الدوال البرمجية
(Functions)

أنواع الحلقات في برمجية بايثون (Python)

هل خطر ببالك يوماً أن تسجل روتينك اليومي ؟ مثل الإستيقاظ،تناول طعام الإفطار، الذهاب الى المدرسة، العودة الى المنزل، تناول طعام الغداء ، حل الواجبات،تناول طعام العشاء، الذهاب الى النوم وغيرها من الأعمال على مدار اليوم؟

لو كتبتها على مدار أسبوع، سوف تجد أن هناك مجموعة من الأعمال تتكرر معك يومياً وبانتظام، أي أنك ممكن أن تجمعها داخل قوسين وتضع جنبها كلمة : تتكرر طوال الأسبوع.

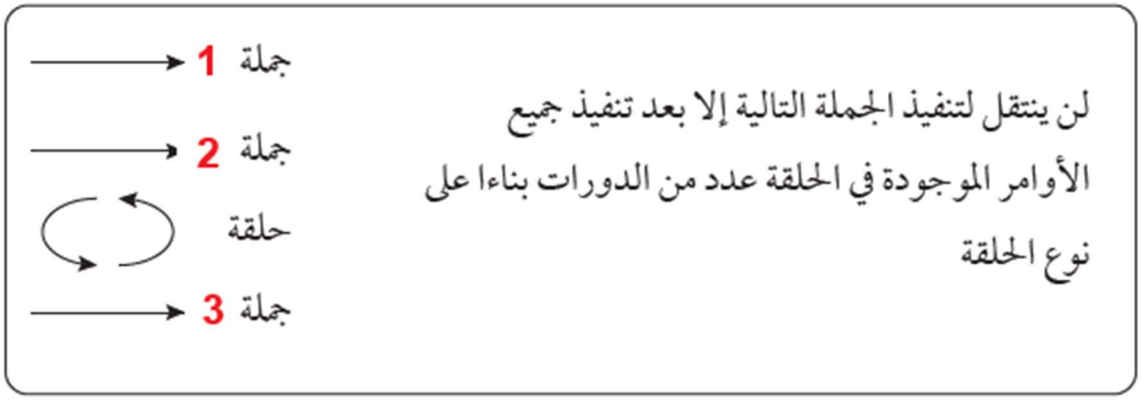
إن مثل هذه الأعمال التي تتكرر تشبه الى حد كبير مجموعة الأوامر والتعليمات التي تريد تكراراها خلال برنامج بايثون، إذ بدل أن تكرر كتابتها بعدد مرات تنفيذها،نستعيض عن ذلك بمفهوم **الحلقة(Loop).**

الحلقة: هي عملية تكرار تنفيذ مجموعة من الأوامر في البرنامج ، حيث يتم إ ستخدام الحلقات) Loops (لجعل الكود يعاد تنفيذه ضمن شروط معينة.

فعند البدء بتشغيل البرنامج،وعندما يمر المفسر على الحلقة التكرارية فإنه يتحقق من الشرط،

إذا كان الشرط (true) ينفذ الأوامر الموجودة بداخل الحلقة التكرارية ثم يعود مرة أخرى لبداية الحلقة التكرارية ليتحقق مرة أخرى من الشرط، فإذا كان الشرط ينتج عنه (true) سيقوم المفسر بتنفيذ الأوامر الموجودة بداخل الحلقة التكرارية مرة أخرى وتتكرر هذه العملية إلى أن يكون ناتج الشرط (false) فيتوقف المفسر عن التكرار ويمر على باقي الكود.

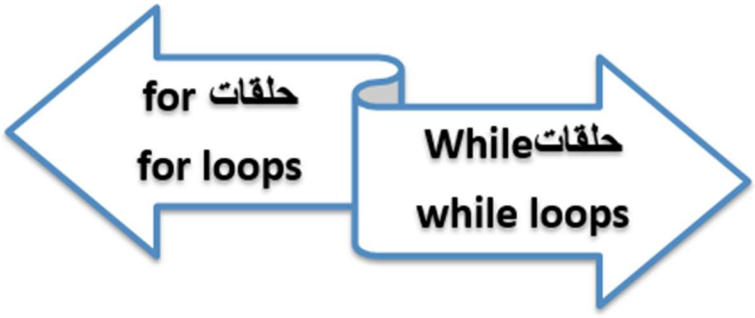
ويمثل الرسم التالي مبدأ عمل الحلقات في برمجية بايثون (Python):



حيث انّ البرنامج يقوم بتنفيذ الجملة **رقم 1** ثم الجملة **رقم 2** وعندما يصل الى الحلقة يقوم البرنامج بتنفيذ الأوامر الموجودة داخل الحلقة التكرارية عدد من المرات طالما كانت نتيجة الشرط صحيحة، وفي حال كانت نتيجة الشرط خطأ، فإن البرنامج يقوم بالانتقال الى الجملة

رقم 3.

تُصنَّف الحلقات في برمجية بايثون (Python) إلى نوعين، هما:



سوف نتعرف الآن على أنواع الحلقات وجمل التحكم بشكل عام، وسوف نتناولها بالتفصيل لاحقاً.

أولاً:حلقات while (while loops) :

تُستعمَل حلقة **(while)** لتكرار تنفيذ جملة واحدة أو أكثر طالما تحقَّق شرط مُعيَّن .وفي حال لم يعد هذا الشرط مُتحقِّقًا، فإنَّ البرنامج ي توقَّف عن تنفيذ هذه الجملة أو الجمل.

ثانيا: حلقات for (for loops):

يستخدم هذا النوع من الحلقات لغايات تكرار مجموعة من الجمل البرمجية عدداً محددًا من المرات.

جمل التحكم (Control Statements)

وحتى يتم ضبط هذه الحلقات، فإن برمجية بايثون تحتوي على عدد من جمل التحكم (**Control Statements**) التي تضبط آلية تنفيذ الحلقة. وسوف نتعرف هنا على نوعين من جمل التجكم هما:

1.جملة التحكم (**break**) :والتي يتم استخدامها لغايات إيقاف الحلقة اذا تحقق شرط معين ومن ثم يتم تنفيذ الجملة التي تلي الحلقة في البرنامج

2.جملة التحكم (**continue**): تستخدم لإيقاف الدورة الحالية في الحلقة و الإنتقال إلى الدورة التالية فيها غذا تحقق شرط معين، و يفترض أن تكون موضوعة بداخل جملة شرطية.

وفيما يلي شرح توضيحي حول الحلقات المستخدمة في يرمجية بايثون وآلية استخدامها مع الجمل الشرطية:

حلقات(while loops) while

1.الكلمة المحجوزة: while

2.الصيغة العامة :

```
while condition:
statements
increment or decrement
```

الكلمة المحجوزة (while) حيث تمثل كلمة

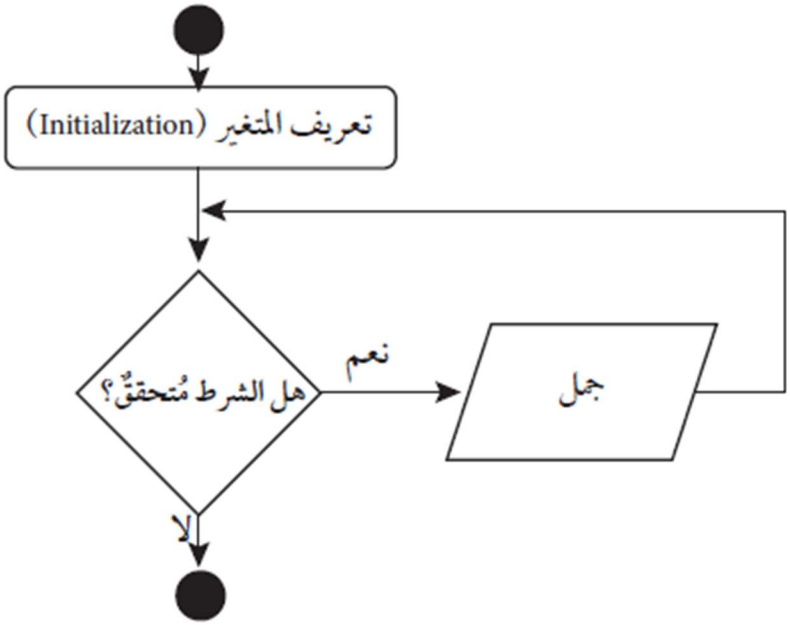
وكلمة (condition) :الشرط الذي يحدد استمرارية تنفيذ الجمل الموجودة في الحلقة،وفي كل مرة يمر البرنامج على الحلقة يتم التحقق من الشرط،إذ يتم توقف تنفيذ الجمل حين يصبح الشرط غير متحقق.

وكلمة (statement) :الجمل الموجودة داخل الحلقة والمراد من البرنماج تكرار تنفيذها ما دام الشرط متحققاً.

وكلمة (increment):الآلية التي من خلالها يتم زيادة قيمة العداد

) : الآلية التي من خلالها يتم تنقيص قيمة العدادdecrementوكلمة (

3.مخطط سير العمليات للحلقة:



مثال

لنكتب الآن برنامجًا يقوم بطباعة قيمة العداد (count) من (1) الى (5) باستخدام الحلقة (while)

```
Count = 1
While count < 6 :
        Print("count is ", count)
        Count+= 1
else:
      print("Loop has ended")
```

والآن، لنتتبع الكود الذي في الأعلى لنرى كيف يتم تنفيذ البرنامج:

1. في البداية،يعمل البرنامج على تعريف المتغير الذي يحمل الإسم count ويقوم بإعطائه قيمة أولية هي 1
2. الآن يقوم المفسر بتنفيذ الحلقة، حيث يعمل على التحقق من الشرط: هل قيمة المتغير (count) أقل من (6) ؟

   اذا كانت الإجابة نعم، فإن البرنماج يقوم بتنفيذ الجملة التالية في داخل الحلقة.
3. الجملة التي في داخل الحلقة ، تطلب طباعة قيمة المتغير () ، أي ان البرنامج يقوم بطباعة الرقم 1
4. يذهب البرنامج الى تنفيذ الجملة التالية وهي العمل على زيادة قيمة المتغير () بمقدار (1) ثم يعود الى تنفيذ الجلقة مرة أخرى

5.في كل مرة يعود البرنامج لتنفيذ الحلقة،يعمل على التحقق من الشرط مرة أخرى، وطالما كانت نتيجة تطبيق الشرط صحيحة،يعمل البرنامج على تكرار الخطوة رقم (3) والخطوة رقم (4).

