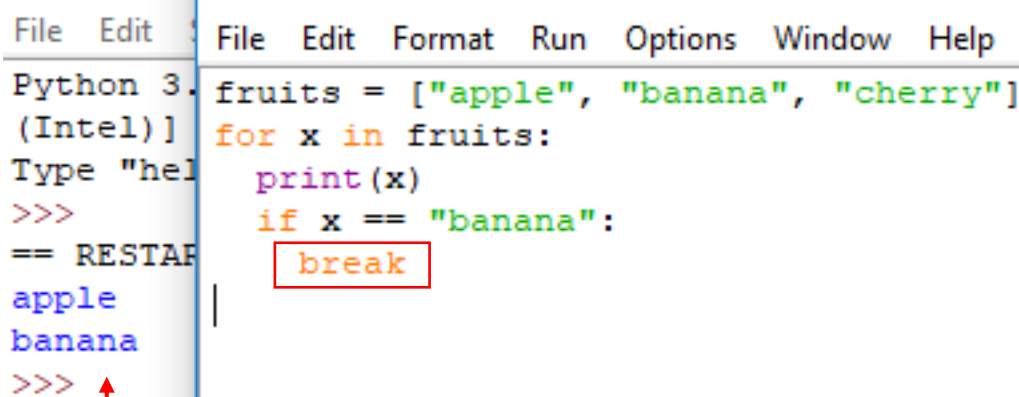
في هذا المثال

Exit the loop when x is "banana"

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
    if x == "banana":
        break
```

الحلقة **for** كانت ستطبع جميع العناصر الموجودة في القائمة **fruits**

لكننا استخدمنا الجملة **break** لإيقاف الحلقة عند تحقق هذا الشرط (**x == "banana"**)



```
File Edit Format Run Options Window Help
Python 3. (Intel)
Type "help" for more
>>>
== RESTART ==
apple
banana
>>>
```

نتيجة تشغيل الكود

هذا المثال شبيهه بالمثال السابق

لكن الاختلاف هو في موقع وضع الجملة **break**

هنا كتبناها قبل جملة الطباعة `print()`

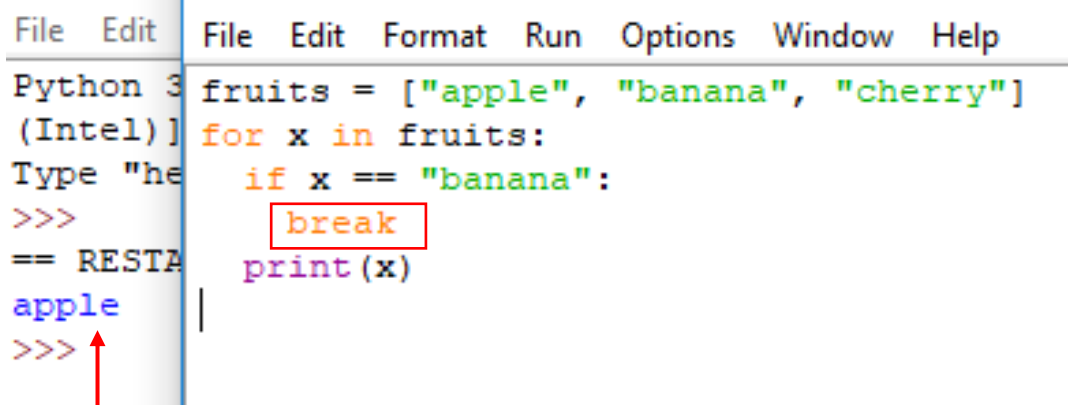
Example

Exit the loop when `x` is "banana", but this time the **break** comes before the print.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

الحلقة `for` كانت ستطبع جميع العناصر الموجودة في القائمة `fruits`

لكن بإستخدامنا الجملة **break** توقفت الحلقة عند تحقق الشرط (`x == "banana"`)



