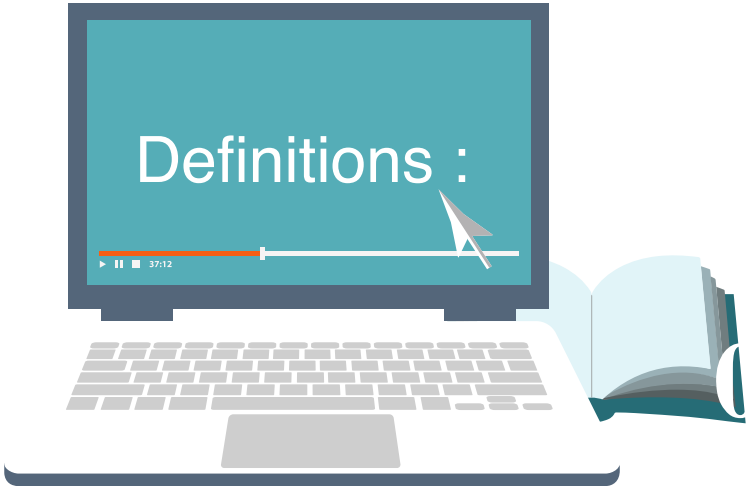
python loops

- while loops
- for loops



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

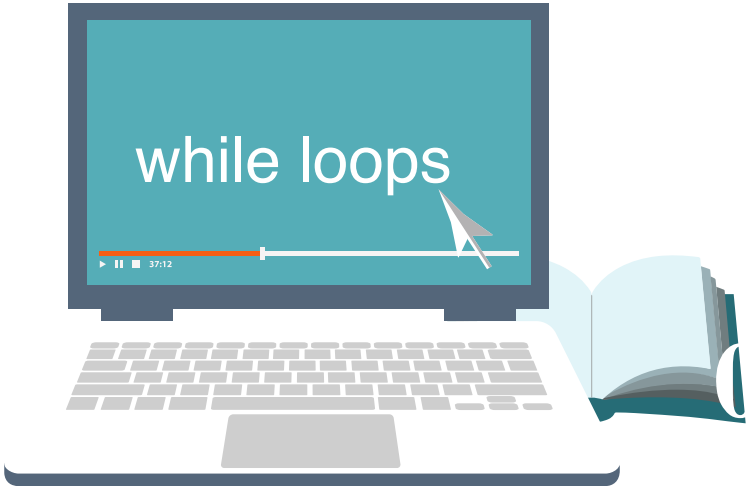
```
i =1

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

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

---

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

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



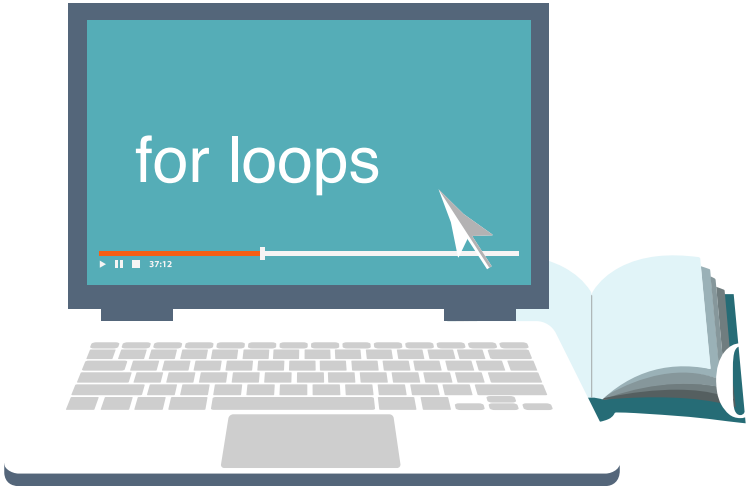
break statement : we can stop the loop even if the while condition is true

```
i =1
while i < 6:

  print(i)
  if i == 3:
break

i += 1
```

---

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)
```

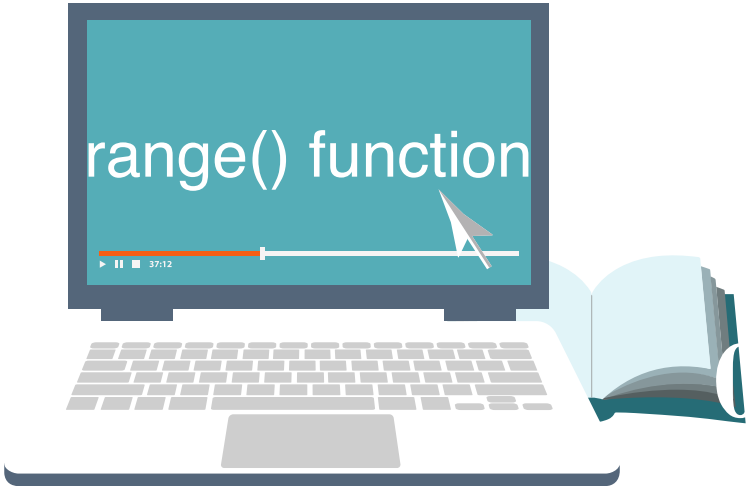


break statement : we can stop the loop before it has looped through all the items

