- Basics
- Data types

  14 data types

  int, float, strings,list,tuple,set,dictionary complex,range,bytearray
- Eval input
- type casting
- Print statements
- Basic python codes
- Conditional statements
- Try-except
- Functions

In [ ]:
```
python+statistics: EDA
python+ML
python+DL
python+NLP
python+GenAI

# Non IT : Practice more
# Spend time in the Lab

# AI python
# 5months ====6months
# Sat
```

Functions

- A block of code
- Reuseable
- a block of statements that return the specific task
- a named section of a code that performs a specific task

In [ ]:
```
bill pay and tax amount

bill_amount=20000
tax_per=10
tax_amount
total_bill
```

syntax

In [ ]:
```
#def  <write your function name>():
     #<write your code>
```

In [1]:
```
# write addition program
a=10
b=20
c=a+b
print(c)
```

```
30
```

In [5]:
```
def add():
    a=10
    b=20
    c=a+b
    print(c)
```

In [ ]:
```
# in order to see the output
# we need to call the function
# function name : add
```

In [3]:
```
add
```

Out[3]:
```
<function __main__.add()>
```

In [4]:
```
import random
random.randint
```

Out[4]:
```
<bound method Random.randint of <random.Random object at 0x000001D9FD2F8C40>>
```

- whenever we see function
- whenever we see bound method
- then add normal brackets
- functions or methods always with normal brackets only

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

```
30
```

In [7]:
```
#defining the function
def add():
    a=10
    b=20
    c=a+b
    print(c)

# call the function
add()
```

```
30
```

In [8]:
```
print(add())
# don't print the function name
# print is used for only see the output
```

```
30
None
```

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


add1()
```

```
the addition of 100 and 200 is: 300
```

In [11]:
```
a=eval(input('enter a num1:'))
b=eval(input('enter a num2:'))
c=a+b
print(f"the addition of {a} and {b} is: {c}")
```

```
the addition of 100 and 200 is: 300
```

In [12]:
```
import random
num1=random.randint(1,100)
num2=random.randint(2,200)
add=num1+num2
print(f"The addition of {num1} and {num2} is {add}")
```

```
The addition of 38 and 149 is 187
```

In [13]:
```
import random
def add2():
    num1=random.randint(1,100)
    num2=random.randint(2,200)
    add=num1+num2
    print(f"The addition of {num1} and {num2} is {add}")
```

In [14]:
```
add2()
```

```
The addition of 56 and 178 is 234
```

- Function name should not same as variable name
- Function name rules same as variable rules
- Function name and your file name should not be same
- Function name should not be same as package name

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

def add1():
    a=eval(input('enter a num1:'))
    b=eval(input('enter a num2:'))
    c=a+b
    print(f"the addition of {a} and {b} is: {c}")
```

```
def add2():
    num1=random.randint(1,100)
    num2=random.randint(2,200)
    add=num1+num2
    print(f"The addition of {num1} and {num2} is {add}")
```

In [15]:
```
def add3():
    a=eval(input('enter a num1:'))
    b=eval(input('enter a num2:'))
    c=a+b
    print(f"the addition of {aa} and {b} is: {c}")
```

In [16]:
```
add3()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 add3()

Cell In[15], line 5, in add3()
      3 b=eval(input('enter a num2:'))
      4 c=a+b
----> 5 print(f"the addition of {aa} and {b} is: {c}")

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

- when you define the function , if any error is there It will not show
- If you want to see the error or output
- You need to call the function only

In [ ]:
```
#Q2) WAP ask the user take height of the traingle
#                       breadth of the traingle
# calculate the area  of the right angle traingle
# Formual: 1/2*b*h

#Q3) Wap ask the user take the radius of circle
#    calculate the area of the circle
# Formula: pi*r*r  where pi=3.14

#Q4) wap ask the user take the length and breadth
#    calculate area of the rectangle
# Formulae:  lenghth * breadth

#Q5) Wap ask the how much bill he wants to pay
#                how much tip percentage he wants to give
#                calculate the total bill amount

# bill amount =1000
#  you want give 10% tip = 1000*10/100
# total bill= 1000+100=1100
```

In [19]:
```
#Q2) WAP ask the user take height of the traingle
#                       breadth of the traingle
# calculate the area  of the right angle traingle
```

```
# Formual: 1/2*b*h
def area_of_tri():
    height=eval(input('enter the height:'))
    breadth=eval(input("enter the breadth:"))
    area=0.5*height*breadth
    print(f"The Area of the Traingle is:",area)
```

In [20]:
```
area_of_tri()
```

```
The Area of the Traingle is: 300.0
```

In [21]:
```
# Q3)

def circle():
    radius=eval(input('enter the radius:'))
    pi=3.14
    area=pi*radius*radius
    print(f"The area of circle having {radius} is {area}")
    print("The area of a circle having {} is {}".format(radius,area))
```

In [22]:
```
circle()
```

```
The area of circle having 20 is 1256.0
The area of a circle having 20 is 1256.0
```

In [23]:
```
# Q4)

def rec_area():
    length=eval(input('enter the length:'))
    breadth=eval(input("enter the breadth:"))
    area=length*breadth
    print(f"The area of the rectangle  having {breadth},{length} is {area}")
    print("The area of the rectangle having {},{} is {}".format(breadth,length,a
```

In [24]:
```
rec_area()
```

```
The area of the rectangle  having 40,30 is 1200
The area of the rectangle having 40,30 is 1200
```

In [26]:
```
# Q5)
def bill_pay():
    bill_amount=eval(input('enter the bill_amount:'))
    tip_percentage=eval(input("enter the tip_percentage:"))
    tip_amount= bill_amount*tip_percentage/100
    total_bill= bill_amount+tip_amount
    print(f"The total bill is : {total_bill}")
```

In [27]:
```
bill_pay()
```

```
The total bill is : 1100.0
```

In [29]:
```
add1()
add2()
add3()
area_of_tri()
circle()
rec_area()
bill_pay()
```

```
the addition of 10 and 20 is: 30
The addition of 48 and 86 is 134
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[29], line 3
      1 add1()
      2 add2()
----> 3 add3()
      4 area_of_tri()
      5 circle()

Cell In[15], line 5, in add3()
      3 b=eval(input('enter a num2:'))
      4 c=a+b
----> 5 print(f"the addition of {aa} and {b} is: {c}")

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

- some thing we are not providing inside the brackets means
- It is called as **Functions with out arguments**

Function with arguments

- First check how many variables are there inside the function
- There are 3 varaibles
- In the how many input varaibles are there: 2 a,b
  - input varaible means user will provide
  - here a,b provided by user only
- In that how many output variables are there: 1 c
  - output variables are provided by code by using input variable
- Dont touch **Output variable**

In [ ]:
```
def add1():
    a=eval(input('enter a num1:'))
    b=eval(input('enter a num2:'))
    c=a+b
    print(f"the addition of {a} and {b} is: {c}")
```

In [37]:
```
def add4(a1,b1):
    c1=a1+b1
    print(f"the addition of {a1} and {b1} is: {c1}")

add4(1000,2000)
```

```
the addition of 1000 and 2000 is: 3000
```

In [38]:
```
def average():
    a=eval(input('enter a :'))
    b=eval(input('enter b:'))
```

```
    c=eval(input('enter c :'))
    avg=(a+b+c)/3
    avg1=round(avg,2)
    print(f"the average of {a} and {b} and {c} is:",avg1)

average()
```

```
the average of 10 and 20 and 30 is: 20.0
```

In [42]:
```
def average(a,b,c):
    avg=(a+b+c)/3
    avg1=round(avg,2)
    print(f"the average of {a} and {b} and {c} is:",avg1)

average(101,20,30)
```

```
the average of 101 and 20 and 30 is: 50.33
```

In [3]:
```
def area_of_tri(height,breadth):
    area=0.5*height*breadth
    print(f"The Area of the Traingle is:",area)

area_of_tri(20,20)
```

```
The Area of the Traingle is: 200.0
```

In [4]:
```
def circle(radius,pi):
    area=pi*radius*radius
    print(f"The area of circle having {radius} is {area}")
    print("The area of a circle having {} is {}".format(radius,area))

circle(30,3.14)
```

```
The area of circle having 30 is 2826.0
The area of a circle having 30 is 2826.0
```

In [5]:
```
def rec_area(length,breadth):
    area=length*breadth
    print(f"The area of the rectangle  having {breadth},{length} is {area}")
    print("The area of the rectangle having {},{} is {}".format(breadth,length,a

rec_area(40,50)
```

```
The area of the rectangle  having 50,40 is 2000
The area of the rectangle having 50,40 is 2000
```

In [6]:
```
def bill_pay(bill_amount,tip_percentage):
    tip_amount= bill_amount*tip_percentage/100
    total_bill= bill_amount+tip_amount
    print(f"The total bill is : {total_bill}")

bill_pay(1000,20)
```

```
The total bill is : 1200.0
```

In [7]:
```
num=eval(input('enter the number:'))
if num%2==0:
    print(f"the {num} is an even")

else:
    print(f"the {num} is an odd")
```

```
the 19 is an odd
```

In [8]:
```
def even_odd():
    num=eval(input('enter the number:'))
    if num%2==0:
        print(f"the {num} is an even")

    else:
        print(f"the {num} is an odd")

even_odd()
```

```
the 20 is an even
```

In [10]:
```
def even_odd1(num):
    if num%2==0:
        print(f"the {num} is an even")

    else:
        print(f"the {num} is an odd")

even_odd1(20)
```

```
the 20 is an even
```

- Try-exception should add inside the function call
- we already know that , defining the function will not give any error

In [12]:
```
def average(a,b,c):
    try:
        avg=(a+b+c)/3
        avg1=round(avg,2)
        print(f"the average of {a} and {b} and {c} is:",avg1)
    except Exception as e:
        print(e)
average('A',20,30)
```

```
can only concatenate str (not "int") to str
```

In [11]:
```
('A'+20+30)/3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 ('A'+20+30)/3

TypeError: can only concatenate str (not "int") to str
```

In [13]:
```
def circle(radius,pi):
    try:
        area=pi*radius*radius
        print(f"The area of circle having {radius} is {area}")
        print("The area of a circle having {} is {}".format(radius,area))
    except Exception as e:
        print(e)

circle(30,3.14)
```

```
The area of circle having 30 is 2826.0
The area of a circle having 30 is 2826.0
```

In [16]:
```python
import random
def evenoddd(num):
    try:
        if num%2==0:
            print(f"the given {num} is even")
        else:
            print(f"the given {num} is odd")

    except Exception as e:
        print(e)

evenoddd(10)
# what is the variable name: num
# Function with arguments
# Function with out
# what are the rules

# Inside function we need to provide variable names
# I want to see your notes
# Share the screenshot

# Im doing the mistake here
# Im doing the mistake (already one chance over )
#
```

```
the given 10 is even
```

In [14]:
```python
def average1():
    a=10
    b=20
    c=30
    avg=(a+b+c)/3
    print(avg)

average1()
```

```
20.0
```

In [15]:
```python
def average1(a):
    print("the value of a is:",a)
    b=20
    c=30
    avg=(a+b+c)/3
    print(avg)

average1(10)
```

```
the value of a is: 10
20.0
```

In [17]:
```python
def average1(a,b):
    print("the value of a is:",a)
    print("the value of b is:",b)
    c=30
    avg=(a+b+c)/3
    print(avg)

average1(10,20)
```

```
the value of a is: 10
the value of b is: 20
20.0
```

- Function with out arguments
- Function with arguments

Function with Default arguments

- There is a situation some variable values always a fixed
- for example tip percentage for a hote it is fixed 20
- These type varaibles are default parameters

In [19]:
```
def add(a,b,c):
    summ=a+b+c
    print(summ)

add(10,20,30)
```

```
60
```

- while defining function we will provide arguments
- at that time only we will provide value also
- that argument is default argument
- below examples there 3 arguments a,b,c
- but we given c=100 which means c is our default argument
- once you provide the default argument
- while calling the function no need to provide c value again

In [21]:
```
def add(a,b,c=100):
    print("the value a is:",a)
    print("the value b is:",b)
    print("the value b is:",c)
    summ=a+b+c
    print(summ)

add(10,20)
```

```
the value a is: 10
the value b is: 20
the value b is: 100
130
```

In [ ]:
```
# how many variables are there: 3
# how many default arguments are there: 1
```

```
# how many non default arguments are there:2
# how many values we need to pass while calling the function:2
```