```
File Edit Format Run Options Window Help
Python 3 fruits = ["apple", "banana", "cherry"]
(Intel)] for x in fruits:
Type "he if x == "banana":
>>> break
== RESTA print(x)
apple
>>>
```

نتيجة تشغيل الكود

➤ The **continue** Statement

جملة التحكم **استمر**

الجملة **continue** تقوم بإيقاف الدورة الحالية في الحلقة التكرارية عند شرط معين ولا تنفذها، والانتقال إلى الدورة التالية

With the **continue** statement we can stop the current iteration of the loop and continue with the next.

Example

Do not print banana.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        continue
    print(x)
```

في المثال التالي

قمنا بتعريف حلقة تكرارية **for** ستطبع جميع العناصر في القائمة

لكن باستخدام الأمر **continue** جعلنا الحلقة **for** تنتقل للدورة التالية عندما يتحقق هذا الشرط ($x == \text{"banana"}$) وبالتالي لن تتم طباعة هذا العنصر

| File | Edit | Format | Run | Options | Window | Help |
|-------------------|--|--------|-----|---------|--------|------|
| Python 3 (Intel)] | fruits = ["apple", "banana", "cherry"] | | | | | |
| Type "he | for x in fruits: | | | | | |
| >>> | if x == "banana": | | | | | |
| == RESTA | continue | | | | | |
| apple | print(x) | | | | | |
| cherry | | | | | | |
| >>> | | | | | | |

نتيجة تشغيل الكود

مبرمج الغد!

أتممت درسك

روابط قد تهلك

Useful links

- [Learn Python in Arabic #27 - القاموس و الصفوف و القوائم و tuple dictio](#)
- [Learn Python in Arabic #33 - break infinity loop Python الخروج من التكرار](#)
- [Learn Python in Arabic #34 - continue in loop Python الاستمرار في التكرار](#)
- [Python tutorial Loop for use Break and Continue](#)
- [25- Python|| Loop Control التحكم بالعبارات التكرارية](#)
- [#22 Python Tutorial for Beginners | Break Continue Pass in Python](#)

طبق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم الثلاثون

الحلقات التكرارية في لغة البايثون

Python Loops 3

For Loop

➤ The **range()** Function

دالة المجال/المدى

To loop through a set of code a specified number of times, we can use the **range()** function, The **range()** function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

هذه الدالة تقوم بترجييع سلسلة أرقام صحيحة (مدى محدد) لرقم معين، من الأصغر للأكبر حسب رقم البداية والنهاية تبدأ السلسلة من الصفر (افتراضيا) إذا لم يحدد لها رقم البدء وتزيد بواحد في الدورة التالية (افتراضيا) إذا لم يحدد لها قيمة الزيادة ثم تنتهي بقيمة الرقم الصحيح الذي يمثل نهاية السلسلة

عند استدعاء هذه الدالة يجب تمرير قيمة لها، أو قيمتين، أو ثلاث ... ستري ذلك من خلال الأمثلة

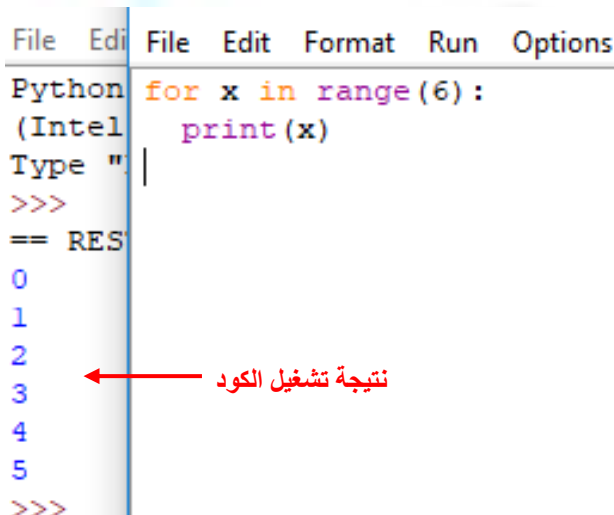
Example

Using the **range()** function

في هذا المثال

```
for x in range(6):
    print(x)
```

قمنا باستخدام الدالة **range()** مع الحلقة **for** لطباعة سلسلة الأرقام من 0 إلى 5 في كل دورة في الحلقة سيتم تخزين رقم من هذه السلسلة في المتغير **x** ثم تتم طباعته



```
File Edit File Edit Format Run Options
Python for x in range(6):
(Intel print(x)
Type "
>>>
== RES
0
1
2
3
4
5
>>>
```

لاحظ إذا تم تمرير قيمة واحدة كما في المثال **range(6)**

فإن الدالة ستبدأ بترجييع سلسلة الأرقام من الرقم 0 وتنتهي عند الرقم 5

Note that **range(6)** is not the values of 0 to 6, but the values 0 to 5

The **range()** function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: **range(2, 6)**, which means values from 2 to 6

(but not including 6)

يمكنك تحديد رقم البداية، بدلا من أن يكون افتراضيا ويبدأ بالـ 0

عند تمرير قيمتين للدالة **range(2, 6)** كما في المثال التالي

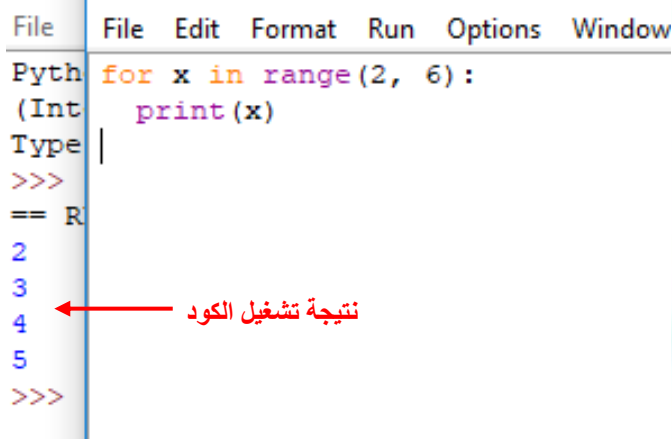
فإن الدالة ستبدأ بترجيع سلسلة الأرقام بدايةً من الرقم الذي قمت بتحديد قيمته وهو هنا 2 وتنتهي عند الرقم 5

Example

Using the start parameter

قمنا بتحديد رقم البداية

```
for x in range(2, 6):  
    print(x)
```



```
File  File  Edit  Format  Run  Options  Window  
Pyth for x in range(2, 6):  
(Int  print(x)  
Type |  
>>>  
== R  
2  
3  
4 ← نتيجة تشغيل الكود  
5  
>>>
```

The **range()** function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: **range(2, 30, 3)**

يمكنك تحديد مقدار الزيادة في كل دورة، بدلاً من أن يكون افتراضياً بـ 1

ويكون بتمرير ثلاث قيم للدالة **range(2, 30, 3)** كما في المثال التالي

فإن الدالة ستبدأ بترجيع سلسلة الأرقام بدايةً من الرقم الذي قمت بتحديد قيمته وهو هنا 2 وتنتهي عند الرقم 29 ومقدار الزيادة يكون 3 في كل مرة

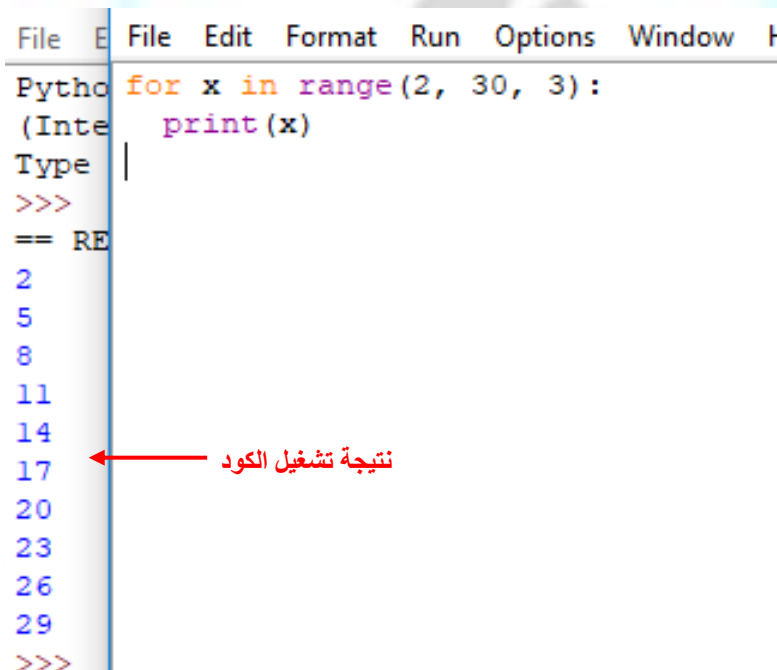
Example

Increment the sequence with 3 (default is 1)

```
for x in range(2, 30, 3):
    print(x)
```

قمنا بتحديد رقم البداية

ومقدار الزيادة في كل مرة (عدد الأرقام التي يجب زيادتها في الدورة التالية)



```
File Edit Format Run Options Window
Python for x in range(2, 30, 3):
(Inte print(x)
Type |
>>>
== RE
2
5
8
11
14
17
20
23
26
29
>>>
```

نتيجة تشغيل الكود

➤ Else in For Loop

استخدام الأمر **else** مع الحلقة التكرارية **for**

The **else** keyword in a **for** loop specifies a block of code to be executed when the loop is finished

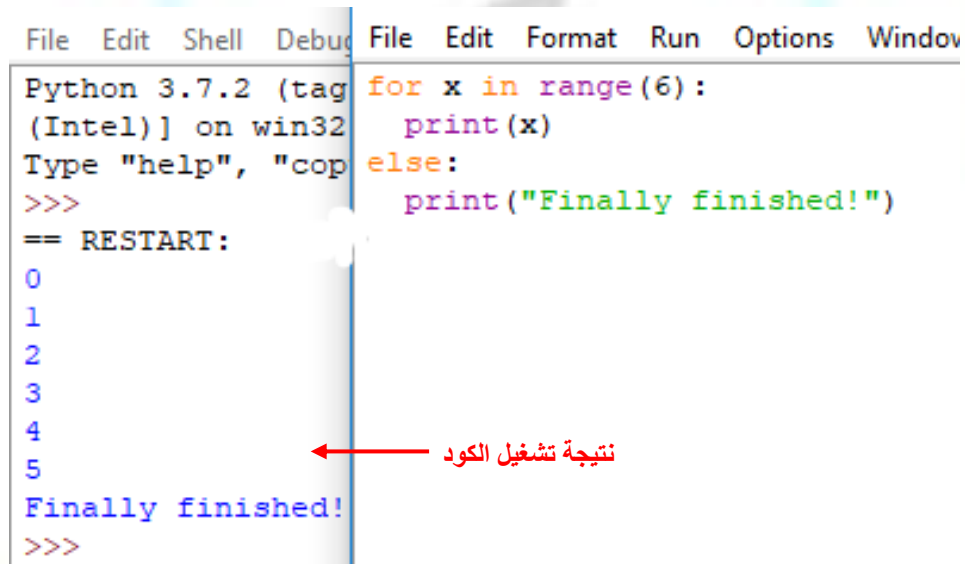
الجملة الشرطية **else** ستقوم بتنفيذ (مجموعة أوامر) بعدما تنتهي الحلقة التكرارية **for** من العمل
سيتضح لك المفهوم مع المثال ..

Example

Print all numbers from 0 to 5, and print a message when the loop has ended

```
for x in range(6):  
    print(x)  
else:  
    print("Finally finished!")
```

تم كتابة الأمر **else** مباشرة بعد الحلقة التكرارية
وسيتنفيذ دائماً بعد انتهاء الحلقة **for**



```
File Edit Shell Debug File Edit Format Run Options Window  
Python 3.7.2 (tag for x in range(6):  
(Intel)] on win32 print(x)  
Type "help", "cop else:  
>>> print("Finally finished!")  
== RESTART:  
0  
1  
2  
3  
4  
5  
Finally finished!  
>>>
```

➤ Nested Loops

الحلقات التكرارية المتداخلة/المتشعبة

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop"

عبارة عن حلقة تكرار موجودة داخل حلقة تكرار أخرى، وهي شبيهة بـ الجمل الشرطية **if** المتداخلة
حلقة التكرار الداخلية سيتم تنفيذها مرة واحدة وتنفذ بالكامل لكل دوران من الحلقة التكرارية الخارجية

Example

Print each adjective for every fruit.

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj: ← Outer loop
    for y in fruits: ← Inner loop
        print(x, y)
```

يبدأ البرنامج بتنفيذ حلقة التكرار الخارجية وينفذ أول دورة فيها

وأول دورة ستؤدي إلى دخول حلقة التكرار الداخلية التي سيتم تنفيذها كاملة إلى أن تنتهي

ثم يعود البرنامج إلى حلقة التكرار الخارجية مرة أخرى ليبدأ بتنفيذ الدورة الثانية .. وهكذا

إلى أن ينتهي تنفيذ حلقة التكرار الخارجية

| File | Edit | Shell | File | Edit | Format | Run | Options | Window | Help |
|--|------|-------|---|------|--------|-----|---------|--------|------|
| Python 3.7.2 (Intel) on Windows Type "help", or >>> | | | adj = ["red", "big", "tasty"] fruits = ["apple", "banana", "cherry"] for x in adj: for y in fruits: print(x, y) | | | | | | |
| == RESTART: C:\Python37\Python.exe red apple red banana red cherry big apple big banana big cherry tasty apple tasty banana tasty cherry >>> | | | نتيجة تشغيل الكود | | | | | | |

أتممت درسك بنجاح!
واصل التعلّم

روابط قد تهمك

Useful links

- [Loops: For loop, while loop](#)
- [Python tutorial Loop for](#) العبارة التكرارية
- [Python tutorial Loop for](#) طباعة مجموعة قيم
- [Learn Python in Arabic #25 - Python in Arabic](#) شرح بايثون بالعربي 25 -
- [Learn Python in Arabic #26 - Python in Arabic](#) for loop التكرار مع جملة فور بايثون ب - statement Python
- [11 - Python - Beginners Tutorial - For loop statement](#)
- [Python Programming #9 - For Loops](#)
- [23- Python|| iterator For Loop](#) العبارات التكرارية
- [08 - بايثون بالعربي - الجمل الشرطية والخوارزميات](#)
- [Loops in Python - Coderbyte](#)
- [#21 Python Tutorial for Beginners | For Loop in Python](#)
- [#24 Python Tutorial for Beginners | For Else in Python](#)
- [Python tutorial Nested For Loops](#)
- [24- Python|| Nested Loop](#) العبارات التكرارية متداخلة

طبق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم الواحد والثلاثون

الدوال في لغة البايثون

Python Functions

الدالة عبارة عن مجموعة من الأوامر (أكواد) في مكان واحد **block** يمكن مناداتها واستدعائها عند الحاجة إليها
ويمكنك تمرير قيم لها عبر الـ **parameters** ، ويمكن أن نجعل الدالة تُعيد نتيجة

A **function** is a block of code which only runs when it is called. You can pass data, known as parameters, into a **function**. A **function** can return data as a result.

الدوال في بايثون نوعين :

١ / هناك دوال جاهزة في بايثون (دوال معرفة مسبقا) وتسمى **Built-In Functions**
بايثون توفر مجموعة كبيرة منها وهي ما استخدمت كثير منها في الدروس السابقة مثل : `print()` – `len()` – `range()`

٢ / دوال يقوم المبرمج بتعريفها وإنشائها تسمى **User-Defined Functions**
وهي ما سنبدأ بمناقشتها وشرحها في درسنا هذا وإكمال سلسلة دروس الدوال في الأسبوع القادم .. بإذن الله

➤ Creating a Function

تعريف / إنشاء دالة

لتعريف دالة جديدة في بايثون نستخدم الكلمة المحجوزة **def**

In **Python** a function is defined using the **def** keyword

Example

كتابة القوسين لتمرير قيم الـ **parameters**

اسم الدالة، والذي بواسطته نستدعي الدالة

وضع النقطتين بعد القوسين مباشرة

```
def my_function():
    print("Hello from a function")
```

نستخدمها لتعريف الدالة

الأوامر التي تكتبها في الدالة، وسيتم تنفيذها عند مناداة الدالة

➤ Calling a Function

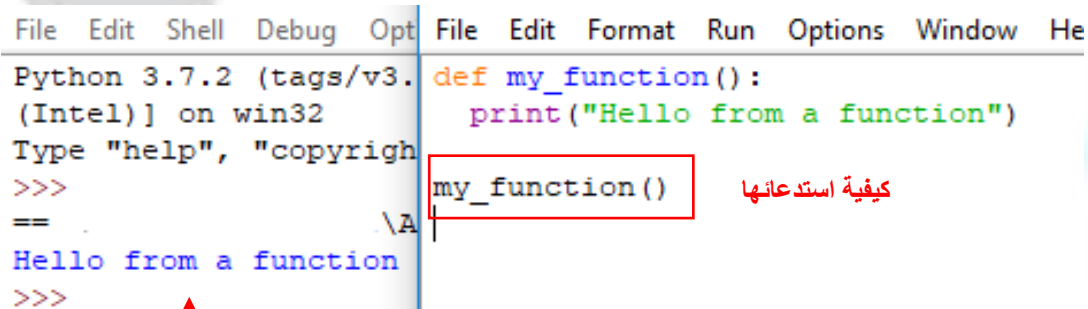
استدعاء الدالة

الاستدعاء يكون بكتابة اسم الدالة مع القوسين

To call a function, use the function name followed by parenthesis

Example

```
def my_function():  
    print("Hello from a function")  
  
my_function() ←
```



```
File Edit Shell Debug Opt File Edit Format Run Options Window He  
Python 3.7.2 (tags/v3.7.2.1 on win32  
Type "help", "copyright", and  
>>>  
==  
Hello from a function  
>>>
```

كيفية استدعائها

نتيجة تشغيل الكود

Parameters

المعاملات

Information can be passed to functions as parameter.

Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

يمكن تمرير قيم عن طريق المعاملات في الدالة (وضع المعاملات هو أمر اختياري)
يتم تحديد وكتابة المعاملات عند تعريف الدالة بعد اسم الدالة داخل القوسين () ويتم الفصل بين المعاملات بواسطة الفاصلة

The following example has a function with one parameter (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name

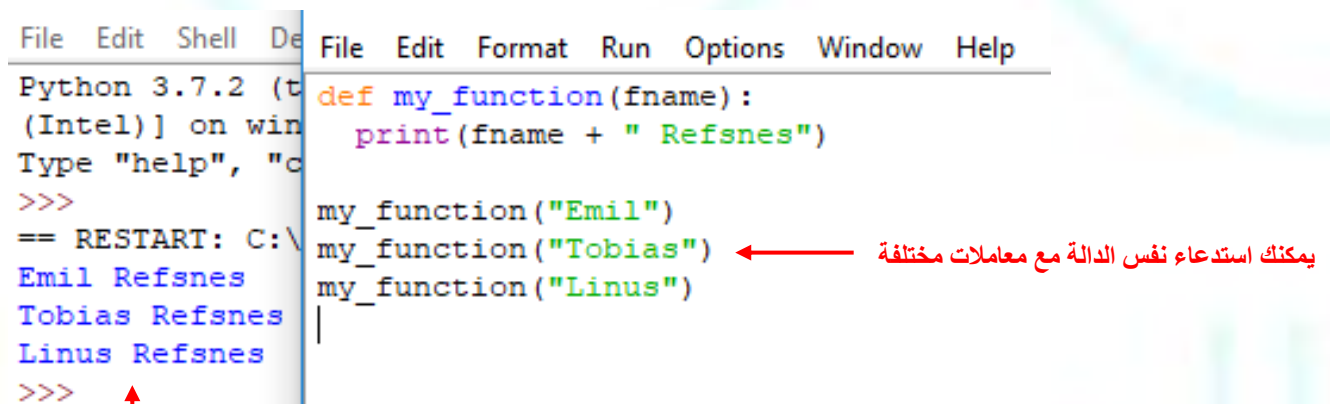
Example

