Functions in python

- why do we need function lets understand one scenario you might be working on simple or complex project
- when you work on complex project break down complex project to smaller task.
- when you talk about smaller task of course you need to writing multiple statement.
- e.g if you want to operate payment operation in that you need to multiple statement all that statement keep it together & re useable later.
- all the statement of the tasks are stays together thats why we are creation function

FUNCTION -->

- Inbuild function -- print(), type(), id(), sqrt(), max(), min(), sqrt(), ceil(), etc
- User defined function
- function is collection of statement
- 2 main property of the function is -- define the function & calling the function
- function always define with def & function always declares as ()

difference between variable & function || a - variable || b() - function

In [1]:
```
def greet(): # define the function hear
    print('good evening') # print the statement
```

In [2]:
```
def greet():
    print('good evening')
greet()
```

```
good evening
```

In [3]:
```
def greet():
    print('good evening')
greet()

def greet():
    print('good evening')
```

```
greet()

def greet():
    print('good evening')
greet()
```

```
good evening
good evening
good evening
```

In [4]:
```
def greet():
    print('good evening')
greet()
print() # print means next line
greet()
print()
greet()
```

```
good evening

good evening

good evening
```

In [5]:
```
def greet():
    print('good evening')
greet()

greet()

greet()

greet()
```

```
good evening
good evening
good evening
good evening
```

In [6]:
```
def add(x,y):
    c=x+y
    print(c)
add(5,6)
```

```
11
```

In [7]:
```
def add(x):
    c=x+y
    print(c)
add(5,6)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 4
      2     c=x+y
      3     print(c)
----> 4 add(5,6)

TypeError: add() takes 1 positional argument but 2 were given
```

In [ ]:
```
def add(x,y,z):
    c=x+y
    print(c)
add(5,6)
```

In [ ]:
```
def add(x,y):
    c=x+y
    print(c)
add(5,6,7)
```

In [8]:
```
def greet():
    print('good evening')
greet()

def add(x,y):
    c=x+y
    print(c)
add(5,6)
```

```
good evening
11
```

In [9]:
```
def greet():   # It provides space
    print('good evening')
greet()
```

```
print()

def add(x,y):
    c=x+y
    print(c)
add(5,6)
```

```
good evening

11
```

In [10]:
```
def greet():
    print('good evening')
def add(x,y):
    c=x+y
    print(c)

greet()
add(5,6)
```

```
good evening
11
```

In [11]:
```
def greet():
    print('good evening')
def add(x,y):
c=x+y
    print(c)

greet()
add(5,6)
```

```
  Cell In[11], line 4
    c=x+y
    ^
IndentationError: expected an indented block after function definition on line 3
```

In [12]:
```
def greet():
    print('good evening')
def add(x,y):
    c=x+y
    print(c)
def sub(x,y):
```

```
    c=x-y
    print(c)

greet()
add(5,6)
sub(5,6)
```

```
good evening
11
-1
```

In [13]:
```
def greet():
    print('good evening')
def add(x,y):
    c=x+y
    print(c)
def sub(x,y):
    c=x-y
    print(c)

greet()
add(5,6)
sub(5,6)

greet()
add(10,20)
sub(10,20)
```

```
good evening
11
-1
good evening
30
-10
```

In [14]:
```
def add(x,y):
    c=x+y
    print(c)
add(5,6)
```

```
11
```

In [15]:
```
def add(x,y):
    c=x+y
    return c
add(5,6)
```

Out[15]: 11

- as a functioni we have 2 choice
- 1- whenever we call the function - function is do the task for you greet() & add()
- 2- we have another type of function it will return you the value

In [16]:
```
def add(x,y):
    c=x+y
    return c
def sub(x,y):
    d=x-y
    return d

add(20,10)
sub(20,10)
```

Out[16]: 10

In [17]:
```
def add(x,y):
    c=x+y
    return c
def sub(x,y):
    d=x-y
    return d

print(add(20,10))
print(sub(20,10))
```

```
30
10
```

In [18]:
```
def add_sub(x,y): # what if i want to return 2 values add_sub & i want to return 2 values & function can access
    c = x+y
    d = x-y
```

```
    return c, d

print(add_sub(4,5))
```

```
(9, -1)
```

In [19]:
```
def add_sub(x,y):
    c = x+y
    d = x-y
    return c, d

print(add_sub(20,10))
```

```
(30, 10)
```

In [20]:
```
def add_sub(x,y):
    c = x+y
    d = x-y
    return c, d

result = add_sub(20,10)
print(type(result))
```

```
<class 'tuple'>
```

