FUNCTION & THEIR TYPES

- USER DEFINED FUNCTION
- BUILT-IN FUNCTION

FUNCTION

A function is a block of code that performs a specific task.

Which take some input known as Input and return some Output ,known as the result.

A function can be **reused multiple times**, which makes the code more **modular and efficient**.

That’s why we call it reduce duplication(redundancy)and increase Reusability.

To create a function in Python, we use the **def** keyword, followed by the function name and a pair of parentheses. Inside the parentheses, we can optionally specify the parameters that the function expects. After the parentheses, we add a colon and then indent the body of the function.

Here is an example of creating a simple function that prints "Hello World!":

```
# Define the function
def greet():
    print("Hello World!")

# Call the function
greet()
```

To call a function, we use the function name followed by parentheses. If the function has parameters, we need to pass the arguments inside the parentheses

Syntax :

def function_name(parameter1,parameter2……,parameter n): → Defination of function

process

functioname() → Function Call

WAP using function to print your name:

```
def takename():
        name=str(input("Enter your name")
        print("Hello ",name)

takename()
```

Here,

takename(): is function name

name is an attribute /parameter

takename()-function call

WAP OF ADDTION USING FUNCTION

```
def Addition():#Default Function without parameter
    a=10
    b=30
    print("Addition :",a+b)
Addition()
```

O/P
Addition :40

OR

```
def Addition(a,b): #parameterized function
    print("Addition :",a+b)
Addition(10,30)
```

OR

```
def Addition():  #Default constructor using runtime value
        a=int(input("Enter value of a")
        b=int(input("Enter value of b")
        print("Addition :",a+b)
Addition()
```

O/P
Enter value of a
10
Enter value of b
30
Addition :40

BUILT-IN FUNCTIONS

Python built-in functions are **functions** that are always available in the Python interpreter and do not require any import statements.

They provide some common and useful functionality for various data types, objects, and operations

Some examples of Python built-in functions are:
abs(), len(), print(), type(), and zip()

BUILT-IN FUNCTIONS

Mathematical functions: These functions perform some mathematical operations on numbers, such as `abs()`, `pow()`, `round()`, and `divmod()`

```
# Use the abs() function to get the absolute value of a number eg:
x = -5
y = abs(x)
# o/p y is 5
```

Conversion functions: These functions convert one data type to another, such as int(), float(), str(), bool(), and complex()

Iterable functions: These functions operate on iterable objects, such as lists, tuples, sets, dictionaries, and strings. They can perform filtering, mapping, sorting, aggregating, or iterating operations, such as all(), any(), filter(), map(), sorted(), sum(), and enumerate()

Object-oriented functions: These functions deal with objects and their attributes, methods, and classes. They can create, modify, inspect, or delete objects, such as object(), classmehod(), staticmethod(), property(), getattr(), setattr(), delattr(), and hasattr()[1]

Reflection functions: These functions allow the user to access or manipulate the internal structure or behavior of the Python interpreter or code, such as globals(), locals(), dir(), id(), hash(), help(), eval(), and exec()[1]