In [1]:
```
# def function_name(<parameter list>):
#     """ Docstring """
#     statements
#     return value
```

In [3]:
```
def greet():
    print("Hello World.")
```

In [5]:
```
# None
```

In [7]:
```
greet()
```

```
Hello World.
```

In [9]:
```
def greet(name):
    return "Hello" + " " + name
```

In [11]:
```
greet("XYZ")
```

Out[11]:
```
'Hello XYZ'
```

In [13]:
```
def volume(l,b,h):
    return l * b * h
```

In [15]:
```
volume(10,15,20)
```

Out[15]:
```
3000
```

In [17]:
```
l1 = [10,20,30,40,10,10,10,20,20,30,40,50,60,60,50]
```

In [19]:
```
def frequency_count(l):
    d = {}
    for i in l:
        if i not in d:
            d[i] = 1
        else:
            d[i] += 1
    return d
```

In [21]:
```
frequency_count(l1)
```

Out[21]:
```
{10: 4, 20: 3, 30: 2, 40: 2, 50: 2, 60: 2}
```

In [23]:
```
def calc(a,b):
    return [a+b, a-b, a*b, a/b]
```

In [25]:
```
calc(20,10)
```

Out[25]:
```
[30, 10, 200, 2.0]
```

In [27]:
```
def calc(a,b):
    return a+b, a-b, a*b, a/b
```

In [29]:
```
calc(20,10)
```

```
Out[29]:  (30, 10, 200, 2.0)
```

```
In [31]: t = (1,2,3,4,5)
```

```
In [33]: def cube(x):
             return [i**3 for i in x]
```

```
In [35]: cube(t)
```

```
Out[35]:  [1, 8, 27, 64, 125]
```

```
In [37]: def factors(num):
             return [i for i in range(1,num+1) if num % i == 0]
```

```
In [39]: def isPrime(num):
             for i in range(2,num):
                 if num % i == 0:
                     return False
             else:
                 return True
```

```
In [41]: isPrime(5)
```

```
Out[41]:  True
```

Types of Function Arguments

```
In [44]: def fun1(a,b,c):
             print(a,b,c)
```

Positional Arguments

```
In [47]: fun1(2,3,1)
```

```
2 3 1
```

Keyword Arguments

```
In [50]: fun1(c=3,b=2,a=1)
```

```
1 2 3
```

```
In [52]: fun1(1,b=2,c=3)
```

```
1 2 3
```

```
In [54]: fun1(1,c=3,2)
```

```
  Cell In[54], line 1
    fun1(1,c=3,2)
                ^
SyntaxError: positional argument follows keyword argument
```

```
In [58]: fun1(1,2,c=3)
```

```
1 2 3
```

```
In [60]: fun1(1,a=1,c=3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[60], line 1
----> 1 fun1(1,a=1,c=3)

TypeError: fun1() got multiple values for argument 'a'
```

Defualt Arguments

```
In [63]: def addition(a,b,c=0):
             return a+b+c
```

```
In [65]: addition(1,2)
```

```
Out[65]: 3
```

```
In [67]: addition(1,2,3)
```

```
Out[67]: 6
```

```
In [69]: def addition(a,b=0,c=0):
             return a+b+c
```

```
In [71]: def addition(a=0,b=0,c=0):
             return a+b+c
```

variable length positional argument

```
In [73]: t1 = 1,2,3,4,5,6
```

```
In [75]: t1
```

```
Out[75]: (1, 2, 3, 4, 5, 6)
```

```
In [77]: a,b,*c = t1
```

```
In [79]: a
```

```
Out[79]: 1
```

```
In [81]: b
```

```
Out[81]: 2
```

```
In [83]: c
```

```
Out[83]: [3, 4, 5, 6]
```

In [85]:
```
def fun1(*args):
    print(args)
    print(type(args))
```

In [89]:
```
fun1()
```

```
()
<class 'tuple'>
```

In [91]:
```
fun1(1)
```

```
(1,)
<class 'tuple'>
```

In [93]:
```
fun1(1,2,3,4,5)
```

```
(1, 2, 3, 4, 5)
<class 'tuple'>
```

In [95]:
```
fun1(1,2,3,4,c=5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[95], line 1
----> 1 fun1(1,2,3,4,c=5)

TypeError: fun1() got an unexpected keyword argument 'c'
```

variable length keyword argument

In [98]:
```
def fun3(**kwargs):
    print(kwargs)
    print(type(kwargs))
```

In [100…
```
fun3(a=1, b=2, c=3, d=4)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
<class 'dict'>
```

positional only argument

In [103…
```
def fun4(a,b,c,d,/):
    print(a,b,c,d)
```

In [105…
```
fun4(1,2,3,4)
```

```
1 2 3 4
```

keyword only argument

In [108…
```
def fun5(*,a,b,c,d):
    print(a,b,c,d)
```

In [110…
```
fun5(a=1,b=2,c=3,d=4)
```

```
1 2 3 4
```

In [112…
```
def fun6(a,b,/,c,d,*,e,f):
    print(a,b,c,d,e,f)
```

In [114…
```
fun6(1,2,3,d=4,e=5,f=6)
```

```
1 2 3 4 5 6
```

In [116…
```
def fun(a,b,*args,c,d,**kwargs):
    print(a,b,args,c,d,kwargs)
```

positional -> variable length positional -> keyword -> variable length keyword

In [119…
```
fun(1,2,c=3,d=4)
```

```
1 2 () 3 4 {}
```

In [121…
```
fun(1,2,3,4,5,6,d=7,c=8,e=9,f=10)
```

```
1 2 (3, 4, 5, 6) 8 7 {'e': 9, 'f': 10}
```

lambda expression/function

In [124…
```
# lambda arguments : expression
```

In [126…
```
def double(x):
    return x*2
```

In [128…
```
db = lambda x : x*2
```

In [130…
```
db(7)
```

Out[130…
```
14
```

In [132…
```
(lambda x : x*3)(7)
```

Out[132…
```
21
```

In [134…
```
add = lambda x,y : x + y
```

In [136…
```
add(10,20)
```

Out[136…
```
30
```

In [138…
```
add("Hello","students")
```

Out[138…
```
'Hellostudents'
```

In [140…
```
li = [18,12,16,78,11]
```

In [142…
```
li.sort()
```

In [144…
```
li
```

```
Out[144…  [11, 12, 16, 18, 78]

In [146…  sorted(li, reverse = True, )

Out[146…  [78, 18, 16, 12, 11]

In [148…  li

Out[148…  [11, 12, 16, 18, 78]

In [150…  l1 = ["cat","apple","Catch","Ball","Dog"]

In [152…  sorted(l1, key = str.upper)

Out[152…  ['apple', 'Ball', 'cat', 'Catch', 'Dog']

In [154…  l2 = [(105,"Riya"), (103,"Ajay"), (101,"Rahul"), (110,"Charmi"), (106,"Bina")]

In [156…  sorted(l2)

Out[156…  [(101, 'Rahul'), (103, 'Ajay'), (105, 'Riya'), (106, 'Bina'), (110, 'Charmi')]

In [158…  sorted(l2, key = lambda x: x[1] )

Out[158…  [(103, 'Ajay'), (106, 'Bina'), (110, 'Charmi'), (101, 'Rahul'), (105, 'Riya')]

In [160…  student = dict([(105,"Riya"), (103,"Ajay"), (101,"Rahul"), (110,"Charmi"), (106,

In [162…  student

Out[162…  {105: 'Riya', 103: 'Ajay', 101: 'Rahul', 110: 'Charmi', 106: 'Bina'}

In [164…  student = dict(sorted(student.items(), key = lambda x : x[1]))

In [166…  student

Out[166…  {103: 'Ajay', 106: 'Bina', 110: 'Charmi', 101: 'Rahul', 105: 'Riya'}
```

map(function , iterable)

```
In [169…  li = [1,2,3,4,5,6]

In [171…  list(map(str,li))

Out[171…  ['1', '2', '3', '4', '5', '6']

In [173…  l2 = ["python","is","fun","at","programming"]

In [175…  list(map(len,l2))

Out[175…  [6, 2, 3, 2, 11]

In [177…  l3 = list(range(1,6))
In [179…  list(map(lambda x:x**3, l3))

Out[179…  [1, 8, 27, 64, 125]
```

filter(function, iterable)

```
In [182…  l1 = ["abc","aba","liril","python","lambda"]
```

```
In [184…  def isPalndrome(s):
              return s == s[::-1]
```

```
In [186…  list(filter(isPalndrome,l1))

Out[186…  ['aba', 'liril']
```

```
In [188…  tuple(filter(lambda s: s == s[::-1],l1))

Out[188…  ('aba', 'liril')
```

```
In [190…  def outer():
              def inner():
                  print("Inside the function.")
              inner()
```

```
In [192…  x = 10

          def change():
              global x
              x = 20

          change()
          print(x)

         20
```

```
In [194…  outer()

         Inside the function.
```

```
In [196…  inner()

         ---------------------------------------------------------------------------
         NameError                                 Traceback (most recent call last)
         Cell In[196], line 1
         ----> 1 inner()

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

DOCSTRING

```
In [198…  def add(a,b):
              """This function takes two objects and returns their addition"""
              return a+b #adds two numbers/strings
```

```
In [200…  add.__doc__
```

Out[200…

```
'This function takes two objects and returns their addition'
```

In [202…

```python
help(add)
```

```
Help on function add in module __main__:

add(a, b)
    This function takes two objects and returns their addition
```

pass by object reference in Python

In [231…

```python
def modify(l):  #mutable object passed, so behave like pass by reference
    l.append(100)
```

In [233…

```python
l1 = [1,2,3,4]
modify(l1)
print(l1)
```

```
[1, 2, 3, 4, 100]
```

In [235…

```python
def modify_data(x): #immutable object passed, so behave like pass by value
    x = x+10
```

In [237…

```python
a = 20
modify_data(a)
print(a)
```

```
20
```

In [239…

```python
def demo(l):
    print("Inside the function :",id(l))
```

In [241…

```python
l1 = [1,2,3]
print("Outside the function :",id(l1))
demo(l1)
```

```
Outside the function : 1902926095104
Inside the function : 1902926095104
```

In [243…

```python
x = 10
print("Outside the function :",id(x))
demo(x) #both are same because value is not changed yet
```

```
Outside the function : 140729994324696
Inside the function : 140729994324696
```

In [245…

```python
def demo(l):
    l *= 10
    print("Inside the function :",id(l))
```

In [247…

```python
x = 10
print("Outside the function :",id(x))
demo(x) #both are different because value is changed inside the function
```

```
Outside the function : 140729994324696
Inside the function : 140729994327576
```