Python Loops

Python has two primitive loop commands:

- while loops
- for loops

The while Loop

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

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

```
# Print i as long as i is less than 6

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

```
1
2
3
4
5
```

The break Statement

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

```
# Exit the loop when i is 3:

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

```
1
2
3
```

The continue Statement

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

```
# Continue to the next iteration if i is 3:

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

```
1
2
4
5
6
```

The else Statement

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

```
# 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")
```

```
1
2
3
4
5
i is no longer less than 6
```

Python 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.

```
# Print each fruit in a fruit list:

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

```
apple
banana
cherry
```

Looping Through a String

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

```
# Even strings are iterable objects, they contain a sequence of characters:

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

```
b
a
n
a
n
a
```

The break Statement

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

```
# Exit the loop when x is "banana":

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

```
apple
banana
```

```
# 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)
```

```
apple
```

The continue Statement

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

```
# Do not print banana:

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

```
apple
cherry
```

The range() Function

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.

```
# Using the range() function:

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

# Note: that range(6) is not the values of 0 to 6, but the values 0 to 5.
```

```
0
1
2
3
4
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).

```
# Using the start parameter:

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

```
2
3
4
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).

```
# Increment the sequence with 3 (default is 1):

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

```
2
5
8
11
14
17
20
23
26
29
```

Else in For Loop

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

```
# 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!")
```

```
0
1
2
3
4
5
Finally finished!
```

Note: The else block will NOT be executed if the loop is stopped by a break statement.

```
# Break the loop when x is 3, and see what happens with the else block:

for x in range(6):
  if x == 3: break
  print(x)
```

```
else:
```

```
0
1
2
```

Nested Loops

A nested loop is a loop inside a loop.

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

```
# Print each adjective for every fruit:

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

for x in adj:
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

```
  for x in [0, 1, 2]:
    pass
```