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

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

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



range() : To loop through a set of code a specified number of times

1- using range() function

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

2- using start parameter (default 1)

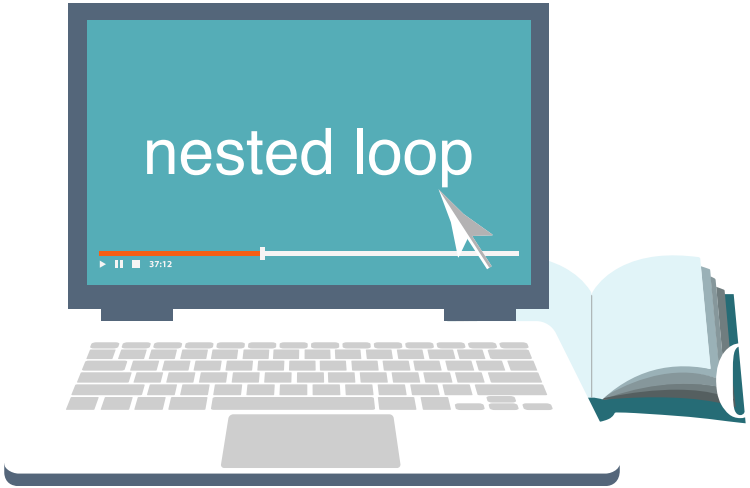
```
for x in range(2, 6):
  print(x)
```

3- using start parameter(increment with 3)

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

4- else

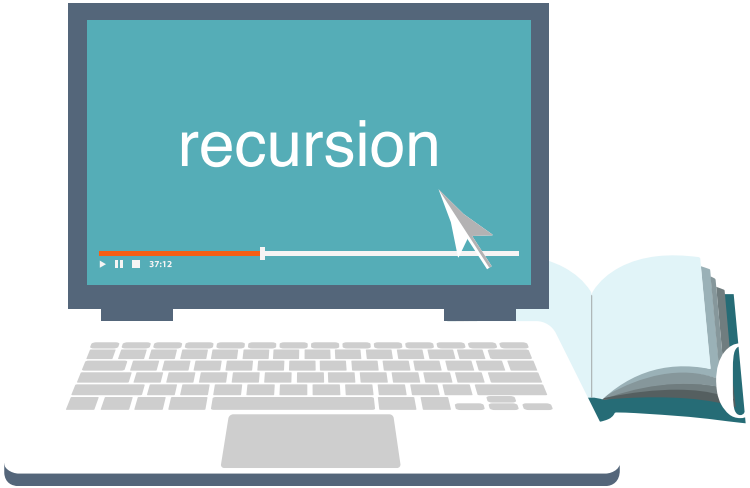
```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```



nested loop : is a loop inside a loop.

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

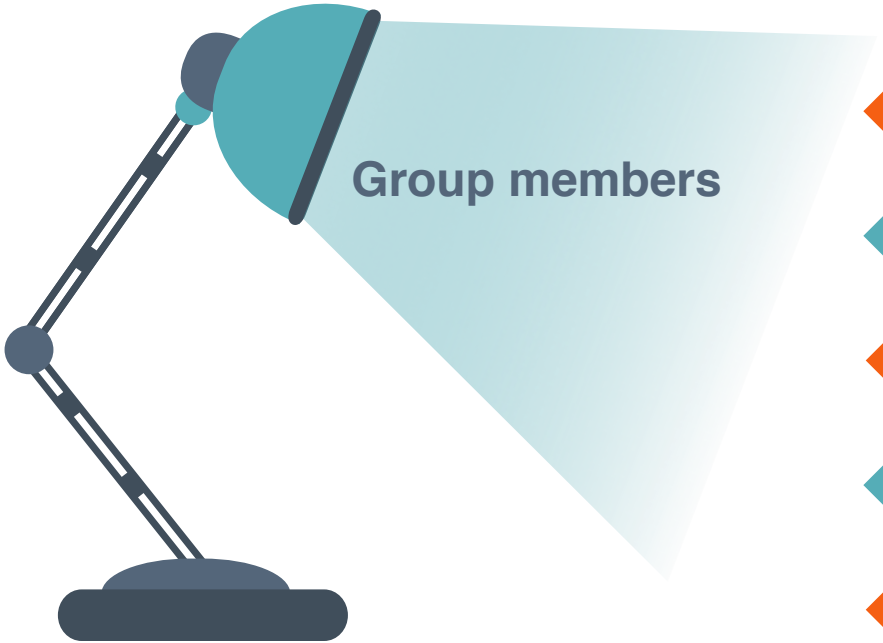
```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
for x in adj:
  for y in fruits:
print(x, y)
```



recursion : It means that a function calls itself.

```
def tri_recursion(k):

  if(k>0):
    result = k+tri_recursion(k-1)
    print(result)
  else:
    result = 0
  return result
print("\n\nRecursion Example Results")
tri_recursion(6)
```

Note :tri_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).



Group members

01 Ghadir aljafen

02 Rema alsmaeel

03 Hadeel alrashed

04 Rema alajlan

05 Lama altwijri

Thank you