While loops will continue to execute a block of code while some condition reamins True

For example, while my pool is not full, keep filling my pool with water.

Or while my dogs are still hungry, keep feeding my dogs

while some_boolean_condition:

    # do something

else:

    # do something different

In [1]:
```
x = 0

while x < 5:
    print(f'The current value of x is {x}')
    x = x + 1
```

```
The current value of x is 0
The current value of x is 1
The current value of x is 2
The current value of x is 3
The current value of x is 4
```

In [2]:
```
x = 0

while x < 5:
    print(f'The current value of x is {x}')
    x = x + 1
else:
    print("X IS NOT LESS THAN 5")
```

```
The current value of x is 0
The current value of x is 1
The current value of x is 2
The current value of x is 3
The current value of x is 4
X IS NOT LESS THAN 5
```

In [3]:
```
x = 50

while x < 5:
    print(f'The current value of x is {x}')
    x = x + 1
else:
    print("X IS NOT LESS THAN 5")
```

```
X IS NOT LESS THAN 5
```

In [4]:
```
x = 50

while x < 5:
    print(f'The current value of x is {x}')
    x = x + 1
```

break, continue, pass

We can use break, continue, and pass statements in our loops to add additional functionality for various cases.

The three statements are defined by:

break: Breaks out of the current closest enclosing loop
continue: Goes to the top of the closest enclosing loop
pass: Does nothing at all

In [5]:
```
x = [1,2,3]

for item in x:
    # comment
```

```
  File "C:\Users\Keegz\AppData\Local\Temp/ipykernel_15956/1264013902.py", line 4
    # comment
               ^
IndentationError: expected an indented block
```

We can an error because the syntax expects something to happen after the colon, therefore we put 'pass'

In [6]:
```
x = [1,2,3]

for item in x:
    # comment
    pass

print('end of my script')
```

```
end of my script
```

In [7]:
```
mystring = 'Sammy'
```

In [8]:
```
for letter in mystring:
    print(letter)
```

```
S
a
m
m
y
```

In [9]:
```
for letter in mystring:
    if letter == 'a':
        continue
    print(letter)
```

```
S
m
m
y
```

So if that letter is equal to a, continue, meaning go back to the top of the closest enclosing loop.

Lets try it with break instead:

In [10]:

```
    if letter == 'a':
        break
    print(letter)
```

```
S
```

With break, instead of going to the top of the closest enclosing loop, it breaks once letter == a

In [11]:

```
x = 0

while x < 5:
    print(x)
    x += 1
```

```
0
1
2
3
4
```

In [12]:

```
x = 0

while x < 5:

    if x == 2:
        break
    print(x)
    x += 1
```

```
0
1
```

In [ ]:

```

```