Learn Python part 2

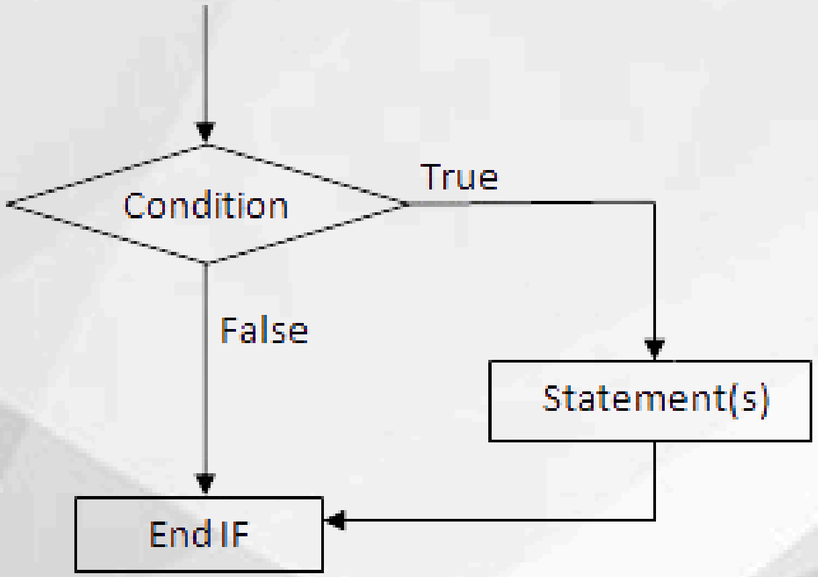
control statements

python™

If ... Else

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b



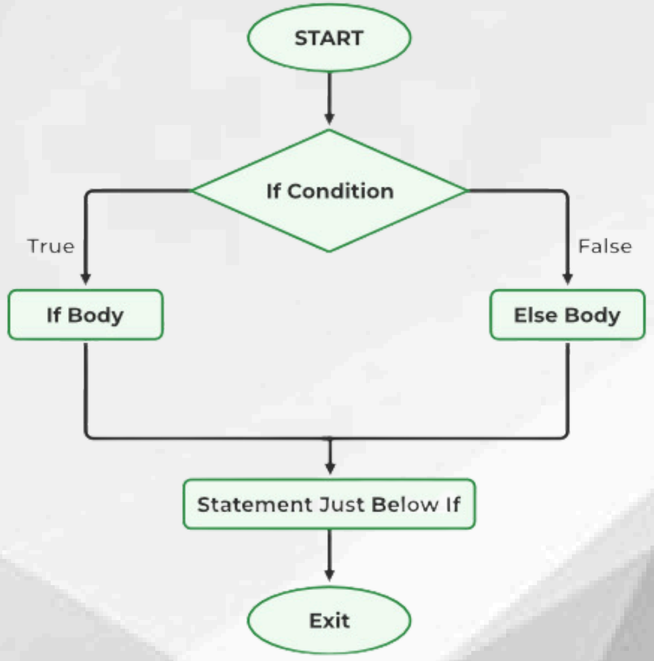
An "if statement" is written by using the **if** keyword.

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

Else

The else keyword catches anything which isn't caught by the preceding conditions.

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
else:
    print('a is greater than b')
```



Elif

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Else :

The else keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

```
if a > b: print("a is greater than b")
```

Short Hand If ... Else

```
print("A") if a > b else print("B")
```

And

The **and** keyword is a logical operator, and is used to combine conditional statements:

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

Or

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Not

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

Nested If

You can have if statements **inside** if statements, this is called nested if statements.

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

The pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
a = 33
b = 200

if b > a:
  pass
```

While Loops

Python has two primitive loop commands:

- **while** loops
- **for** loops
- loop is a control structure that allows you to **repeatedly execute a block** of code as long as a certain condition

```
i = 1
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")
```