//q1-Write a program to print the number of lines, words, and characters present in the given file.
#before hand create a file and keep and then type this code

```
import os

filename=input("enter file name: ")
if os.path.isfile(filename):
    print("File exists with name: ",filename)
    f= open(filename,'r')
else:
    print("file doesnt exist")

lcount = wcount=ccount=0

for line in f:
    lcount+=1
    ccount+=len(line)
    words=line.split()
    wcount+=len(words)

print("The no of lines: ",lcount)
print("The no of words: ",wcount)
print("The no of characters: ",ccount)
```

---

//q2- Write a python program to find the longest word in a file. Get the file name from the user.
#before hand create a file and keep and then type this code

```
import os
filename = input("Enter the filename: ")

if os.path.isfile(filename):
    print("File exists with name: ",filename)
    f= open(filename,'r')
    words = f.read().split()  # Split the contents into words
    longest_word = max(words, key=len)  # Find the longest word
    if longest_word:
        print(f"The longest word in the file '{filename}' is: {longest_word}")

else:
    print("File not found.")
```

---

//q3-Write a Python program to create a dictionary in a CSV file with the heading USN, NAME, Attendance ,Marks, then read and display the content of that CSV file

```
import csv

with open("IAT1.csv", "w", newline='') as f:
    w = csv.writer(f)
    w.writerow(["usn", "stud name", "attendance", "marks"])
    n = int(input("Enter number of students: "))
    for i in range(n):
        usn = input("Enter USN: ")
```

```
        studname = input("Enter student name: ")
        attend = input("Enter attendance: ")
        marks = input("Enter marks: ")
        w.writerow([usn, studname, attend, marks])
    print("Total student marks written to CSV file successfully")

# Reading and displaying the content of the CSV file
print("\nContent of CSV file:")
with open("IAT1.csv", "r") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

---

```
//q4- Write a Python Program to count the number of vowels present in a string using sets
def count_vowels(string):
    count = 0
    vowels={'a', 'e', 'i', 'o', 'u'}
    lowercase_string = string.lower()
    for char in lowercase_string:
        if char in vowels:
            count += 1
    return count

string = input("Enter a string: ")
num_vowels = count_vowels(string)
print("Number of vowels in the string:", num_vowels)
```

(or)

```
string = input("Enter a string: ")
count = 0
vowels={'a', 'e', 'i', 'o', 'u','A','E','I','O','U'}
for char in string:
    if char in vowels:
        count += 1

print("Number of vowels in the string: ", count)
```

---

```
//q5-Write Python Program to Calculate Area and Perimeter of different Shapes using Polymorphism

import math

class Shape:
    def area(self):pass
    def perimeter(self):pass

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width
```

```
    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)

class Circle(Shape):
    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

class Triangle(Shape):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def perimeter(self):
        return self.a + self.b + self.c

    def area(self):
        s = self.perimeter() / 2
        return math.sqrt(s * (s - self.a) * (s - self.b) * (s - self.c))


rectangle = Rectangle(4, 5)
print("Rectangle - Area:", rectangle.area())
print("Rectangle - Perimeter:", rectangle.perimeter())

circle = Circle(3)
print("Circle - Area:", circle.area())
print("Circle - Perimeter:", circle.perimeter())

triangle = Triangle(3, 4, 5)
print("Triangle - Area:", triangle.area())
print("Triangle - Perimeter:", triangle.perimeter())



---------------------------------------------------------------------------------------------
//q6-Write a python program to demonstrate polymorphism using classes Vehicle, Bike, Car, Aeroplane

class Vehicle:
    def __init__(self, name):
        self.name = name

    def move(self):pass

    def display_info(self):
        print(f"Vehicle Type: {self.__class__.__name__}")
```

```
        print(f"Name: {self.name}")
        self.move()


class Bike(Vehicle):
    def move(self):
        print("Bike is moving with two wheels.")


class Car(Vehicle):
    def move(self):
        print("Car is moving with four wheels.")


class Aeroplane(Vehicle):
    def move(self):
        print("Aeroplane is flying in the sky.")


bike = Bike("Harley Davidson")
car = Car("Toyota Corolla")
aeroplane = Aeroplane("Boeing 747")

vehicles = [bike, car, aeroplane]

for vehicle in vehicles:
    print("\n")
    vehicle.display_info()
```

---

//q7- Write a program to simulate a bank account with support for DepositMoney, WithdrawMoney and ShowBalance operations.

```
class bankacc:
    def __init__(self,name):
        self.username=name
        self.balance=0.0
    def showbalance(self):
        print(f"{self.username} has balance of {self.balance}")
    def withdrawmoney(self,amount):
        if (amount>self.balance):
            print("u dont hv suff balance")
        else:
            self.balance-=amount
            print(f"{self.username} has withdrawn a amt of {self.balance}")
    def depositmoney(self,amount):
                self.balance+=amount
                print(f"{self.username} has deposited an amt of {self.balance}")

acc=bankacc("xyz")
acc.showbalance()
acc.depositmoney(1000)
acc.showbalance()
acc.withdrawmoney(500)
```

```
acc.showbalance()
```

---

//q8- Write a program to demonstrate multiple inheritance

```
class Parent1:
    def method1(self):
        print("Parent 1 Method")


class Parent2:
    def method2(self):
        print("Parent 2 Method")


class Child(Parent1, Parent2):
    def method3(self):
        print("Child Method")

child = Child()
child.method1()  # Method from Parent1
child.method2()  # Method from Parent2
child.method3()  # Method from Child
```

---

//q11. Write a program to compute DFT and IDFT for a discrete signal

```
import numpy as np
import matplotlib.pyplot as plt
from math import pi

def DFT(x):
    N = len(x)
    n = np.arange(N)
    k = n.reshape((N, 1))
    e = np.exp(-2j * pi * k * n / N)
    X = np.dot(e, x)
    return X

def IDFT(X):
    N = len(X)
    k = np.arange(N)
    n = k.reshape((N, 1))
    e = np.exp(2j * pi * k * n / N)
    x = np.dot(e, X) / N
    return x

# Sample discrete signal
x = np.array([0, 1, 2, 3])

# Compute DFT
X = DFT(x)
```

```
# Compute IDFT
x_reconstructed = IDFT(X)

# Plot original and reconstructed signals
plt.figure()

plt.subplot(2, 1, 1)
plt.stem(np.arange(len(x)), x, 'b')
plt.title('Original Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid(True)

plt.subplot(2, 1, 2)
plt.stem(np.arange(len(x_reconstructed)), np.real(x_reconstructed), 'r')
plt.title('Reconstructed Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid(True)


plt.tight_layout()
plt.show()
```

---

//q12. Write a program to extract In-phase and quadrature phase signals from a random data.

```
from math import pi
import numpy as np
import matplotlib.pyplot as plt

ts=np.arange(0,2,0.01)
fc=5
AI=2*np.sin(2*pi*fc*ts);
AO=2*np.cos(2*pi*fc*ts);
t=np.arange(0,2,0.01)
f=1
x=2*np.sin(2*pi*f*t)
f=4
x+=np.sin(2*pi*f*t)
f=7
x=0.5*np.sin(2*pi*f*t)
inph=x*AI
ouph=x*AO
plt.plot(ts,inph)
plt.plot(ts,ouph)
plt.legend(['quad','inph'])
plt.show()
```