Python While Loops

- With the while loop we can execute a set of statements as long as a condition is true
- Example
  - Print i as long as i is less than 6

In [1]:
```
i = 1
while i < 6:
  print(i)
  i += 1  # as same as i = i + 1
```

```
1
2
3
4
5
```

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

- 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

- With the break statement we can stop the loop even if the while condition is true
- Example
  - Exit the loop when i is 3

In [2]:
```
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

- Example
  - Continue to the next iteration if i is 3

In [3]:
```
i = 0
while i < 6:
  i += 1 # i = 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
- Example
  - Print a message once the condition is false

In [3]:
```
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
```

Nested While Loop

- A nested while loop is a loop structure that contains another while loop within its body
- Example

In [1]:
```
outer_count = 1

while outer_count <= 3:
    inner_count = 1

    while inner_count <= 3:
        print(f"Outer count: {outer_count}, Inner count: {inner_count}")
        inner_count += 1

    outer_count += 1
```

```
Outer count: 1, Inner count: 1
Outer count: 1, Inner count: 2
Outer count: 1, Inner count: 3
Outer count: 2, Inner count: 1
Outer count: 2, Inner count: 2
Outer count: 2, Inner count: 3
Outer count: 3, Inner count: 1
Outer count: 3, Inner count: 2
Outer count: 3, Inner count: 3
```