```
    print('The value is',e)
```

```
Enter the value:45 90,04.8-45/45
SyntaxError
```

```
# OS module - Operating system

import os

# mkdir - going to make the directory(Folder)

os.mkdir('D:\\Empty')

# getcwd - get the name of the current working directory

os.getcwd()
```

```
'C:\\Users\\ELCOT\\Desktop\\Python Anaconda'
```

```
# rmdir - its going to remove the directory

os.rmdir('D:\\Empty')

os.rename('D:\\python programming\\sec.txt','D:\\python programming\\
ABC.txt')

os.remove('D:\\python programming\\ABC.txt')
```

## DAY-23 AUG-12

```
#OOPs - class is a key word. class is a collection of object

# Function in python

def is a key word for function

# Function Program


def find_max(x,y,z):
    max_number = max(x,y,z)
    return max_number

maximum = find_max(56,78,90)
print('The maximum number is:',maximum)
```

```
The maximum number is: 90
```

```
# write a python function to find the max of three numbers
```

```python
def max_of_two(a,b):
    if a>b:
        return a
    return b
def max_of_three(a,b,c):
    return max_of_two(a, max_of_two(b,c))
print(max_of_three(23,45,67))
```

```
67
```

```python
# write a python function to sum all the numbers in a list.

def sum(numbers):
    total = 0
    for x in numbers:
        total+= x
    return total
print(sum((2,5,7,8,9,45,77)))
```

```
153
```

```python
# write a python function that accepts a string and calculate the
number of upper case letters
# and lower case letters

def string_test(s):
    d = {'UPPER_CASE':0, 'LOWER_CASE':0}
    for c in s:
        if c.isupper():
            d['UPPER_CASE']+=1
        elif c.islower():
            d['LOWER_CASE']+=1
        else:
            pass
    print ('original string:',s)
    print('No. of upper case characters : ',d['UPPER_CASE'])
    print('No. of lowere case characters : ',d['LOWER_CASE'])

string_test('Battle Through The Heaven')
```

```
original string: Battle Through The Heaven
No. of upper case characters :  4
No. of lowere case characters :  18
```

```python
# write a python function to calculate the factorial of a number

def factorial(n):
    result = 1
    for i in range(1, n+1):
        result *=i
    return result
print(factorial(7))
```

```
5040
```

```
# write a python function to check whether a given number is prime.


def is_prime(n):
    if n<2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n%i ==0:
            return False
    return True
print(is_prime(11))
```

```
True
```

```
# write a python fumction function to reverse a string

def reverse_string(s):
    return s[::-1]
print(reverse_string('ShinChan'))
```

```
nahCnihS
```

```
# write a python function to find the sum of all elementys in a list

def sum_list(l):
    total = 0
    for num in l:
        total += num
    return total
print (sum_list([12,34,56,78,90]))
```

```
270
```

```
# write a python function to count the number of vowels in a string

def count_vowels(s):
    vowels = 'aeiouAEIOU'
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count
print (count_vowels('Enter Backspace'))
```

```
5
```

```python
# write a python function to sum all even number in a list

def sum_even_numbers(list):
    even_sum= 0
    for num in list:
        if num%2==0:
            even_sum += num
    return even_sum
print(sum_even_numbers([12,23,34,45,56,67,45,33,44,24,80]))
```

```
250
```

```python
# write a function to remove vowels from the string

def remove_vowels(string):
    vowels = 'aeiouAEIOU'
    new_string = ''
    for char in string:
        if char.lower() not in vowels:
            new_string += char
    return new_string
print(remove_vowels('Apple is good for Health'))
```

```
ppl s gd fr Hlth
```

```python
# filter long list

def filter_long_string(s):
    long_string = []
    for string in s:
        if len(string)>4:
            long_string.append(string)
    return long_string
print(filter_long_string(['bun','jam','Butter','Dynamite']))
```

```
['Butter', 'Dynamite']
```

```python
# pic out the 2nd largest number

def second_largest(l):
    largest = None
    second_largest = None
    for num in l:
        if largest is None or num > largest:
            second_largest = largest
            largest = num
        elif second_largest is None or num>second_largest:
            second_largest = num
    return second_largest
print (second_largest([12,45,55,77]))
```

```
55
```

## DAY-24 AUG-13

```
# Create Empty class

class dress:
    pass

# d - object for class name dress
d=dress()
d.dressname = 'Chudi'
d.dresscolour = 'Black'
d.dresssize = 'XL'

print('I want', d.dressname,'in ', d.dresscolour,'colour with size
',d.dresssize)

print(d.__dict__)
```

```
I want Chudi in  Black colour with size  XL
{'dressname': 'Chudi', 'dresscolour': 'Black', 'dresssize': 'XL'}
```

```
# how to declare variable inside the class

class phone:
    phonename = 'Samsung Galaxy'
    model = 'S21'
    colour = 'Black'
ph = phone()
print(f'I got a {ph.phonename} phone model {ph.model} in a {ph.colour}
colour')
```

```
I got a Samsung Galaxy phone model S21 in a Black colour
```

```
# Method declaration

class fan:
    def __init__(self,group,fan):
        self.g=group
        self.f=fan
    def display (self):
        print(f'My fav k-pop group is {self.g}.And their fandom name 
is {self.f}')
f = fan('BTS','ARMY')
f.display()
```

```
My fav k-pop group is BTS.And their fandom name is ARMY
```

```
# Creating class and objects with methods

class dog:
    attr1 = 'Mammal'
    def __init__(self,name):
        self.n = name
    def speak(self):
        print('My dog name is {}'.format(self.n))
Bam = dog('Bam')
Tan = dog('Tan')
Bam.speak()
Tan.speak()
```

```
My dog name is Bam
My dog name is Tan
```

## DAY-25 AUG-19

```
# sum of two no:

class add:
    #Method Declaration
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def display (self):
        print(f'sum of {self.x} and {self.y} is {self.x+self.y}')
x=int(input('Enter the number x:'))
y=int(input('Enter the number y:'))
ad=add(x,y)
ad.display()
```

```
Enter the number x:5
Enter the number y:7
sum of 5 and 7 is 12
```

```
# class div:
    #Method Declaration
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def display (self):
        print(f'sum of {self.x} and {self.y} is {self.x /self.y}')
x=int(input('Enter the number x:'))
y=int(input('Enter the number y:'))
ad=div(x,y)
ad.display()
```

## INHERITANCE

### Single Inheritance

```
class parent:
    def __init__(self,Fname,Mname):
        self.F = Fname
        self.M = Mname
    def show (self):
        print(f'My Father name is {self.F}, and my Mother name is
{self.M}')

class child(parent):
    def __init__(self,Fname,Mname,Dname):
        self.F = Fname
        self.M = Mname
        self.D = Dname
    def display(self):
        print(f'My name is {self.D} My Father name is {self.F}, and
Mother name is {self.M}')
ch=child('Rajan','Vijey Lakshmi','Lavanya Sree')
ch.display()

My name is Lavanya Sree My Father name is Rajan, and Mother name is
Vijey Lakshmi
```

### Multiple Inheritance

```
class parent1:
    def method1(self):
        print ('This is parent class 1')
class parent2:
    def method2(self):
        print('This is parent class 2')
class child (parent1,parent2):
    def method3(self):
        print ('This is child class inherited from parent1 and
parent2')

ch = child()
ch.method1()
ch.method2()
ch.method3()
```

```
This is parent class 1
This is parent class 2
This is child class inherited from parent1 and parent2
```

## DAY - 26 AUG-26

## MULTIPLE INHERITANCE

```
class parent():
    def __init__(self,gname):
        self.gn=gname
class child1(parent):
    def __init__(self,fname):
        self.fn=fname

class child2 (child1):
    def __init__(self,sname,fname,gname):
        self.gn=gname
        self.fn=fname
        self.sn=sname
    def display (self):
        print (f'My Grand Father name is {self.gn}, My Father name is 
{self.fn}, And My name is {self.sn}')
ch=child2('Vinayak','Krishna','Shiva')
ch.display()
```

```
My Grand Father name is Shiva, My Father name is Krishna, And My name 
is Vinayak
```

```
class parent:
    def func1(self):
        print('This function is in parent class.')

class child1 (parent):
    def func2(self):
        print('This function is in child class 1.')

class child2 (parent):
    def func3(self):
        print('This function is in child class 2.')
```

```
ch1=child1()
ch2=child2()
ch1.func1()
ch1.func2()
ch2.func1()
ch2.func3()
```

```
This function is in parent class.
This function is in child class 1.
This function is in parent class.
This function is in child class 2.
```

```
class parent:
    def func1(self):
        print('This function is in parent class.')

class child1 (parent):
    def func2(self):
        print('This function is in child class 1.')

class child2 (child1):
    def func3(self):
        print('This function is in child class 2.')


ch1=child1()
ch2=child2()
ch1.func1()
ch1.func2()
ch2.func1()
ch2.func3()
ch2.func2()
```

```
This function is in parent class.
This function is in child class 1.
This function is in parent class.
This function is in child class 2.
This function is in child class 1.
```

## HIERARCHIAL INHERITANCE

```
class worker():
    def __init__ (self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary
```

```
class worker1(worker):
    def  __init__ (self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary


class worker2(worker1):
    def  __init__ (self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary
w1 = worker('Lavanya Sree',20,50000)
w2 = worker ('Anusuya',23,520000)

print(w1.name)
print(w2.salary)

print(f'My name is {w1.name}, {w1.age} years old, My salary is
{w1.salary}')

print(f'My name is {w2.name}, {w2.age} years old, My salary is
{w2.salary}')

Lavanya Sree
520000
My name is Lavanya Sree, 20 years old, My salary is 50000
My name is Anusuya, 23 years old, My salary is 520000
```

## HYBRID INHERITANCE

```
class Arith:
    def __init__ (self,a,b):
        self.a = a
        self.b = b
class cal1 (Arith):
    def add (self):
        return self.a+self.b
class cal2 (cal1):
    def sub (self):
        return self.a-self.b
class cal3 (cal2):
    def mul (self):
        return self.a*self.b
class cal4 (cal3):
    def div (self):
        return self.a/self.b
```

```python
a=int(input('Enter the number a:'))
b=int(input('Enter the number b:'))
cal=cal4(a,b)

print(f'Addition of a & b is {cal.add()}')
print(f'Subtraction of a & b is {cal.sub()}')
print(f'Multiplication of a & b is {cal.mul()}')
print(f'division of a & b is {cal.div()}')
```

```
Enter the number a:70
Enter the number b:7
Addition of a & b is 77
Subtraction of a & b is 63
Multiplication of a & b is 490
division of a & b is 10.0
```

```python
class school:
    def func1(self):
        print('This function is in school.')
class student1(school):
    def func2(self):
        print('This function is mainly for student1.')
class student2(student1):
    def func3(self):
        print('This function is mainly for student2.')
class student3(student2):
    def func4(self):
        print('This function is mainly for student3.')


st=student3()
st.func1()
st.func3()
st.func2()
st.func4()
```

```
This function is in school.
This function is mainly for student2.
This function is mainly for student1.
This function is mainly for student3.
```

## DAY-27 AUG-27

## POLYMORPHISM

```python
class India:
    def Capital(self):
        print('New Delhi is the capital city of India')
    def Language(self):
        print('Hindi is the National Language of India')
    def Type(self):
        print('India is The Developing Country')
class USA:
    def Capital(self):
        print('Washington is the Capital city of USA')
    def Language(self):
        print('English is the commonly used Language')
    def Type(self):
        print('USA is Developed Country')
Ind = India()
USA =USA()
for country in (Ind,USA):
    country.Capital()
    country.Language()
    country.Type()
```

```
New Delhi is the capital city of India
Hindi is the National Language of India
India is The Developing Country
Washington is the Capital city of USA
English is the commonly used Language
USA is Developed Country
```

## ENCAPSULATION(HIDE THE INFO - INTERNAL FUNCNALITY)

```
class Bank:
    def __init__(self,name,Accno,salary):
        self.n = name
        self.acc = Accno
        self.salary = salary
    def display(self):
        print(f'My name is {self.n} am getting salary of 
{self.salary}, and My Account no is {self.acc}')
B = Bank('Lavanya Sree',1234567891007,70000)
B.display()
```

```
My name is Lavanya Sree am getting salary of 70000, and My Account no 
is 1234567891007
```

```
# HIDE .__

class Bank:
    def __init__(self,name,Accno,salary):
        self.n = name
        self.__acc = Accno # HIDE INFO NY .__
        self.salary = salary
    def display(self):
        print(f'My name is {self.n} am getting salary of
{self.salary}')
        print(f'My Account no is {self.acc}')
B = Bank('Lavanya Sree',1234567891007,70000)
B.display()

My name is Lavanya Sree am getting salary of 70000

----------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
Cell In[3], line 12
     10         print(f'My Account no is {self.acc}')
     11 B = Bank('Lavanya Sree',1234567891007,70000)
---> 12 B.display()

Cell In[3], line 10, in Bank.display(self)
      8 def display(self):
      9     print(f'My name is {self.n} am getting salary of
{self.salary}')
---> 10     print(f'My Account no is {self.acc}')

AttributeError: 'Bank' object has no attribute 'acc'
```

## ABSTRACTION - ABC - Abstract Bsae Class

```
from abc import ABC, abstractmethod
class car (ABC):
    def mileage(self):
        pass
class Hyundai(car):
    def mileage(self):
        print('Hyundai gives the mileage of 27 kmpl')
class BMW(car):
    def mileage(self):
        print('BMW gives the mileage of 25 kmpl')
H = Hyundai()
H.mileage()

B = BMW()
B.mileage()
```

```
Hyundai gives the mileage of 27 kmpl
BMW gives the mileage of 25 kmpl

# SUBCLASS

class parent:
    def mother(self):
        print('My Mother name is: Vijey Lakshmi')
class child(parent):
    def son (self):
        print('Son name is : Danoj Kumar')
issubclass(child,parent)


True

class parent:
    def mother(self):
        print('My Mother name is: Vijey Lakshmi')
class child():
    def son (self):
        print('Son name is : Danoj Kumar')
issubclass(child,parent)


False
```

## DAY - 28 SEP - 2

```
# program to illustrate class variable to keep count of number of
object created

class sample:
    num = 0 # initialize the value as 0
    def __init__(self,vrbls):
        sample.num +=1
        self.vrbls = vrbls
        print('The object value is:',self.vrbls)
        print('The count of object created is:',self.num)
v1 = sample(5)
v2 = sample(7)
v3 = sample(77)

The object value is: 5
The count of object created is: 1
The object value is: 7
The count of object created is: 2
The object value is: 77
The count of object created is: 3
```

```python
# CONSTRUCTOR & DESTRUCTOR IN OOPS


class icecream:

    # CONSTRUCTOR
    def __init__(self,flavour):
        print('Inside Constructor')
        self.flavour = flavour
        print('Object Initialized')
    def show(self):
        print(f'I want {self.flavour} icecream')



    # DESTRUCTOR
    def __del__(self):
        print('Inside Destructor')
        print('Object Destroyed')
# create object
f1 = icecream('Vennila falvour')
f1.show()
# destroy object
del f1
f2 = icecream('Chocolate falvour')
f2.show()
f1.show()
```

```
Inside Constructor
Object Initialized
I want Vennila falvour icecream
Inside Destructor
Object Destroyed
Inside Constructor
Object Initialized
I want Chocolate falvour icecream
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[2], line 27
     25 f2 = icecream('Chocolate falvour')
     26 f2.show()
---> 27 f1.show()

NameError: name 'f1' is not defined
```

```python
class calculation:
    def __init__(self,a,b):
        self.a = a
        self.b = b
    def add(self):
        return self.a+self.b
a=int(input('Enter the number a:'))
b=int(input('Enter the number y:'))
cal=calculation(a,b)
print(cal.add())
print(cal+cal)
```

```
Enter the number a:23
Enter the number y:34
57

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[3], line 11
      9 cal=calculation(a,b)
     10 print(cal.add())
---> 11 print(cal+cal)

TypeError: unsupported operand type(s) for +: 'calculation' and
'calculation'
```

```
# OPERATOR OVERLOADING

class calculation:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def __add__(self,other):
        return self.x+other.y
x=int(input('Rnter the number x:'))
y=int(input('Enter the number y:'))
cal=calculation(x,y)
print('The Addition of x and y is:',cal+cal)
print('The Multiplication of x and y is:',cal.x*cal.y)
print('The division of x and y is:',cal.x/cal.y)
print('The Modulus of x and y is:',cal.x%cal.y)
```

```
Rnter the number x:70
Enter the number y:5
The Addition of x and y is: 75
The Multiplication of x and y is: 350
The division of x and y is: 14.0
The Modulus of x and y is: 0
```

```
class car:
    def __init__(self,make,model):
        self.make= make
        self.model= model
    def start(self):
        print(f'The {self.make} {self.model} is starting.')
    def stop(self):
        print(f'The {self.make} {self.model} is stopping.')


class ElectricCar(car):
    def __init__(self,make,model,battery_capacity):
        super().__init__(make, model)
        self.battery_capacity = battery_capacity
    def start(self):
        print(f'The {self.make} {self.model} is starting.')
    def display_battery_capacity(self):
        print(f'The battery capacity of {self.make} {self.model} is 
{self.battery_capacity}.')
my_car = car('BMW','k7')
my_car.start()
my_car.stop()

my_electric_car = ElectricCar('Hyundai','Jk26',100)
my_electric_car.start()
my_electric_car.stop()
my_electric_car.display_battery_capacity()
```

```
The BMW k7 is starting.
The BMW k7 is stopping.
The Hyundai Jk26 is starting.
The Hyundai Jk26 is stopping.
The battery capacity of Hyundai Jk26 is 100.
```

```
class BankAccount:
    def __init__(self,name,balance):
        self.name = name
        self.balance = balance

    def deposit(self,amount):
        self.balance += amount
        print(f'{amount} deposited in your account. New balance: 
{self.balance}')
    def withdraw (self, amount):
        if self.balance >= amount:
            self.balance -= amount
            print(f'{amount} withdraw from your account. New balance: 
{self.balance}')
```

```
        else:
            print('Insufficient funds.')
    def display_balance(self):
        print(f'Your current balance is: {self.balance}')
# CREATE AN OBJECT OF THE BANK ACCOUNT CLASS

account = BankAccount('LS',150000)

# DEPOSIT 5000 IN THE ACCCOUNT
account.deposit(5000)

# WITHDREAW 500 FROPM THE ACCOUNT
account.withdraw(500)

# DISPLAY THE CURRENT BALANCE
account.display_balance()

5000 deposited in your account. New balance: 155000
500 withdraw from your account. New balance: 154500
Your current balance is: 154500
```

## DAY =29 SEP -3

```
# A PYTHON PROGRAM TO CREATE A STATIC METHOD THAT CALCULATION THE
SQUARE ROOT OF A GIVEN NUMBER

import math

class square:
    # Static method declaration
    def calculation(x):
        result=math.sqrt(x)
        return result
# Accept a number from keyboard
num= float(input('Enter the number:'))
sq=square.calculation(num)
print(f'square root of {num} is {sq:.2f}')


Enter the number:49
square root of 49.0 is 7.00

# A python program to access base class construct and method in the
sub class using super().
# super(). = Value Inherited


class square:
    def __init__(self,x):
        self.x=x
    def area (self):
        print('Area of square is=',self.x*self.x)
class rectangle(square):
    def __init__(self,x,y):
        super().__init__(x)
        self.y=y
    def area(self):
        super().area()
        print('Area of Rectangle is=',self.x*self.y)
a,b=[float(x) for x in input('Enter TWO measures:').split()]
r=rectangle(a,b)
r.area()
```

```
Enter TWO measures:5 7
Area of square is= 25.0
Area of Rectangle is= 35.0
```

```
# A python program to overload greater that(>) operator to make it act
on class objects
#gt-(>), lt-(<)

class Ramayan:
    def __init__(self,pages):
        self.pg = pages
    def __gt__(self,other):
        return self.pg>other.pg

class Mahabharat:
    def __init__(self,pages):
        self.pg=pages
Ram = Ramayan(1200)
Mah = Mahabharat(2000)

if (Ram>Mah):
    print('Ramayan has more pages')
else:
    print('Mahabharat has more pages')
```

```
Mahabharat has more pages
```

## DAY -30 SEP- 9

```
# REULAR EXPRESSION

import re

# SEARCH
```