IOTA ACADEMY

Become Job Ready...

Function Arguments

When we define and call a Python function, the term parameter and argument is used to pass information to the function.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

```
In [1]: def greet(name, msg):  # it takes exactly 2 arguments
            """
            This function greets to person with the provided message
            """
            print(f"Hello {name} , {msg}")

        # call the function with arguments
        greet( "Good Morning","IOTA")
```

```
Hello Good Morning , IOTA
```

```
In [3]: # suppose if we pass one argument

        greet("IOTA") # will get an error

        # how to avoid this type of errors? default arguments.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 3
      1 # suppose if we pass one argument
----> 3 greet("IOTA")

TypeError: greet() missing 1 required positional argument: 'msg'
```

Different Forms of Arguments

1. Positional Arguments

Positional arguments are those arguments where values get assigned to the arguments by their position when the function is called.

By default, Python functions are called using the positional arguments.

Example:

```
In [8]: def add(a, b):
            print(a - b)

        add(100, 20) # 100 assigned to a and 20 assigned to b
```

```
80
```

2. Default Arguments

We can provide a default value to an argument by using the assignment operator (=).

```
In [9]: def greet(name ,phonenumber,msg="Good Morning"):
            """
            This function greets to person with the provided message
            if message is not provided, it defaults to "Good Morning"
            """
            print(f"Hello {name} , {msg}")
            print(phonenumber)

        greet("IOTA","3455425673") # with out msg argument
```

```
Hello IOTA , Good Morning
3455425673
```

```
In [10]: greet("Hemant")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 greet("Hemant")

TypeError: greet() missing 1 required positional argument: 'phonenumber'
```

Once we have a default argument, all the arguments to its right must also have default values.

def greet(msg="Good Morning", name)

will get a SyntaxError : non-default argument follows default argument

3. Keyword Arguments

Keyword arguments are those arguments where values get assigned to the arguments by their keyword (name) when the function is called.

Example:

```
In [12]: def greet(name, msg):  # it takes exactly 2 arguments
             """
             This function greets to person with the provided message
             """
             print(f"Hello {name} , {msg}")

         # default function call (positional argumnets)
         greet("IOTA","Good Morning")

         # with keyword argument
         greet(name="IOTA",msg="Good Morning")

         greet(msg="Good Morning",name="IOTA")

         # 1 positional and 1 keyword
         greet('IOTA', msg="Good Morning")
```

```
Hello IOTA , Good Morning
Hello IOTA , Good Morning
Hello IOTA , Good Morning
Hello IOTA , Good Morning
```

4. Arbitrary arguments (variable-length arguments)

Sometimes, we do not know in advance the number of arguments that will be passed into a function. Python allows us to handle this kind of situation through function calls with arbitrary number of arguments.

A) arbitrary positional arguments (*args)

Example:

```
In [13]: def average(*numbers):   # Internally all these values are represented in the form of a tuple.
             addition = 0
             for i in numbers:
                 addition += i

             print(f"Average is {addition/len(numbers)}")
```

```
In [14]: average(1,2,3,4,5,6)
```

```
Average is 3.5
```

```
In [15]: def greet(msg,*names):
             """
             This function greets all persons in the names tuple
             """
             print(names)

             for name in names:
                 print(f"Hello,  {name}, {msg}")

         greet("thank you for coming","ranveer", "ranbir", "kartik", "nawaj")
```

```
('ranveer', 'ranbir', 'kartik', 'nawaj')
Hello,  ranveer, thank you for coming
Hello,  ranbir, thank you for coming
Hello,  kartik, thank you for coming
Hello,  nawaj, thank you for coming
```

B) arbitrary keyword arguments (**kwargs)

The **kwargs allow you to pass multiple keyword arguments to a function. Use the** kwargs if you want to handle named arguments in a function.

```
In [17]: def greet(**kwargs):
             """
             This function greets to person with the provided message
             """

             # kwargs are accessed using key-value pair (same as accessing a dictionary in Python).
             if kwargs:
                 print(f"Hello {kwargs['name']} , {kwargs['msg']}")


         greet(name="IOTA",msg="Good Night")
```

```
Hello IOTA , Good Night
```

Great Job!