13. Write a Program to illustrate the use of str(), repr(), new, doc, dict, name and bases methods

In [31]:

```
import math
class Person:
    '''This is a DPerson class doc-string'''

    def __new__(cls):
        print("__new__ method called.. ")
        return super().__new__(cls)

    def __init__(self):
        self.name=input("Enter Your Name: ")
        self.age=int(input("Enter your age: "))

    def __str__(self):
        return f"Name of Person= {self.name}"

    def __repr__(self):
        return f"Age of Person = {self.age}"



p=Person()
print(str(p))
print(repr(p))
print(Person.__doc__)
print(p.__dict__)
```

```
__new__ method called..
Enter Your Name: Abhinay
Enter your age: 18
Name of Person= Abhinay
Age of Person = 18
This is a DPerson class doc-string
{'name': 'Abhinay', 'age': 18}
```

15. Write a program to illustrate the use of following built-in methods:

a. hasattr(obj,attr) b. getattr(object, attribute_name [, default]) c. setattr(object, name, value) d. delattr(class_name, name)

In [ ]:

```

```

16. Write a program to create class Employee. Display the personal information and salary details of 5 employees using

single inheritance.

In [1]:

```
class Employee:

    def __init__(self,name,designation,salary):
        self.name= name
        self.designation = designation
        self.salary= salary

    def display(self):
        print(f"Name ={self.name}, Designation = {self.designation}, Salary ={self.sala

class Member(Employee):

    def __init__(self,name,designation,salary):
        self.name= name
        self.designation=designation
        self.salary=salary
        super().__init__(self.name,self.designation,self.salary)

m1=Member('Abhinay','CEO',28662)
m2=Member('Jack','abc',28662)
m3=Member('Abhinay','CEO',28662)
m4=Member('Abhinay','CEO',28662)
m5=Member('Abhinay','CEO',28662)
m1.display()
m2.display()
m3.display()
m4.display()
m5.display()

```

```
Name =Abhinay, Designation = CEO, Salary =28662
Name =Jack, Designation = abc, Salary =28662
Name =Abhinay, Designation = CEO, Salary =28662
Name =Abhinay, Designation = CEO, Salary =28662
Name =Abhinay, Designation = CEO, Salary =28662
```

In [ ]:

```

```

17

In [ ]:

```
class Employee:
    data={"Name":['Abhinay','Babli','Abhinav'],
          "Designation":['CEO','Saaf safai karmchari','CO-Founder'],
          "Salary":[10000000,501,1000],
            'TL_Name':['Meenal','Meenal','Abhinay'],
           "M_name":['Prateek','Abhinay','Abhinay']
        }
    def __init__(self):
        self.name= input("Enter Employee Name: ")
        self.des= input("Enter Employee Designation: ")
        self.sal = int(input("Enter Salary: "))
        self.TL_name= input("Enter Team Leader name of Employee: ")
        self.M_name= input("Enter Manager Name of Employee: ")


        ##ADding EMp info to data
        Employee.data['Name'].append(self.name)
        Employee.data['Designation'].append(self.des)
        Employee.data['Salary'].append(self.sal)
        Employee.data['TL_Name'].append(self.TL_name)
        Employee.data['M_name'].append(self.M_name)



class Manager(Employee):
    def __init__(self):
        super().__init__()

    def get_name(self):
        name=input("Enter Manager name: ")
        n=len(Manager.data["M_name"])
        for i in range(n):

            if Manager.data["M_name"][i]==name:
                print(Manager.data["Name"][i])


class TeamLeader(Employee):
    def __init__(self):
        super().__init__()

    def get_name(self):
        name=input("Enter Team Leader name: ")
        n=len(TeamLeader.data["TL_Name"])
        for i in range(n):

            if TeamLeader.data["TL_Name"][i]==name:
                print(TeamLeader.data["Name"][i])




t=TeamLeader()
t.get_name()
```

```
Enter Employee Name: abhi
```

18 Write a program that has a class Point. Define another class Location which has two objects (Location and destination) of class Point. Also, define a function in Location that prints the reflection on the y-axis

In [3]:

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y


class Location(Point):
    def __init__(self,x,y):
         super().__init__(x,y)

    def location(self):
        print(f"Reflection on y axis = ({-self.x},{self.y})")

    def destination(self):
        print(f"Destination from origin = {  ((self.x)**2 + (self.y)**2 )**0.5  }")

p=Location(3,4)
p.location()
p.destination()


```

```
Reflection on y axis = (-3,4)
Destination from origin = 5.0
```

19.

In [ ]:

```
class Student:

    def __init__(self,marks):
        self.name = input("Enter Name: ")
        self.age= int(input("Enter your age: "))


    def display_info(self):
        print(f" Student Name= {self.name}, Student Age= {self.age}")


class Marks(Student):

    def __init__(self):

        self.marks1 = int(input("Enter marks in first subject: "))
        self.marks2= int(input("Enter marks in first subject: "))
        self.marks3= int(input("Enter marks in first subject: "))
        super().__init__([self.marks1,self.marks2,self.marks3])



    def display_marks(self):
        print(f"Marks1 ={self.marks1}, Marks2 ={self.marks2},Marks3  ={self.marks3}")

class Result(Marks):
    def __init__(self,total):
        self.t= total
        super().__init__()


    def display_result(self):
        print("Result =", (   (self.marks1 + self.marks2 +  self.marks3 )  / self.t ) *

r=Result(300)
r.display_info()
r.display_marks()
r.display_result()
```

20

In [2]:

```
class Distance:
    def __init__(self,km,m):
        self.km=km
        self.m=m

    def distance(self):
        return self.km*1000 + self.m

class School(Distance):
    def __init__(self,km,m):
        super().__init__(km,m)

    def display(self):
            print(f"Distance Between House and School = {self.distance()} m")

class Office(Distance):
    def __init__(self,km,m):
        super().__init__(km,m)

    def display(self):
            print(f"Distance Between House and Office = {self.distance()} m")
a=School(10,5)
b=Office(2,100)
a.display()
b.display()
```

```
Distance Between House and School = 10005 m
Distance Between House and Office = 2100 m
```

21

In [1]:

```
from abc import ABC,abstractmethod


class Vehicle(ABC):

    @abstractmethod
    def display(self):
        pass

    @abstractmethod
    def applybreak(self):
        pass

class Car(Vehicle):
    def __init__(self,Brand,Model,price):

        self.brand= Brand
        self.model= Model
        self.price = price

    def display(self):
        print(f"Car Brand = {self.brand}, Car model = {self.model}, Car price = {self.p

    def applybreak(self):
        print("Break is Applied.....")



class Motercycle(Vehicle):
    def __init__(self,Brand,Model,price):

        self.brand= Brand
        self.model= Model
        self.price = price

    def display(self):
        print(f"Motercycle Brand = {self.brand}, Motercycle model = {self.model}, Moter

    def applybreak(self):
        print("Break is Applied.....")

class Truck(Vehicle):
    def __init__(self,Brand,Model,price):

        self.brand= Brand
        self.model= Model
        self.price = price

    def display(self):
        print(f"Truck Brand = {self.brand}, Truck model = {self.model}, Truck price = {

    def applybreak(self):
        print("Break is Applied.....")

c=Car('BMW','M30',10000)
c.display()
c.applybreak()

m=Motercycle('Bullet','xyz',1000)
```

```
m.display()
m.applybreak()

t=Truck('Pqr','dhsv',103200)
t.display()
t.applybreak()
```

```
Car Brand = BMW, Car model = M30, Car price = 10000
Break is Applied.....
Motercycle Brand = Bullet, Motercycle model = xyz, Motercycle price = 1000
Break is Applied.....
Truck Brand = Pqr, Truck model = dhsv, Truck price = 103200
Break is Applied.....
```

22. Write a program that has a class Polygon. Derive two classes Rectangle and triangle from polygon and write methods to get the details of their dimensions and hence calculate the area

In [9]:

```
class Polygon:
    def __init__(self,length,height):
        self.length = length
        self.height = height

    def Dimension(self):
        print("Length of Polygon =",self.length)
        print("Height of polygon =",self.height)

class Rectangle(Polygon):
    def __init__(self,length,height):
        super().__init__(length,height)

    def Area(self):
        print("Area of rectangle =",self.length * self.height)

class Triangle(Polygon):
    def __init__(self,length,height):
        super().__init__(length,height)

    def Area(self):
        print("Area of Triangle =",self.length * self.height * 0.5)

T=Triangle(2,4)
T.Area()
T.Dimension()
R=Rectangle(2,4)
R.Area()
R.Dimension()
```

```
Area of Triangle = 4.0
Length of Polygon = 2
Height of polygon = 4
Area of rectangle = 8
Length of Polygon = 2
Height of polygon = 4
```

23. Write a program that extends the class Shape to calculate the area of a circle and a cone .(use super to inherit base class methods)

In [ ]:

```

```

24. Write a program to demonstrate hybrid inheritance and show MRO for each class.

In [17]:

```
class Source1:
    def source(self):
        print("Source of youre Power is XYZ")
class Source2:
    def source(self):
        print("Source of youre Power is PQR")
class Father(Source1):

    def SuperPower(self):
        print("I have Speed power.")

    def Suit(self):
        print("I have a Blue color suit")

class Mother(Source2):

    def SuperPower(self):
        print("I have Mind Power")
    def Suit(self):
        print("I have a yellow Color Suit")

class child1(Father,Mother):
    pass

class child2(Mother,Father):
    pass

c1=child1()
c1.SuperPower()
c1.Suit()
c1.source()

c2=child2()
c2.SuperPower()
c2.Suit()
c2.source()

print("METHOD RESOLUTION ")
print(child1.__mro__)
print(child2.__mro__)
```

```
I have Speed power.
I have a Blue color suit
Source of youre Power is XYZ
I have Mind Power
I have a yellow Color Suit
Source of youre Power is PQR
METHOD RESOLUTION
(<class '__main__.child1'>, <class '__main__.Father'>, <class '__main__.Sour
ce1'>, <class '__main__.Mother'>, <class '__main__.Source2'>, <class 'objec
t'>)
(<class '__main__.child2'>, <class '__main__.Mother'>, <class '__main__.Sour
ce2'>, <class '__main__.Father'>, <class '__main__.Source1'>, <class 'objec
t'>)
```

25. Write a program to overload + operator to multiply to fraction object of fraction class which contain two instance variable numerator and denominator. Also, define the instance method simplify() to simplify the fraction objects.

In [34]:

```
class Fraction:

    def __init__(self,num,den):
        self.n=num
        self.d=den

    def __add__(self,other):
        return (self.n * other.n,  self.d*other.d )

f1=Fraction(2,3)
f2=Fraction(3,5)
print(f1+f2)
```

```
(6, 15)
```

26. Write a program to compare two-person object based on their age by overloading > operator.

In [1]:

```
class Person:
    def __init__(self, age):
        self.age = age
    def __ge__(self, other):
        return self.age >= other.age
p1=Person(18)
p2=Person(15)
p3=Person(50)

print(p1>=p3)
print(p1>=p2)
print(p2>=p3)
```

```
False
True
False
```

Compare two object

In [48]:

```
class Person:
    def __init__(self, num,den):
        self.n = num
        self.d=den
    def __cmp__(self,other):
        return self.n*other.n, self.d*other.d

p1=Person(2,3)
p2=Person(5,3)


v=p2.__cmp__(p1)
print(v)
```

```
(10, 9)
```

27. Write a program to overload inoperator.

In [ ]:

```

```

28. WAP to create a Complex class having real and imaginary as it attributes. Overload the +,-,/,* and += operators for objects of Complex class.

`

In [29]:

```
class Complex:
    def __init__(self,r1,i1,r2,i2):
        self.r1 = r1
        self.i1= i1
        self.r2 = r2
        self.i2 =i2
        self.c1 = complex(r1,i1)
        self.c2 = complex(r2,i2)


    def add(self):
        return self.c1 + self.c2

    def subtract(self):
        return self.c1 - self.c2

    def div(self):
        return self.c1 / self.c2

    def multiply(self):
        return self.c1 * self.c2
    def increment(self):

c=Complex(3,3,4,5)
print("Sum =",c.add())
print("Difference =",c.subtract())
```

```
Sum = (7+8j)
Difference = (-1-2j)
```

29. Write a program to inspect the object using type() ,id(), isinstance(), issubclass() and callable() built-in function.

In [87]:

```
class Person:
    total=0
    def __init__(self,name,age):
        self.name=name
        self.age=age

    def display(self):
        print(f"Name = {self.name}")
        print(f"Age = {self.age}")

class Engineer(Person):
    def __init__(self,name,age):
        super().__init__(name,age)
    def profession(self):
        print(f"{self.name} is an Engineer")

p=Person("Abhinay",18)

print("Type =",type(p))
print("ID",id(p))
print(isinstance(p,Person))
print(isinstance('ps',Person))
print(issubclass(Engineer,Person))
print(issubclass(Person,Engineer))
print(callable(Person))

```

```
Type = <class '__main__.Person'>
ID 2297825504656
True
False
True
False
True
```

30. WAP to inspect the program code using the functions of inspect module. `

In [5]:

```
import inspect
import numpy
import collections

def fun(a):
    return 2*a

class A(object):
    pass

class B(A):
    pass

class C(B):
    pass


print(inspect.getclasstree(inspect.getmro(C)))

print(inspect.ismethod(collections.Counter))
print(inspect.isclass(A))
print(inspect.ismodule(numpy))
print(inspect.isfunction(fun))
```

```
[(<class 'object'>, ()), [(<class '__main__.A'>, (<class 'object'>,)), [(<cl
ass '__main__.B'>, (<class '__main__.A'>,)), [(<class '__main__.C'>, (<class
'__main__.B'>,))]]]]
False
True
True
True
```

31. Write a program to create a new list containing the first letters of every element in an already existing list.

In [7]:

```
l=['My','name','is','Abhinay','Prakash']

print([i[0] for i in l ])
```

```
['M', 'n', 'i', 'A', 'P']
```

32. Write a program using reduce() function to calculate the sum of first 10 natural numbers

In [10]:

```
import functools
print(functools.reduce(lambda a, b: a+b, range(1,11)))
```

```
55
```

33. Write a program that convert a list of temperatures in Celsius into Fahrenheit using map() function.

In [12]:

```
temp_list=[0,37,65,33,75,34,22,43]
def C_to_F(x):
    return x*9/5+32
print(list(map(C_to_F,temp_list)))
```

```
[32.0, 98.6, 149.0, 91.4, 167.0, 93.2, 71.6, 109.4]
```

34. Write a program that creates an iterator to print squares of numbers.

In [43]:

```

sq_list= [i**2 for i in range(10)]
a = iter(sq_list)
print(next(a))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
print(next(a))


```

```
0
1
4
9
16
25
36
```

35. Write a program that create a custom iterator to create even numbers

In [52]:

```
class Even_num:
    def __init__(self):
        self.start=0

    def __iter__(self):
        return self

    def __next__(self):
        x=self.start
        self.start+=2
        return x



a=Even_num()
b=iter(a)
print(next(b))
print(next(b))
print(next(b))
print(next(b))
print(next(b))

```

```
0
2
4
6
8
```

36. Write a program to create a generator that starts counting from 0 and raise an exception when counter is equal to 10.

In [59]:

```
def count():
    i=0
    while True:
        if i<10:
            yield i
            i+=1
        else:
            raise StopIteration
            break
for i in count():
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9


---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
C:\Users\ABHINA~1\AppData\Local\Temp/ipykernel_16068/1176165129.py in count
()
      7         else:
----> 8             raise StopIteration
      9             break

StopIteration:

The above exception was the direct cause of the following exception:

RuntimeError                              Traceback (most recent call last)
C:\Users\ABHINA~1\AppData\Local\Temp/ipykernel_16068/1176165129.py in <modul
e>
      8             raise StopIteration
      9             break
---> 10 for i in count():
     11     print(i)

RuntimeError: generator raised StopIteration
```

37. Write a program to create a generator to print the Fibonacci number

In [7]:

```
def my_gen(x):
    a=0
    b=1
    while(b<=x):
        yield a
        t=a
        a=b
        b=t+b

k=my_gen(5)


for i in k:
    print(i)
```

```
0
1
1
2
3
```

38. Write a program to create an arithmetic calculator using tkinter.

In [1]:

```
import tkinter as tk
window= tk.Tk()
window.geometry('300x150')
window.title("CALCULATOR")

tk.Label(window,text="First Number:").grid(row=0,column=0)
n1=tk.Entry(window,width=17)
tk.Label(window,text="Second Number:").grid(row=1,column=0)
n2=tk.Entry(window,width=17)
tk.Label(window,text="result =").grid(row=2,column=0)
r=tk.Label(window,bg='grey',width=15)


def add():
    num1=n1.get()
    num2=n2.get()
    s=float(num1)+float(num2)
    r.configure(text=s)
def sub():
    num1=n1.get()
    num2=n2.get()
    s=float(num1)-float(num2)
    r.configure(text=s)
def mul():
    num1=n1.get()
    num2=n2.get()
    s=float(num1)*float(num2)
    r.configure(text=s)
def div():
    num1=n1.get()
    num2=n2.get()
    s=float(num1)/float(num2)
    r.configure(text=s)
bt1=tk.Button(window,text="+",command=add,width=4).grid(row=0,column=2)
bt2=tk.Button(window,text="-",command=sub,width=4).grid(row=1,column=2)
bt3=tk.Button(window,text="*",command=mul,width=4).grid(row=0,column=3)
bt4=tk.Button(window,text="/",command=div,width=4).grid(row=1,column=3)
n1.grid(row=0,column=1)
n2.grid(row=1,column=1)
r.grid(row=2,column=1)
window.mainloop()
```

39. Write a program to draw colored shapes (line, rectangle, oval) on canvas.

In [33]:

```
from tkinter import *
root = Tk()

C = Canvas(root,bg='hotpink')

line = C.create_line(200,50,200, 200,fill="Red")
oval = C.create_oval(100,200, 150,100,fill="blue")
rect=C.create_rectangle(250,50,350,200,fill="yellow")
C.pack()
mainloop()

```

40. Write a program to create a window that disappears automatically after 5 seconds.

In [7]:

```

from tkinter import Tk, mainloop, TOP
from tkinter.ttk import Button
from time import time
root = Tk()

button = Button(root, text = 'HEllo THERE!!!')
button.pack(side = TOP, pady = 5)

print('Running...')
start = time()
root.after(5000, root.destroy)

mainloop()
end = time()
print('Destroyed after % d seconds' % (end-start))

```

```
Running...
Destroyed after  5 seconds
```

41. Write a program to create a button and a label inside the frame widget. Button should change the color upon hovering over the button and label should disappear on clicking the button.

In [22]:

```
from tkinter import *
root=Tk()
def disappear():
    l.destroy()

l=Label(text="This is a Label")
b=Button(text="Click Me",bg='Blue',activebackground='red',command=disappear)
b.bind('<Enter>', func=lambda e :b.config(bg="pink"))
b.bind('<Leave>', func=lambda e :b.config(bg="orange"))
b.pack()
l.pack()

mainloop()
```

42. Write a program to create radio-buttons (Male, Female, and Transgender) and a label. Default selection should be on Female and the label must display the current selection made by user

In [66]:

```
from tkinter import *
root=Tk()

def p():
    if v.get()==1:
        l.config(text='Male')
    elif  v.get()==2:
        l.config(text='Female')
    elif  v.get()==3:
        l.config(text='Transgender')

v=IntVar(root,2)
l=Label(text='Female',bg='grey',width='10')
Radiobutton(text='Male',value=1,variable=v,command=p).grid(row=0,column=0)
Radiobutton(text='Female',value=2,variable=v,command=p).grid(row=0,column=1)
Radiobutton(text='Transgender',value=3,variable=v,command=p).grid(row=0,column=2)
Label(text="Gender :").grid(row=1,column=0)
l.grid(row=1,column=1)
mainloop()
```

43. Write a program to display a menu on the menu bar.

In [2]:

```
from tkinter import Toplevel, Button, Tk, Menu

top = Tk()
menubar = Menu(top)
file = Menu(menubar, tearoff=0)
file.add_command(label="New")
file.add_command(label="Open")
file.add_command(label="Save")
file.add_command(label="Save as...")
file.add_command(label="Close")

file.add_separator()

file.add_command(label="Exit", command=top.quit)

menubar.add_cascade(label="File", menu=file)
edit = Menu(menubar, tearoff=0)
edit.add_command(label="Undo")

edit.add_separator()

edit.add_command(label="Cut")
edit.add_command(label="Copy")
edit.add_command(label="Paste")
edit.add_command(label="Delete")
edit.add_command(label="Select All")

menubar.add_cascade(label="Edit", menu=edit)
help = Menu(menubar, tearoff=0)
help.add_command(label="About")
menubar.add_cascade(label="Help", menu=help)

top.config(menu=menubar)
top.mainloop()
```

44. Write a NumPy program to create an array of (3, 4) shape, multiply every element value by 3 and display the new array.

In [12]:

```
import numpy as np
arr=np.array([[1,2,3,4],[3,5,2,6],[4,2,6,1]])
new_arr= arr*3
print(new_arr)
```

```
[[ 3  6  9 12]
 [ 9 15  6 18]
 [12  6 18  3]]
```

45. Write a NumPy program to compute the multiplication of two given matrixes

In [25]:

```
import numpy as np
n1=int(input("Enter No. of rows of first matrix:"))
m1=int(input("Enter No. of Column of first matrix:"))
n2=int(input("Enter No. of rows of second matrix:"))
m2=int(input("Enter No. of Column of second matrix:"))

if n1==m2:
    arr1=np.zeros((n1,m1))
    arr2=np.zeros((n2,m2))
    print("Enter Elements for first Matrix")
    for i in range(n1):
        for j in range(m1):
            a=int(input(f' Enter Element as position A{i+1}{j+1}: '))
            arr1[i][j]=a

    print("Enter Elements for Second Matrix")
    for i in range(n2):
        for j in range(m2):
            a=int(input(f' Enter Element as position A{i+1}{j+1}: '))
            arr2[i][j]=a

    print(arr1*arr2)
else:
    print("Multiplication not possible")
```

```
Enter No. of rows of first matrix:3
Enter No. of Column of first matrix:3
Enter No. of rows of second matrix:3
Enter No. of Column of second matrix:3
Enter Elements for first Matrix
 Enter Element as position A11: 5
 Enter Element as position A12: 4
 Enter Element as position A13: 2
 Enter Element as position A21: 5
 Enter Element as position A22: 3
 Enter Element as position A23: 2
 Enter Element as position A31: 3
 Enter Element as position A32: 2
 Enter Element as position A33: 4
Enter Elements for Second Matrix
 Enter Element as position A11: 6
 Enter Element as position A12: 7
 Enter Element as position A13: 3
 Enter Element as position A21: 5
 Enter Element as position A22: 1
 Enter Element as position A23: 8
 Enter Element as position A31: 3
 Enter Element as position A32: 5
 Enter Element as position A33: 3
[[30. 28.  6.]
 [25.  3. 16.]
 [ 9. 10. 12.]]
```

46. Write a Program to create a series from a list, numpy array and dict.

In [13]:

```
import pandas as pd
import numpy as np

my_list=[1,2,4,2]
series1= pd.Series(my_list)
print("Series using List")
print(series1)

arr= np.array([1,6,2,8])
series2=pd.Series(arr)
print("Series using Numpy")
print(series2)

my_dict= {'x':2,'y':6,'z':9}

series3=pd.Series(my_dict)
print("Series using Dict")
print(series3)


```

```
Series using List
0    1
1    2
2    4
3    2
dtype: int64
Series using Numpy
0    1
1    6
2    2
3    8
dtype: int32
Series using Dict
x    2
y    6
z    9
dtype: int64
```

47. Write a Program to convert a numpy array to a dataframe of given shape

In [27]:

```
import numpy as np
import pandas as pd

arr=np.array([[1,2,3],[4,5,3],[7,5,3]])
pd.DataFrame(arr)
```

Out[27]:

	0	1	2
0	1	2	3
1	4	5	3
2	7	5	3

48. Write a program to count number of missing values in each column.

In [ ]:

```
1
```

49. Write a program to replace missing values in a column of a dataframe by the mean value of that column.

In [ ]:

```
1
```

50. Write a Pandas program to create a line plot of the opening, closing stock prices of Alphabet Inc. between two specific dates. Use the alphabet_stock_data.csv file to extract data.

In [ ]:

```
1
```