Control Statements

if - else

In [19]:
```
a = 3
b = 4
if a > b:
  print('a is greater than b')
else:
  print('b is greater than a')
```

```
b is greater than a
```

if - elif - else

In [20]:
```
a = 3
b = 4
if a > b:
  print('a is greater than b')
elif a < b:
  print('a is less than b')
else:
  print('a is equal to b')
```

```
a is less than b
```

nested if

In [21]:
```
a = 1
b = 2
c = 3

if a > b:
  if a > c:
    print('a is max')
  else:
    print('c is max')
else:
  if b > c:
    print('b is max')
  else:
    print('c is max')
```

```
c is max
```

Loops

While Loop

Program to print Factorial of a number

In [22]:
```
n = int(input('Enter a number'))
f = 1
while n > 0:
  f *= n
  n = n - 1
print('Factorial is', f)
```

```
Enter a number5
Factorial is 120
```

Range Function

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Syntax

```
range(start, stop, step)
```

Parameter Values

Parameter	Description
start	Optional. An integer number specifying at which position to start. Default is 0
stop	Required. An integer number specifying at which position to stop (not included).
step	Optional. An integer number specifying the incrementation. Default is 1

For Loop

In [23]:
```
for i in range(1,10):
  print(i)
```

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

In [24]:
```
for i in range(10):
  print(i)
```

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

In [25]:
```
for i in range(1,10,2):
  print(i)
```

```
1
3
5
7
9
```

In [26]:
```
for i in range(10,1,-1):
  print(i)
```

```
10
9
8
7
6
5
4
3
2
```

break

The break keyword is used to break out a for loop, or a while loop.

In [27]:
```
for i in range(9):
  if i > 3:
    break
  print(i)
```

```
0
1
2
3
```

In [28]:
```
i = 1
while i < 9:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

Continue

The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.

In [29]:
```
for i in range(9):
  if i == 3:
    continue
  print(i)
```

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

In [30]:
```
i = 0
while i < 9:
  i += 1
  if i == 3:
    continue
  print(i)
```

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

Pass

The pass statement is used as a placeholder for future code.

When the pass statement is executed, nothing happens, but you avoid getting an error when empty code is not allowed.

Empty code is not allowed in loops, function definitions, class definitions, or in if statements.

In [31]:
```
a = 33
b = 200

if b > a:
  pass
```

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 [32]:
```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
3
4
5
Finally finished!
```

Note: The else block will NOT be executed if the loop is stopped by a break statement.

In [33]:
```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
```

In [34]:
```
#program to print prime numbers between a given range

st = int(input('Enter the value of start '))
en = int(input('Enter the value of end '))

import math

for n in range(st, en+1):
  for d in range(2, n//2 + 1):
    if n % d == 0:
      break
  else:
    #executes only if the for loop is not broken by break statement
    if n>1:
      print(n)
```

```
Enter the value of start 0
Enter the value of end 25
2
3
5
7
11
13
17
19
23
```

Else in While Loop

Execute if the while loop is not broken due to break statement

In [35]:
```
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
```