08th-nov

Functions

- Reuse of the code

- functions without arguments
- functions with arguments
- functions with defaluts arguments
- Global variables vs loacl variables
- Functions with return statements
- functions with keywords arguments
- functions in functions

Function without arguments

- The function we defined does not have any values or variables
- those also called as arguments or parameters
- simply it is a empty bracktes

Syntax

In [ ]:
```
def <function_name> ():
    <write your code here>
```

In [1]:
```
n1 = eval(input("enter the n1:"))
n2 = eval(input("enter the n2:"))
add = n1+n2
print(f"The addition of {n1} and {n2} is: {add}")
```

```
The addition of 12 and 45 is: 57
```

In [3]:
```
def addition():
    n1 = eval(input("enter the n1:"))
    n2 = eval(input("enter the n2:"))
    add = n1+n2
    print(f"The addition of {n1} and {n2} is: {add}")
```

In [4]:
```
# Function is ready
# creating function means it will not give the answer
# you need to call the function
addition()
```

```
The addition of 45 and 78 is: 123
```

Note

- Wilecalling the function we need to use the function name
- also we need to provide bracketes
- in python bracktes indicates it is a function or method
- if we miss the brackets, it will display as
  - function
  - bound method

Note

- Once we create the function we can use anywhere in the same notebook
- If we want to use in another notebook
- we need to create as package(OOPs)

In [5]:
```
addition1()
def addition1():
    n1 = eval(input("enter the n1:"))
    n2 = eval(input("enter the n2:"))
    add = n1+n2
    print(f"The addition of {n1} and {n2} is: {add}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 addition1()
      2 def addition1():
      3     n1 = eval(input("enter the n1:"))

NameError: name 'addition1' is not defined
```

1st define and then call

In [6]:
```
def name1():
    n1 = 20
    n2 = 30
    print(n1-n2)
name1()
```

```
-10
```

- Que

In [7]:
```
def avg():
    n1 = eval(input("enter a number:"))
    n2 = eval(input("enter a number:"))
    n3 = eval(input("enter a number:"))
    average = (n1+n2+n3)/3
    print(average)
avg()
```

```
53.0
```

- Que

In [8]:
```
def Area():
    breadth = eval(input("Enter a breadth"))
    height = eval(input("Enter a height"))
    area = 0.5*breadth*height
    print(area)
Area()
```

```
1755.0
```

- Que

In [10]:
```
import math
def circle():
    radius = eval(input("enter a radius:- "))
    pi = math.pi
    area1 = pi*radius*radius
    area = round(area1,2)
    print(area)
circle()
```

```
17671.46
```

- Que

In [11]:
```
def bonus():
    bill = eval(input("Enter your bill amount"))
    tip = eval(input("Enter your tip amount"))
    total_bill = bill + tip
    print(total_bill)
bonus()
```

```
500
```

In [15]:
```
def bouns_tip():
    bill = eval(input("Enter your bill amount"))
    tip = eval(input("Enter your tip percentage"))
    total_bill = (bill/100)*tip+bill
    print(total_bill)
bouns_tip()
```

```
440.0
```

Note

- Variable names do not use to function names

In [19]:
```
try:
    def avg():
        n1 = eval(input("enter a number:"))
        n2 = eval(input("enter a number:"))
        n3 = eval(input("enter a number:"))
        average = (n1+n2+n3)/3
        print(average)
```

```
except Exception as e:
    print(e)

avg()
```

```
458.0
```

In [27]:
```
def avg1():
    try:
        n1 = eval(input("enter a number:"))
        n2 = eval(input("enter a number:"))
        n3 = eval(input("enter a number:"))
        average1 = (n1+n2+n32)/3
        average = round(average1,2)
        print(average)
    except Exception as e:
        print(e)
```

In [28]:
```
avg1()
```

```
name 'n32' is not defined
```

Every time you create a new function give differnt name otherwise you will confused

In [29]:
```
try:
    def avg():
        n1 = eval(input("enter a number:"))
        n2 = eval(input("enter a number:"))
        n3 = eval(input("enter a number:"))
        average = (n1+n2+n23)/3
        print(average)

except Exception as e:
    print(e)

avg()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[29], line 12
      9 except Exception as e:
     10     print(e)
---> 12 avg()

Cell In[29], line 6, in avg()
      4 n2 = eval(input("enter a number:"))
      5 n3 = eval(input("enter a number:"))
----> 6 average = (n1+n2+n23)/3
      7 print(average)

NameError: name 'n23' is not defined
```

Note

- always **try and except** apply for only on code
- Not for function
- inside the function write try-except

- try and except works inside the function only

- step-1:- define
- step-2:- function name
- step-3:- :
- step-4:- indentation
- step-5:- write the code

Note

- when we define the function, it will not give any error or any answer
- If we wnat to see any error or any answer we need to call the function

In [ ]:
```
bouns_tip() # here the bracket was empty thats why it is functions without agrs
bonus()
```

9th Nov

Functions with arguments

- Let's cerate an addition program code

In [31]:
```
def addition():
    n1 = eval(input("Enter a n1:- "))
    n2 = eval(input("Enter a n2:- "))
    add = n1+n2
    print(add)
addition()
```

```
9
```

- How many variables are there in the above code: 3
  - n1,n2,add
- what are the input variables
  - n1,n2
- What are the output variables
  - add
- pass the input variableds inside the functions brackets
- pass n1,n2 inside the brackets these n1,n2, behaves as functions arguments

In [33]:
```
def addition1(n1,n2):
    add = n1+n2
    print(add)
addition1(10,20)
```

```
30
```

- This is called as functions with arguments

In [34]:
```
addition1(124)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 addition1(124)

TypeError: addition1() missing 1 required positional argument: 'n2'
```

- error will always display down
- TypeError: addition1() missing 1 required positional argument: 'n2'

In [35]:
```
from math import pi

def average10(n1,n2,n3):
    avg = (n1+n2+n3)/3
    print(avg)

average10(123,345,567)
```

```
345.0
```

In [36]:
```
def area_of_triangel(b,h):
    area = 0.5*b*h
    print(area)
area_of_triangel(45,12)
```

```
270.0
```

In [38]:
```
def area_of_circle(r):
    area = pi*r*r
    print(round(area,2))
area_of_circle(42)
```

```
5541.77
```

In [41]:
```
def avg1(n1,n2):
        n3 = eval(input("enter a number:"))
        average = (n1+n2+n3)/3
        print(average)
avg1(15,20)

# while calling the function we want to provide n1,n2
# while running the function we want to provide n3
```

```
26.666666666666668
```

- you never no need to provied all parameter or argumnets inside the function
- remeaning parameter you can provide like this also

In [43]:
```
def avg1(n1):
    n2 = eval(input("enter a number:"))
    n3 = eval(input("enter a number:"))
```

```
    average = (n1+n2+n3)/3
    print(average)
avg1(130)

# while calling the function we want to provide n1
# while running the function we want to provide n2,n3
```

```
84.33333333333333
```

In [45]:
```
def avg1():
    n1 = eval(input("enter a number:"))
    n2 = eval(input("enter a number:"))
    n3 = eval(input("enter a number:"))
    average = (n1+n2+n3)/3
    print(average)
avg1()

# while calling the function we want to provide zero parameter
# while running the function we want to provide n1,n2,n3
```

```
64.0
```

In [48]:
```
2**2
2**3
2**4

number**power
```

Out[48]: 16

In [50]:
```
number = eval(input("enter the number:"))
power = eval(input("how much power we want:"))
print(number**power)
```

```
1600
```

In [51]:
```
def Pow():
    number = eval(input("enter the number:"))
    power = eval(input("how much power we want:"))
    print(number**power)

Pow()
```

```
91125
```

In [49]:
```
def logic(num,pow):
    print(num**pow)
logic(10,2)
```

```
100
```

Function with Default argumnet

In [53]:
```
def bill(bill_amt, tip_amt=500):
    total_bill = bill_amt+tip_amt
    print(total_bill)
bill(1000)
```

```
1500
```

In [54]:
```
def a(n1,n2,n3=40):
    print('n1:',n1)
```

```
    print('n2:',n2)
    print('n3:',n3)
    avg = (n1+n2+n3)/3
    print(f"The avg of {n1},{n2} and {n3} is {avg}")
a(19,45)
```

```
n1: 19
n2: 45
n3: 40
The avg of 19,45 and 40 is 34.666666666666664
```

In [55]:
```
def a(n1,n2=50,n3):
    print('n1:',n1)
    print('n2:',n2)
    print('n3:',n3)
    avg = (n1+n2+n3)/3
    print(f"The avg of {n1},{n2} and {n3} is {avg}")
a(35,45)
```

```
  Cell In[55], line 1
    def a(n1,n2=50,n3):
                   ^
SyntaxError: parameter without a default follows parameter with a default
```

- **why error**
  - because python is a step by step process
  - In above code we giving n2=50 insteaded of n3
  - always remember
    - **always write the default arguments at last**

In [56]:
```
def a(n1,n3,n2=50):
    print('n1:',n1)
    print('n2:',n2)
    print('n3:',n3)
    avg = (n1+n2+n3)/3
    print(f"The avg of {n1},{n2} and {n3} is {avg}")
a(35,45)
```

```
n1: 35
n2: 50
n3: 45
The avg of 35,50 and 45 is 43.333333333333336
```

In [57]:
```
def a(n3,n2,n1=50):
    print('n1:',n1)
    print('n2:',n2)
    print('n3:',n3)
    avg = (n1+n2+n3)/3
    print(f"The avg of {n1},{n2} and {n3} is {avg}")
a(35,45)
```

```
n1: 50
n2: 45
n3: 35
The avg of 50,45 and 35 is 43.333333333333336
```

In [ ]:
```
(n1,n2,n3=100) # W
n1,n2=100,n3 # F
n1=100,n2,n3 # F
n1,n2=100,n3=200 # W
n1=100,n2,n3 #F
n2=100,n3=200,n1=23 #W
```

11th-nov

In [58]:
```
# with out arguments
# with arguments
# default arguments

# wap ask the user enter a number
# ask the user get a random number
# if both numbers are match print won
# otherwise lost
```

In [59]:
```
import random
num1 = eval(input("enter a number"))
num2 = random.randint(1,10)
if num1==num2:
    print("Won")
else:
    print("Lost")
```

```
Won
```

with out args

In [61]:
```
import random
def game():
    num1 = eval(input("enter a number"))
    num2 = random.randint(1,10)
    if num1==num2:
        print("Won")
    else:
        print("Lost")

game()
```

```
Lost
```

with args

In [67]:
```
import random
def game1(num1):
    num2 = random.randint(1,10)
    if num1==num2:
        print("Won")
    else:
        print("Lost")

game1(4)
```

```
Won
```

defaul args

In [89]:
```
import random
def game2(num1=6):
    num2 = random.randint(1,10)
    if num1==num2:
        print("Won")
    else:
        print("Lost")

game2()
```

```
Lost
```

Case-1

- step-1:- Define the function
- step-2:- call the function
- step-3:- Run the function

In [2]:
```
def ave(n1,n2,n3=40):
    print('n1:',n1)
    print('n2:',n2)
    print('n3:',n3)
    avg = (n1+n2+n3)/3
    print(f'the avg of {n1},{n2} and {n3} is {avg}')

ave(45,57)
```

```
n1: 45
n2: 57
n3: 40
the avg of 45,57 and 40 is 47.333333333333336
```

In [5]:
```
def add(a,b,c=50):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    print(d)

add(10,20)
```

```
a: 10
b: 20
c: 50
80
```

In [6]:
```
def add(a,b,c=50):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    print(d)

add(10,20,100)
```

```
a: 10
b: 20
c: 100
130
```

In [7]:
```
def add(a,b,c=50):
    c=500
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    print(d)

add(10,20,100)

# define = 50
# calling = 100
# running = 500
```

```
a: 10
b: 20
c: 500
530
```

In [8]:
```
def add(a,b,c=50):
    c=500
    print('a:',a)
    print('b:',b)
    print('c:',c)
    c=1000
    d=a+b+c
    print(d)

add(10,20,100)
```

```
a: 10
b: 20
c: 500
1030
```

In [11]:
```
def add(a,b,c=50):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    print(d)
c = 700
add(10,20,100)

# defining === 50
# c=700
# calling == 100
# running ==100
```

```
a: 10
b: 20
c: 100
130
```

In [12]:
```
def add(a,b,c=50):
    c = 800
```

```
    print('a:',a)
    print('b:',b)
    print('c:',c)
    c = 900
    d=a+b+c
    print(d)
c = 700
add(10,20,100)

# defining = 50
# c= 700
# calling = 100
# running = 800
# c = 900
```

```
a: 10
b: 20
c: 800
930
```

In [13]:
```
c = 900
def add(a,b,c=50):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    print(d)

add(10,20,100)

# c = 900
# defining c = 50
# calling c= 100
# running =#= 100
```

```
a: 10
b: 20
c: 100
130
```

In [14]:
```
c = 200
def add(a,b,c=50):
    c = 400
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    c=500
    print(d)
c = 300
add(10,20,100)

# c =200
# defining = 50
# c = 300
# calling c = 100
# c = 400
```

```
a: 10
b: 20
c: 400
430
```

In [15]:
```
c = 200
def add(a,b,c=50):
    c = 400
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    c=500
    print(d)
    print(c)
c = 300
add(10,20,100)

# c =200
# defining = 50
# c = 300
# calling c = 100
# c = 400
```

```
a: 10
b: 20
c: 400
430
500
```

In [16]:
```
c = 200
def add(a,b,c):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    d=a+b+c
    print(d)
add(10,20)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 8
      6     d=a+b+c
      7     print(d)
----> 8 add(10,20)

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

local variables vs Gloable variables

- local variables are used inside the function
- gloable variables are used outside the function
- gloable variables can be used at anywhere **inside the function** or **outside the function**

In [17]:
```
n1 = eval(input("Enter the n1:"))
n2 = eval(input("Enter the n2:"))
```

```
n3 = eval(input("Enter the n3:"))
def average():
    avg = (n1+n2+n3)/3
    print(f'the avg of {n1}, {n2} and {n3} is {avg}')

average()
```

```
the avg of 4, 7 and 8 is 6.333333333333333
```

In [18]:
```
print(n1,n2,n3)
```

```
4 7 8
```

In [19]:
```
def average1():
    N1 = eval(input("Enter the n1:"))
    N2 = eval(input("Enter the n2:"))
    N3 = eval(input("Enter the n3:"))
    avg = (N1+N2+N3)/3
    print(f'the avg of {n1}, {n2} and {n3} is {avg}')

average1()
```

```
the avg of 4, 7 and 8 is 45.0
```

In [20]:
```
print(N1,N2,N3)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[20], line 1
----> 1 print(N1,N2,N3)

NameError: name 'N1' is not defined
```

Note

- Local variables can not use outside the function
- in above code N1,N2,N3 is a local variable
- We are trying to use outside the function it is throwing the error

In [21]:
```
A = eval(input("Enter the n1:"))
B = eval(input("Enter the n2:"))
C = eval(input("Enter the n3:"))
def average3():
    AvG = (A+B+C)/3
    print(f'the avg of {A}, {B} and {C} is {AvG}')

average3()
```

```
the avg of 45, 78 and 12 is 45.0
```

In [22]:
```
AvG
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 AvG

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

- **Note**
- Gloable variables are intialized outside the function
- gloabl variables can be use anywhere
- local variables are intialized inside the function
- local variables can not use outside the function

In [24]:
```
A = 100
B = 200
def summ():
    C = 300
    add = A+B+C
    print(f'the avg of {A}, {B} and {C} is {add}')

summ()

# A = 100
# B = 200
# defining
# call C =300
```

```
the avg of 100, 200 and 300 is 600
```

In [25]:
```
A = 100
B = 200
def summ(A=400):
    C = 300
    add = A+B+C
    print(f'the avg of {A}, {B} and {C} is {add}')

summ()

# A = 100
# B = 200
# defining A = 400
# call C =300
```

```
the avg of 400, 200 and 300 is 900
```

In [27]:
```
A = 100
B = 200
def summ():
    C = 300
    add = A+B+C
    print(f'the avg of {A}, {B} and {C} is {add}')
A = 2000
summ()

# A = 100
# B = 200
# defining
# A = 2000
# call C =300
```

```
the avg of 2000, 200 and 300 is 2500
```

In [28]:
```
A = 100
B = 200
def summ(A=4000):
    C = 300
    A = 8000
    add = A+B+C
    print(f'the avg of {A}, {B} and {C} is {add}')
A = 2000
summ(A=3000)

# A = 100
# B = 200
# defining A = 4000
# A = 2000
# call A = 3000
# Running c = 300
# A = 8000
```

```
the avg of 8000, 200 and 300 is 8500
```

In [29]:
```
A = 100
B = 200
#########################################################
def summ(A=4000):
    C = 300
    A = 8000
    add = A+B+C
    print(f'the avg of {A}, {B} and {C} is {add}')
#########################################################
A = 2000
#########################################################
summ(A=3000)
#########################################################
```

```
the avg of 8000, 200 and 300 is 8500
```

how to use local variables outside the function

In [32]:
```
A = eval(input("Enter the n1:"))
B = eval(input("Enter the n2:"))
C = eval(input("Enter the n3:"))
avg=0
def average3():
    avg = (A+B+C)/3
    print(f'the avg of {A}, {B} and {C} is {avg}')

average3()
```

```
the avg of 122, 456 and 789 is 455.6666666666667
```

In [33]:
```
avg
```

Out[33]:
```
0
```

In [34]:
```
A = eval(input("Enter the n1:"))
B = eval(input("Enter the n2:"))
C = eval(input("Enter the n3:"))
def add1():
    ADD1 = (A+B+C)
    print(f'the avg of {A}, {B} and {C} is {ADD1}')
```

```
add1()
```

```
the avg of 100, 200 and 300 is 600
```

In [35]:
```
ADD1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 ADD1

NameError: name 'ADD1' is not defined
```

In [36]:
```
A = eval(input("Enter the n1:"))
B = eval(input("Enter the n2:"))
C = eval(input("Enter the n3:"))
def add1():
    global ADD1
    ADD1 = (A+B+C)
    print(f'the avg of {A}, {B} and {C} is {ADD1}')

add1()
```

```
the avg of 100, 200 and 250 is 550
```

In [37]:
```
ADD1
```

Out[37]:
```
550
```

- Local variables can be use outside the funtion in two ways
  - By making local variables as global by using **global**
  - By using return statement also

12th-Nov

- completed
  - with out args
  - with args
  - default args
  - local variable vs global variable

In [42]:
```
def summ(a,b):
    add2 = a+b
    sub2 = a-b
    print(add2,sub2)

summ(20,30)
```

```
50 -10
```

In [43]:
```
add2
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[43], line 1
----> 1 add2

NameError: name 'add2' is not defined
```

In [44]:
```
def summ(a,b):
    global add2,sub2
    add2 = a+b
    sub2 = a-b
    print(add2,sub2)

summ(20,30)
```

```
50 -10
```

In [45]:
```
add2,sub2
```

Out[45]: (50, -10)

In [46]:
```
def fun1():
    a = 10
    b=b+a
    print(b)
fun1()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[46], line 5
      3     b=b+a
      4     print(b)
----> 5 fun1()

Cell In[46], line 3, in fun1()
      1 def fun1():
      2     a = 10
----> 3     b=b+a
      4     print(b)

UnboundLocalError: cannot access local variable 'b' where it is not associated wi
th a value
```

above code

- addition and saving in the same variable which is not defining is called unbound local error.
- b only input and b only output and b is not defining
- **unbound means it is not exists**

unbound loal error

- unbound local error means the variable is not defined

- that variable is used as both input and output

In [47]:
```
def fun2():
    a = 10
    c=b+a
    print(c)
fun2()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[47], line 5
      3     c=b+a
      4     print(c)
----> 5 fun2()

Cell In[47], line 3, in fun2()
      1 def fun2():
      2     a = 10
----> 3     c=b+a
      4     print(c)

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

above code

- Only adding a variable that snot defining that's why error coming

return Statement

- till now above we have created a function with print statement in order
- The funtion output we are not able to use outside the function
- because it is a local variable
- one method we used to is **global**
- another method is return statement
- return means something we are recieving
- print means only we can see the answer

In [50]:
```
def add(a,b):
    CC = a+b
    print(CC)
add(20,30)
```

```
50
```

In [51]:
```
CC
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[51], line 1
----> 1 CC

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

In [56]:
```
def add(a,b):
    D = a+b
    return(D)
D=add(20,30)
D
```

Out[56]: 50

In [57]:
```
D
```

Out[57]: 50

In [61]:
```
def xyz(a,b):
    val1 = a+b
    return(val1)

val1 = xyz(20,30)
```

In [62]:
```
val1
```

Out[62]: 50

- print is use for what only see the answer
- answer is coming properly or not
- print is use only for testing or checking our code is run or not
- But when the final answer is come then we will use **return**

In [67]:
```
def add_sub(a,b):
    aDD = a+b
    sUB = a-b
    return(aDD,sUB)

val1 = add_sub(200,30)
```

In [68]:
```
val1
```

Out[68]: (230, 170)

- one variable use in above code

In [70]:
```
def add_sub(a,b):
    aDD = a+b
    sUB = a-b
    return(aDD,sUB)
```

```
val1,val2 = add_sub(200,30)
```

In [72]:
```
val1,val2
```

Out[72]:
```
(230, 170)
```

- Two variable use in above code

In [86]:
```
def add_sub(a,b):
    aDD = a+b
    sUB = a-b
    return(aDD)
    return(sUB)
```

In [90]:
```
n1 = 100,200 # works
n1,n2 = 100,200 # works
n1,n2 = 100 # FAils
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[90], line 3
      1 n1 = 100,200 # works
      2 n1,n2 = 100,200 # works
----> 3 n1,n2 = 100 # FAils

TypeError: cannot unpack non-iterable int object
```

In [91]:
```
def add_summ(a,b):
    summm = a+b
    subbb = a-b
    return(summm,subbb)

op = add_summ(233,363)
```

In [92]:
```
op
```

Out[92]:
```
(596, -130)
```

Homework

35 total questions write

1. with
2. without
3. def
4. try
5. local
6. global
7. return

global vs return

- global variable can be use we can take the inside variable to outside the function

- return can return any value outside the function
- the value you return can be used another notebook

- global and return work was same but the difference is only you can use global variable value in same notebook
- **but return value you can use in different notebook also and it will run**

function in functions

In [94]:
```
def greet1():
    print("Hello good morning")

def greet2():
    print("Hello good night")


greet1()
greet2()
```

```
Hello good morning
Hello good night
```

In [96]:
```
def greet1():
    print("Hello good morning")

def greet2():
    print('greet1 function run')
    greet1()
    print("Hello good night")

greet2()
```

```
greet1 function run
Hello good morning
Hello good night
```

In [97]:
```
def greet1():
    print('greet2 function run')
    greet2()
    print("Hello good morning")

def greet2():
    print("Hello good night")

greet1()
```

```
greet2 function run
Hello good night
Hello good morning
```

In [ ]:
```
def greet1():
    print('greet2 function run')
    greet2()
    print("Hello good morning")

def greet2():
```

```
    print('greet1 function run')
    greet1()
    print("Hello good night")

greet2()
```

- the above code is run continuously

In [106…
```
def outside_function(name):
    def inside_function(name):
        print(f"hello {name} good morning")

outside_function('p')
```

In [112…
```
############ Error Code ###################
def inside_function():
    print(f"hello {name} good morning")

# we need to sense here
# name is not defined not even globally
# if i call this function definetly i will get error


def outside_function(name):
    inside_function()

outside_function("python")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[112], line 13
     10 def outside_function(name):
     11     inside_function()
---> 13 outside_function("python")

Cell In[112], line 11, in outside_function(name)
     10 def outside_function(name):
---> 11     inside_function()

Cell In[112], line 3, in inside_function()
      2 def inside_function():
----> 3     print(f"hello {name} good morning")

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

In [110…
```
def inside_function(name):
    print(f"hello {name} good morning")
        # is the variable defined anywhere

def outside_function(name):
    inside_function(name)

outside_function("python")
```

```
hello python good morning
```

In [114…
```
def outside_function(name):
    def inside_function(name):
        print(f"hello {name} good morning")
```

```
outside_function("naresh")
```

- In above code we run outside_function but not calling inside_function
- Thats why it will not see any answer

In [115…
```
def inside_function(name):
        print(f"hello {name} good morning")

inside_function('Python')
```

```
hello Python good morning
```

In [119…
```
def outside_function(name):
    def inside_function(name):
        print(f"hello {name} good morning")
    inside_function('Naresh')

outside_function('X')
```

```
hello Naresh good morning
```

In [120…
```
def add(a,b):
    print(a+b)

add(12,78)
```

```
90
```

In [121…
```
def MATH():
    def add(a,b):
        print(a+b)

    add(12,78)
MATH()
```

```
90
```

In [122…
```
def MATH1():
    def MATH():
        def add(a,b):
            print(a+b)
        add(12,78)
    MATH()
MATH1()
```

```
90
```

In [123…
```
def MATH2():
    def MATH1():
        def MATH():
            def add(a,b):
                print(a+b)
            add(12,78)
        MATH()
    MATH1()
MATH2()
```

```
90
```

In [124…
```
def MATH(a,b):
    def add():
        print(a+b)
    add()
MATH(20,80)
```

```
100
```

In [ ]:
```
# calculator program

# first create 4 function
# add function
# def add(a,b):
# print(a+b)
# sub function
# mul function
# div function
# print("Enter the value 1,2,3,4")
# main function
# a= number
# b=number
# operator=
# if operator == 1:
# add function add()
```

In [ ]:
```
# in main function we add 4 sub functions
```