Day 25 - Functions in Python

Functions are reusable blocks of code that perform a specific task. They help make code **modular**, **organized**, and **easy to debug**.

We use functions to avoid repetition and improve clarity in large programs.

In this notebook, I explored how to:

- Define and call functions
- Pass arguments to functions
- Return values from functions

In [1]:
```
def greet():
 print('Hello')
 print('Good Morning Team')
```

In [2]:
```
def greet():
 print("Hello")
 print("Good Morning Team")
greet()
```

```
Hello
Good Morning Team
```

In [3]:
```
def greet():
 print("Hello")
 print("Good Morning Team")
greet()
def greet():
 print("Hello")
 print("Good Morning Team")
greet()
```

```
Hello
Good Morning Team
Hello
Good Morning Team
```

In [4]:
```
def greet():
 print("Hello")
 print("Good Morning Team")
greet()

print()

def greet():
 print("Hello")
 print("Good Morning Team")
greet()
```

```
Hello
Good Morning Team

Hello
Good Morning Team
```

In [5]:
```
def greet():
 print("Hello")
 print("Good Morning Team")
greet()
print()

def greet():
 print("Hello")
```

```
 print("Good Morning Team")
greet()
print()

def greet():
 print("Hello")
 print("Good Morning Team")
greet()
print()

def greet():
 print("Hello")
 print("Good Morning Team")
greet()
```

```
Hello
Good Morning Team

Hello
Good Morning Team

Hello
Good Morning Team

Hello
Good Morning Team
```

Function Without Arguments

These are the simplest functions — they don't take any input from the user.

The function is defined with `def`, and its code runs only when you call it.

In [6]:
```
def greet(): #Declare function Without Argument
 print("Hello")
 print("Good Morning Team")
greet()
print('******')
greet()
print('******')
greet() #function calling without argument
```

```
Hello
Good Morning Team
******
Hello
Good Morning Team
******
Hello
Good Morning Team
```

In [7]:
```
#Function Without Argument
def greet():
 print("Hello")
 print("Good Morning Team")
greet()
```

```
Hello
Good Morning Team
```

Function With Arguments

Functions can accept **inputs/arguments** which let us pass data when we call them.

In [8]:
```
#Function With Argument
def add(x,y):
 c= x + y
```

```
 print(c)
add(8,9)
```

```
17
```

In [9]:
```
#Function With Argument
def add(x,y):
 c= x + y
 return c
print("Sum is:", add(8,9))
```

```
Sum is: 17
```

In [10]:
```
#Function With Argument
def add(x,y,z):
 c=x + y
 return c
add(5,6,7)
```

Out[10]: 11

In [11]:
```
#function with Argument
def add(x,y,z,n):
 c = x+y+z+n
 return c
add(5,6,7,8)
```

Out[11]: 26

In [12]:
```
def greet():
 print("Hello")
 print("Good Morning Team")
greet()

def add(x,y):
 c = x + y
 return c
add(8,9)
```

```
Hello
Good Morning Team
```

Out[12]: 17

In [13]:
```
def grret():
 print("Hello")
 print("Good Morning Team")

def add(x,y):
 c = x + y
 return c

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

greet()
print(add(7,5))
print(sub(9,5))
```

```
Hello
Good Morning Team
12
4
```

Function With Multiple Operations

Functions can do more than one task. For example, this function adds and subtracts two numbers in a single call.

In [14]:
```
def add_sub(x,y):
 c = x + y
 d = x - y
 return c, d

result= add_sub(7,5)
print(result)
print(type(result))
```

```
(12, 2)
<class 'tuple'>
```

In [15]:
```
def add_sub(x,y):
 c= x+y
 d= x-y
 return c, d

result, result1 = add_sub(7,5)
print(result)
print(result1)
print(type(result))
```

```
12
2
<class 'int'>
```

Function Returning Multiple Outputs

A function can return multiple values using `return`, separated by commas.

This is useful when you want to perform multiple operations and keep the result.

Function Returning Multiple Outputs

A function can return multiple values using `return`, separated by commas.

This is useful when you want to perform multiple operations and keep the result.

In [16]:
```
def add_sub_mul(x,y):
 c= x+y
 d= x-y
 e= x*y
 return c, d, e

add, sub, mul = add_sub_mul(4,5)
print('The addition of two numbers:', add)
print('The subtraction of two numbers:', sub)
print('The multiplication of two numbers:', mul)
```

```
The addition of two numbers: 9
The subtraction of two numbers: -1
The multiplication of two numbers: 20
```

Update

In [17]:
```
def update():
 x = 8
 print(x)
update()
```

```
8
```

In [18]:
```
def update(x):
 x = 8
```

```
 return(x)
update(100)
```

Out[18]: 8

In [19]:
```
def update(x):
 x = 8
 return x
a = 15
update(a)
print(a)
```

```
15
```