#loop is used to repeat some code again and again.

#We use it when we want to do the same thing multiple times without writing it again and again.

#type of loop:

# 1.while loop

# 2. for loop

# While loop:

# A while loop keeps running as long as a condition is true.As soon as the condition becomes false, the loop stops.

```
i = 0

while i < 6:

    print(i)

    i += 1
```

# The break Statement

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

```
i = 1

while i < 6:

  print(i)

  if i == 3:

    break

  i += 1
```

# The continue Statement

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

```
i = 0

while i < 6:

  i += 1

  if i == 3:

    continue

  print(i)
```

The else Statement

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

```
i = 1

while i < 6:

  print(i)

  i += 1

else:

  print("i is no longer less than 6")
```

# 2.for loop:

# A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

# range - The range() is the in-build function in Python generates a sequence of numbers.

# syntax for range: range(start, stop, step)

# start → Where to start (bydefault = 0)

# stop → Where to stop (this number is NOT included)

# step → How much to increment by (bydefault = 1)

# eg.1

```
for a in range(5):
   print(a)
```

# number 5 is not included. The range() function in Python does not go up to the last number, it stops one number before it.

# eg.2

```
for i in range(1,10,2):
   print(i)
```

# to print the list through loop

```
my_list = [10, 20, 30, 40, 50]
for item in my_list:
   print(item)
```

Using range and len to access list with index

```
my_list = ['apple', 'banana', 'cherry']
for i in range(len(my_list)):
   print(i, my_list[i])
```

# Looping Through a String

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

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

# The break Statement

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

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  print(x)

  if x == "banana":

    break
```

# 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)
```

# 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":

```
adj = ["red", "yellow", "tasty"]

fruits = ["apple", "banana", "cherry"]


for x in adj:

  for y in fruits:

    print(x, y)
```

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 range(3):
  if x == 2:
  else:
    print(x)
```