The range() function in Python

The range() function will create a ordered list of numbers.

range() function can take 1, or 2, or 3 input arguments in the form of "range(start, end, step)" Note: the end number is not included in the result list. Step can be a positive or negative number

For example:

range(10): means a list of (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

range(5, 10): means a list of (5, 6, 7, 8, 9)

range(5, 10, 2): means a list of (5, 7, 9)

range(10, 5, -2): means a list of (10, 8, 6)

for loop

A "for loop" can be used to execute a set of statement for a certain number of times.

For example:

In [1]:
```
for x in range(10):
    print(x)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [2]:
```
for y in range(5, 10):
    print(y)
```

```
5
6
7
8
9
```

In [6]:
```
# print all of the positive odd numbers less than 10
for z in range(1, 10, 2):
    print(z)
```

```
1
3
5
7
9
```

The continue Statement

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

In [9]:
```
# print integers from 0 to 9, but skip 7
for x in range(10):
    if x > 6 and x < 8:
        continue
    else:
        print(x)
```

```
0
1
2
3
4
5
6
8
9
```

The break statement

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

In [10]:
```
# find the integer that when it is added by 4 equal to 10
for x in range(10):
    if x + 4 == 10:
        print("Found the number: ", x)
        break
```

```
Found the number:  6
```

Else in For Loop

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

In [7]:
```
# only print integers from 0, to 5
for x in range(10):
    if x < 6:
        print(x)
    else:
        continue
else:
    # the following line will only be executed when the loop finished without
    print("for loop has completed")
print("after the for loop")
```

```
0
1
2
3
4
5
for loop has completed
after the for loop
```

Example 2: else in for loop

When the for loop is terminated before it reaches the full range, "else:" statement will not be executed.

In [12]:
```
for x in range(10):
    if x < 6:
        print(x)
    elif x > 8:
        print('break since x=', x)
        break
    else:
        print('skip since x=', x)
        continue
else:
    # the following line will only be executed when the loop finished without
    print("for loop has completed")
print("after the for loop")
```

```
0
1
2
3
4
5
skip since x= 6
skip since x= 7
skip since x= 8
break since x= 9
after the for loop
```

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":

In [15]:
```
for x in range(3):
    for y in range(3, 6):
        print('(', x, ',', y, ')')
```

```
( 0 , 3 )
( 0 , 4 )
( 0 , 5 )
( 1 , 3 )
( 1 , 4 )
( 1 , 5 )
( 2 , 3 )
( 2 , 4 )
( 2 , 5 )
```

In [ ]:
```

```