# Conditionals

16 > 18 → false

→ if (age > 18):

else cond:

else:

**Basic rule:** Truthy & falsy values

→ when ever a value evaluated it returns boolean value either

True / false

**Comparison Operators & logical operators**

↳ (==) → equality , != (not equal)

>, <, >=, <=

"25" == "25"

25 == 25

↑ ↑

(=) assignment operator

27 != 25 ✓

(A == B) → T

>= <=

* we usually compare the values & take decisions

**logical operators:**

and , or , not

if (cond1 and cond2 . . . . condn)

T T . . . . F

| A | B | A AND B |
|---|---|---|
| false | false | false |
| false | true | false |
| true | false | false |
| true | true | true |

and : Both conditions must be True

or : At least one condition must be True

if (cond1 or cond2 or cond3 . . . . condn):

T

F T

×

enter

② True and True and True and True

True and True and True

(True) and True

True

not : invert the boolean value.

# Loops

* It helps us to execute the same piece of code repeatedly just by writing only once

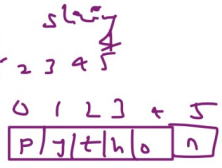
  * For loop
  * While
  * Loop Control

**for loop :** iterates over a sequence

→ range of numbers

→ String — "python"

→ [ ], tuple, set, { }




**while loop :** repeat until condition is true

**Note :**

don't forget to update the condition

```
counter = 1
while counter <= 5:
  print(f"Hello welcome, How are you - {counter}")
```

false

counter += 1

① 1 <= 5 ✓

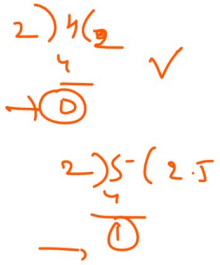
② 1 <= 5 ✓

③ 1 <= 5

**loop Control :**

break : exit the loop immediately

continue : skip to the next iteration



# Functions

→ It helps us to execute a piece of code(same logic) as when required

* Defining a function
* Calling a function
* parameters & arguments
* return
* function scope

## Defining a function:

default void

```
def func_name(parameters):
    ____
    ____
    ____
    ____
    return value
```

↑ which holds the value from called

```
def add(a, b, c):
    return a+b+c
```

## Calling a function:

→ which contains actual value

```
merage = func_name(arguments)
print(func_name(arguments))
```

```
add(1, 2, 3)
```

## Parameters & arguments

→ default parameters

→ Required parameters

### decorators

* default argument
* positional arguments
* named argument
* keyword arguments

## Return values:

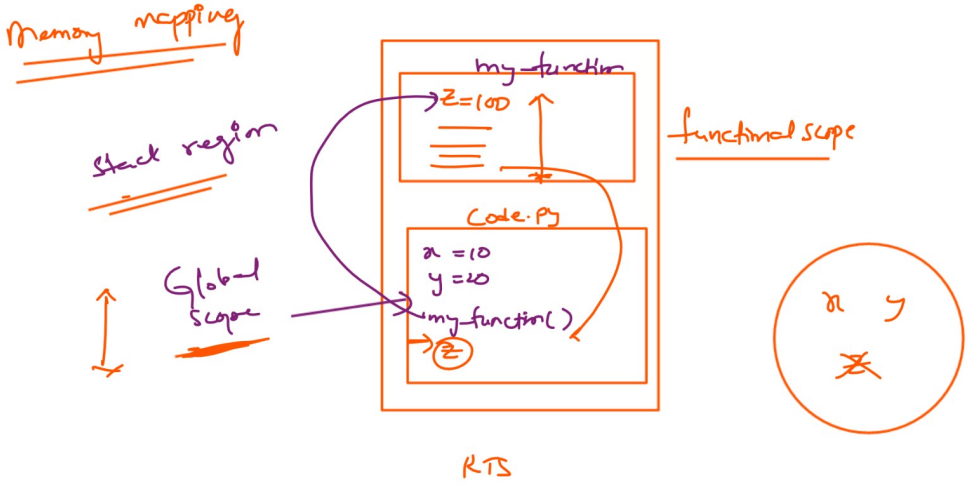
* no return value
* return a value
* return a multiple value

## functional scope :

* Global variable & global scope

access
- function
- condition
- any where

* local variable has local scope
- function




## Note:

→ when inside a function, scope of variable assessment

→ local variable
→ Global variable

} neither it exists → Error throws