7) FUNCTION

In [2]:

```
def _1st_function():
    print("this is the 1st function")
```

In [3]:

```
_1st_function()
```

```
this is the 1st function
```

In [7]:

```
#concatining function with a string
_1st_function() + "fun"
# not possible as inside function there is a print function which returns none type
```

```
this is the 1st function

--------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_5660\1961727740.py in <module>
      1 #concatining function with a string
----> 2 _1st_function() + "fun"
      3 # not possible as inside function there is a print function which
returns none type

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

return

In [10]:

```
def _2nd_fun():
    return "this is my fun with return"
#return will return the data as it is and not as none type as in case of print()
```

In [9]:

```
print(_2nd_fun()+"asit")
```

```
this is my fun with returnasit
```

In [11]:

```
#returning multiple values
def fun3():
    return 1, 4, "pwskills" , 34.56
```

In [22]:

```
print(fun3())
print(type(fun3()))
#note- return return muptile values as a tuple

#slicing
print(fun3()[2])
```

```
(1, 4, 'pwskills', 34.56)
<class 'tuple'>
pwskills
```

In [20]:

```
#holding multiple data by a variable
a=1,2,3,4
print(a)
print(type(a)) #hold values as a tuple

#multiplle varible assignment
a,b,c,d=1,2,3,4
print(a,b,c,d)
```

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

In [23]:

```
#create a sum function
def add(a,b):
    c= a+b
    return c
```

In [24]:

```
add(1,2)
```

Out[24]:

```
3
```

In [26]:

```
add("asit","shastri")
```

Out[26]:

```
'asitshastri'
```

In [27]:

```
add([1,2,3,4] , [4,5,6,7,8])
```

Out[27]:

```
[1, 2, 3, 4, 4, 5, 6, 7, 8]
```

In [29]:

```
#pass named parameter in a function
add(b="asit",a="shastri") #observe no need of to order the argument
```

Out[29]:

```
'shastriasit'
```

In [41]:

```
"""
create a function which will take list as a input and give me a
final list with all the numeric value
"""

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

In [42]:

```
l = [1,2,3,4,5,"sudh" , "pwskills" , [1,2,3,34,45]]
lst_filter(l)
```

Out[42]:

```
[1, 2, 3, 4, 5, 1, 2, 3, 34, 45]
```

(*args)- to create a function that can take any no. of arguments

In [43]:

```
def fun4(a,b,c,d,e):
    pass

fun4(1,2,3,4,5,67) #this function can only take 5 arguments
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_5660\2465101979.py in <module>
      2     pass
      3 #this function can only take 5 arguments
----> 4 fun4(1,2,3,4,5,67)

TypeError: fun4() takes 5 positional arguments but 6 were given
```

In [44]:

```
def fun5(*args):
    pass
fun5(1,2,3,4,5,6,7,7,8,9)
```

In [45]:

```
#or
def fun6(*asit):
    pass
fun6(20,85,64,"string")
```

In [46]:

```
#fun with * and one other argument
def fun7(*args , a ):
    return args ,a
fun7(3)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_5660\331099969.py in <module>
      2 def fun7(*args , a ):
      3     return args ,a
----> 4 fun7(3)

TypeError: fun7() missing 1 required keyword-only argument: 'a'
```

In [48]:

```
fun7(1,2,3,4,5,a=3)
```

Out[48]:

```
((1, 2, 3, 4, 5), 3)
```

In [51]:

```
#creating a function that checks arg and returns a list
def fun8(*args):
    l = []
    for i in args:
        if type(i) == list :
            l.append(i)

    return l

fun8(1,2,3,[1,2,3,4,4] , (1,2,3,4,4) , "sudh" , [4,5,6] , [6,7,8])
```

Out[51]:

```
[[1, 2, 3, 4, 4], [4, 5, 6], [6, 7, 8]]
```

In [53]:

```
#function with key and value as argument
def fun9(**kwargs): #double astrix means it will take all arguments as key-value pair
    return kwargs
print(fun9)
print(type(fun9()))
```

```
<function fun9 at 0x000002063D063790>
<class 'dict'>
```

In [55]:

```
#function that returns key its value whiich is list
def fun10(**kwargs):
    for i in kwargs.keys():
        if type(kwargs[i] ) == list :
            return i , kwargs[i]
print(fun10(a = 34 , b = 23 , c = [1,2,3,4] , d = ("sudh" , "pwskills")))
print(type(fun10()))
```

```
('c', [1, 2, 3, 4])
<class 'NoneType'>
```

In [62]:

```
#functions that take any argument and both key-valye pairs as input
def fun11(*args , **kwargs) :
    return args , kwargs

fun11(2,3,4,5,a= 34, b = 98)
```

Out[62]:

```
((2, 3, 4, 5), {'a': 34, 'b': 98})
```

In [64]:

```
type(fun11()) #tuple of arg and key value pairs
```

Out[64]:

```
tuple
```

7.2) Generator Function

the range() function is a generating function as it work on itself. It can be only used inside the for loop

> So,How to produce this typr of function?

In [1]:

```
range(1,10)
```

Out[1]:

```
range(1, 10)
```

In [3]:

```
for i in range(1,10): #range()only works inside for loop
    print(i)
```

```
1
2
3
4
5
6
7
8
9
```

What is the advantage of Generating function?

> As in ML we work on millions of data. Function which gives the filnal list as an outcome will only not communicate unless it prepares the whole list which will take a long time cuz of billions of enteries and thus creating a bottleneck.
>
> So a function which shows only when we try to iterate over it (like range) will be helpfull. as it does not remember the whole data it only remembers the last data it generated.

usning -Yield- to make a generator function

note using generator function we generate data in an optimised way by not blocking the whole memory

In [66]:

```
#Q) Create a Generator function whi produces a fibonacci series
```

In [72]:

```
def fibo_gen(n):
    a,b=0,1
    for i in range(n):
        yield a #passes the value of a without storing it
        a,b=b,a+b
```

In [73]:

```
fibo_gen(1000) #as we can see it does not create a list of thousand finachi nos.
```

Out[73]:

```
<generator object fibo_gen at 0x000002063D19A7B0>
```

In [79]:

```
lst=[]
for i in fibo_gen(100):
    if i==1134903170:
        break
    else: lst.append(i)
print(lst)
print(len(lst))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2
584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 31781
1, 514229, 832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352,
24157817, 39088169, 63245986, 102334155, 165580141, 267914296, 433494437,
701408733]
45
```

7.3) Lambda Function

In [3]:

```
#creating a function that returns power of a no.
def pwr(m,p):
    return m**p

print(pwr(2,4))
```

```
16
```

Adhoc Function (below)

In [4]:

```
# Storing lambda finction inside a variable

a= lambda n,p:n**p  #note lambda finction can be assigned to a variable

print(a(5,3))
```

```
125
```

In [5]:

```
add=lambda x,y:x+y
```

In [6]:

```
c_2_f = lambda c:(9/5)*c+32
```

In [7]:

```
 max_no = lambda x,y:c if x>y else x
```

In [8]:

```
max(4,7)
```

Out[8]:

```
7
```

7.4) Map, Reduce & Filter Function

7.4.1) Map Function

In [9]:

```
#Create a function that squares every element of the list below
l = [1,2,3,4,45,5]

#solution
def sqr(l):
    l1=[]
    for i in l:
        l1.append(i**2)
    return l1
print(sqr(l))
```

```
[1, 4, 9, 16, 2025, 25]
```

Map Function Approach
it take functions and iterables as argument, and iterates all elements of the iterable through the function

In [14]:

```
#solving above problem using map function
def sqr2(l):
    return l**2
```

In [15]:

```
list(map(sqr2,l))
```

Out[15]:

```
[1, 4, 9, 16, 2025, 25]
```

In [16]:

```
#mapper function with a lambda function
list(map(lambda x:x**2,l))
```

Out[16]:

```
[1, 4, 9, 16, 2025, 25]
```

In [17]:

```
list(map(lambda x : x+10 , l ))
```

Out[17]:

```
[11, 12, 13, 14, 55, 15]
```

In [18]:

```
list(map(lambda x : str(x) , l ))
```

Out[18]:

```
['1', '2', '3', '4', '45', '5']
```

In [19]:

```
l1 = [1,2,3,4,5]
l2 = [6,7,8,9,10]
```

In [20]:

```
list(map(lambda x,y :x+y , l1,l2))
```

Out[20]:

```
[7, 9, 11, 13, 15]
```

In [22]:

```
#using adhoc function inside map()
l1 = [1,2,3,4]
l2 = [6,7,8,9,10]
f= lambda x,y:x+y  #adhoc function
list(map(f,l1,l2))
```

Out[22]:

```
[7, 9, 11, 13]
```

7.4.2) Reduce Function

By default reduce function is not available in python. It needs to be imported library "functools"

In [23]:

```
from functools import reduce
```

In [24]:

```
l = [1,2,3,4,5,4]
reduce( lambda x,y:x+y,l) #it take 1st arg. as function and 2nd as iterable
```

Out[24]:

```
19
```

Type *Markdown* and LaTeX: α^2

In [27]:

```
reduce(lambda x , y , z : x+y+z , l)
#does not work as explained in below img
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_8776\3759722890.py in <module>
----> 1 reduce(lambda x , y , z : x+y+z , l)
      2 #does not work as explained in below img

TypeError: <lambda>() missing 1 required positional argument: 'z'
```

Type *Markdown* and LaTeX: α^2

In [28]:

```
reduce(lambda x , y : x+y , [])
#passing empty iterable inside reduce doesnt work
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_8776\279108227.py in <module>
----> 1 reduce(lambda x , y : x+y , [])
      2 #passing empty iterable inside reduce doesnt work

TypeError: reduce() of empty sequence with no initial value
```

In [29]:

```
reduce(lambda x , y : x+y , [1])
```

Out[29]:

```
1
```

In [34]:

```
#using reduce function calucate max no. element inside a list
l = [1,2,3,4,5,4]
reduce( lambda x,y: x if x>y else y,l)
```

Out[34]:

```
5
```

Type *Markdown* and LaTeX: α^2

7.4.3) Filter Function

- takes function and iterable as argument
- print the element of iterable if condition is true

In [35]:

```
#Filter even nos. from list below
l = [1,2,3,4,5,4]
list(filter( lambda x:x%2==0,l))
```

Out[35]:

```
[2, 4, 4]
```

In [36]:

```
#Filter odd nos. from list below
l = [1,2,3,4,5,4]
list(filter( lambda x:x%2!=0,l))
```

Out[36]:

```
[1, 3, 5]
```

In [38]:

```
#Filter -ve nos. from list below
l1 = [-2,4,5,6,-3,-6,-7]
list(filter( lambda x:x<0,l1))
```

Out[38]:

```
[-2, -3, -6, -7]
```

In [39]:

```
#Filter +ve nos. from list below
l1 = [-2,4,5,6,-3,-6,-7]
list(filter( lambda x:x>0,l1))
```

Out[39]:

```
[4, 5, 6]
```

In [41]:

```
#Filter strings with length < 6 from list below
l2 = ["sudh" , "pwskills" , "kumar" , "bengalore" , "krish"]
list(filter( lambda x:len(x)<6,l2))
```

Out[41]:

```
['sudh', 'kumar', 'krish']
```

In [ ]: