

# Python If ... Else

---

## Python Conditions and If statements

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`

These conditions can be used in several ways, most commonly in "if statements" and loops.

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

### Example

If statement:

```
a = 33
b = 200

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

In this example we use two variables, `a` and `b`, which are used as part of the if statement to test whether `b` is greater than `a`. As `a` is 33, and `b` is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

### Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

## Example

If statement, without indentation (will raise an **error**):

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

---

## Elif

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

## Example

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

In this example `a` is equal to `b`, so the first condition is not true, but the `elif` condition is true, so we print to screen that "a and b are equal".

---

# Else

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

## Example

```
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")
```

In this example `a` is greater than `b`, so the first condition is not true, also the `elif` condition is not true, so we go to the `else` condition and print to screen that "a is greater than b".

You can also have an `else` without the `elif`:

## Example

```
a = 200
b = 33

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

```
print("b is not greater than a")
```

---

## Short Hand If

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

### Example

One line if statement:

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

---

## Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

### Example

One line if else statement:

```
a = 2
```

```
b = 330
```

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

This technique is known as Ternary Operators, or Conditional Expressions.

You can also have multiple else statements on the same line:

### Example

One line if else statement, with 3 conditions:

```
a = 330
```

```
b = 330
```

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

---

## And

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

### Example

Test if `a` is greater than `b`, AND if `c` is greater than `a`:

```
a = 200
```

```
b = 33
```

```
c = 500
```

```
if a > b and c > a:
```

```
    print("Both conditions are True")
```

---

## Or

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

### Example

Test if `a` is greater than `b`, OR if `a` is greater than `c`:

```
a = 200
```

```
b = 33
```

```
c = 500

if a > b or a > c:

    print("At least one of the conditions is True")
```

---

## Nested If

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

### Example

```
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.

### Example

```
a = 33

b = 200
```

```
if b > a:  
    pass
```