Function Arguments

In [3]:

```
def greet(name, msg):
    """
    This function greets to person with the provided message
    """
    print("Hello {0} , {1}".format(name, msg))

#call the function with arguments
greet("likhit","test")
```

```
Hello likhit , test
```

In [ ]:

```
#suppose if we pass one argument

greet("veera") #will get an error
```

Different Forms of Arguments

1. Default Arguments

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

In [7]:

```
def greet(msg="Good Morning",name):
    """
    This function greets to person with the provided message
    if message is not provided, it defaults to "Good Morning"
    """
    print("Hello {0} , {1}".format(name, msg))

greet("veera", "Good Night")
```

```
  File "<ipython-input-7-26cef80c4543>", line 1
    def greet(msg="Good Morning",name):
              ^
SyntaxError: non-default argument follows default argument
```

In [6]:

```
#with out msg argument
greet("likhit")
```

```
Hello likhit , Good Morning
```

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

2. Keyword Arguments

2. Keyword Arguments

kwargs allows you to pass keyworded variable length of arguments to a function. You should use **kwargs if you want to handle named arguments in a function

Example:

In [ ]:

```
def greet(**kwargs):
    """
    This function greets to person with the provided message
    """
    if kwargs:
        print("Hello {0} , {1}".format(kwargs['name'], kwargs['msg']))
greet(name="satish", msg="Good Morning")
```

3. Arbitary 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.

Example:

In [8]:

```
def greet(*names):
    """
    This function greets all persons in the names tuple
    """
    print(names)

    for name in names:
        print("Hello,  {0} ".format(name))

greet("satish", "murali", "naveen", "srikanth")
```

```
('satish', 'murali', 'naveen', 'srikanth')
Hello,  satish
Hello,  murali
Hello,  naveen
Hello,  srikanth
```

In [ ]:

```
#Few More Examples
```

In [15]:

```
def myfunc(*mul_arg):
    print (type(mul_arg))
    print(mul_arg)

tuple_vec = (1, 0, 1,)
dict_vec = {'x': 1, 'y': 0, 'z': 1}


myfunc(tuple_vec)
myfunc(dict_vec)
```

```
<class 'tuple'>
((1, 0, 1),)
<class 'tuple'>
({'x': 1, 'y': 0, 'z': 1},)
```

In [3]:

```
def myfunc(x, y, z):
    print(x, y, z)

tuple_vec = (1, 0, 1,)
dict_vec = {'x': 1, 'y': 0, 'z': 1}


myfunc(*tuple_vec)
myfunc(**dict_vec)
```

```
1 0 1
1 0 1
```