In [21]:
```
def add_sub(x,y):
    c = x+y
    d = x-y
    return c, d

result = add_sub(20,10)
print(add_sub(20,10))
print(type(result))
```

```
(30, 10)
<class 'tuple'>
```

In [22]:
```
def add_sub(x,y):
    c = x+y
    d = x-y
    return c, d

result, result2 = add_sub(20,10)
print(result)
```

```
print(result2)
print(type(result))
print(type(result2))
```

```
30
10
<class 'int'>
<class 'int'>
```

In [23]:
```
def add(x,y):
    c=x+y
    return c

def sub(x,y):
    d=x-y
    return d

result = add(20,10)
result2 = sub(20,10)

print(result)
print(result2)

print(type(result))
print(type(result2))
```

```
30
10
<class 'int'>
<class 'int'>
```

function arguments

- FUNCTION ARGUMENT
- How to pass parameter to a function & what happend to the variable when you pass to a function & if you modify

the value then what happen

- every code check with debug

In [24]:
```
def update():
    x = 8
    print(x)

update(8)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[24], line 5
      2     x = 8
      3     print(x)
----> 5 update(8)

TypeError: update() takes 0 positional arguments but 1 was given
```

In [25]:
```
def update(x): # update function take the value from the user
    x = 8
    return x

update(8)
```

Out[25]: 8

In [26]:
```
def update(x): # user want to update the value from 8 to 10
    x = 8
    print(x)

update(10)
```

```
8
```

In [27]:
```
def update(x):
    x = 8
    print(x)

a = 10
update(a)
print(a)
```

```
8
10
```

In [28]:
```
def update(x):
    x = 8
    return x

a = 10
update(a)
```

Out[28]: 8

In [29]:
```
def update(x):
    x = 8
    return x

a = 10
update(a)

print(a)
```

```
10
```

In [30]:
```
def update(x):
    x = 8
    return x

a = 10

print(update(a))

print(a)
```

```
8
10
```

function without argument - not required any thing

function with argument -- divide into 2 part

- formal argument
- actual argument

In [31]:
```
def add(x,y): #x & y is called -- FORMAL AGRUMENT
    c=x+y
    return c

add(4,5) #4 & 5 is called - ACTUAL ARGUMENNT
```

Out[31]: 9

In [32]:
```
def add(a,b): # a & b called formal argument
    c = a+b
    print(c)

add(5,6) # 5 and 6 we called as actual argument
```

11

In [33]:
```
def add(a,b,d,e): # a & b called formal argument
    c = a+b+d
    print(c)

add(5,6,7,8) # 5 and 6 we called as actual argument
```

18

POSITIONAL ARGUMENT

In [34]:
```
def add(x,y): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c

add(4,5) #4 & 5 is called - ACTUAL ARGUMENT
```

Out[34]: 9

In [35]:
```
def add(x,y): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c
```

```
add(4) #4 & 5 is called - ACTUAL ARGUMENT
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 5
      2     c=x+y
      3     return c
----> 5 add(4)

TypeError: add() missing 1 required positional argument: 'y'
```

In [36]:
```
def add(y,x): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c

add(4) #4 & 5 is called - ACTUAL ARGUMENT
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 5
      2     c=x+y
      3     return c
----> 5 add(4)

TypeError: add() missing 1 required positional argument: 'x'
```

In [37]:
```
def add(x,y,z): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c

add(4,5) #4 & 5 is called - ACTUAL ARGUMENT
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[37], line 5
      2     c=x+y
      3     return c
----> 5 add(4,5)

TypeError: add() missing 1 required positional argument: 'z'
```

In [38]:
```
def add(x,y,z): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c

add(4,5,0) #4 & 5 is called - ACTUAL ARGUMENT
```

Out[38]: 9

In [39]:
```
def add(x,y): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c

add(4,5,0) #4 & 5 is called - ACTUAL ARGUMENT
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[39], line 5
      2     c=x+y
      3     return c
----> 5 add(4,5,0)

TypeError: add() takes 2 positional arguments but 3 were given
```

In [40]:
```
def add(x,y,z,m,n,o,y,u): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c

add(4,5,0,7,4,54,5,68) #4 & 5 is called - ACTUAL ARGUMENT
```

```
  Cell In[40], line 1
    def add(x,y,z,m,n,o,y,u): # x & y is called -- FORMAL ARGUMENT
                         ^
SyntaxError: duplicate argument 'y' in function definition
```

In [41]:
```
def add(x,y,z,m,n,o,a,u): # x & y is called -- FORMAL ARGUMENT
    c=x+y
    return c

add(4,5,0,7,4,54,5,68) #4 & 5 is called - ACTUAL ARGUMENT
```

