In [16]:
```
# 1. (I) Create a person class with:
# i) two instance variable: name, age.
# ii) Create a parameterized constructor

class person:
    def __init__(self,name,age):
        self.name = name
        self.age = age

p1 = person("ajay" , 45)
p1.name
p1.age
```

Out[16]:
```
45
```

In [32]:
```
# 1. (II)Create a student class. Inherit person class in Student class.
# Student class have:
# i) instance variable: rollno and stream.
# ii) Create a parameterized constructor to initialize all instance varia
# student class as well as Person class
# iii)Instance method: display() to print name, age, rollno and stream
# Create an object of Student class and call display method

class person():
    def __init__(self,name , age):
        self.name = name
        self.age = age
class student(person):
    def __init__(self, rollno , stream):
        self.rollno = rollno
        self.stream = stream

class display(student):
    def __init__(self , name , age ,rollno , stream):
        self.name = name
        self.age = age
        self.rollno = rollno
        self.stream = stream

p1 = display("ajay" , 23 , 234 , "computer science")
print(f"\n name = {p1.name} , age = {p1.age} ,rollno = {p1.rollno} , stre
```

```
name = ajay , age = 23 ,rollno = 234 , stream= computer science
```

In [47]:
```
# 2. Write a Python class named Circle. Declare an instance variable, rad
# two methods that will compute the area and the perimeter of a circle.
import math
class circle():
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi*(self.radius**2)

    def perimeter(self):
        return 2*math.pi*self.radius
```

```
c1 = circle(4)
print(f"area:{c1.area()}")
print(f"perimeter : {c1.perimeter()}")
```

```
area:50.26548245743669
perimeter : 25.132741228718345
```

In [60]:

```
# 3. Write a Python program to create a calculator class. Include methods
# basic arithmetic operations.

class calculator():

    def __init__(self , a, b):
        self.a = a
        self.b = b

    def multiply(self):
        return self.a * self.b
    def add(self):
        return self.a + self.b
    def sub(self):
        return self.a - self.b
    def divide(self):
        return self.a / self.b

calcy1 = calculator(5 , 9)
print(f"addition: {calcy1.add()} ")
print(f"subtraction: {calcy1.sub()} ")
print(f"multiplication: {calcy1.multiply()} ")
print(f"division: {calcy1.divide()} ")
```

```
addition: 14
subtraction: -4
multiplication: 45
division: 0.5555555555555556
```

In [14]:

```
# Write a Python program to create a class representing a shopping cart.
# Include methods for adding and removing items, and calculating the tota
# price.

class ShoppingCart():
    def __init__(self):
        self.item = []

    def add_item(self , name_item ,price , quantity=1):
        item = {"name" : name_item ,"price":price , "quantity": quantity}
        self.item.append(item)
        print("the item is added in list" , self.item)

    def remove_item(self , name_item):
        for item in self.item:
            if item["name"] == name_item:
                self.item.remove(item)
                print(f"{name_item} reomve from cart")
                return
        print(f"{name_item} not found")

    def calculate(self):
        total = sum(item["price"]*item["quantity"] for item in self.item)
        return total
    def show_cart(self):
        if not self.item:
            print("the cart is empty")
        else:
            print("the list of cart" , self.item)

shop1 = ShoppingCart()
shop1.add_item("mango" , 600 , 5)
shop1.add_item("apple" , 600 , 5)
shop1.show_cart()
shop1.remove_item("apple")
shop1.calculate()
```

```
the item is added in list [{'name': 'mango', 'price': 600, 'quantity': 5}]
the item is added in list [{'name': 'mango', 'price': 600, 'quantity': 5},
{'name': 'apple', 'price': 600, 'quantity': 5}]
the list of cart [{'name': 'mango', 'price': 600, 'quantity': 5}, {'name':
'apple', 'price': 600, 'quantity': 5}]
apple reomve from cart
```

Out[14]: 3000

In [65]:

```
# 5. Write a Python class Employee with attributes like emp_id, emp_name,
# emp_salary, and emp_department and methods like calculate_emp_salary,
# emp_assign_department, and print_employee_details.
# Sample Employee Data:
# "ADAMS", "E7876", 50000, "ACCOUNTING"
# "JONES", "E7499", 45000, "RESEARCH"
# "MARTIN", "E7900", 50000, "SALES"
# "SMITH", "E7698", 55000, "OPERATIONS"

class employee():
    def __init__(self , emp_name ,emp_id , emp_salary , emp_department):
        self.emp_name = emp_name
        self.emp_id = emp_id
        self.emp_salary = emp_salary
        self.emp_department = emp_department

    def calculate_emp_salary(self, hours_worked):
        if hours_worked>50:
            overtime = hours_worked -50
            overtime_amount = (overtime*(self.emp_salary/50))
            total = self.emp_salary + overtime_amount
        else :
            total = self.emp_salary
        return total

    def emp_assign_department(self , new_department):
        if self.emp_department == new_department:
            print("the new department assign" , self.emp_department)
        else:
            self.emp_department
        return new_department

    def print_employee_details(self):
        print(f"Employee ID: {self.emp_id}")
        print(f"Employee Name: {self.emp_name}")
        print(f"Employee Salary: {self.emp_salary}")
        print(f"Employee Department: {self.emp_department}")
        print("-" * 30)
```

```
emp1 = employee("ADAMS" ,"E7499", 50000 , "accounting")
emp2 = employee("JONES", "E7499", 45000, "RESEARCH")
emp3 = employee("MARTIN", "E7900", 50000, "SALES")
emp4 = employee("SMITH", "E7698", 55000, "OPERATIONS")

emp1.print_employee_details()
emp2.print_employee_details()
emp3.print_employee_details()
emp4.print_employee_details()

emp1.calculate_emp_salary(7)
emp2.calculate_emp_salary(8)
emp3.calculate_emp_salary(9)
emp4.calculate_emp_salary(10)

emp1.emp_assign_department("research")
emp2.emp_assign_department("SALES")
emp3.emp_assign_department("OPERATIONS")
emp4.emp_assign_department("accounting")
```

```
Employee ID: E7499
Employee Name: ADAMS
Employee Salary: 50000
Employee Department: accounting
------------------------------
Employee ID: E7499
Employee Name: JONES
Employee Salary: 45000
Employee Department: RESEARCH
------------------------------
Employee ID: E7900
Employee Name: MARTIN
Employee Salary: 50000
Employee Department: SALES
------------------------------
Employee ID: E7698
Employee Name: SMITH
Employee Salary: 55000
Employee Department: OPERATIONS
------------------------------
```

```
Out[65]:  'accounting'
```

```
In [76]:  # 6. Write a Python class BankAccount with attributes like account_number
          # balance, date_of_opening and customer_name, and methods like deposit,
          # withdraw, and check_balance.
          class bankaccount():
              def __init__(self, acc_no , balance , date_opening , costumer_name):
                  self.acc_no = acc_no
                  self.balance = balance
                  self.date_opening = date_opening
                  self.costumer_name = costumer_name

              def deposit(self , amount):
                  self.balance += amount
                  print("RS." ,amount , "is deposit")

              def debit(self , amount):
                  self.balance -= amount
                  print("Rs.",amount, "is debited")
```

```
    def check_balance(self):
        print("the balance is :" , self.balance)

acc1 = bankaccount(542345523532 , 10000 , 23/11/2006 , "rahul")
acc1.deposit(5000)
acc1.debit(500)
acc1.check_balance()
acc1.deposit(4500)
acc1.check_balance()
```

```
RS. 5000 is deposit
Rs. 500 is debited
the balance is : 14500
RS. 4500 is deposit
the balance is : 19000
```

In [86]:

```
# 7. Create a class hierarchy for different types of geometric shapes, in
# circles, rectangles, and triangles, using inheritance.
# Tasks:
# A. Define a base class called Shape with common attributes
# like colour and area.
# B. Implement subclasses for specific shape types such
# as Circle, Rectangle, and Triangle. Each subclass should inherit
# from the Shape class.
# C. Incorporate additional attributes and methods specific to each
# shape type. For example, a Circle class might have attributes
# like radius and methods like calculate_area.
# D. Use inheritance to create subclasses representing variations within
# each shape type. For example, within the Rectangle class, create
# subclasses for Square and Parallelogram.
# E. Implement methods or attributes in the subclasses to demonstrate
# how inheritance allows for the sharing of attributes and methods
# from parent classes.
# F. Create instances of the various shape classes and test their
# functionality to ensure that attributes and methods work as
# expected.
import math
class shape():
    def __init__(self , colour , area):
        self.colour = colour
        self.area = area

class circle(shape):
    def __init__(self , radius):
        self.radius = radius

    def calculate_area(self , ):
        return math.pi * (self.radius**2)

class rectangle(shape):
    def __init__(self , l , b):
        self.l = l
        self.b = b

    def calculate_area(self):
        return  self.l*self.b

class square(rectangle):
    def __init__(self , l):
        self.l = l

    def calculate_area(self):
        return self.l**2

class parallelogram(rectangle):
    def __init__(self , base , height):
        self.base = base
        self.height = height

    def calculate_area(self):
        return self.base * self.height

class triangle(shape):
    def __init__(self , base , height):
        self.base = base
        self.height = height

    def calculate_area(self):
        return (0.5 * self.base * self.height)

s1 = shape("red" , 51)
s2 = circle(6)
s2.calculate_area()
s3 = square(9)
s3.calculate_area()
s4 = rectangle(6,8)
s4.calculate_area()
```

```
Out[86]: 48
```

```
In [110…  # 8. WAP to find the number of words in the given text file
# Hints:
# Use the split() method to separate words.

f = open("abcd.txt" , "w")
f.write("the python is high level programming language")
f.close()

f = open("abcd.txt" , "r")
words = f.read()
print("the file word" , words)
f.close()

word_count = len(words.split())
print("the number of words in the file ", word_count)
```

```
the file word the python is high level programming language
the number of words in the file  7
```

In [111…

```
# 9. Write a program to write “Happy Programming” in a text file and read

f = open("abcd.txt" , "w")
f.write("HAPPY PROGRAMMING")
f.close()

f = open("abcd.txt" , "r")
print(f.read())
f.close()
```

```
HAPPY PROGRAMMING
```

In [6]:

```
# 10.WAP to demonstrate the working of the following functions:
# i) read()
# ii) read(n)
# iii)readline()
# iv) readlines()

f = open("sample.txt" , "w")
f.write("the programming is very interesting\n")
f.write("the python is not for loosers\n")
f.close()

f= open("sample.txt" , "r")
print("using read()")
print(f.read())
f.close()

f = open("sample.txt" , "r")
print("using readline():")
print(f.readline())
print(f.readline())
f.close()

f = open("sample.txt" , "r")
print("using readlines()")
print(f.readlines())
f.close()

f = open("sample.txt" , "r")
n= 30
print("using read(n):")
print(f.read(n))
f.close
```

```
using read()
the programming is very interesting
the python is not for loosers

using readline():
the programming is very interesting

the python is not for loosers

using readlines()
['the programming is very interesting\n', 'the python is not for loosers
\n']
using read(n):
the programming is very intere
```

```
Out[6]:  <function TextIOWrapper.close()>
```

```
In [19]: # 11.WAP that exhibits the working of the following functions:
         # i. write()
         # ii. writelines()
         f = open("sample.txt" , "w")
         f.write("using write()\n")
         f.write("the python is not for loosers\n")
         f.close()

         f = open("sample.txt" , "r")
         print(f.read())
         f.close()

         f = open("abcd.txt" , "w")
         print("using writelines()\n")
         lines = ("the python is easy to learn\n")
         f.writelines(lines)
         f.close()

         f = open("abcd.txt" , "r")
         print(f.read())
         f.close()
```

```
using write()
the python is not for loosers

using writelines()

the python is easy to learn
```

```
In [22]: # 12.Write a Python program to read first n lines of a file.
         f = open("sample.txt"  , "w")
         f.write("hii whatsup dude\n")
         f.close()

         f = open("sample.txt" , "r")
         n = 30
         print(f.read(n))
         f.close()
```

```
hii whatsup dude
```

```
In [32]: # 13.Write a Python program to append text to a file and display the text

         f = open("abcd.txt" , "w")
         list = (" hii joseph, how are you!\n")
         f.writelines(list)
         f.close()

         f = open("abcd.txt" , "r")
         print(f.read())
         f.close()

         with open("abcd.txt" , "a") as f:
             f.write("the python is great")

         f= open("abcd.txt" , "r")
         print("the updated file is ")
         print(f.read())
         f.close()
```

```
 hii joseph, how are you!

the updated file is
 hii joseph, how are you!
the python is great
```

In [59]:
```
# 14.Write a Python program to read last n lines of a file.

f = open("abcd.txt" , "w")
f.write("the file is open\n")
f.write("hii im great\n")
f.write("me brr aahe\n")
f.close()

f = open("abcd.txt" , "r")
f.readline()
second_line = f.readline()
print("the last line of the list :" , third_line.strip())
f.close()
```

```
the last line of the list  hii im great
```

In [74]:
```
# 15.Write a Python program to read a file line by line and store it into

f = open("abcd.txt" , "w")
list =["hii im great\n" ,"dude how are you doing financially\n"]
f.writelines(list)
f.close()

f = open("abcd.txt" , "r")
print(f.readline())
print(f.readline())
f.close()
```

```
hii im great

dude how are you doing financially
```

In [77]:
```
# 16.Write a program to exhibit these concepts:

# i. try
# ii. except
# iii. finally

try:
    x = int(input("enter your number"))
    i = input("enter your operation")
    y = int(input("enter your number"))
    if i == "/":
        print("the number is division")
    else:
        print("not a division")
except ZeroDivisionError :
    print("y should not be 0")
finally:
    if i == "/":
        print("nice")
    else:
        print("ENTER IT AGAIN")
```

```
the number is division
nice
```

In [81]:
```
# 17.Write a Python program to handle a ZeroDivisionError exception when
# dividing a number by zero.

try:
    x = int(input("enter your number"))
    i = input("enter your operation")
    y = int(input("enter your number"))
    if x and y is x/y:
        print("the number is division")
    else:
        print("not a division")
except ZeroDivisionError :
    print("y should not be 0")
```

```
y should not be 0
```

In [82]:
```
# 18.Write a Python program that prompts the user to input an integer and
# ValueError exception if the input is not a valid integer.
x = int(input("enter your number:"))
y = int(input("enter your number:"))
z = x+y
print(z)
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[82], line 3
      1 # 18.Write a Python program that prompts the user to input an inte
ger and raises a
      2 # ValueError exception if the input is not a valid integer.
----> 3 x = int(input("enter your number:"))
      4 y = int(input("enter your number:"))
      5 z = x+y

ValueError: invalid literal for int() with base 10: '5ijjij'
```

In [89]:
```
# 19.WAP that exhibits multiple except blocks along with default block

try:
    num1 = int(input("enter your number:"))
    num2 = int(input("enter your number:"))
    result = num1/num2
    print("the divison value is " ,result)

    main_list= [1,2,3]
    x = int(input("enter the no. btwn 0,1,2" ))
    print("the index numer is " , main_list[x])
except ValueError:
    print("valueeror: you must enter your correct number")
except ZeroDivisionError:
    print("zerodivisionerror: entr your correct number")
except TypeError:
    print("TypeError: enter your correct number")
else:
    print("the operation is completed")
finally:
    print("HAVE A NICE DAY")
```

```
the divison value is  0.8615384615384616
the index numer is  3
the operation is completed
HAVE A NICE DAY
```

In [97]:

```
# 20.WAP that exhibits except blocks that can catch multiple exceptions.

try:
    num1 = int(input("enter your number:"))
    num2 = int(input("enter your number:"))
    result = num1/num2
    print("the divison value is " ,result)

    main_list= [1,2,3]
    x = int(input("enter the no. btwn 0,1,2" ))
    print("the index numer is " , main_list[x])
except (ValueError ,ZeroDivisionError , IndexError) as f:
    print(f"error occured as:{f}")
else:
    print("the operation is completed")
finally:
    print("HAVE A NICE DAY")
```

```
error occured as:division by zero
HAVE A NICE DAY
```

In [1]:

```
# 21.WAP to demonstrate how to use lambda in map() function.
l1 = [10,20,30 ,40,50]
l2 = [1,2,3,4,5]
l3 =list(map(lambda x,y:x*y,l1 ,l2))
print(l3)
```

```
[10, 40, 90, 160, 250]
```

In [5]:

```
# 22.WAP to demonstrate how to use lambda in filter() function.
def check_number(number):
    if number > 65:
        return True
    else:
        return False
x = [10,70,50,100,110,200,400]
number = filter(check_number , x)
print(list(number))
```

```
[70, 100, 110, 200, 400]
```

In [11]:

```
# 23.Write a Python program to filter a list of integers into list of eve
# and list of odd numbers using Lambda. [Hint: use lambda in filter() ]
def even_odd(num):
    return num%2==0
num = [2,4,1,3,5,7]
x = filter(even_odd , num)
print("even number",list(x))
```

```
even number [2, 4]
```

In [15]:

```
# 24.Write a Python program to square and cube every number in a given li
# integers using Lambda. [Hint: use lambda in map() ].
```

```
l= [1,2,3,4,5]
s = list(map(lambda x :x**2 ,l))
t = list(map(lambda x:x**3 , l))
print("the square of the number is :", s)
print(" the cube of the numner is :",t)
```

```
the square of the number is : [1, 4, 9, 16, 25]
 the cube of the numner is : [1, 8, 27, 64, 125]
```

In [16]:
```
# 25.Write a Python program to create a lambda function that adds 15 to a
# number passed in as an argument.
add_15 = lambda x: x + 15
num = int(input("Enter a number: "))
print("Result after adding 15:", add_15(num))
```

```
Result after adding 15: 21
```

In [18]:
```
# 26.Create a lambda function that multiplies argument x with argument y
# prints the result.
x=[1,2,3,4,5,6]
y =[1,2,3,4,5,6]
result= list(map(lambda t,s:t*s ,x,y))
print(result)
```

```
[1, 4, 9, 16, 25, 36]
```