في حال لم يعد الشرط صحيحاً، فإن البرنامج يغادر الحلقة، ويذهب الى تنفيذ الجملة التي خارج الحلقة ويطبع عبارة " Loop has ended"

نشاط عملي

- أُنفِّذ المثال السابق في بيئة بايثون (Python)، وأُلاحِظ الناتج.
- أُعدِّل المقطع البرمجي بتغيير جملة (count += 1) إلى جملة (count += 2)، ثمَّ أُنفِّذ البرنامج. ما الناتج المُترتِّب على تنفيذ البرنامج؟
- أُعدِّل المقطع البرمجي بتغيير الشرط (count = 6) إلى الشرط (count < 6)، ثمَّ أُنفِّذ البرنامج. ما الناتج المُترتِّب على تنفيذ البرنامج؟

أُقارِن إجاباتي بإجابات زملائي/ زميلاتي في الصف.

- أُنفِّذ المثال السابق في بيئة بايثون Python وأُلاحِظ الناتج

```
count = 1
while count > 6:
    print("Count is", count)
    count += 1
else:
    print("Loop has ended")
```

```
Loop has ended
```

تيجة على شاشة جهاز الحاسوب

- أُعدِّل المقطع البرمجي بتغيير جملة count += 1 إلى جملة count += 2 ثمَّ أُنفِّذ البرنامج. ما الناتج المُترتِّب على تنفيذ البرنامج؟

```
count = 1
while count > 6:
    print("Count is", count)
    count += 2
else:
    print("Loop has ended")
```

```
Loop has ended
```

النتيجة على شاشة جهاز الحاسوب

- أُعدِّل المقطع البرمجي بتغيير الشرط count = 6 إلى الشرط count < 6 ثمَّ أُنفِّذ البرنامج. ما الناتج المُترتِّب على تنفيذ البرنامج؟

```
count = 6
while count < 6:
    print("Count is", count)
    count += 1
else:
    print("Loop has ended")
```

```
Loop has ended
```

النتيجة على شاشة جهاز الحاسوب

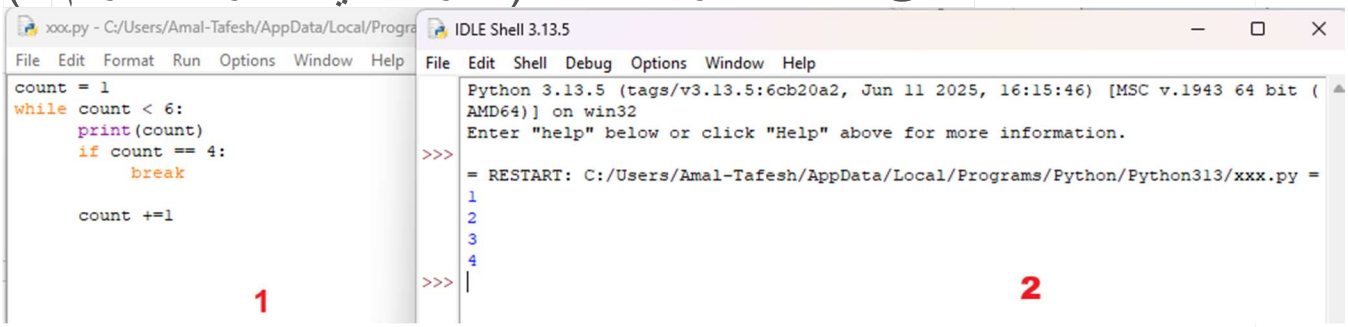
جملة التحكُّم (break) في حلقات (while)

والان لنجعل ذات البرنامج الذي كتبناه سابقا يقوم بطباعة قيمة العداد (count) باستخدام الحلقة (while) ولكن بشرط أن يتوقف البرنامج إذا اصبحت قيمة العداد 4.

في هذه الحالة، يجب استخدام جملة التحكم (break) والتي سوف تعمل على توقف البرنامج عند تحقق الشرط (وهنا قيمة count=4)،بحيث يصبح البرنامج على النحو التالي:

```
count = 1
while count < 6:
    print(count)
    if count == 4:
        break
    count += 1
```

عند تشغيل البرنامج (الجزء الذي يظهلر عليه رقم 1) ستظهر النتيجة الآتية على شاشة جهاز الحاسوب(الجزء الذي يظهر عليه رقم 2):



جملة التحكُّم continue في حلقات while

تُستعمَل continue لإيقاف الدورة الحالية تعرَّفْتُ سابقًا أنَّ جملة التحكُّم في الحلقة، والانتقال إلى الدورة التالية فيها إذا تحقَّق شرط مُعيَّن.

مثال:

على طباعة الأرقام continue تعمل جملة التحكُّم
من: 1 إلى 6 باستثناء الرقم 2 كما يأتي

```
count = 0
while count < 6:
    count += 1
    if count == 2:
        continue
    print(count)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

while مع حلقات else جملة

لتنفيذ مجموعة من الأوامر while مع حلقة else تُستعمَل جملة البرمجية إذا أصبحت قيمة الشرط خطأً False أيْ خارج الحلقة.

مثال:

يطبع البرنامج الآتي قيمة العدّاد إذا كانت القيمة أقل من 4 وخلافًا لذلك، فإنَّ البرنامج سيطبع عبارة "count is no longer less than 4":

```
count = 1
while count < 4:
print(count)
count += 1
else:
print("count is no longer less than 4")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

نشاط عملي

في المثال السابق، إذا حُذِفت جملة (else)، فهل سيختلف الناتج؟ ماذا أتوقَّع أنْ يكون الناتج؟
أُعدِّل المقطع البرمجي بحذف جملة (else)، ثمَّ أُنفِّذ البرنامج المُعدَّل باستخدام برمجية بايثون (Python)، ثمَّ أُقارِن الناتج الحالي بالناتج السابق، وأُلاحِظ الفرق بينهما إن وجد.

أُناقِش زملائي/ زميلاتي في السؤالين الآتيين:

- هل أثَّرت جملة (else) في الناتج؟

عند تعديل المقطع البرمجي،يصبح الكود كما يلي:

```
count = 1
while count < 4:
 print(count)
 count += 1
 print("count is no longer less than 4")
```

```
1
count is no longer less than 4
2
count is no longer less than 4
3
count is no longer less than 4
```

النتيجة على شاشة جهاز الحاسوب

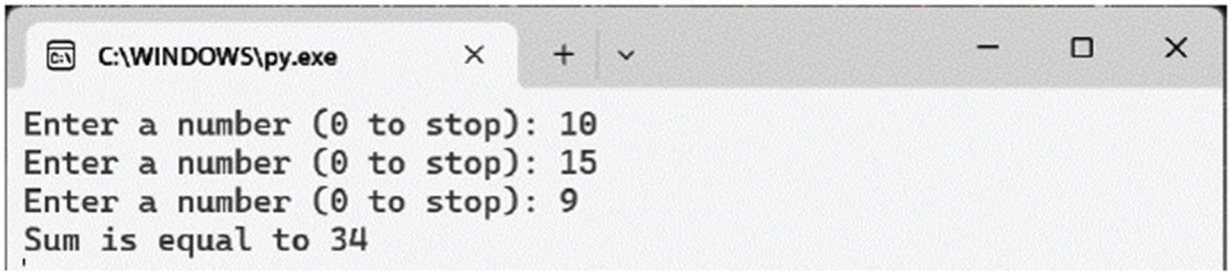
مثال:

البرنامج الآتي يعمل على **حساب مجموع الأعداد التي أدخلها المُستخدِم** حتى يصل المجموع إلى 30 أو أكثر، فإذا وصل المجموع 0 فأكثر، طُبِع المجموع 30 فأكثر، وخرج البرنامج من الحلقة. أمّا إذا أُدخِلت القيمة 0 قبل الوصول إلى المجموع 30

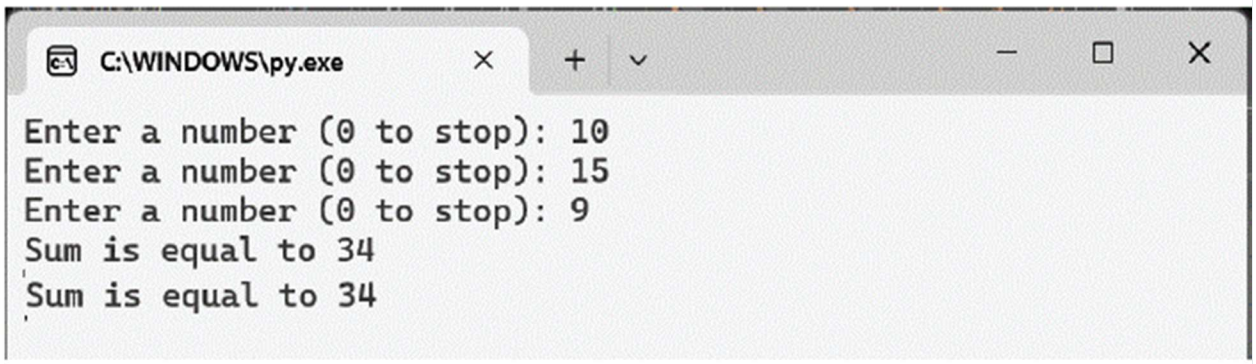
فإنَّ البرنامج يطبع رسالة مفادها أنَّ المجموع أقل من 30 ثمَّ يعرض قيمة المجموع النهائية.

```
s = 0
a = int(input("Enter a number (0 to stop): "))
while a != 0:
s += a
if s >= 30:
print("Sum is equal to", s)
break
a = int(input("Enter a number (0 to stop): "))
else:
print("Sum is equal to", s, ".")
```

عند تشغيل البرنامج، وإدخال العدد 10 ثمَّ العدد 15 ثمَّ العدد 9 ثمَّ العدد 0 ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

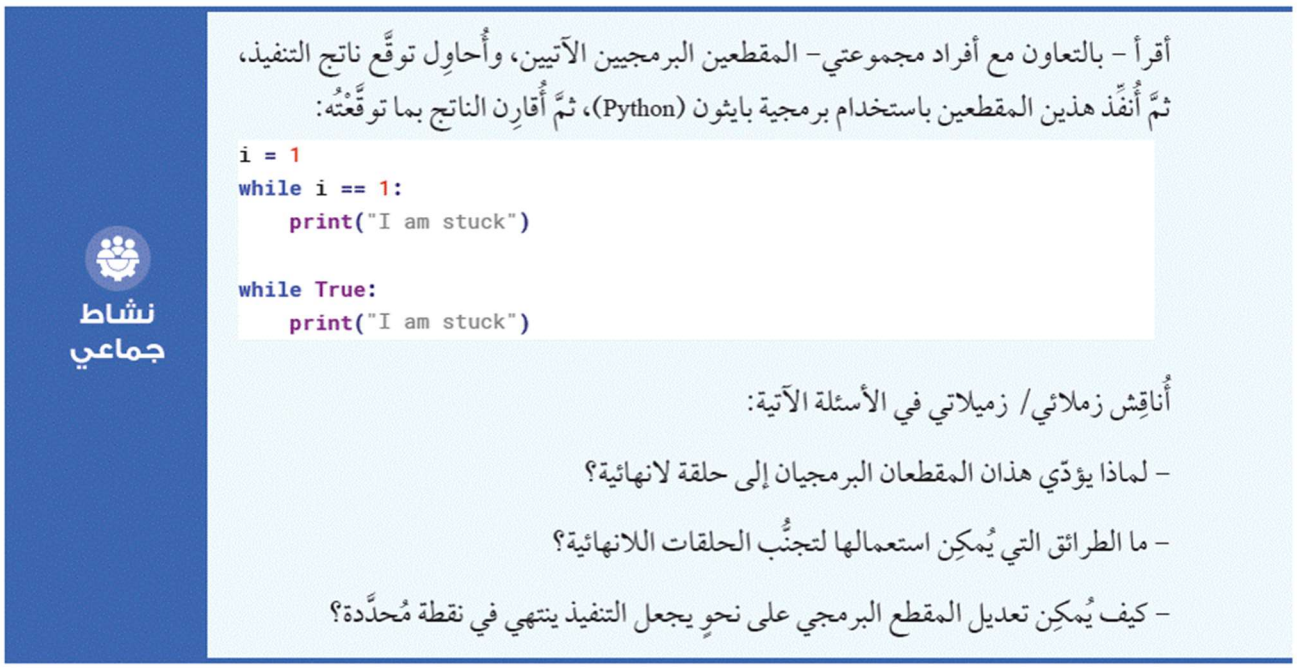


الواردة في المثال السابق، فإنَّ النتيجة الآتية ستظهر else إذا حُذِفت جملة على شاشة جهاز الحاسوب عند تشغيل البرنامج:



يتيح للمُستخدِم التحكَّم في طباعة الجملة else ألاحِظ أنَّ وجود جملة الثانية، بحيث لا تُطبَع إلّا

بعد خروج البرنامج من الحلقة دون أنْ يتحقَّق الشرط في جملة



لماذا يؤدّي هذان المقطعان البرمجيان إلى حلقة لانهائية؟

صحيح وسوف يعمل على تنفيذ ما **i = = 1** لأن الشرط

بدون توقف وذلك بسبب **عدم وجود تغيير في قيمة** داخل while

الخ والتي تساعد على ايقاف التكرار..... i+=2 مثل" **i العداد**.

ما الطرائق التي يُمكِن استعمالها لتجنُّب الحلقات اللانهائية؟

الجواب:

تغيير قيمة العداد بزيادته أو انقاصه حتى يعمل على استمرار عمل الحلقة أو ايقافها .

كيف يُمكِن تعديل المقطع البرمجي على نحوٍ يجعل التنفيذ ينتهي في نقطة مُحدَّدة؟

يمكن تعديل المقطع الأول على النحو التالي: :

```
i = 1
while i == 1:
  print("I am stuck")
  i += 1
```

```
I am stuck
```

النتيجة على شاشة جهاز الحاسوب

ويمكن تعديل المقطع الثاني على النحو التالي: :

```
while True:
    print("I am stuck")
    if True:
        break
```

```
I am stuck
```

النتيجة على شاشة جهاز الحاسوب

حلقاتfor loop) for)

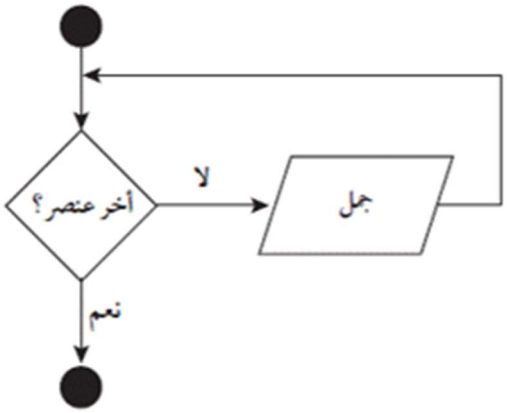
تُعرَّف حلقة **for** باستخدام الكلمتين المحجوزتين **for** و **in** على النحو الآتي:

```
for element in sequence:
  statements
```

حيث:

- **element** : مُتغيِّر يُعرَف داخل الحلقة، وتوضَع فيه إحدى قِيَم المتتابعة sequence التي تُجلَب في كل دورة، وتكون موضوعة بعد هذا المُتغيِّر.
- **sequence** : سلسلة يريد المُستخدِم الوصول إلى جميع عناصرها.
- **statements** : جمل موجودة في حلقة for التي سيُكرِّر البرنامج تنفيذها في كل دورة.

والشكل التالي يظهر مخطط سير العمليات لطريقة عمل حلقات (for)



for مع حلقات range الدالَّة

- تُستعمَل الدالَّة range 0 لإرجاع سلسلة من الأرقام، تبدأ بالرقم (ما لم يُحدَّد مقدار آخر للزيادة) وتنتهي برقم مُحدَّد (لم يُحدَّد رقم آخر)
- تزداد range 1 بمقدار

تُستخدَم الدالَّة range هي، بثلاث طرائق مختلفة:

range(a) : تُرجِع الدالَّة بهذه الطريقة سلسلة من الأرقام، بَدْءًا بالرقم 0 وانتهاءً بالرقم **a-1** حيث **a** تمثل رقمًا

مثال:

إذا كانت قيمة a هي 5 فإنَّ الدالَّة ستُرجِع سلسلة الأرقام الآتية: 0 ، 1 ، 2 ، 3، 4 .

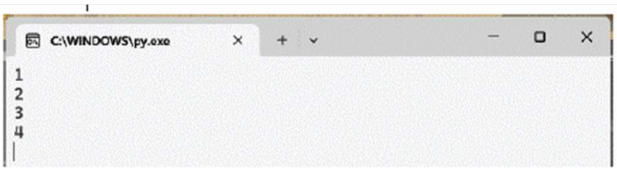
تُرجِع الدالَّة بهذه الطريقة سلسلة من الأرقام، بَدْءًا **range(a, b)** :
بالرقم **a** وانتهاءً بالرقم **b-1**

مثال:
إذا كانت قيمة a وقيمة 1 هي b هي سلسلة ستُرجِع الدالَّة فإنَّ 5 هي
الأرقام الآتية:
1 ، 2، 3 ، 4 .
تُرجِع الدالَّة بهذه الطريقة سلسلة من الأرقام على **range(a, b, c)** :
النحو التالي::
a تبدأ بالرقم
b-1تنتهي بالرقم
(c) تتزايد الأرقام بقفزة مقدارها .
مثال:
إذا كانت قيمة a وقيمة 1 هي b وقيمة 5 هي c 2 هي
a = 1 , b = 5 , c = 2
الأرقام 1 ، 3 ذلك أنَّ السلسلة تبدأ فإن الدالَّة ستُرجِع سلسلة
ثمَّ تزيد القيمة التالية بمقدار 2 وتزيد **a** بالعدد 1 والقيمة الابتدائية
القيمة الابتدائية بمقدار 2 فتصبح 3 ولا يتمُّ تضمين القيمة 5 في
أيْ عند الرقم 4 **b-1** هذه الحالة؛ لأنَّ السلسلة تتوقَّف عند الرقم

مثال:
يطبع البرنامج الآتي القِيَم من 1 إلى 4، حيث القيمة 1 هي
قيمة a هي قيمة 4 والقيمة b-1
وبتزايد c 1 مقداره :

```
for x in range(1, 5, 1):
print(x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط عملي

أُجرِّب وأستنتج:

هل سيختلف الناتج في البرنامج السابق إذا استُخدِم range (1, 5) بدلًا من range (1, 5, 1)؟

الحل :

1- لا ، لن يختلف لأنه مقدار التزايد هو 1

```
for x in range(1, 5, 1):
 print(x)
```

```
1
2
3
4
```

النتيجة على شاشة جهاز الحاسوب

2-

```
for x in range(1, 5):
 print(x)
```

```
1
2
3
4
```

النتيجة على شاشة جهاز الحاسوب

إضاءة

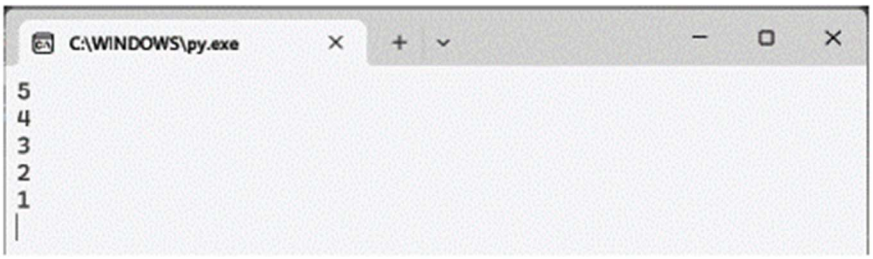
يُمكِن استعمال الدالَّة range(a, b) إذا أراد المُستخدِم أنْ تكون الزيادة بمقدار (1). أمّا إذا أراد المُستخدِم تحديد خطوة مختلفة للزيادة، فيُمكِنه استعمال الدالَّة range(a, b, c) لتحديد قيمة الخطوة (c) . وإذا كانت قيمة الخطوة (c) تساوي (1)، فإنَّ النتيجة لن تختلف؛ لأنَّ range(a, b) تعني ضمنيًّا range (a, b, 1).

مثال:

يُبيِّن البرنامج الآتي استخدام الدالَّة range في العَدِّ العكسي من 5 إلى 1:

```
for x in range(5, 0, -1):
print(x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



إضاءة

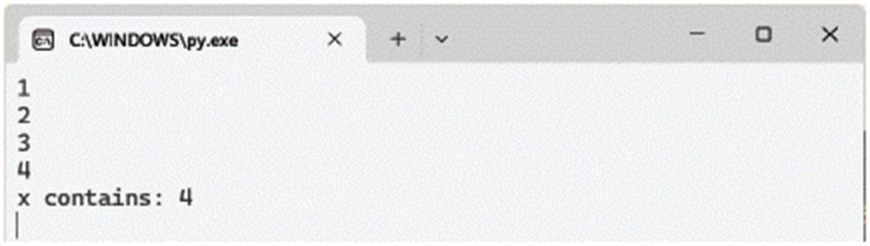
في لغة البرمجة بايثـــون (Python)، يُمكِن الوصول إلى المُتغيِّرات التي عُرِّفت داخل الحلقة من خارج الحلقة.

مثال:

يطبع البرنامج الآتي القِيَم من 1 إلى 4، ثمَّ يطبع عبارة x contains:", x " حيث x هي القيمة النهائية ضمن range

```
for x in range(1, 5, 1):
  print(x)
  print("x contains:",x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



for مع حلقات break جملة التحكُّم

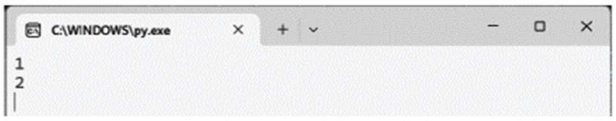
بالطريقة نفسها التي for مع حلقات break تعمل جملة التحكُّم while استُخدِمت فيها مع حلقات

مثال:

ويتوقَّف عن ذلك حين تصبح for يُنفِّذ البرنامج الآتي الحلقة x 2 قيمة تساوي :

```
for x in range(1, 5, 1):
  print(x)
if x == 2:
break
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط جماعي

أُجرِّب وأستنتج:

في المثال السابق، إذا وُضِعت جملة print() بعد جملة (if)، فماذا سيحدث؟

أتوقَّع ناتج البرنامج، ثمَّ أُنفِّذه باستخدام برمجية بايثون (Python)، وأُلاحِظ الناتج.

الحل : سيطبع فقط 2 على شاشة جهاز الحاسوب كما يلي :

```
for x in range(1, 5, 1):
  if x == 2:
    print(x)
    break
```

```
2
```

النتيجة على شاشة جهاز الحاسوب

for مع حلقات continue جملة التحكُّم

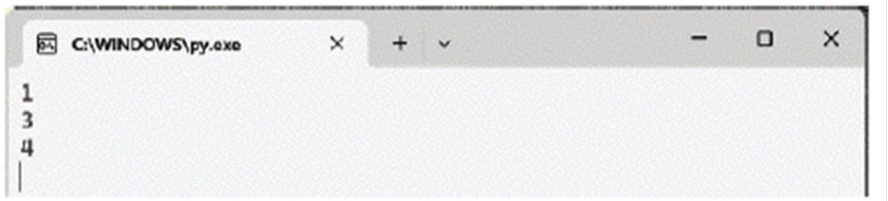
for مع حلقات continue تُستخدَم جملة التحكُّم بالطريقة نفسها التي استُخدِمت فيها مع حلقات while .

مثال:

يطبع البرنامج الآتي الأعداد 1 و 3 و 4 باستخدام جملة التحكُّم continue :

```
for x in range(1, 5, 1):
  if x == 2:
   continue
   print(x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



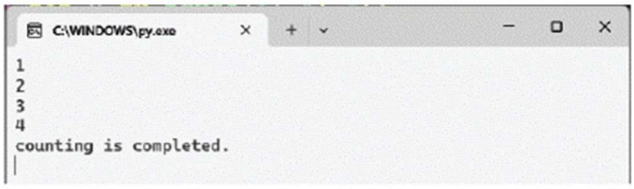
for مع حلقات else جملة

for مع حلقات else يُمكِن استعمال جملة لتنفيذ مجموعة من الأوامر عند الخروج من الحلقة.

مثال:

```
for x in range(1, 5, 1):
   print(x)
  else:
   print("counting is completed.")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط جماعي

أُجرِّب وأستنتج:

في المثال السابق، إذا وُضِعت جملة

```
if x == 2: break
```

قبل جملة print(x) في البرنامج، فماذا سيحدث؟ هل سيُنفَّذ البرنامج جملة (else)؟

أتوقَّع ناتج البرنامج، ثمَّ أُنفِّذه باستخدام برمجية بايثون (Python)، وأُلاحِظ الناتج.

