Python Functions

Function is a group of related statements that perform a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

It avoids repetition and makes code reusable.

Syntax:

```
def function_name(parameters):

    """
    Doc String
    """

    Statement(s)
```

1. keyword "def" marks the start of function header
2. Parameters (arguments) through which we pass values to a function. These are optional
3. A colon(:) to mark the end of funciton header
4. Doc string describe what the function does. This is optional
5. "return" statement to return a value from the function. This is optional

Example:

In [5]:
```
def print_name(name):
    """
    This function prints the name
    """
    print("Hello " + str(name) )           # +" "+ str(name2))      # notice indentation here

    # return statement is optional... we will see how to use it
```

Function Call

Once we have defined a function, we can call it from anywhere

In [6]:
```
print_name('IOTA')
```

```
Hello IOTA
```

Doc String

The first string after the function header is called the docstring and is short for documentation string.

Although optional, documentation is a good programming practice, always document your code

Doc string will be written in triple quotes so that docstring can extend up to multiple lines

In [7]:
```
print(print_name.__doc__) # print doc string of the function  (double underscore)
```

```
    This function prints the name
```

In [8]:
```
print(print.__doc__)
```

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
```

return Statement

The return statement is used to exit a function and go back to the place from where it was called.

Syntax:

```
return [expression]
```

-> return statement can contain an expression which gets evaluated and the value is returned.

-> if there is no expression in the statement or the return statement itself is not present inside a function, then the function will return None Object

In [9]:
```
def get_sum(lst):
    """
    This function returns the sum of all the elements in a list
    """
    # initialize sum
    _sum = 0

    # iterating over the list
    for num in lst:
        _sum += num  # _sum = _sum+num
    return _sum
```

In [10]:
```
s = get_sum((1,2,3,4,5.5))
print(s)
```

```
15.5
```

In [11]:
```
s = get_sum([1, 2, 3, 4])
print(s)
```

```
10
```

In [12]:
```
# print doc string

print(get_sum.__doc__)
```

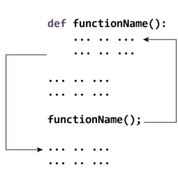
```
    This function returns the sum of all the elements in a list
```

In [13]:
```
def admin_portal(userid,userpass):
    if userid == 'iota' and userpass=='academy':
        print("Login Successful")
    else:
        print("Wrong Details")
```

In [15]:
```
x = input("Enter userid")
y = input("Enter password")
admin_portal(x,y)
print("I am learning")
```

```
Enter useridiota
Enter passwordacademy
Login Successful
I am learning
```

How Function works in Python?



Scope and Life Time of Variables

-> Scope of a variable is the portion of a program where the variable is recognized

-> variables defined inside a function is not visible from outside. Hence, they have a local scope.

-> Lifetime of a variable is the period throughout which the variable exits in the memory.

-> The lifetime of variables inside a function is as long as the function executes.

-> Variables are destroyed once we return from the function.

Example:

In [16]:
```
global_var = "This is global variable"

def test_life_time():
    """
    This function test the life time of a variables
    """
    local_var = "This is local variable"
    print(local_var)       # print local variable local_var

    print(global_var)      # print global variable global_var



# calling function
test_life_time()

# print global variable global_var
print(global_var)

# print local variable local_var
print(local_var)
```

```
This is local variable
This is global variable
This is global variable
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17772/2707600453.py in <module>
     19
     20 #print local variable local_var
---> 21 print(local_var)

NameError: name 'local_var' is not defined
```

Exercise:

Write Below programs as functions

1. Basic Calculator Function: Write a function which takes equation of two numbers and one operator (+,-, /, *) as input and return the correct answer.
    - Ex:- calculator(10,"+",20)
2. Factorial Function: WAP which takes integer as agrument and return the factorial of it as answer.
3. Fibonacci Function: WAP which takes nth_term as argument and return its value.
4. isPrime Function: For telling whether a number is prime or not
5. Python function to print Highest Common Factor (HCF) of two numbers
6. Python function to return all prime numbers as list between two given numbers. Example: 10,20 --->
    - expected output: [11,13,17,19]
7. Python function to print LCM of two numbers.

That's Great