Out[41]: 9

In [42]:
```
def add(x,y,z,m,n,o,a,u): # x & y is called -- FORMAL ARGUMENT
    c=x+y+z+m+n+o+a+u
    return c

add(4,5,0,7,4,54,5,68) #4 & 5 is called - ACTUAL ARGUMENT
```

Out[42]: 147

In [43]:
```
def person(name,age):
    print(name)
    print(age)

person('nit', 22)
```

```
nit
22
```

In [44]:
```
def person(name,age):
    print(name)
    print(age)

person(22, 'nit')
```

```
22
nit
```

In [45]:
```
def person(name,age):
    print(name)
    print(age + 1)

person(22, 'nit')
```

```
22
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[45], line 5
      2     print(name)
      3     print(age + 1)
----> 5 person(22, 'nit')

Cell In[45], line 3, in person(name, age)
      1 def person(name,age):
      2     print(name)
----> 3     print(age + 1)

TypeError: can only concatenate str (not "int") to str
```

keyword agrument

In [46]:
```
def person(name,age):
    print(name)
    print(age)

person(22, 'nit')
```

```
22
nit
```

In [47]:
```
def person(name,age):
    print(name)
    print(age + 1)

person(22, 'nit')
```

```
22
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[47], line 5
      2     print(name)
      3     print(age + 1)
----> 5 person(22, 'nit')

Cell In[47], line 3, in person(name, age)
      1 def person(name,age):
      2     print(name)
----> 3     print(age + 1)

TypeError: can only concatenate str (not "int") to str
```

In [48]:
```
def person(name,age):
    print(name)
    print(age + 1)

person(age = 22, name = 'nit')
```

```
nit
23
```

In [49]:
```
def person(name,age):
    print(name)
    print(age + 1)

person(22, name = 'nit')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[49], line 5
      2     print(name)
      3     print(age + 1)
----> 5 person(22, name = 'nit')

TypeError: person() got multiple values for argument 'name'
```

In [50]:
```
def person(name,age):
    print(name)
    print(age + 1)
```

```
person(age = 22, 'nit')
```

```
  Cell In[50], line 5
    person(age = 22, 'nit')
                     ^
SyntaxError: positional argument follows keyword argument
```

In [51]:
```
def person(name,age):
    print(name)
    print(age + 1)

person(ag = 22, name = 'nit')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[51], line 5
      2     print(name)
      3     print(age + 1)
----> 5 person(ag = 22, name = 'nit')

TypeError: person() got an unexpected keyword argument 'ag'. Did you mean 'age'?
```

In [52]:
```
def person(name,age):
    print(name)
    print(age + 1)
    print('sir job chalagaya')

person(age = 22, name = 'nit')
```

```
nit
23
sir job chalagaya
```

In [53]:
```
def person(name,age,salary):
    print(name)
    print(age + 1)
    print('sir job chalagaya')

person(age = 22, name = 'nit')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 6
      3     print(age + 1)
      4     print('sir job chalagaya')
----> 6 person(age = 22, name = 'nit')

TypeError: person() missing 1 required positional argument: 'salary'
```

In [54]:
```
def person(name,age,salary):
    print(name)
    print(age + 1)
    print('sir job chalagaya')

person(age = 22, name = 'nit', salary = 10000)
```

```
nit
23
sir job chalagaya
```

In [55]:
```
def person(name, age, age2):
    print(name)
    print(age)
    print(age2)

person(age = 20, name = 'nit', age2 = 21)

# this is called keyword arguments
```

```
nit
20
21
```

Default argument

- while you open meta accountk minimum age criterial is so by default age is 18

In [56]:
```
def person(name,age): # in this code we expected to print 2 but we got by default
    print(name)
    print(age)
```

```
person('nit')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[56], line 5
      2     print(name)
      3     print(age)
----> 5 person('nit')

TypeError: person() missing 1 required positional argument: 'age'
```

In [57]:
```
def person(name,age=18):
    print(name)
    print(age)

person('nit')
```

```
nit
18
```

In [58]:
```
def person(name,age=18):
    print(name)
    print(age)

person('nit', age=24)
```

```
nit
24
```

In [59]:
```
def person(name,age=18):
    print(name)
    print(age)

person('nit')
```

```
nit
18
```

In [60]:
```
def person(name,age=18):
    print(name)
    print(age)
```

```
person('nit', age=40)
```

```
nit
40
```

