Functions

In [3]:

```
#Defining a function
def test():
```

```
  File "C:\Users\harsh\AppData\Local\Temp\ipykernel_10164\3938161042.py", line
2
    def test():
              ^
IndentationError: expected an indented block
```

In [4]:

```
# Use pass to not get error
def test():
    pass
```

In [5]:

```
#Print using Function
def test():
    print("Harshith")
test()
```

```
Harshith
```

In [6]:

```
# Concatination with functions output
test()+" Varma"
```

```
Harshith
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_10164\4230364662.py in <module>
      1 # Concatination with functions output
----> 2 test()+" Varma"

TypeError: unsupported operand type(s) for +: 'NoneType' and 'str'
```

In [7]:

```
#use return to get it in str type
def test():
    return "Harshith"
test()+" Varma"
```

Out[7]:

```
'Harshith Varma'
```

In [8]:

```
# Return multiple items
def test():
    return 1,2.0,'Three',True,[1,2,3],(4,5,6),{7,8,9},{1:"hhjj"}
test()
```

Out[8]:

```
(1, 2.0, 'Three', True, [1, 2, 3], (4, 5, 6), {7, 8, 9}, {1: 'hhjj'})
```

In [9]:

```
# Addition
def add():
    return 1+2
add()
```

Out[9]:

```
3
```

In [10]:

```
# Pass own Arguments
def add(a,b):
    return a+b
add(1,4)
```

Out[10]:

```
5
```

In [11]:

```
# set default values as arguments
def add(a=1,b=10):
    return a+b
add()
```

Out[11]:

```
11
```

In [12]:

```
add(16)
```

Out[12]:

```
26
```

In [13]:

```
add(9,7)
```

Out[13]:

```
16
```

In [14]:

```
# Use strings for adding
def add(a,b):
    return a+b
add("Harshith"," Varma")
```

Out[14]:

```
'Harshith Varma'
```

In [15]:

```
add(b=" Varma",a="Harshith")
```

Out[15]:

```
'Harshith Varma'
```

In [16]:

```
# Finding numbers in given List
l=[1,2,3,"Harshith","Varma",[9,8,7]]
```

In [17]:

```
def test(a):
    l=[]
    n=[]
    for i in a:
        if type(i)==list:
            for j in i:
                if type(j)==int or type(j)==float:
                    n.append(j)
            l.append(n)
            n=[]
        elif type(i)==int or type(i)==float:
            l.append(i)
    return l

test(l)
```

Out[17]:

```
[1, 2, 3, [9, 8, 7]]
```

In [18]:

```
l=[1,2,[2.2,55.34,"hqbw"],True,[33.33,False],"hsuwus","fienwfnf",[56,75]]
```

In [19]:

```
test(l)
```

Out[19]:

```
[1, 2, [2.2, 55.34], [33.33], [56, 75]]
```

In [20]:

```
a,b,c,d,e=test(l)
```

In [21]:

```
a
```

Out[21]:

```
1
```

In [22]:

```
b
```

Out[22]:

```
2
```

In [23]:

```
c
```

Out[23]:

```
[2.2, 55.34]
```

In [24]:

```
d
```

Out[24]:

```
[33.33]
```

In [25]:

```
e
```

Out[25]:

```
[56, 75]
```

In [26]:

```
# How to run without passing arguments
def test(a,b):
    pass
test()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_10164\3035369399.py in <module>
      2 def test(a,b):
      3     pass
----> 4 test()

TypeError: test() missing 2 required positional arguments: 'a' and 'b'
```

In [27]:

```
# Use *args
def test(*args):
    return args
test()
```

Out[27]:

```
()
```

In [28]:

```
type(test())
```

Out[28]:

```
tuple
```

In [29]:

```
test(1,2,3,"hudahadihid",True)
```

Out[29]:

```
(1, 2, 3, 'hudahadihid', True)
```

In [30]:

```
# Find the lists in any given number of arguments
def test(*args):
    l=[]
    for i in args:
        if type(i)==list:
            l.append(i)
    return l
test(1,[2,3,4],"jsjs",["jsjs","jsksk"])
```

Out[30]:

```
[[2, 3, 4], ['jsjs', 'jsksk']]
```

In [31]:

```
# Take arguments as Dictionaries
def test(**kwargs):
    return kwargs
test(a=1,b=2,c=3,d=4)
```

Out[31]:

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

In [32]:

```
# Use both *args and **kwargs at the same time.
def test(*args,**kwargs):
    return args,kwargs
test(1,2,3,a=1,b=2,c=3)
```

Out[32]:

```
((1, 2, 3), {'a': 1, 'b': 2, 'c': 3})
```

Generator Functions

In [33]:

```
# Fibanocci Series
def test(n):
    a,b=0,1
    for i in range(n):
        yield a
        a,b=b,a+b
test(10)
```

Out[33]:

```
<generator object test at 0x000001C59310C5F0>
```

In [34]:

```
for i in test(20):
    print(i)
```

```
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
```

In [ ]: