Functions

- Reuse of the code
- Imagine you want to calculate tax of all tax payers
- Suppose there are 100 tax payers are there
- In order to calculate tax we required 5 lines of code
- for 100 tax payers it will required 500 lines
- this is not the good appraoch
- because 5lines we are repaeting for 100 members
- Here Functions will help us how to use only these 5 lines to all 100 tax payers

In [2]:
```
salary=eval(input('enter the salary:'))
tax_per=eval(input('enter the tax per:'))
tax_amount= salary*tax_per/100
print(f"the amount of tax pay is {tax_amount}")
```

```
the amount of tax pay is 6000.0
```

In [ ]:
```
#syntax

# def <function_name>():
#     <code stats here>
```

In [3]:
```
n1=10
n2=20
add=n1+n2
print(add)
```

```
30
```

In [11]:
```
def addition():
    n1=10
    n2=20
    add=n1+n2
    print(add)
```

In [12]:
```
addition()
```

```
30
```

- first mistake you will forget the brackets while define function
  - syntax error
- second mistake you will forget colon
  - syntax error
- third mistake you will not give indetation

    - indetation error
- Fourth mistake, you will copy the code you will forget to make the allignment
    - indetation error
- fifth mistake: variable name and functions name must be different
    - This error you will get in future, when we create apps
- sixt mistake: while calling the function you will forget the brackets

Function also called a method

Whenever function is there brackets must

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

In [13]:
```
def add1():
    n1=eval(input("enter the number1:"))
    n2=eval(input("enter the number2:"))
    add=n1111+n2
    print(add)
```

In [14]:
```
add1()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 add1()

Cell In[13], line 4, in add1()
      2 n1=eval(input("enter the number1:"))
      3 n2=eval(input("enter the number2:"))
----> 4 add=n1111+n2
      5 print(add)

NameError: name 'n1111' is not defined
```

Note

- when we define function will not give the error
- erros will display after calling the function only

Method-1

functions with out argument

- when we define function we will provide brackets
- inside these brackets we can provide the values also

- these values are called as **arguments** or **parameters**
- if we dont provide argumnets it is known as
    - **function with out arguments**

In [ ]:
```
# Q1) Create a function read three numbers find the average

# Q2) Create a function ask the user enter the bill amount
      # enter the tip per
      # calculate total bill pay

# Q3) Create a function ask the user enter enter radius
     # calculate the area of the circle

# Q4) Create a function ask the user enter breadth and height
    # calculate area of the traingle

# Q5) Create a function ask the user enter a number find it is even or odd

#  Q6) create a function ask the user enter number find it is postive or ngeativ

# Q7) cratae a function ask the user enter threes numbers find the greatet numbe
```

In [15]:
```
#Create a function to get 3 numbers and print average
def average():
    num1 = eval(input('Enter number1: '))
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average()
```

```
Average of 20, 30,40 is 30.0
```

In [17]:
```
def billing():
    bill=eval(input("enter the bill amount:"))
    tip_per=eval(input("enter the tip percentage:"))
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing()
```

```
1100.0
```

In [18]:
```
# create a function ask the user enter radius and calculate area of the circle#
import math
def circle():
    r=eval(input("Enter radius of the circle"))
    pi=math.pi
    area=round(pi*r*r,2)
    print(f"area of the circle is {area}")

circle()
```

```
area of the circle is 1256.64
```

In [19]:
```
# Get breadth and height. Calculate are of the triangle
def areaOfTriangle():
```

```
    breadth = eval(input('Enter breadth: '))
    height = eval(input('Enter height: '))
    area = round(breadth * height /2, 2)
    print(f'Area of the Triangle : {area}')
areaOfTriangle()
```

```
Area of the Triangle : 200.0
```

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

In [21]:
```
# first write the orignal code
# then create the function
# then copy the original code inside function

def eve_odd():
    num = eval(input('Enter a number: '))
    if(num%2 == 0):
        print(f'{num} is an even number')
    else:
        print(f'{num} is an odd number')

eve_odd()
```

```
20 is an even number
```

In [24]:
```
# Find if a given number is positive or negative
def positive_negative():
    num = eval(input('Enter a number: '))
    if(num>0):
        print(f'{num} is a positive number')
    elif(num==0):
        print(f'{num} is a zero')
    else:
        print(f'{num} is a negative number')
positive_negative()
```

```
-90 is a negative number
```

In [26]:
```
def greatest():
    num1 = eval(input("Enter the number1:"))
    num2 = eval(input("Enter the number2:"))
    num3 = eval(input("Enter the number3:"))
    if num1>num2 and num1>num3:
        print(f"{num1} is greatest")
    elif num2>num3:
        print(f"{num2} is greatest")
    else:
        print(f"{num3} is greatest")
greatest()
```

```
30 is greatest
```

In [ ]:
```
def average():
def billing():
def circle():
def areaOfTriangle():
def eve_odd():
```

```
def positive_negative():
def greatest():
```

Method-2

function with arguments

- how many variables are there in a given function
  - input variables : user will provide
  - suppose n1 and n2 are two input variables
  - output variables: we are creating a new varible using input variable

concentrate on input variables

In [ ]:
```
def add1():
    n1=eval(input("enter the number1:"))
    n2=eval(input("enter the number2:"))
    add=n1+n2
    print(add)
```

In [35]:
```
def add2(n111,n222):
    add=n111+n222
    print(add)

add2(10,30)
```

```
40
```

In [36]:
```
def average(num1,num2,num3):
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(10,20,30)
```

```
Average of 10, 20,30 is 20.0
```

In [ ]:
```
# Perform the with argumnets concept
# all other 6 problems
```

- Functions with out arguments
- Functions with arguments

In [ ]:
```
#Create a function to get 3 numbers and print average
# with out arguments
def average():
    num1 = eval(input('Enter number1: '))
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average()
```

In [1]:
```
def average(num1,num2,num3):
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(10,20,30)
```

```
Average of 10, 20,30 is 20.0
```

In [4]:
```
def average(num1,num2):
    print("num1:",num1)
    print("num2:",num2)
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(10,20)
```

```
num1: 10
num2: 20
Average of 10, 20,30 is 20.0
```

In [5]:
```
def average(num1):
    print("num1:",num1)
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(10)
```

```
num1: 10
Average of 10, 20,30 is 20.0
```

In [8]:
```
def average(num1):
    print("num1:",num1)
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(eval(input("enter the number1:")))
```

```
num1: 100
Average of 100, 200,400 is 233.33
```

- hard coded
- eval(input())
- random.randint()

In [9]:
```
import random
def average(num1):
    print("num1:",num1)
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(random.randint(1,100))
```

```
num1: 91
Average of 91, 100,20 is 70.33
```

In [ ]:
```
def average(num1,num2,num3):
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(10,20,30)

def average(num1):
    print("num1:",num1)
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(10)

def average(num1):
    print("num1:",num1)
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(eval(input("enter the number1:")))

import random
def average(num1):
    print("num1:",num1)
    num2 = eval(input('Enter number2: '))
    num3 = eval(input('Enter number3: '))
    ave = round((num1+num2+num3)/3,2)
    print(f'Average of {num1}, {num2},{num3} is {ave}')
average(random.randint(1,100))
```

In [ ]:
```
#average(10)
#average(eval(input("enter the number1:")))
#average(random.randint(1,100))
```

In [ ]:
```
# Try above things for bill and tip per problem
def billing():
    bill=eval(input("enter the bill amount:"))
    tip_per=eval(input("enter the tip percentage:"))
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing()
```

In [10]:
```
# Try above things for bill and tip per problem
def billing(bill,tip_per):
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing(1000,10)
```

```
1100.0
```

In [11]:
```
def billing(bill,tip_per):
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)
```

```
billing(eval(input("enter bill:")),
        eval(input("enter the tip per:")))
```

```
1100.0
```

In [13]:
```
import random
def billing(bill,tip_per):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing(random.randint(1000,2000),
        random.randint(10,20))
```

```
the bill is: 1256
the tip per is: 10
1381.6
```

In [16]:
```
import random
def billing():
    bill=eval(input("enter the bill amount:"))
    tip_per=eval(input("enter the tip percentage:"))
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing()
```

```
1100.0
```

Default argument

- argument values are fixed

In [19]:
```
# Try above things for bill and tip per problem
def billing(bill,tip_per=20):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing(1000)
```

```
the bill is: 1000
the tip per is: 20
1200.0
```

In [18]:
```
def bill(amount, tip=20):
    print(f"Total bill with tip is {amount+ (amount*tip/100)}")

bill(random.randint(100,200) )
```

```
Total bill with tip is 156.0
```

case-1: Make all the arguments as default

- all the arguments are default

- so no need to pass the value while calling the function

In [ ]:
```
def billing(bill=1000,tip_per=20):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing()
```

Case-2: All default parameters should assign after non default arguments

In [20]:
```
# here non default argument: tip_per
# assigned after default argument: bill=1000
# when we call the function function will assign the value for bill only
def billing(bill=1000,tip_per):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing(20)
```

```
  Cell In[20], line 1
    def billing(bill=1000,tip_per):
                          ^
SyntaxError: non-default argument follows default argument
```

In [21]:
```
def billing(tip_per,bill=1000):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing(20)
```

```
the bill is: 1000
the tip per is: 20
1200.0
```

In [ ]:
```
average(n1,n2,n3=10)  # c
average(n1,n2=10,n3)  # f
average(n1=10,n2,n3) # f
average(n1,n2=10,n3=10) # c
average(n1=10,n2,n3=10) # f
average(n1=10,n2=10,n3) # f
average(n1=10,n2=10,n3=10) # c
```

Case-3: Default argumnents values will be override

In [23]:
```
def billing(bill,tip_per=20):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)
```

```
billing(1000,40)
```

```
the bill is: 1000
the tip per is: 40
1400.0
```

In [22]:
```
n1=100
n1=200
n1
```

Out[22]: 200

In [24]:
```
def billing(bill=1000,tip_per=20):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing(2000,40)
```

```
the bill is: 2000
the tip per is: 40
2800.0
```

In [25]:
```
def billing(bill=1000,tip_per=20):
    bill=5000
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

billing(2000,40)
```

```
the bill is: 5000
the tip per is: 40
7000.0
```

In [ ]:
```
# step-1: define the function: 1000
# step-2: call the function : 2000
# step-3: run the function  : 5000
```

In [26]:
```
def billing(bill=1000,tip_per=20):
    bill=5000
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

bill=7000
billing(2000,40)

#bill=1000
#bill=7000
#bill=2000
#bill=5000
```

```
the bill is: 5000
the tip per is: 40
7000.0
```

In [27]:
```python
def billing(bill=1000,tip_per=20):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

bill=7000
billing(2000,40)
```

```
the bill is: 2000
the tip per is: 40
2800.0
```

In [28]:
```python
bill=8000
def billing(bill=1000,tip_per=20):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

bill=7000
billing(2000,40)

# bill=8k
#  1k
# 7k
#  2k
```

```
the bill is: 2000
the tip per is: 40
2800.0
```

In [29]:
```python
bill=8000
def billing(tip_per=20):
    print("the bill is:",bill)
    print("the tip per is:",tip_per)
    tip_amount=(bill*tip_per)/100
    total=bill+tip_amount
    print(total)

bill=7000
billing(40)
```

```
the bill is: 7000
the tip per is: 40
9800.0
```

Global variable - Local variable

- local variable means the variables define inside the function
- local variables can not access outside the function
- global variables means the variables define outside the function

- global variable can access anywhere anytime

In [32]:
```
def addition():
    NUMBER1=10
    NUMBER2=20
    print(NUMBER1+NUMBER2)

addition()
```

```
30
```

In [33]:
```
NUMBER1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[33], line 1
----> 1 NUMBER1

NameError: name 'NUMBER1' is not defined
```

In [37]:
```
NUMBER11=100
NUMBER22=200
def addition():
    print(NUMBER11+NUMBER22)


def mul():
    print(NUMBER11*NUMBER22)
mul()
addition()
```

```
20000
300
```

In [35]:
```
NUMBER11
```

Out[35]:
```
100
```

In [39]:
```
NUMBER11=100
NUMBER22=200
def addition():
    print("number1:",NUMBER11)
    print("number2:",NUMBER22)
    add=NUMBER11+NUMBER22
    print(add)
NUMBER11=1000
NUMBER22=2000
addition()
```

```
number1: 1000
number2: 2000
3000
```

In [41]:
```
NUMBER11=100
NUMBER22=200
def addition(NUMBER22):
    NUMBER11=1000
    print("number1:",NUMBER11)
    print("number2:",NUMBER22)
    add=NUMBER11+NUMBER22
```

```
    print(add)
NUMBER11=1000
NUMBER22=2000
addition(500)
```

```
number1: 1000
number2: 500
1500
```

Spl case

- we want to use the local variable outside function also

In [46]:
```
def addition():
    global number_one1
    number_one1=10
    number_two=20
    print(number_one1+number_two)
print(number_one1)
addition()

# i defined variable as global inside function
# with out call the function also
# i can use that variable anywhere
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[46], line 6
      4     number_two=20
      5     print(number_one1+number_two)
----> 6 print(number_one1)
      7 addition()

NameError: name 'number_one1' is not defined
```

In [45]:
```
number_one
```

Out[45]: 10

In [47]:
```
def addition():
    global number_one1,number_two
    number_one1=10
    number_two=20
    print(number_one1+number_two)

addition()
print(number_one1)
```

```
30
10
```

In [ ]:
```
if we want to declare all the define variables as global sir what we can do
```

In [ ]:

In [ ]:

In [ ]:

In [38]:
```
addition()
```

```
300
```

- Functions with out arguments
- Functions with arguments
- Functions with default arguments
- Local variable and global variable
- Return statements
- Function in functions

Average Program

- Functions with out arguments
- Functions with arguments
- Functions with default arguments
- Local variable and global variable

In [2]:
```
def avg():
    n1=eval(input("enter the number1:"))
    n2=eval(input("enter the number2:"))
    n3=eval(input("enter the number3:"))
    average=(n1+n2+n3)/3
    average1=round(average,2)
    print(f"the average of {n1}, {n2} and {n3} is {average1}")

avg()

# input:n1  n2  n3
# output:average average1
```

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

In [4]:
```
n1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 n1

NameError: name 'n1' is not defined
```

In [5]:
```
n1=eval(input("enter the number1:"))
n2=eval(input("enter the number2:"))
n3=eval(input("enter the number3:"))
def avg():
    average=(n1+n2+n3)/3
    average1=round(average,2)
    print(f"the average of {n1}, {n2} and {n3} is {average1}")
```

```
avg()
```

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

In [6]:
```
n1
```

Out[6]:
```
10
```

In [7]:
```
def avg(n1,n2,n3):
    average=(n1+n2+n3)/3
    average1=round(average,2)
    print(f"the average of {n1}, {n2} and {n3} is {average1}")

avg(10,20,30)
```

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

In [8]:
```
def avg(n1,n2,n3=30):
    average=(n1+n2+n3)/3
    average1=round(average,2)
    print(f"the average of {n1}, {n2} and {n3} is {average1}")

avg(10,20)
```

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

In [ ]:
```
def avg():
    average=(n1+n2+n3)/3
    average1=round(average,2)
    print(f"the average of {n1}, {n2} and {n3} is {average1}")

n1=eval(input("enter the number1:"))
n2=eval(input("enter the number2:"))
n3=eval(input("enter the number3:"))
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [9]:
```
def add():
    print(a+b)

add()
a=10
b=20
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[9], line 4
      1 def add():
      2     print(a+b)
----> 4 add()
      5 a=10
      6 b=20

Cell In[9], line 2, in add()
      1 def add():
----> 2     print(a+b)

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

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

```
30
```

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

add()
a=10
b=20
###################Works####################
def add():
    print(a+b)
a=10
b=20
add()
####################Fail#####################
def add():
    print(a+b)
    a=10
    b=20
add()
#####################Works#####################
a=10
b=20
def add():
    print(a+b)
add()
#####################works#####################
def add():
    a=10
    b=20
    print(a+b)
add()
```

In [11]:
```
def add():
    print(a1+b1)

add()
```

```
a1=10
b1=20
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 4
      1 def add():
      2     print(a1+b1)
----> 4 add()
      5 a1=10
      6 b1=20

Cell In[11], line 2, in add()
      1 def add():
----> 2     print(a1+b1)

NameError: name 'a1' is not defined
```

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

In [ ]:
```
~~~
num=10;
def show(num):
    #print(num) ##  Here I want to print global variable value
    print(num)

show(20)
~~~

Q# Is there any way to print global variable value  inside function , if there i
```

In [27]:
```
s1=10
def add1():
    s2=s1+20   # 10+20
    print(s2)
add1()
```

```
30
```

In [26]:
```
s
```

Out[26]:
```
20
```

In [ ]:
```
x=5
def add():
    print(x+y)
    x=10
    y=20
    print(f"Value of X is : {x}")
add()
```

In [30]:
```
sum1=10
def add1():
    global sum1
```

```
    sum1=20
add1()
```

In [31]:
```
sum1
```

Out[31]: 20

In [ ]:
```
def show():
    global a
    a=20
    a=10
    print(a)
a=10
show()
```

In [32]:
```
a=10
b=20
def add():
    a=16
a=17
add()
print(a)

# a=10 17  16
```

17

In [35]:
```
NUMBER11=100
NUMBER22=200
def addition():
    NUMBER11=2000
    print("number1:",NUMBER11)
    print("number2:",NUMBER22)
    add=NUMBER11+NUMBER22
    print(add)
NUMBER11=1000
NUMBER22=2000
addition()

print(NUMBER11)
```

```
number1: 2000
number2: 2000
4000
1000
```

In [34]:
```
def addition():
    NUMBER11=2000
    print("number1:",NUMBER11)
    print("number2:",NUMBER22)
    add=NUMBER111+NUMBER22
addition()
```

```
number1: 2000
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[34], line 6
      4     print("number2:",NUMBER22)
      5     add=NUMBER111+NUMBER22
----> 6 addition()

Cell In[34], line 4, in addition()
      2 NUMBER11=2000
      3 print("number1:",NUMBER11)
----> 4 print("number2:",NUMBER22)
      5 add=NUMBER111+NUMBER22

NameError: name 'NUMBER22' is not defined
```

Use the local variables outside function with out using global keywords

- We already know that local variables cant use outside the function directly
- If we want to use local variables outside the function we need to use global keyword
- We can also use the local variables with out using global keyword
- that concept is called **return**

In [37]:
```
def add1():
    num1=10
    num2=20
    summ=num1+num2
    return(num1)

var1=add1()
```

In [38]:
```
var1
```

Out[38]: 10

In [45]:
```
def add1():
    num1=10
    num2=20
    s11=num1+num2
    return(s11,num1,num2)

s11,num1,num2=add1()
print(s11)
print(num1)
print(num2
```

```
30
10
20
```

In [55]:
```
def avg():
    n1=eval(input("enter the number1:"))
    n2=eval(input("enter the number2:"))
    n3=eval(input("enter the number3:"))
    average=(n1+n2+n3)/3
    average1=round(average,2)
```

```
    print(f"the average of {n1}, {n2} and {n3} is {average1}")
    print(f"the average of {n1}, {n2} and {n3} is {average}")
    return(average1,average)

AVERAGE1,AVERAGE=avg()
```

```
the average of 1, 10 and 9 is 6.67
the average of 1, 10 and 9 is 6.666666666666667
```

In [51]:
```
print(AVERAGE)
```

```
40.0
```

In [56]:
```
def add1():
    num1 = 10
    num2 = 20
    s111 = num1+num2
    print(s111, num1, num2)
s111 = add1()
print(s111)
```

```
30 10 20
None
```

In [ ]:
```
import random
def add1():
    num1=10
    num2=20
    s11=eval(input("enter the number"))
    s22=random.randint(1,1)
    s33=num1+num2
    s44=num1*s11
    return(s11,s22,s33,s44,num1,num2) #############
s11,s22,s33,s44,num1,num2=add1()####################
print(s11)
print(s22)
print(s33)
print(s44)
print(num1)
print(num2)
```

In [57]:
```
def avg():
    n1=eval(input("Enter the 1st number: "))
    n2=eval(input("Enter the 2nd number: "))
    n3=eval(input("Enter the 3rd number: "))
    avg=(n1+n2+n3)/3
    avg1=round(avg,2)
    return(avg,avg1)

print(f"The average of the three numbers is:, avg")
```

```
The average of the three numbers is:, avg
```

In [ ]:
```
# Now the assignment 7qns
# Return statmenst

def mul():
    a=100
    b=700

    multiplication=a*b
```

```
    return(a,multiplication)
multilpication,a=mul()
print(multiplication,a)
```