2nd JULY 2025

Variable length

In [61]:
```
def sum(a, b):
    c = a+b
    print(c)

sum(5,6)
```

```
11
```

In [62]:
```
def sum(a, b):
    c = a+b
    return(c)

sum(5,6)
```

Out[62]: 11

In [63]:
```
def sum(a, b):
    c = a+b
    return(c)

sum(5,6,7,8,9)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[63], line 5
      2     c = a+b
      3     return(c)
----> 5 sum(5,6,7,8,9)

TypeError: sum() takes 2 positional arguments but 5 were given
```

In [64]:
```
def sum(a, *b):
    c = a+b
    return(c)

sum(5,6,7,8,9)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[64], line 5
      2     c = a+b
      3     return(c)
----> 5 sum(5,6,7,8,9)

Cell In[64], line 2, in sum(a, *b)
      1 def sum(a, *b):
----> 2     c = a+b
      3     return(c)

TypeError: unsupported operand type(s) for +: 'int' and 'tuple'
```

In [65]:
```
def sum(a, *b):  #1st argument is fixed but for 2nd argument
    #c = a+b
    print(type(a))
    print(type(b))

sum(5,6,7,8,9)
```

```
<class 'int'>
<class 'tuple'>
```

In [66]:
```
def sum(a, *b): # 1st argument is fixed & we fetch each value from the tuple & we can add them.
    c = a
```

```
    for i in b:
        c = c + i
    print(c)

sum(5,6,7,8)
```

```
26
```

kwargs

- KWARGs (key worded variable length arguments)
- Keyworded Variable Length Arguments

In [67]:
```
def person():
    person('Alex', 36, 'John', 987767)
```

In [68]:
```
def person(name,*data):
    print(name)
    print(data)

person('Alex', 36, 'John', 987767)

# hear what is name - is it southcit or AAA thats why we assigned keywords variable arguments
```

```
Alex
(36, 'John', 987767)
```

In [69]:
```
def person(name,*data):
    print('name')
    print(data)

person('Alex', age = 36, home_place ='southcity', mob = 987767)
# we got error as keyword argument thats why we add another *
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[69], line 5
      2     print('name')
      3     print(data)
----> 5 person('Alex', age = 36, home_place ='southcity', mob = 987767)

TypeError: person() got an unexpected keyword argument 'age'
```

In [70]:
```
def person(name,**data):
    print('name')
    print(data)

person('Alex', age = 36, home_place ='southcity', mob = 987767)
# we got error as keyword argument thats why we add another *
```

```
name
{'age': 36, 'home_place': 'southcity', 'mob': 987767}
```

3rd JULY 2025

LOCAL VARIABLE & GLOBAL VARIABLE

In [71]:
```
a = 10
print(a)
```

```
10
```

In [72]:
```
a = 10 # GLOBAL

def something():
    b = 15 # LOCAL VARIABLE

    print('in function',b)
    print('out function',a)
```

In [73]:
```
a = 10
```

```
def something():
    b = 15
    print('in function',b)

print('out function',a)
```

```
out function 10
```

In [75]:
```
a = 10

def something():
    b = 15
    print('in function',b)

something()

print('out function',a)
```

```
in function 15
out function 10
```

In [76]:
```
a = 10

def something():
    a = 15

print('in function',a)
print('out function',a)
```

```
in function 10
out function 10
```

In [80]:
```
a = 10

def something():
    b = 15

something()

print('in function',a)
print('out function',a)
```

```
in function 10
out function 10
```

In [81]:
```python
a = 10

def something():
    b= 25
    # if we remove this variable then can default it consider as global variable
    print('in function',b)

something()
print('out function',a)
# if we dont assign any variable inside the function by default both considered as local variable
```

```
in function 25
out function 10
```

In [82]:
```python
a = 10

def something():
    a = 55
    print('in function',a)

something()

print('out function',a)
```

```
in function 55
out function 10
```

In [84]:
```python
# if i want to define global variable inside the function
a = 10

def something():
    global a
    b = 15 # 15 is converted to local when user assigned global a
    print('in function',b)
    print('global variable',a)

something()

print('out function',a)
```

```
in function 15
global variable 10
out function 10
```

In [85]:
```python
# if i want to define global variable inside the function
a = 10

def something():
    global a
    b = 15 # 15 is converted to local when user assigned global a
    print('in function',b)
    print('global variable',a)
    a += 100
something()

print('out function',a)
```

```
in function 15
global variable 10
out function 110
```

In [87]:
```python
a = 10
def something():
    global a
    b = 15
    print('in function',b)
    a += 100
    print('global variable',a)
something()

print('out function',a)
```

```
in function 15
global variable 110
out function 110
```

In [ ]:
```python

```