الحل : عند قيمة x = 2 سيتوقف عمل جملة for ولن ينفذ ما بعد else كما يلي :

```
for x in range(1, 5, 1):
  if x == 2:
    break
  print(x)
else:
  print("counting is completed.")
```

النتيجة على شاشة جهاز الحاسوب ←

```
1
```

حلقات for المُتداخِلة (Nested for Loops)

يُمكِن كتابة حلقة for في البرنامج داخل حلقة for أُخرى، عندئذٍ سيُنفِّذ البرنامج الحلقة الداخلية في كل دورة من دورات الحلقة الخارجية.

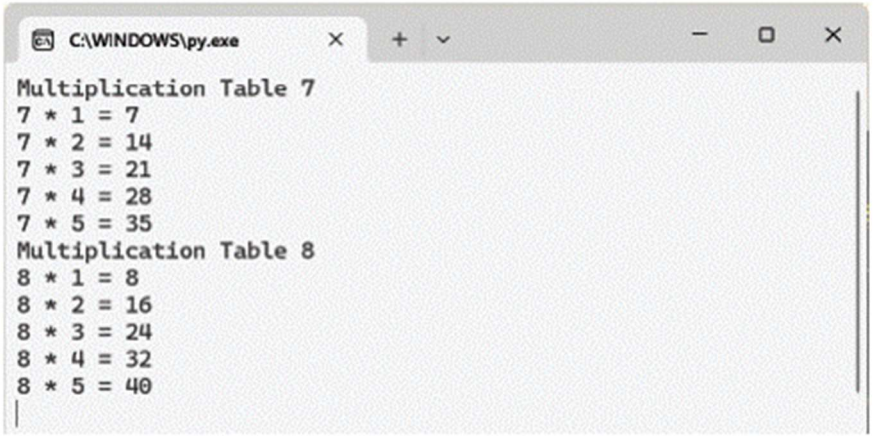
مثال:

يطبع البرنامج الآتي العناصر الخمسة الأولى من جدولي الضرب للعددين 8 و 7 باستخدام حلقات for المُتداخِلة:

```
for x in range(7, 9):
  print("Multiplication Table", x)
```

```
  for y in range(1, 6):
    print(x, "*", y, "=", x * y)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط فردي

أتتبَّع المقطع البرمجي السابق، ثمَّ أُدوِّن عدد مَرّات تنفيذ الحلقة الخارجية والحلقة الداخلية.

- أُعدِّل على المقطع البرمجي لطباعة جداول الضرب للأعداد: (7)، (8)، (9)، (10).
- أُنفِّذ المقطع البرمجي في بيئة بايثون (Python)، وأستكشف الأخطاء، ثمَّ أُصحِّحها.

أُقارِن إجاباتي بإجابات زملائي/ زميلاتي في الصف.

الحل:

```
for x in range(7, 11):
    print("Multiplication Table", x)
    for y in range(1, 11):
        print(x, "*", y, "=", x * y)
```

إضاءة

إذا أراد المُستخدِم كتابة حلقة (for) فارغة (أيْ لا تحتوي على أيِّ جملة)، فإنَّه يضع جملة (pass) داخل هذه الحلقة؛ لكيلا تصله رسالة تفيد بوجود خطأ في البرنامج.

نشاط عملي

أُجرِّب وأستنتج:

أُجرِّب وأستنتج: ما ناتج تنفيذ المقطع البرمجي الآتي مع وجود جملة (pass) ومن دون وجودها؟

```
for x in range(5):
    pass
```

الحل:

لن يظهر أي شيء عند تنفيذ المقطع البرمجي

```
for x in range(5):
pass
```

أما عند تنفيذ المقطع بدون pass سوف يعطي Syntax error لعدم اكتمال جملة الدوران

نشاط جماعي

اكتشاف الأخطاء في المقطع البرمجي ضمن لغة البرمجة بايثون (Python)

أُحلِّل – بالتعاون مع أفراد مجموعتي – المقطع البرمجي الآتي، ثمَّ أكتشف الأخطاء الواردة فيه من دون أنْ أُنفِّذه، ثمَّ أكتب المقطع البرمجي الصحيح، وأعمل على تنفيذه:

```
count = 0
for i in range(10)
    if i % 2 = 0:
        while count < 5:
        print("Count is:", count)
        count += 1
else:
print("Done with inner loop")
    if count == 5:
    break
print("Loop ended.")
```

الحل : الخطأ بكتابة اشارة المقارنة =

```
if i % 2 = 0:
```

الصواب

```
if i % 2 == 0:
```

انظر الشكل :

```
count = 0
for i in range(10):
    if i % 2 == 0:
        while count < 5:
            print("Count is:", count)
            count += 1
    else:
        print("Done with inner loop")
        if count == 5:
            break
        print("Loop ended.")
```

```
Count is: 0
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Done with inner loop
```