Python Function Arguments

Variable Function Arguments

- Up until now functions had fixed number of arguments. In Python there are other ways to define a function which can take variable number of arguments.

Three different forms of this type are:

1. Python Default Arguments
2. Python Keyword Arguments
3. Python Arbitrary Arguments

1. Python Default Arguments

- Function arguments can have default values in Python.
- We can provide a default value to an argument by using the assignment operator (=). #### Here is an example:

In [2]:
```
def greet(name, msg = "Good Morning!"):
    """
    This function greetsunction_x(max=c, remote=a, filename=b) to
    the person with the
    provided message.

    If message is not provided,
    it defaults to "Good
    morning!"
    """

    print("Hello",name + ', ' + msg)

greet("Students")
greet("Students","Good Evening!")
```

```
Hello Students, Good Morning!
Hello Students, Good Evening!
```

- In this function, the parameter name does not have a default value and is required (mandatory) during a call.
- On the other hand, the parameter msg has a default value of "Good morning!". So, it is optional during a call. If a value is provided, it will overwrite the default value.
- Any number of arguments in a function can have a default value. But once we have a default argument, all the arguments to its right must also have default values.
- This means to say, non-default arguments cannot follow default arguments. For example, if we had defined the function header above as:

```
def greet(msg = "Good morning!", name):
```

We would get an error

```
In [3]: def greet(msg = "Good morning!", name):
           print("Hello",name + ', ' + msg)
```

```
  File "<ipython-input-3-0462dc30711a>", line 1
    def greet(msg = "Good morning!", name):
              ^
SyntaxError: non-default argument follows default argument
```

2. Python Keyword Arguments

- When we call a function with some values, these values get assigned to the arguments according to their position.
- For example, in the above function greet(), when we called it as greet("Students","Good Evening!"), the value "Students" gets assigned to the argument name and similarly "Good Evening!" to msg.
- Python allows functions to be called using keyword arguments. When we call functions in this way, the order (position) of the arguments can be changed. #### Example:

```
In [1]: def power(x,y):
            return x**y
```

```
In [2]: power(x=2,y=3)
```

```
Out[2]: 8
```

3. Python Arbitrary 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.
- In the function definition we use an asterisk (*)before the parameter name to denote this kind of argument.

In [ ]:
```
def greet(*names):
   """This function greets all
   the person in the names tuple."""

   # names is a tuple with arguments
   for name in names:
       print("Hello",name)

greet("Monica","Luke","Steve","John")
```

Here, we have called the function with multiple arguments. These arguments get wrapped up into a tuple before being passed into the function. Inside the function, we use a for loop to retrieve all the arguments back.