```
def my_function(fname):  
    print(fname + " Refsnes")
```

في المثال التالي

```
my_function("Emil")  
my_function("Tobias")  
my_function("Linus")
```

لدينا دالة مع معامل واحد وهو fname ووضعنا داخل الدالة جملة طباعة
لتطبع قيمة العامل fname مع جملة نصية
عند استدعاء هذه الدالة يتم تمرير اسم لها
ثم تنفذ جملة الطباعة التي بداخل الدالة



```
File Edit Shell De File Edit Format Run Options Window Help  
Python 3.7.2 (t def my_function(fname):  
(Intel)] on win print(fname + " Refsnes")  
Type "help", "c  
>>>  
== RESTART: C:\ my_function("Emil")  
Emil Refsnes my_function("Tobias")  
Tobias Refsnes my_function("Linus")  
Linus Refsnes  
>>>
```

نتيجة تشغيل الكود

قيم المُعاملات الافتراضية

➤ Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without parameter, it uses the default value.

يمكنك القيام بوضع قيمة افتراضية للمُعاملات، فعندما يتم استدعاء الدالة بدون تمرير للمُعاملات ستستعمل الدالة قيم المُعاملات الافتراضية، انظر المثال التالي :

Example

```
def my_function(country = "Norway"):  
    print("I am from " + country)
```

هذه الدالة فيها مُعامل واحد هو country

يحمل القيمة Norway كقيمة افتراضية

في حال لم يتم تمرير قيمة للمُعامل اثناء استدعاء الدالة

```
my_function("Sweden")  
my_function("India")  
my_function()  
my_function("Brazil")
```

عند استدعاء هذه الدالة سيتم طباعة جملة "I am from" مع قيمة المُعامل

```
File Edit Shell Debu File Edit Format Run Options Window Help
Python 3.7.2 (tag def my_function(country = "Norway"):  
(Intel)] on win32 print("I am from " + country)  
Type "help", "cop  
>>>  
== RESTART: C:\Us my_function("Sweden")  
my_function("India")  
I am from Sweden my_function() ←  
I am from India my_function("Brazil")  
I am from Norway  
I am from Brazil  
>>>
```

لم يتم تمرير قيمة للمُعامل

نتيجة تشغيل الكود

رائع!
أتممت درسك الأخير لهذا الأسبوع

طبّق ما تعلمته في هذا الدرس
ولا تنسى مشاركتنا أكوادك

اليوم الثاني والثلاثون & اليوم الثالث والثلاثون

تحدي الأسبوع (يتم حله ورفعته على Github)

قم بطباعة كل عدد من القائمة أ (A) المتكونة من الأعداد الفردية بدءاً من العدد 3 وانتهاءً بالعدد 17

مع كل عدد من القائمة ب (B) المتكونة من الأعداد الزوجية بدءاً من العدد 2 وانتهاءً بالعدد 16

وذلك باستخدام حلقات التكرار loops و دالة تحديد المجال range() (مجال الأعداد)

بحيث العدد الواحد من القائمة أ (A) يقابل كل عدد من القائمة ب (B)

تذكير :

- دالة الـ range() لا تطبع العدد الأخير، لذلك لن تظهر لك الأعداد الأخيرة في كلا القائمتين .. راجع الدرس الثلاثون
- ربما تحتاج لإستخدام الحلقات التكرارية المتداخلة Nested Loop

انتظرونا في دروس الأسبوع القادم