Functions in Python

> A function in any language is similar to that in mathematics
>
> **Defination:**
>
> A function is a block of code executing specified task only when it is called and return some result.

1. Defining Function

```
def function_name(parameters):
        statments(s)
```

'parameters' and 'return' in a function are optional

Parameter or argument?

The terms 'parameter' and 'argument' are used for same, information passed in the function

1.1 with no parameters and no return type:

```
def function_name():
    statments()
```

In [2]:
```
def hello():     #    its work initiated
```

```
    print("hello")#           ^
          #                   |
hello()   # Here we called the function with its name and there
```

```
hello
```

1.2 with one parameter and no return type:

```
def function_name(param1):
    stat(param1)
```

In [3]:
```
def hello1(param1):
    print(param1)

hello1("hello1")

## parameter can be of any data type

def listFunction(l):
    print(l)
a = [2,4,53]
listFunction(a)
```

```
hello1
[2, 4, 53]
```

But remember one thing the number of parameters should be equal on both calling time and defination time

In [4]:
```
def misMatch(a,b,c):
    print(a,b,c)
misMatch(4,5)
```

```
---------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
```

```
ast)
<ipython-input-4-9f4977e4a5f3> in <module>
      1 def misMatch(a,b,c):
      2     print(a,b,c)
----> 3 misMatch(4,5)

TypeError: misMatch() missing 1 required positional argument: 'c'
```

In [5]:
```
def misMatch(a,b):
    print(a,b)
misMatch(4,5,6)
```

```
---------------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-5-2e92f08658cf> in <module>
      1 def misMatch(a,b):
      2     print(a,b)
----> 3 misMatch(4,5,6)

TypeError: misMatch() takes 2 positional arguments but 3 were given
```

function with default value of the parameter

Benefit: if you do not pass any value to the fuction, then it will assign the default value

In [6]:
```
def hello2(param1='siddhesh'):
    print(param1)
hello2()
hello2("hello2")
```

```
siddhesh
hello2
```

1.2.1 with multiple parameter and no return type:

```
def function_name(*param):
    statements
```

This type of argumennts/parameters are also known as **Arbitrary arguments, *args**

In [7]:
```
def names(*name):
    print(name)
    print(type(name))
names('rina','josaph','dev')
names('rina',1,2)
```

```
('rina', 'josaph', 'dev')
<class 'tuple'>
('rina', 1, 2)
<class 'tuple'>
```

In [8]:
```
def multiPara(*name):
    for i in name:
        print(i)
multiPara('Tony','Steave','Haimdal')
```

```
Tony
Steave
Haimdal
```

In [ ]:
```
```

In [9]:
```
def names1(hello,*name):
    for i in name:
        print(hello,i)
names1("hello",'rina','ramesh','kishan')
#       hello <-----name------------>
names1('rina','ramesh','kishan',1)
#       hello <-----name--------->
```

```
hello rina
hello ramesh
hello kishan
```

```
rina ramesh
rina kishan
rina 1
```

In [ ]:

Lets check what a function returns when no **return** statement is defiend

In [11]:
```
def call():
    pass
print(call())
```

```
None
```

1.3 with no parameter and a return type

```
def function_name():
    stat()
    return x,y,z...or list,tuple... or function(recursion)
```

In [13]:
```
def hello3():
    return "hello3"

def hello4_5():
    return "hello4","hello5"

def hello_list():
    return [1,2,3,4]



print(hello3())
print(hello4_5())
#or
x,y=hello4_5()      # Must remember one thing that the number of return
```

```
ed values should be
                    # equal to the number of variables at calling sid
e.
print(x,y)
print(hello_list())
```

```
hello3
('hello4', 'hello5')
hello4 hello5
[1, 2, 3, 4]
```

In [14]:
```
def misMatch_return():
    return 1,2,3
a,b = misMatch_return()
```

```
---------------------------------------------------------------------
----
ValueError                                Traceback (most recent call l
ast)
<ipython-input-14-b10f647144ad> in <module>
      1 def misMatch_return():
      2     return 1,2,3
----> 3 a,b = misMatch_return()

ValueError: too many values to unpack (expected 2)
```

1.4 with one param and a return type:

```
def function_name(param1):
    state()
    return x,y,z... as above
```

In [15]:
```
def hello6(hello6):
    return hello6

hello6("hello6")
```

Out[15]: 'hello6'

Positional parameter

In [19]:
```
def sum_(a=5,b=7):
    print("a=",a,"b=",b)
    return a+b
print(sum_(10,11))
print(sum_(a=5,b=6))
print(sum_(b=5,a=6))
print(sum_(0,6))
print(sum_())
```

```
a= 10 b= 11
21
a= 5 b= 6
11
a= 6 b= 5
11
a= 0 b= 6
6
a= 5 b= 7
12
```

Arbitrary keyword arguments **kwargs

= Arbitrary arg + keyword arg

In [21]:
```
def ArKey(**student):
    print(student)
ArKey(fname = 'siddhesh', lname = 'jain',Branch = 'IT')
```

```
{'fname': 'siddhesh', 'lname': 'jain', 'Branch': 'IT'}
```

In [ ]:

Recursion Function:

```
def func_name(param):
    stat()
    return func_name(param)
```

In [24]:
```
def factorial(number):
    if number==1:
        return 1
    else:
        return number*factorial(number-1)
print(factorial(5))

def sum_of_n(n):
    if n==0:
        return 0
    else:
        return n+sum_of_n(n-1)
factorial(sum_of_n(3))
```

```
120
```

Out[24]:
```
720
```

In [ ]:
```
5 *f(4)
   4*f(3)
     3*f(2)
        2*f(1)
          1
```

In [28]:
```
def s():
    return 1,2,3
l = []
l = list(s())
```

In [29]:
```
l
```

Out[29]:
```
[1, 2, 3]
```

In [ ]:
```

```

In [40]:
```
def fib(n):
    a = 0
    b = 1
    if n == 1:
        return 0
    if n == 2:
        return [0,1]
    lst = []
    for i in range(n):
        lst.append(a)
        c = a+b
        a = b
        b = c
    return lst

print(fib(5))
```

```
[0, 1, 1, 2, 3]
```

0 1 1 2 3 5 8 13 1 2 3.....

In [ ]:
```

```

In [41]:
```
def fib(n):
    if n < 3:
        return n
    else:
        return fib(n-1),fib(n-2)
fib(5)
```

Out[41]:
```
(((2, 1), 2), (2, 1))
```

In [ ]: