

# Function

- Reuse of the code block
- Function also called as method
- function with out arguments
- function with arguments
- function with default arguments
- function with keyword arguments (kwargs)
- function with global and local variable
- function with return
- function in function

```
In [1]: income=eval(input('enter the income:'))
tax=eval(input('enter the tax percentage:'))
tax_pay=income*tax/100
print(tax_pay)
```

5000.0

```
In [ ]: # 4 lines of code is there
# 100 employess
# for each employee will use this 4 lines of code
# 100*4=400 lines
# we will make these 4 lines as single call that is function
# we will call these line 100 mem 100 times
```

## Functions with out arguments

```
In [ ]: # syntax
def <function name>():
    <write your lines here>
```

```
In [2]: num1=eval(input('enter a num1:'))
num2=eval(input('enter a num2:'))
add=num1+num2
print(f'the addition of {num1} and {num2} is: {add}')
```

the addition of 100 and 200 is: 300

```
In [3]: def addition():
num1=eval(input('enter a num1:'))
num2=eval(input('enter a num2:'))
add=num1+num2
print(f'the addition of {num1} and {num2} is: {add}')
```

## call the function

```
In [6]: addition()
```

the addition of 100 and 200 is: 300

```
In [5]: import random
        random.randint
```

```
Out[5]: <bound method Random.randint of <random.Random object at 0x000002394F1CD240>>
```

```
In [ ]: def addition():
        num1=eval(input('enter a num1:'))
        num2=eval(input('enter a num2:'))
        add=num1+num2
        print(f'the addition of {num1} and {num2} is: {add}')

        addition()
```

### Note

- Function name rules same as variable rules
  - No special characters
  - No numbers before
  - No space between words etc
- Function name should not be any keywords
- Function name should not be any package names
- Function name should not be your notebook name
- Function name should not be any variable which is using inside that function

```
In [7]: def addition1():
        n1=eval(input('enter a num1:'))
        n2=eval(input('enter a num2:'))
        add1=n1+n2
        print(f'the addition of {n1} and {n2} is: {add1}')
```

```
In [8]: addition1()
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[8], line 1
----> 1 addition1()

Cell In[7], line 4, in addition1()
      2 n1=eval(input('enter a num1:'))
      3 n2=eval(input('enter a num2:'))
----> 4 add1=n1+n2
      5 print(f'the addition of {n1} and {n2} is: {add1}')
```

**NameError:** name 'n11' is not defined

- While define the function we does not know whether the error is there or not
- For that we need to call the function

```
In [ ]: try:
    def addition1():
        n1=eval(input('enter a num1:'))
        n2=eval(input('enter a num2:'))
        add1=n1+n2
        print(f'the addition of {n1} and {n2} is: {add1}')
    except Exception as e:
        print(e)
```

```
In [9]: try:
    def addition2():
        n1=eval(input('enter a num1:'))
        n2=eval(input('enter a num2:'))
        add1=n1+n2
        print(f'the addition of {n1} and {n2} is: {add1}')
    except Exception as e:
        print(e)

    addition2()
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[9], line 10
      7 except Exception as e:
      8     print(e)
--> 10 addition2()

Cell In[9], line 5, in addition2()
      3 n1=eval(input('enter a num1:'))
      4 n2=eval(input('enter a num2:'))
----> 5 add1=n1+n2
      6 print(f'the addition of {n1} and {n2} is: {add1}')
```

**NameError:** name 'n11' is not defined

```
In [10]: def addition3():
    try:
        n1=eval(input('enter a num1:'))
        n2=eval(input('enter a num2:'))
        add1=n1+n2
        print(f'the addition of {n1} and {n2} is: {add1}')
```

**NameError:** name 'n11' is not defined

- try-except should write inside the function

```
In [ ]: # 1)wap ask the user enter 3 numbers n1,n2,n3 from ketboard
        # calculate average
```

```

# 2)wap ask the user enter name age city
# print my name is python im 10 years old and came from hyd

# 3)wap ask the user to enter radidus of a circle calculate area of the circle
# var: radidus var: pi=3.14
# formule: pi*radius*radius#
# print the answers using f string and format

# 4)wap ask the user enter breadth and height of a right angle triangle
# calculate the area
# var1: bredath var2: height
# formuale : 0.5*breadth*heigh

# 5)wap ask the user the bill amount and tip amount
# calculate total bill
# var1: bill amount var2: tip amount
# formuale

# 6)wap ask the user the bill amount and tip percentage
# take tip percentage as 10
# calculate total bill= bill amount+ bill amount*tip per/100
# var1: bill amount var2: tip amount
# formuale

# 7)wap ask the length and breadth of a rectangle calculate area
# var1: length var2: breadth
# formulae: length * breadth

# 8) wap ask the user take the radius and calculate volume of sphere
# formulae: pi*r**3(pi*r*r*r)

# 9) wap ask the user enter amount in dollars convert into rupees
# 1$=85rs

# 10)wap ask the user enter weight in kgs convert into pounds
# 1kg= 2.2pounds

```

In [11]: # 1)wap ask the user enter 3 numbers n1,n2,n3 from keyboard  
# calculate average

```

def AVG():
    n1=eval(input('enter a num1:'))
    n2=eval(input('enter a num2:'))
    n3=eval(input('enter a num2:'))
    avg=(n1+n2+n3)/3
    avg1=round(avg,2)
    print(f'the avg of {n1},{n2} and {n3} is: {avg1}')

AVG()

```

the avg of 20,22 and 35 is: 25.67

In [ ]: # Mistake-1: exception Exception

how can we open all the keywords available in function list?

Sir, For example we have Right Angle Triangle Area. I want this to be stored in function name any name: virat

```
In [12]: # 2)wap ask the user enter name age city
# print my name is python im 10 years old and came from hyd

def intro():
    name=input("enter name")
    age=eval(input("enetr age"))
    city=input("Enter city:")
    print(f"my name is {name} i am {age} years old and coming from {city}")

intro()
```

my name is vaibav i am 20 years old and coming from MH

```
In [13]: import random

def area_of_circle():
    radius=random.randint(1,20)
    pi=3.14
    area=pi*radius*radius
    print(f"the area of a circle for radius {radius} is: {area}")

area_of_circle()
```

the area of a circle for radius 14 is: 615.44

```
In [14]: def area_of_traiangle():
    breadth=random.randint(1,20)
    height=random.randint(1,10)
    area=0.5*breadth*height
    print(f"for breadth {breadth} and height {height} are is {area}")

area_of_traiangle()
```

for breadth 9 and height 6 are is 27.0

```
In [15]: def BILL():
    bill_amount=random.randint(500,1000)
    tip_amount=random.randint(20,100)
    total_bill=bill_amount+tip_amount
    print("the total bill is:",total_bill)

BILL()
```

the total bill is: 591

```
In [16]: def BILL_AMOUNT():
    bill_amount=random.randint(500,1000)
    tip_per=random.randint(1,10)
    tip_amount=bill_amount*tip_per/100
    total_bill=bill_amount+tip_amount
    print("the bill amount is:",bill_amount)
    print("the tip percentage is:",tip_per)
    print("the total bill is:",total_bill)

BILL_AMOUNT()
```

the bill amount is: 722  
the tip percentage is: 5  
the total bill is: 758.1

```
In [18]: def area_of_rectangle():
    length=random.randint(1,20)
```

```

breadth=random.randint(1,10)
area=length*breadth
print(f"for breadth {breadth} and length {length} are is {area}")

area_of_rectangle()

```

for breadth 5 and length 3 are is 15

```

In [19]: def sphere():
          radius=random.randint(1,100)
          pi=3.14
          area=pi*radius**3
          print(f"the volumn of a sphere for radius {radius} is: {area}")

          sphere()

```

the volumn of a sphere for radius 13 is: 6898.58

```

In [20]: def money_convert():
          dollar=random.randint(1,100)
          rupees=dollar*85
          print(f"number of {dollar}$={rupees}INR")

          money_convert()

```

number of 34\$=2890INR

```

In [21]: def weight_convert():
          weight=random.randint(1,100)
          pounds=weight*2.2
          print(f"{weight}Kg={pounds} pounds")

          weight_convert()

```

72Kg=158.4 pounds

```

In [22]: addition()
          AVG()
          area_of_circle()
          area_of_rectangle()
          area_of_rectangle()
          money_convert()
          weight_convert()

```

the addition of 20 and 30 is: 50  
the avg of 40,50 and 60 is: 50.0  
the area of a circle for radius 9 is: 254.34  
for breadth 4 and length 9 are is 36  
for breadth 4 and length 5 are is 20  
number of 20\$=1700INR  
14Kg=30.800000000000004 pounds

## Functions with argumnts

```

In [ ]: def AVG():
          n1=eval(input('enter a num1:'))
          n2=eval(input('enter a num2:'))
          n3=eval(input('enter a num2:'))
          avg=(n1+n2+n3)/3
          avg1=round(avg,2)
          print(f'the avg of {n1},{n2} and {n3} is: {avg1}')

```

```
AVG()
```

```
In [2]: def addition():
        n1=eval(input('enter a num1:'))
        n2=eval(input('enter a num2:'))
        add=n1+n2
        print(f'the sum of {n1} and {n2} is:{add}')

        addition()
```

the sum of 100 and 300 is:400

```
In [ ]: addition()
```

```
In [4]: from random import randint
        randint()
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[4], line 2
      1 from random import randint
----> 2 randint()

TypeError: Random.randint() missing 2 required positional arguments: 'a' and 'b'
```

```
In [ ]: def addition():
        n1=eval(input('enter a num1:'))
        n2=eval(input('enter a num2:'))
        add=n1+n2
        print(f'the sum of {n1} and {n2} is:{add}')

        addition()
```

- How many variables are there inside the function
  - in above **addition** function 3 variables are there : **n1,n2,add**
- How many input variables are there
  - input variables means the variable access the values from user as input: **n1,n2**
- How many output variables are there
  - output variables means the variable created because of some operations: **add**
- **Dont touch the output variable**

```
In [8]: def addition1(n1,n2):
        print('n1:',n1)
        print('n2:',n2)
        add=n1+n2
        print(f'the sum of {n1} and {n2} is:{add}')
```

```
In [10]: addition1(1000,2000)
```

n1: 1000  
n2: 2000  
the sum of 1000 and 2000 is:3000

```
In [14]: addition1(500)
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[14], line 1
----> 1 addition1(500)

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

```
In [ ]: TypeError: addition1() missing 1 required positional arguments: n2'
```

```
In [15]: def addition2(n1):
        n2=eval(input('enter a num2:'))
        add=n1+n2
        print(f'the sum of {n1} and {n2} is:{add}')
```

```
In [16]: addition2(1000)
```

```
the sum of 1000 and 20 is:1020
```

```
In [ ]: def addition():
        n1=eval(input('enter a num1:'))
        n2=eval(input('enter a num2:'))
        add=n1+n2
        print(f'the sum of {n1} and {n2} is:{add}')

        def addition1(n1,n2):
            add=n1+n2
            print(f'the sum of {n1} and {n2} is:{add}')

        def addition2(n1):
            n2=eval(input('enter a num2:'))
            add=n1+n2
            print(f'the sum of {n1} and {n2} is:{add}')
```

```
In [20]: print('===== AVG function starts=====')
        def AVG():
            n1=eval(input('enter a num1:'))
            n2=eval(input('enter a num2:'))
            n3=eval(input('enter a num2:'))
            avg=(n1+n2+n3)/3
            avg1=round(avg,2)
            print(f'the avg of {n1},{n2} and {n3} is: {avg1}')

        AVG()

        print('===== AVG1 function starts=====')
        def AVG1(n1):
            print('n1:',n1)
            n2=eval(input('enter a num2:'))
            n3=eval(input('enter a num3:'))
            avg=(n1+n2+n3)/3
            avg1=round(avg,2)
            print(f'the avg of {n1},{n2} and {n3} is: {avg1}')

        AVG1(100)

        print('===== AVG2 function starts=====')
```



```
def AVG2(n1,n2):
    print('n1:',n1)
    print('n2:',n2)
    n3=eval(input('enter a num3:'))
    avg=(n1+n2+n3)/3
    avg1=round(avg,2)
    print(f'the avg of {n1},{n2} and {n3} is: {avg1}')

AVG2(20,30)

print('===== AVG3 function starts=====')
def AVG3(n1,n2,n3):
    print('n1:',n1)
    print('n2:',n2)
    print('n3:',n3)
    avg=(n1+n2+n3)/3
    avg1=round(avg,2)
    print(f'the avg of {n1},{n2} and {n3} is: {avg1}')
AVG3(10,20,30)
```

```
===== AVG function starts=====
the avg of 100,200 and 300 is: 200.0
===== AVG1 function starts=====
n1: 100
the avg of 100,200 and 300 is: 200.0
===== AVG2 function starts=====
n1: 20
n2: 30
the avg of 20,30 and 300 is: 116.67
===== AVG3 function starts=====
n1: 10
n2: 20
n3: 30
the avg of 10,20 and 30 is: 20.0
```

```
In [ ]: # 1)wap ask the user enter 3 numbers n1,n2,n3 from ketboard
        # calculate average

        # 2)wap ask the user enter name age city
        # print my name is python im 10 years old and came from hyd

        # 3)wap ask the user to enter radidus of a circle calculate area of the circle
        # var: radidus var: pi=3.14
        # formuale: pi*radius*radius#
        # print the answers using f string and format

        # 4)wap ask the user enter breadth and height of a right angle triangle
        # calculate the area
        # var1: bredath var2: height
        # formuale : 0.5*breadth*heigh

        # 5)wap ask the user the bill amount and tip amount
        # calculate total bill
        # var1: bill amount var2: tip amount
        # formuale

        # 6)wap ask the user the bill amount and tip percentage
        # take tip percentage as 10
        # calculate total bill= bill amount+ bill amount*tip per/100
```

```
# var1: bill amount var2: tip amount
# formule

# 7)wap ask the length and breadth of a rectangle calculate area
# var1: length var2: breadth
# formulae: length * breadth

# 8) wap ask the user take the radius and calculate volume of sphere
# formulae: pi*r**3(pi*r*r*r)

# 9) wap ask the user enter amount in dollars convert into rupees
# 1$=85rs

# 10)wap ask the user enter weight in kgs convert into pounds
# 1kg= 2.2pounds
```

```
In [21]: def intro(name,age,city):
          print(f"my name is {name} i am {age} years old and coming from {city}")

          intro("python",10,"hyd")
```

my name is python i am 10 years old and coming from hyd

```
In [23]: # 9) wap ask the user enter amount in dollars convert into rupees
          # 1$=85rs

          # 10)wap ask the user enter weight in kgs convert into pounds
          # 1kg= 2.2pounds

          dollars=eval(input('enter the dollars:'))
          rate=eval(input('enter the rupees for one dollar:'))
          rupees=dollars*rate
          print(f'{dollars}$={rupees}/-')
```

100\$=8500/-

```
In [ ]: dollars=eval(input('enter the dollars:'))
          rate=eval(input('enter the rupees for one dollar:'))
          rupees=dollars*rate
          print(f'{dollars}$={rupees}/-')
```

### Functions with Default arguments

```
In [25]: n1=12345//10
          d1=12345%10
          d1
```

Out[25]: 5

```
In [27]: n2=n1//10
          d2=n1%10
          d2
```

Out[27]: 4

```
In [30]: str(d1)+str(d2)
```

Out[30]: '54'

```
In [ ]: in future we will learn easy methods
        then you will understand what is the use of method

        the same qn i have done lengthy way
        this time i am doing in simple way
```

```
In [32]: n1=12345
        str(n1[::-1])
```

```
Out[32]: '54321'
```

```
In [34]: def bill_amount():
        bill=eval(input('enter the bill:'))
        tip_per=eval(input('enter the tip_per'))
        tip_amount=bill*tip_per/100
        total_bill=bill+tip_amount
        print('total bill is:',total_bill)

        bill_amount()
```

```
total bill is: 1100.0
```

```
In [35]: def bill_amount1(bill,tip_per):
        tip_amount=bill*tip_per/100
        total_bill=bill+tip_amount
        print('total bill is:',total_bill)

        bill_amount1(1000,10)
```

```
total bill is: 1100.0
```

```
In [36]: def bill_amount1(bill,tip_per=20):
        tip_amount=bill*tip_per/100
        total_bill=bill+tip_amount
        print('total bill is:',total_bill)

        bill_amount1(1000)
```

```
total bill is: 1200.0
```

- In above function the tip\_per is fixed as 20
- this is called default arguments
- while calling the function no need to provide tip\_per again
- if you provide again the value will be change

```
In [37]: def bill_amount1(bill,tip_per=20):
        tip_amount=bill*tip_per/100
        total_bill=bill+tip_amount
        print('total bill is:',total_bill)

        bill_amount1(1000,30)
```

```
total bill is: 1300.0
```

```
In [38]: def AVG3(n1,n2,n3=30):
          print('n1:',n1)
          print('n2:',n2)
          print('n3:',n3)
          avg=(n1+n2+n3)/3
          avg1=round(avg,2)
          print(f'the avg of {n1},{n2} and {n3} is: {avg1}')
          AVG3(10,20)
```

```
n1: 10
n2: 20
n3: 30
the avg of 10,20 and 30 is: 20.0
```

```
In [39]: def AVG3(n1,n2=20,n3=30):
          print('n1:',n1)
          print('n2:',n2)
          print('n3:',n3)
          avg=(n1+n2+n3)/3
          avg1=round(avg,2)
          print(f'the avg of {n1},{n2} and {n3} is: {avg1}')
          AVG3(10)
```

```
n1: 10
n2: 20
n3: 30
the avg of 10,20 and 30 is: 20.0
```

```
In [40]: def AVG3(n1,n2=20,n3):
          print('n1:',n1)
          print('n2:',n2)
          print('n3:',n3)
          avg=(n1+n2+n3)/3
          avg1=round(avg,2)
          print(f'the avg of {n1},{n2} and {n3} is: {avg1}')
          AVG3(100,200)
```

Cell In[40], line 1

```
def AVG3(n1,n2=20,n3):
```

**SyntaxError:** non-default argument follows default argument

**Note: Default arguments always at last**

```
In [41]: def AVG3(n1,n3,n2=20):
          print('n1:',n1)
          print('n2:',n2)
          print('n3:',n3)
          avg=(n1+n2+n3)/3
          avg1=round(avg,2)
          print(f'the avg of {n1},{n2} and {n3} is: {avg1}')
          AVG3(100,200)
```

```
n1: 100
n2: 20
n3: 200
the avg of 100,20 and 200 is: 106.67
```

```
In [ ]: n1,n2,n3=100 # w
         n1,n2=100,n3 # F
```

```

n1=100,n2,n3    # F
n1,n2=100,n3=100 # W
n1=100,n2,n3=100 # f
n1=100,n2=100,n3 # f
n1=100,n2=100,n3=100 # w

```

- Functions with arguments
- Function with out arguments
- Function with Default arguments

```

In [2]: print("Function With out arguments")
def tax_cal():
    salary=eval(input('enter the salary:'))
    tax_per=eval(input('enter the tax:'))
    tax_pay=salary*tax_per/100
    print("the total tax pay:",tax_pay)

tax_cal()

print("Function With arguments")
def tax_cal1(salary,tax_per):
    tax_pay=salary*tax_per/100
    print("the total tax pay:",tax_pay)

tax_cal1(100000,20)

print("Function With default arguments")
def tax_cal2(salary,tax_per=10):
    tax_pay=salary*tax_per/100
    print("the total tax pay:",tax_pay)

tax_cal2(50000)

```

Function With out arguments

the total tax pay: 10000.0

Function With arguments

the total tax pay: 20000.0

Function With default arguments

the total tax pay: 5000.0

### analogy with the packages

```

In [3]: from random import randint,random

```

```

In [ ]: random()
        tax_cal()
        randint()
        tax_cal1()

```

```

In [4]: complex(3,4) # real=3 imag=4 3+4j

```

```

Out[4]: (3+4j)

```

```

In [5]: complex(3) # real=3 imag=0 3+0j

```

Out[5]: (3+0j)

```
In [6]: complex() # real=0 imag=0 0+0j
```

Out[6]: 0j

- Now onwards whenever you see any package
- always do shift+tab inside the brackets and observe
- With or without or default

```
In [7]: def add(a,b,c):  
        print('a:',a)  
        print('b:',b)  
        print('c:',c)  
        summ=a+b+c  
        print(f'the summation of {a},{b} and {c} is : {summ}')
```

```
add(100,200,300)
```

a: 100  
b: 200  
c: 300  
the summation of 100,200 and 300 is : 600

#### case-1

```
In [8]: def add(a,b,c=400):  
        print('a:',a)  
        print('b:',b)  
        print('c:',c)  
        summ=a+b+c  
        print(f'the summation of {a},{b} and {c} is : {summ}')
```

```
add(100,200)
```

a: 100  
b: 200  
c: 400  
the summation of 100,200 and 400 is : 700

#### case-2

```
In [9]: def add(a,b,c=400):  
        print('a:',a)  
        print('b:',b)  
        print('c:',c)  
        summ=a+b+c  
        print(f'the summation of {a},{b} and {c} is : {summ}')
```

```
add(100,200,600)
```

a: 100  
b: 200  
c: 600  
the summation of 100,200 and 600 is : 900

- In above case we already provided default value c=400
- but while we call the function c value become=600
- step-1: Define the function: while define what is the value 400
- step-2: call the function: while call the function what is the value 600
- step-3: Running the function : while running what is the value : 600

In [10]: `add1(100,200,600)`

```
def add1(a,b,c=400):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+c
    print(f'the summation of {a},{b} and {c} is : {summ}')
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[10], line 1
----> 1 add1(100,200,600)
      3 def add1(a,b,c=400):
      4     print('a:',a)

NameError: name 'add1' is not defined
```

In [11]: `def add(a,b,c=400):`

```
    c=1000
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+cccc
    print(f'the summation of {a},{b} and {c} is : {summ}')
```

In [12]: `def add(a,b,c=400):`

```
    c=1000
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+c
    print(f'the summation of {a},{b} and {c} is : {summ}')
```

`add(100,200,600)`

```
# while define c=400
# while calling c=600
# while running c=1000
```

```
a: 100
b: 200
c: 1000
the summation of 100,200 and 1000 is : 1300
```

In [13]: `def add(a,b,c):`

```
    c=1000
```

```

print('a:',a)
print('b:',b)
print('c:',c)
summ=a+b+c
print(f'the summation of {a},{b} and {c} is : {summ}')

add(100,200,600)

```

a: 100  
b: 200  
c: 1000  
the summation of 100,200 and 1000 is : 1300

```

In [14]: c=3000
def add(a,b,c=600):
    c=1000
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+c
    print(f'the summation of {a},{b} and {c} is : {summ}')
c=2000
add(100,200,800)

```

a: 100  
b: 200  
c: 600  
the summation of 100,200 and 600 is : 900

```

In [15]: c=3000

def add(a,b):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+c
    print(f'the summation of {a},{b} and {c} is : {summ}')
add(100,200)

```

```

# c=3000
# add() there is no c : c=3000
# call no c : c=3000
# inside the any c : c=3000

```

a: 100  
b: 200  
c: 2000  
the summation of 100,200 and 2000 is : 2300

### global variable vs Local variable

- The variables outside the function is called as **Global variable**
  - Global variables can use anywhere
  - global variables can be use inside the function also outside the function
- The variables inside the function is called as **Local variable**
  - local variables can be use only inside function



- local variables can not be use outside the function

```
In [16]: def addition():  
         n1=100  
         n2=200  
         summ=n1+n2  
         print(f'the summation of {n1},{n2} is : {summ}')
```

```
         addition()
```

the summation of 100,200 is : 300

```
In [18]: n1
```

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[18], line 1  
----> 1 n1  
  
NameError: name 'n1' is not defined
```

```
In [19]: n1=1000  
         n2=2000  
         def addition():  
             summ=n1+n2  
             print(f'the summation of {n1},{n2} is : {summ}')
```

```
         addition()
```

the summation of 1000,2000 is : 3000

```
In [20]: n1
```

```
Out[20]: 1000
```

### How to convert local variable to Global variable

- if we want to use summ variable outside the function
- Then intialise summ as global also using **global** keyword

```
In [21]: n1=1000  
         n2=2000  
         def addition():  
             global summ  
             summ=n1+n2  
             print(f'the summation of {n1},{n2} is : {summ}')
```

```
         addition()
```

the summation of 1000,2000 is : 3000

```
In [22]: summ
```

```
Out[22]: 3000
```

### unbound local error

- in below example s is not intialised

- we might think name error will come
- but inside function we will get **unbound local error**

```
In [32]: def add1():
          n1=100
          summ1=summ1+n1
          print(s1)
          add1()
```

```
-----
UnboundLocalError                                Traceback (most recent call last)
Cell In[32], line 5
      3     summ1=summ1+n1
      4     print(s1)
----> 5 add1()

Cell In[32], line 3, in add1()
      1 def add1():
      2     n1=100
----> 3     summ1=summ1+n1
      4     print(s1)

UnboundLocalError: cannot access local variable 'summ1' where it is not associated with a value
```

```
In [33]: def add1():
          n1=100
          s1=summ1+n1
          print(s1)
          add1()
```

```
-----
NameError                                         Traceback (most recent call last)
Cell In[33], line 5
      3     s1=summ1+n1
      4     print(s1)
----> 5 add1()

Cell In[33], line 3, in add1()
      1 def add1():
      2     n1=100
----> 3     s1=summ1+n1
      4     print(s1)

NameError: name 'summ1' is not defined
```

```
In [ ]: def add1():
          n1=100
          s1=summ1+n1
          print(s1)
          add1()      # Name error summ1
          #####
          def add1():
              n1=100
              summ1=summ1+n1
              print(s1)
          add1()      # unbound local
          #####
```

```
def add1():
    summ1=0
    n1=100
    s1=summ1+n1
    print(s1)
add1()
#####
def add1():
    n1=100
    summ1=0
    summ1=summ1+n1
    print(s1)
add1()
```

```
In [34]: #####
def add1():
    summ1=0
    n1=100
    s1=summ1+n1
    print(s1)
add1()
```

100

```
In [36]: def add2():
    n1=100
    summ1=0
    summ1=summ1+n1
    print(s1)
add2()
```

100

```
In [37]: def add3():
    n1=100
    summ2=100
    summ2=summ2+n1
    print(summ2)
add3()
```

200

```
In [42]: def avg(a,b,c):
    global ADD,AVG1
    ADD=a+b+c
    AVG1=ADD/3
    print(ADD,AVG1)
avg(10,20,30)
```

60 20.0

```
In [43]: ADD,AVG1
```

Out[43]: (60, 20.0)

```
In [45]: c=3000
def add(a,b):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+c
```

```

    print(f'the summation of {a},{b} and {c} is : {summ}')
c=5000
add(100,200)

# st-1: c=3000
# st-2: def
# st-3: c=5000
# st-4: call
# st-5: run

```

a: 100  
b: 200  
c: 5000  
the summation of 100,200 and 5000 is : 5300

In [47]:

```

c=3000
def add(a,b):
    c=7000
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+c
    c=8000
    print(f'the summation of {a},{b} and {c} is : {summ}')
c=5000
add(100,200)

```

a: 100  
b: 200  
c: 7000  
the summation of 100,200 and 8000 is : 7300

In [48]:

```

c=3000
def add(a,b,c=10000):
    print('a:',a)
    print('b:',b)
    print('c:',c)
    summ=a+b+c
    c=8000
    print(f'the summation of {a},{b} and {c} is : {summ}')
c=5000
add(100,200,9000)

```

a: 100  
b: 200  
c: 7000  
the summation of 100,200 and 8000 is : 7300

In [1]:

```

a=10
b=20
a,b=b,a
print(a)
print(b)

```

20  
10

- Functions with out arguments
- Function with arguments

- Function default arguments
- Local vs global variable

```
In [ ]: # wap ask the user enter a number
        # print it is an even or odd

        #a) Function with out argument with default
```

```
In [3]: def even_odd():
        num=eval(input('enter a num:'))
        if num%2==0:
            print(f'{num} is an even')
        else:
            print(f'{num} is an odd')

        even_odd()
```

20 is an even

```
In [4]: def even_odd1(num):
        if num%2==0:
            print(f'{num} is an even')
        else:
            print(f'{num} is an odd')

        even_odd1(101)
```

101 is an odd

```
In [9]: def even_odd2(num1=202):
        global num1
        if num1%2==0:
            print(f'{num1} is an even')
        else:
            print(f'{num1} is an odd')

        even_odd2()
```

```
Cell In[9], line 2
    global num1
    ^
SyntaxError: name 'num1' is parameter and global
```

```
In [8]: num1
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[8], line 1
----> 1 num1

NameError: name 'num1' is not defined
```

```
In [12]: def even_odd():
        global num2
        num2=eval(input('enter a num:'))
        if num2%2==0:
            print(f'{num2} is an even')
        else:
            print(f'{num2} is an odd')
```

```
even_odd()
```

205 is an odd

```
In [13]: num2
```

```
Out[13]: 205
```

```
In [ ]: def even_odd2(num1=202):
        global num1
        if num1%2==0:
            print(f'{num1} is an even')
        else:
            print(f'{num1} is an odd')

even_odd2()

def even_odd():
    global num2
    num2=eval(input('enter a num:'))
    if num2%2==0:
        print(f'{num2} is an even')
    else:
        print(f'{num2} is an odd')

even_odd()
```

### parameter vs Variable

- Parameters also called as Arguments
  - This always inside the function brackets
- Variables either outside the function or inside the function

```
In [ ]: **return**

- return means function is giving something

- we know that local variables can not use outside the function

- in order to use local variables outside the function, we have two methods

  - global

  - return
```

```
In [18]: def add():
        a=10
        b=20
        c=a+b
        return(c)

c=add()
```

```
In [19]: value
```

Out[19]: 30

```
In [ ]: value=10,20 # conf
        val1,val2=10,20 # works
        val1,val2=10 # fail
```

```
In [21]: value=10,20
        value # tuple
```

Out[21]: (10, 20)

```
In [22]: val1,val2=10,20
        val1
```

Out[22]: 10

```
In [23]: val2
```

Out[23]: 20

```
In [24]: val1,val2=10
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[24], line 1
----> 1 val1,val2=10

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

```
In [25]: def add():
        a=10
        b=20
        c=a+b
        return(a,c)

        a,c=add()
        print(a)
        print(c)
```

10  
30

```
In [28]: def avg(a,b,c):
        summ=a+b+c
        AVG=summ/3
        return(summ,AVG)

        summ,AVG=avg(10,20,30)
        print(summ)
        print(AVG)
```

60  
20.0

```
In [33]: def even_odd():
        num2=eval(input('enter a num:'))
        if num2%2==0:
            val=10
            print(f'{num2} is an even')
```

```

        return(num2+10)
    else:
        val=20
        print(f'{num2} is an odd')
        return(num2+20)

output=even_odd()
print(output)

```

1000 is an even  
1010

```

In [ ]: def even_odd():
        num2=eval(input('enter a num:'))
        if num2%2==0:
            val=10
            print(f'{num2} is an even')
        else:
            val=20
            print(f'{num2} is an odd')
        return(num2)

output=even_odd()
print(output)

```

```

In [34]: def func_event_odd2(n1):

        # n1=eval(input('Enter the number :'))
        if n1%2==0:
            return('Event Number')
        else:
            return('Odd Number')

Event_Odd = func_event_odd2(80)

```

```

In [36]: print Event_Odd

```

```

Cell In[36], line 1
    print Event_Odd
    ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?

```

### function in function

```

In [38]: def greet1():
        print('hello good morning')

        def greet2():
            print('hello good night')

        greet1()
        greet2()

```

hello good morning  
hello good night

```

In [39]: def greet1():
        print('hello good morning')

```



```
def greet2():  
    print('hello good night')  
    greet1()  
  
greet2()
```

hello good night  
hello good morning

In [40]:

```
In [41]: def greet11():  
        print('hello good morning')  
        greet22() # first time seeing error  
  
        def greet22():  
            print('hello good night')  
  
        greet11()
```

hello good morning  
hello good night

```
In [ ]: def greet11():  
        print('hello good morning')  
        greet22() # first time seeing error  
  
        def greet22():  
            print('hello good night')  
            greet11()  
  
        greet11()
```

```
In [42]: def greet3():  
        print('i dont know what im doing')  
        greet3()  
  
        greet3()
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

```

-----
RecursionError                                Traceback (most recent call last)
Cell In[42], line 5
      2     print('i dont know what im doing')
      3     greet3()
----> 5     greet3()

Cell In[42], line 3, in greet3()
      1 def greet3():
      2     print('i dont know what im doing')
----> 3     greet3()

Cell In[42], line 3, in greet3()
      1 def greet3():
      2     print('i dont know what im doing')
----> 3     greet3()

[... skipping similar frames: greet3 at line 3 (2967 times)]

Cell In[42], line 3, in greet3()
      1 def greet3():
      2     print('i dont know what im doing')
----> 3     greet3()

Cell In[42], line 2, in greet3()
      1 def greet3():
----> 2     print('i dont know what im doing')
      3     greet3()

File ~\anaconda3\Lib\site-packages\ipykernel\iostream.py:649, in OutStream.write
(self, string)
    646     msg = "I/O operation on closed file"
    647     raise ValueError(msg)
--> 649 is_child = not self._is_master_process()
    650 # only touch the buffer in the IO thread to avoid races
    651 with self._buffer_lock:

File ~\anaconda3\Lib\site-packages\ipykernel\iostream.py:520, in OutStream._is_ma
ster_process(self)
    519 def _is_master_process(self):
--> 520     return os.getpid() == self._master_pid

RecursionError: maximum recursion depth exceeded while calling a Python object

```

```

In [ ]: 5
        5+4 ===== 5+(5-1)
        5+4+3 ===== 5+4+(4-1)
        5+4+3+2
        5+4+3+2+1
        5+4+3+2+1+0

        n+n-1

```

```

In [ ]: def add(n):
        if n==0:
            return(0)
        else:
            return(n+add(n-1))

```

```

add(5)
step-1:  n=5  if F  else 5+add(4)
              5+4+add(3)
              5+4+3+add(2)
              5+4+3+2+add(1)
              5+4+3+2+1+add(0)
              5+4+3+2+1+0

```

```

In [43]: def add(n):
          if n==0:
              return(0)
          else:
              return(n+add(n-1))

add(5)

```

Out[43]: 15

```

In [ ]: # Q1) Factorial recursion 5! = 5*4*3*2*1*
        # Q2) calculator

        # step-1: create 4 functions
        # def add(a,b):
        #     return(a+b)
        # def mul
        # def div
        # def sub

        # print('enter 1 for add 2 sub so on')

        def main():
            option=eval(input('enter between 1 to 4'))
            if option==1:
                a=
                b=
                add()
                return
            elif option==2:
                a=
                b=
                mul()
                return

```