Case-1

- Default arguments always at last
- Do not provide in middile

In [22]:
```
def add(a,b=100,c):
    print("the value a is:",a)
    print("the value b is:",b)
    print("the value b is:",c)
    summ=a+b+c
    print(summ)

add(10,20)
```

```
  Cell In[22], line 1
    def add(a,b=100,c):
                    ^
SyntaxError: non-default argument follows default argument
```

In [ ]:
```
a,b,c=100 ============ c
a,b=100,c ============ W
a=100,b,c ============ w
a,b=100,c=100 ======= C
a=100,b=100,c ====== w
a=100,b,c=100 ===== w
a=100,b=100,c=100 === c
```

Case-2

In [24]:
```
def add(a,b,c=100):
    print("the value a is:",a)
    print("the value b is:",b)
    print("the value c is:",c)
    summ=a+b+c
    print(summ)

add(10,20,200)
```

```
the value a is: 10
the value b is: 20
the value c is: 200
230
```

- If we already provided default value
- But while calling the fucntion again we are providing the value
- The value will be overwrite
- In above example while define the function c=100
- but while calling the function c=200

- python is a step by step process
- In above there 3 operations
  - define the function
  - call the function
  - running the function
- while define the function c=100
- while call the function c=200
- so the latest values is 200

In [27]:
```
def add(a,b,c=100):
    print('hai')
    c=1000
    summ=a+b+c
    print(summ)

add(10,20,200)
```

```
hai
1030
```

In [28]:
```
def add(a,b,c=100):
    print('hai')
    c=10
    summ=a+b+c
    print(summ)

add(10,20,1000)

# define: 100
# call: 1000
# Run: 10
```

```
hai
40
```

In [29]:
```
add1(10,20,1000)  # call

def add1(a,b,c=100):
    print('hai')
    c=10
    summ=a+b+c
    print(summ)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 add1(10,20,1000)  # call
      3 def add1(a,b,c=100):
      4     print('hai')

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

In [30]:
```
def add(a,b,c=100):
    print('hai')
    c=10
    summ=a+b+2000
    print(summ)

add(10,20,1000)
# 100 === > 1000 === > 10 ==== > 2000
```

```
hai
2030
```

- Functions with out argument
- Functions with argument
- Functions with Default argunment

In [1]:
```
def average(a,b,c=100):
    avg=(a+b+c)/3
    avg1=round(avg,2)
    print(f"The average of {a},{b} and {c} is {avg1}")
```

In [2]:
```
average
```

Out[2]:
```
<function __main__.average(a, b, c=100)>
```

In [4]:
```
average(30,40)
```

```
The average of 30,40 and 100 is 56.67
```

In [5]:
```
def value(a=10):
    print(a)
```

In [11]:
```
value(100)
```

```
100
```

Analogy with packages

In [ ]:
```
#########################################################
# A data type is complex
#########################################################
# By default complex has real=0 imag=0
# When we dont provide any value by default
# 0+0j === > 0j
################################################
# Take your curosr inside the bracket
# apply shift+tab
################################################
# If any function or method
# We dont give any values still is giving output
# Which means that function has default values
```

In [10]:
```
complex()
```

Out[10]:
```
0j
```

In [ ]:
```
0+0J
0
0j
NT
```

In [17]:
```
# I want 2+3j
complex(2+3j,4)
```

Out[17]: (2+7j)

In [13]:
```
complex()
```

Out[13]: 0j

In [15]:
```
value(),value(100)
```

```
10
100
```

Out[15]: (None, None)

In [18]:
```
complex(2,2+4j)
```

Out[18]: (-2+2j)

In [ ]:
```
value()  # 10 ?
# already we given sir  where?
# while define the function
# what is this method: Default argument
```

In [19]:
```
# 100 will come
# We need to change the function default value
# while calling the function we need to provide the value
value(100)
```

```
100
```

In [20]:
```
complex()
```

Out[20]: 0j

In [21]:
```
complex(2,3)
```

Out[21]: (2+3j)

In [23]:
```
import random
random.randint(10,20)

# why it is not giving answers like complex
```

Out[23]: 16

In [26]:
```
random.random()
```

Out[26]: 0.03417216421176261

In [25]:
```
def value1():
    print(100)
```

```
value1()
```

```
100
```

In [ ]:
```
- complex

- random.randint

- random.random
```

In [27]:
```
complex('2','2')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 complex('2','2')

TypeError: complex() can't take second arg if first is a string
```

In [ ]:
```
# type your code
# paste in chat
# i will take it
# type in my laptop

# where you defined
```

In [28]:
```
def value():
    print(100)

value()
```

```
100
```

In [ ]:
```
#default value is 10 but I am not getting it
```

In [29]:
```
complex(2,2+4j)
```

Out[29]: (-2+2j)

In [30]:
```
complex(3,2+4j)
```

Out[30]: (-1+2j)

In [ ]:
```
4j*j === 4j^2 ==== 4*-1 === -4

2-4= -2
```

Global variable - Local Variable

In [33]:
```
def add1():
    a=10
    b=20
    c=a+b
    print(c)
add1()
```

```
30
```

In [34]:
```
a1=100
b1=200
def add2():
    c1=a1+b1
    print(c1)
add2()
```

```
300
```

- Variables inside function is called as **Local variables**
- Variables outside function is called as **Global variable**
- Global variable can use any where at any time
- Imagine that you created 10 functions all 10 functions using same variables
- So why we want intialize every time for evey function
- we will intailize one time at top , and all 10 functions wil use

In [36]:
```
def add():
    a=10
    b=20
    print(a+b)

def mul():
    a=10
    b=20
    print(a*b)

add()
mul()
```

```
30
200
```

In [37]:
```
a=100
b=200
def add():
    print(a+b)

def mul():
    print(a*b)

add()
mul()
```

```
300
20000
```

In [39]:
```
# Case-4
#######################################
c=500
###############################################
def add(a,b,c=100):
    summ=a+b+c
    print(summ)
```

```
add(10,20)
# intializtion=c
# Define the function
# call the function
# Run the function
# c=500 === 100 === 100 === 100
```

```
130
```

In [40]:
```
# Case-5

c=500
def add(a,b):
    summ=a+b+c
    print(summ)

add(10,20)
```

```
530
```

In [41]:
```
# Case-6

c=500
def add(a,b):
    summ=a+b+c
    print(summ)
c=700
add(10,20)
# Python is a step by step process
# First c=500
# When we defined there is no c, c=500
# after that c=700 ===
# while calling the function No c value= 700
# While running the c value not there c=700
# 730
```

```
730
```

In [43]:
```
# Case-7
c=500
def add(a,b,c=1000):
    summ=a+b+c
    print(summ)
c=700
add(10,20,2000)
c=800
```

```
2030
```

In [45]:
```
# Case-8
c=500
def add(a,b,c=1000):
    c=9000
    summ=a+b+c
    print(summ)
c=700
add(10,20,2000)
c=800
```

```
9030
```

In [ ]:
```
# Global means outside the function
# local varaibles
```

In [1]:
```
def add():
    a=10
    b=20
    summ=30
    print(summ)
```

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

```
30
```

In [3]:
```
a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 a

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

In [4]:
```
summ
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 summ

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

- local variables can not use outside the function
- here a, b, summ are local variables
- there are working and we are able to print the answer also
- but when we want print **summ**, **a**, **b**
- we are not able to access the values

In [10]:
```
n1=20
n2=40
def add1():
    summ1=n1+n2
    print(summ1)

add1()
print('n1:',n1)
print('n2:',n2)
print(summ1)
```

```
60
n1: 20
n2: 40
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 10
      8 print('n1:',n1)
      9 print('n2:',n2)
---> 10 print(summ1)

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

In [5]:
```
sum([1,2,3])

# do not use sum as variable
# keep that in mind
```

Out[5]: 6

How to use local variable outside the function

- we already know that local variable can not use outside the function call
- But we can use those local variables outside function call
- by using a keyword called **global**

In [14]:
```
def add():
    global summ

    a=10
    b=20
    summ=30
    print(summ)

add()
print("summ is:",summ)
```

```
30
summ is: 30
```

In [17]:
```
def add_sub():
    global summ,sub
    #global sub
    a=10
    b=20
    summ=a+b
    sub=a-b
    print(summ,sub)

add_sub()
print("summ is:",summ)
print("sub is:",sub)
```

```
30 -10
summ is: 30
sub is: -10
```

use case

- imagine we intialize a value with sum number

- we updated that value inside function
- but when we want to use that value outisde function
- we will not able to use

In [18]:
```
age=0

def age_update():
    age=30
    print(age)

age_update()
```

```
30
```

In [19]:
```
age
```

Out[19]:
```
0
```

In [20]:
```
age=0

def age_update():
    global age
    age=30
    print(age)

age_update()
```

```
30
```

In [21]:
```
age
```

Out[21]:
```
30
```

In [27]:
```
x=10

def fun1():
    print(x)
    x=30

fun1()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[27], line 7
      4     print(x)
      5     x=30
----> 7 fun1()

Cell In[27], line 4, in fun1()
      3 def fun1():
----> 4     print(x)
      5     x=30

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

- First check local variable and global variable both have same name

- Inside function with out intializing are we using local variable

In [32]:
```
s=0
def fun2():
    print(s)
    s=s+10
    print(s)

fun2()
```

```
20
```

case-1

In [33]:
```
x=10

def fun1():
    print(x)

fun1()

# No error
```

```
10
```

case-2

In [34]:
```
x=10

def fun1():
    x=30
    print(x)

fun1()

# No error
```

```
30
```

Case-3

In [ ]:
```
x=10

def fun1():
    print(x)
    x=30

fun1()

# Error will come
```

Case-4

In [ ]:
```
s=0
def fun2():
    s=s+10
    print(s)
```

```
fun2()

# Error
```

In [35]:
```
s=0
def fun2():
    global s
    s=s+10
    print(s)

fun2()
```

```
10
```

In [ ]:
```
x2=20
def add2():
    print(x2)
add2()


#First check local variable and global variable both have same name

#Inside function with out intializing are we using local variable
```

In [ ]:
```
n1=20
n2=40
def add1():
    summ1=n1+n2
    print(summ1)
```

In [36]:
```
x=10

def fun1():
    global x
    print(x)
    x=30

fun1()
```

```
10
```

In [ ]:
```
x=10

def fun1():
    x=10
    print(x)
    x=30

fun1()
```

In [ ]:
```
x=10

def fun1():
    print(x)
    x=30

fun1()


x=10
```

```
def fun1():
    x=10
    print(x)
    x=30

fun1()


x=10

def fun1():
    global x
    print(x)
    x=30

fun1()
```

In [37]:
```
n1=10
n2=20
n3=30
def fun3():
    n1=100
    n2=n1+n2  ####### Error #####
    n3=n1+n2
    print(n1,n2,n3)

fun3()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[37], line 10
      7     n3=n1+n2
      8     print(n1,n2,n3)
---> 10 fun3()

Cell In[37], line 6, in fun3()
      4 def fun3():
      5     n1=100
----> 6     n2=n1+n2
      7     n3=n1+n2
      8     print(n1,n2,n3)

UnboundLocalError: cannot access local variable 'n2' where it is not associated w
ith a value
```

In [40]:
```
n1=10
n2=20
n3=30
def fun3():
    n1=n2  # 20  n1=20
    n2=n2+n3 # 20+30=50

fun3()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[40], line 8
      5     n1=n2
      6     n2=n2+n3
----> 8 fun3()

Cell In[40], line 5, in fun3()
      4 def fun3():
----> 5     n1=n2
      6     n2=n2+n3

UnboundLocalError: cannot access local variable 'n2' where it is not associated w
ith a value
```

In [43]:
```
a=10
b=20
c=30
def fun4():
    a=b+c
    b=c+a
    c=c
    print(a,b,c)

fun4()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[43], line 10
      7     c=c
      8     print(a,b,c)
---> 10 fun4()

Cell In[43], line 5, in fun4()
      4 def fun4():
----> 5     a=b+c
      6     b=c+a
      7     c=c

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

In [45]:
```
c=0
def fun5():
    a=c+1
    print(c)

fun5()
```

```
0
```

Return

- As of now we are defining functions
- we are calling the functions
- if any operation we are performing inside function
- we are able to use that values using **global** keyword

- we also can use that values using **return** statement

In [54]:
```
def average(a,b,c):
    avg=(a+b+c)/3
    return(avg)

# The above function performing
# avg calculation and it is returning
# that return value we are storing in a variable
# called AVG

# generally we will keep the variable same as
# retutn variable only

avg=average(10,20,30)
```

In [53]:
```
avg
```

Out[53]: 20.0

In [55]:
```
def fun5(a,b,c):
    avg=(a+b+c)/3
    summ=a+b+c
    return(avg,summ)

fun5(20,30,40)
```

Out[55]: (30.0, 90)

In [ ]:
```
a=10
b=20
a=10,20
a,b=10,20
a,b=10
```

In [57]:
```
a=10,20
a
```

Out[57]: (10, 20)

In [59]:
```
a,b=10,20
a
b
```

Out[59]: 20

In [60]:
```
a,b=10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[60], line 1
----> 1 a,b=10

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

In [61]:
```
def fun5(a,b,c):
    avg=(a+b+c)/3
```

```
    summ=a+b+c
    return(avg,summ)

values=fun5(20,30,40)
values
```

Out[61]: (30.0, 90)

In [62]:
```
def fun5(a,b,c):
    avg=(a+b+c)/3
    summ=a+b+c
    return(avg,summ)

value1,value2=fun5(20,30,40)
print(value1)
print(value2)
```

```
30.0
90
```

In [1]:
```
100<10
```

Out[1]: False

In [2]:
```
100>10
```

Out[2]: True

In [4]:
```
# WAP create the function
# For bill pay
# bill amount
# tip per
# calculate total bill
# return the total bill

def bill_pay(bill_amount,tip_per):
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(total_bill)

total_bill=bill_pay(1000,10)
print("The total bill is:",total_bill)
```

```
The total bill is: 1100.0
```

In [6]:
```
def bill_pay(bill_amount,tip_per):
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(tip_amount,total_bill)

tip_amount,total_bill=bill_pay(1000,10)
print("The total bill is:",total_bill)
print("The tip amount is:",tip_amount)
```

```
The total bill is: 100.0
The tip amount is: 1100.0
```

In [7]:
```
def bill_pay(bill_amount,tip_per):
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
```

```
    return(tip_amount,total_bill)

total_bill=bill_pay(1000,10)
print("The total bill is:",total_bill)
```

```
The total bill is: (100.0, 1100.0)
```

In [ ]:
```
def bill_pay(bill_amount,tip_per):
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(total_bill)

total_bill=bill_pay(1000,10)
print("The total bill is:",total_bill)
#####################################################
def bill_pay(bill_amount,tip_per):
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(tip_amount,total_bill)

tip_amount,total_bill=bill_pay(1000,10)
print("The total bill is:",total_bill)
print("The tip amount is:",tip_amount)
#####################################################
def bill_pay(bill_amount,tip_per):
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(tip_amount,total_bill)

total_bill=bill_pay(1000,10)
print("The total bill is:",total_bill)
```

In [9]:
```
def bill_per():
    bill = eval(input("Enter bill amount: "))
    tip_per = eval(input("Enter tip % wrt bill (1 - 100): "))
    tip = (bill/100)*tip_per
    total_bill = bill+tip
    return (total_bill, tip)

total_bill=bill(1000,10)
print('Total bill:',total_bill)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 7
      5     total_bill = bill+tip
      6     return (total_bill, tip)
----> 7 total_bill=bill_per(1000,10)
      8 print('Total bill:',total_bill)

TypeError: bill_per() takes 0 positional arguments but 2 were given
```

In [14]:
```
def fun():
    a=10
    b=20
    return(a,b)

v1,v2=fun()
```

In [15]:
```
v1
```

Out[15]: 10

In [16]:
```
v2
```

Out[16]: 20

In [ ]:
```
fun() ==== with out argument
fun(a,b) === with arg
```

In [18]:
```
def bill_pay():
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(total_bill)

bill_amount= eval(input())
tip_per= eval(input())

total_bill=bill_pay()
print("The total bill is:",total_bill)

# Step-1: Define the function
# Step-2: Taking the bill amount
# Step-3: taking the tip per
# Step-4: call
# Step-5: run

#you defined the funciton, but you are not passing input parameters to the funct
```

```
The total bill is: 2400.0
```

In [ ]:
```
bill_amount= eval(input())
tip_per= eval(input())


def bill_pay():
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(total_bill)


total_bill=bill_pay()
print("The total bill is:",total_bill)

# Step-1: Taking the bill amount
# Step-2: taking the tip per
# Step-3: Define the function
# Step-4: call
# Step-5: run
```

- Global variables can intialise anywhere
- By the time of the calling function
- Is variables are available or not ?
- If the variables are available then that function works

In [ ]:
```
def bill_pay(bill_amount,tip_per):
    tip_amount=bill_amount*tip_per/100
    total_bill=tip_amount+bill_amount
    return(total_bill)
total_bill=bill_pay(1000,10)
print('the total bill is:',total_bill)
print('the tip amount is:',tip_amount)

# gobal var
# not returned
# you are trying to use
# local varaibles can not use outside function call
```

In [20]:
```
def bill_pay1():
    tip_amount1=bill_amount1*tip_per1/100
    total_bill1=tip_amount1+bill_amount1
    return(total_bill1)


total_bill1=bill_pay1()

bill_amount1= eval(input())
tip_per1= eval(input())

print("The total bill is:",total_bill1)

# Step-1: Taking the bill amount
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[20], line 7
      3     total_bill1=tip_amount1+bill_amount1
      4     return(total_bill1)
----> 7 total_bill1=bill_pay1()
      9 bill_amount1= eval(input())
     10 tip_per1= eval(input())

Cell In[20], line 2, in bill_pay1()
      1 def bill_pay1():
----> 2     tip_amount1=bill_amount1*tip_per1/100
      3     total_bill1=tip_amount1+bill_amount1
      4     return(total_bill1)

NameError: name 'bill_amount1' is not defined
```

- Function with arguments
- Function with arguments
- Function with default arguments
- Local variable vs global variable
- global keyword
- unbound local error

- return
- function in functions

In [24]:
```
def greet1():
    print('hello good morning')

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

greet1()
greet2()
```

```
hello good morning
good night!
```

In [25]:
```
def greet1():
    print('hello good morning')
    greet2()

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

greet1()

# Step-1: define the greet1
# Step-2: define the greet2
# step-3: calling the greet1()
# step-4: run the greet1()
#         print ==== hello good mo
#         calling the greet2()
#         print ===== good night
```

```
hello good morning
good night!
```

In [26]:
```
def greet1():
    print('hello good morning')

def greet2():
    print('good night!')
    greet1()

greet2()
```

```
good night!
hello good morning
```

In [ ]:
```
################## Infinite loop#################
def greet1():
    print('hello good morning')
    greet2()

def greet2():
    print('good night!')
    greet1()

greet2()
```

task-1

- Create a function1
- take the 3 values from user inside the function
- num1= eval(input()),num2= eval(input()),num2= eval(input())
- return those three values
- Create a function2
- inside function2 call the function1
- we alreday know function1 returning 3 values
- so here store in three variables
- and find the average and return it

In [30]:
```
def fun1():
    n1=eval(input('enter the num1:'))
    n2=eval(input('enter the num2:'))
    return(n1,n2)

def Average():
    print("======== Average starts ==============")
    n1,n2=fun1()
    avg=(n1+n2)/2
    return(avg)
avg=Average()
print("The avergae is:",avg)

def addition():
    print("======== Addition stats ==============")
    n1,n2=fun1()
    add=(n1+n2)
    return(add)
add=addition()
print("The addition is:",add)

def subutraction():
    print("======== Subtraction starts ==============")
    n1,n2=fun1()
    sub=(n1-n2)
    return(sub)
sub=subutraction()
print("The subtraction is:",sub)

def multiplicaton():
    print("======== Multiplication stats ==============")
    n1,n2=fun1()
    mul=(n1*n2)
    return(mul)

mul=multiplicaton()
print("The multiplication is:",mul)
```

```
======== Average starts ==============
```

```
The avergae is: 15.0
======== Addition stats ==============
The addition is: 50
======== Subtraction starts ==============
The subtraction is: 10
======== Multiplication stats ==============
The multiplication is: 200
```

In [ ]:

```
# Step-1: Create a add function
          #return the values
# Example
     #def add(a,b):
         #return(a+b)

# step-2: mul
# step-3: sub
# step-4: div

# step-5: if-else calculator
#         print the statement
#         enter 1 ==== add
#         enter 2 ==== sub
#         enter 3===== mul
#         enter 4 ===== div

# step-6: ask the user enter the option
# step-7 : ask the user enter two values a,b
# step-8: if option==1:
#            =add()
#            print the value
# step-9
# step-10
# step-11
```