In [1]:
```
#####functions in python
def test():
    pass ## it does nothing at all
```

In [2]:
```
type(test())
```

Out[2]:
```
NoneType
```

In [3]:
```
def test(): ## mandatory function should return something else give pass otherwise we will get an error

```

```
  File "C:\Users\furqa\AppData\Local\Temp\ipykernel_14796\3866855815.py", line 2

    ^
IndentationError: expected an indented block
```

In [ ]:
```
def add():
    a = 1
    b = 2
    print(a+b)
```

In [ ]:
```
add()
```

In [ ]:
```
def add(a,b):
    print(a+b)
```

In [ ]:
```
add(2,3)
```

In [ ]:
```
add(2.5,3)
```

In [ ]:
```
add(2.5,3.567)
```

In [ ]:
```
add(2,"anand")  # error cannot add int and str,need type conversion
```

In [ ]:
```
"2" + "anand"
```

In [ ]:
```
add("2","anand")
```

In [ ]:
```
type(print("xyz"))
```

In [ ]:
```
def add(a,b):
    return str(a)+str(b)
```

In [ ]:
```
add(3,4)
```

In [ ]:
```
add(3,"Virat Kohli")
```

In [ ]:
```
def add(a,b):
    a = 1
    b = 2
    print(a+b)
```

In [ ]:
```
add(5,6)
```

In [ ]:
```
def add(a,b):
    print(a+b)
```

In [ ]:
```
add(5,6)
```

In [ ]:
```
def add(a,b):
    a=1
    b=2
    c=3
    return a+b+c
```

In [ ]:
```
c=10
add(5,6) + c
```

In [ ]:
```
c=10
add(5,6) + c
```

In [ ]:
```
def add(a,b,c):
    a=1
    b=2
    c=3
    return a+b+c,a+b,a+b*c ##multiple return statement is allowed
```

In [ ]:
```
add(2,3) #error, 1 argument misiing
```

In [ ]:
```
 add(1,3)
```

In [ ]:
```
type(add(1,3))
```

In [ ]:
```
add(1,2,3)
```

In [ ]:
```
add(6,5)+c
```

In [ ]:
```
c=5
c=1
def add(a,b):
    a=1
    b=2
    c=4
    return a+b+c
```

In [ ]:
```
add(2,3)
```

In [ ]:
```
add() #error,need to give 2 parameters as defined in ftn
```

In [ ]:
```
def my_func():
    x = 10
    print("Value inside function:",x)

x = 20
my_func()
print("Value outside function:",x)
```

In [ ]:
```
def abc(a,b,c):
    return a+b+c, a*b*c, a+b*c
```

In [ ]:
```
abc(2,3,4)
```

In [4]:
```
def test():
    return "Sudhanshu" ,[4,5,6,7],{'a':5,'b':6}
```

In [5]:
```
test()
```

Out[5]:
```
('Sudhanshu', [4, 5, 6, 7], {'a': 5, 'b': 6})
```

In [6]:
```
int(4.567)
```

Out[6]:
```
4
```

In [7]:
```
float(3)
```

Out[7]:
```
3.0
```

In [8]:
```
str(3)
```

Out[8]:
```
'3'
```

In [9]:
```
### create a list of all the employess with different name say 5 (from user using input)
### create a uyser defined function which returns the largest member(whoese length is more) from a list
```

In [10]:
```
def add(a,b,c=2):
    c = 1
    return a+b+c
```

In [11]:
```
add(2,3)
```

Out[11]:
```
6
```

In [12]:
```
add(2,3,4)
```

Out[12]:
```
6
```

In [13]:
```
add(2,3,4)
```

Out[13]:
```
6
```

In [14]:
```
test_list = ['anand', 'harsh', 'manisha', 'aishwarya', 'hanuman']
```

In [15]:
```
test_list
```

Out[15]:
```
['anand', 'harsh', 'manisha', 'aishwarya', 'hanuman']
```

In [16]:
```
max_len = 1
for ele in test_list:
    if len(ele) > max_len:
        max_len = len(ele)
        res = ele

print("Maximum length string is : " + res)
```

```
Maximum length string is : aishwarya
```

In [17]:
```
def biglen():
    li=[]
    res=0
    for i in range(5):
        li.append(input('enter a name: '))
        if len(li[i])>res:
            res=len(li[i])
            temp=li[i]
    return temp
```

In [18]:
```
biglen()
```

```
enter a name: furqaan
enter a name: ahmad
enter a name: jazu
enter a name: eleza
enter a name: ehal
```

Out[18]:
```
'furqaan'
```

In [22]:
```
emp_list = []
def emp_list_func():
    for i in range(5):
        emp_name = input("Enter Employee Name ")
        test_list.append(emp_name)
```

In [23]:
```
emp_list_func()
```

```
Enter Employee Name fsb
Enter Employee Name ahamd
Enter Employee Name burhan
Enter Employee Name jazlan
Enter Employee Name ubaid
```

In [ ]:
```
#########LAMBDA FUNCTION ###############################################
```

In [ ]:
```
#### its a one line function which can take any number of arguments but only one expression,which is ev
### its an anonoymous function i.e without a name
### lambda kleyword is used to define it
## no def keyword is used
## one is free to use lambda funcctions wherever funcxtions object are required
## You need to keep in your knowledge that lambda function are syntactically restricted to a single exp
## the lambda operator => separates the input parameters on the left side from the lambda body on the r
```

In [24]:
```
z = lambda a,b,c:a*b/c
```

In [25]:
```
z(8,9,3)
```

Out[25]:
```
24.0
```

In [26]:
```
z(8,9) #error passed only 2 arguments
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14796\3239207005.py in <module>
----> 1 z(8,9)

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

In [27]:
```
cube = lambda x : x**3
```

In [28]:
```
cube(7)
```

Out[28]:
```
343
```

In [29]:
```
### craete an empty list
## ask the user 5 times to input any number
## use lambda function to test for odd and even and pass that number4 to another list calling odd /even
```

In [30]:
```
Check_No= lambda mylist[i]: if mylist[i]%2==0 print("Even") else ("Odd")
```

```
  File "C:\Users\furqa\AppData\Local\Temp\ipykernel_14796\1296789662.py", line 1
    Check_No= lambda mylist[i]: if mylist[i]%2==0 print("Even") else ("Odd")
                                ^
SyntaxError: invalid syntax
```

In [31]:
```
number = int(input("Enter a number >> "))
odd_even = lambda : "Even Number" if number % 2 == 0 else "Odd Number"
print(odd_even())
```

```
Enter a number >> 22
Even Number
```

In [32]:
```
#################################filter function##############################################
```

In [33]:
```
list_test = (4,5,6,7,8,6,7,8,8)
```

In [34]:
```
a = list(filter(lambda x:x%2==0,list_test))
b = list(filter(lambda x:x%2!=0,list_test))
```

In [35]:
```
a,b
```

Out[35]:
```
([4, 6, 8, 6, 8, 8], [5, 7, 7])
```

In [36]:
```
def even_check(n):
    if n%2==0:
        return True
```

In [37]:
```
list(filter(even_check,list_test))
```

Out[37]:
```
[4, 6, 8, 6, 8, 8]
```

In [38]:
```
z = lambda a,b,c:a**2+b*3+c
```

In [39]:
```
z(5,6,7)
```

Out[39]:
```
50
```

In [40]:
```
from functools import reduce
```

In [41]:
```
test = [4,5,6,7,8,9]
reduce(lambda x,y:x+y,test)
```

Out[41]:
```
39
```

In [42]:
```
test = [4,5,6,7,8,9]
reduce(lambda x,y:x/y,test)
```

Out[42]:
```
0.0002645502645502645
```

In [ ]:

In [ ]: