1.a)Develop a program that accept the list and iterates over a list of numbers first and prints the numbers in the list whose square is divisible by 8.

```
num_list = [2, 3, 5, 7, 8, 9, 12, 16]

for num in num_list:

    if (num * num) % 8 == 0:

        print(num)
```

OUTPUT:

8 12 16

1.b) Develop a program that illustrates: SyntaxError, IndexError, ValueError, TypeError in a statement.

SyntaxError: Raised when the parser encounters a syntax error.

```
# initialize the amount variable amount = 10000

 # check that You are eligible to

# purchase an item or not

 if(amount > 2999)

 print("You are eligible to purchase an item")


if(amount > 2999)
```

OUTPUT:

SyntaxError: invalid syntax

SEARCH STACK OVERFLOW

IndexError: Raised when a index or subscript in a sequence is out of range.

```
num_list = [2, 3, 5, 7, 8, 9, 12, 16]
```

num_list[8]

OUTPUT:

----> 3 num_list[8] IndexError: list index out of range

SEARCH STACK OVERFLOW

ValueError: An error caused by an illegal value.

Int (’67.5’)

OUTPUT:

ValueError

----> 1 int('67.5')

ValueError: invalid literal for int() with base 10: '67.5'

SEARCH STACK OVERFLOW

a='67.5'

type(a)

str

TypeError: An error caused by incorrect operand types.

a = 2

b = 'DataCamp'

a + b

'sd'

OUTPUT:

TypeError

1 a = 2

2 b = 'DataCamp'

----> 3 a + b

TypeError: unsupported operand type(s) for +: 'int' and 'str'

SEARCH STACK OVERFLOW

2.Develop a program that accept an input string, count occurrences of all characters within a string.

#To accept input string:

input_string = "Python is a high-level, general-purpose programming language."

print("The input string is:", input_string)

The input string is: Python is a high-level, general-purpose programming language

#To count occurrences of all characters within a string:

```
mySet = set(input_string)
# To create a set of all the characters present in the input for element in mySet:
countOfChar = 0
 for character in input_string:
        if character == element:
            countOfChar += 1
    print("Count of character '{}' is {}".format(element, countOfChar))
```

OUTPUT:

Count of character 'P' is 1

Count of character 'u' is 2

Count of character 'y' is 1

Count of character 'e' is 6

Count of character 'r' is 4

Count of character 'h' is 3

Count of character 'm' is 2

Count of character ',' is 1

Count of character 'v' is 1

Count of character 'p' is 3

Count of character '.' is 1

Count of character 'i' is 3

Count of character '-' is 2

Count of character ' ' is 6

Count of character 's' is 2

Count of character 'l' is 4

Count of character 'n' is 4

Count of character 't' is 1

Count of character 'g' is 6

Count of character 'a' is 5

Count of character 'o' is 3

3. Define a dictionary called agencies that stores a mapping of acronyms CBI, FB), NIA, SSB, WBA (keys) to Indian government agencies Central Bureau of investigation Foreign direct investment',

'National investion agency', 'Service selection board' and 'Works Progress administration' (the values) created by PM during new deal. Then

a)add the map of acronym BSE"Bombay stock exchange"

b)change the value of key SSB to social security administra

c)Remove the pairs with keys (B1 and WBA.

Create dict:

agencies = {"CBI": "Central Bureau of investigation""FBI": "Foreign direct investment", "NIA": "National investion agency",”SSB":"Service selection board","WBA" : "Works Progress administration" }

print (agencies)

print ("******")

type (agencies)

Add the map of acronym BSE "Bombay stock exchange":

agencies ["BSE"] = "Bombay stock exchange"

print (agencies)

OUTPUT:

{*CBI":"central bureau of investigation", "FBI": "Foreign

direct investment", "NIA": "National investion agency","

"SSB":"Service selection board", "WBA": "Works administration"}

{"CBI": "central bureau of investigation", "FBI": "Foreign direct investment", "NIA": "National investion agency","SSB": "Service selection board", "WBA":"Works Progress administration", "B5A": "Bombay stock exchange"}

4. Develop a program uring turtle graphic, write a program that asks the user for the number of sides, the length of the side, the colour, and fill colour of a regular polygon. The program should draw the polygon and then fill it in.

```
import turtle
# Ask user for number of sides, length of side, color, and fill color
num_sides = int(input("Enter the number of sides: "))
side_length = int(input("Enter the length of each side: "))
pen_color = input("Enter the pen color: ")
fill_color = input("Enter the fill color: ")

# Set up turtle
t = turtle.Turtle()
t.color(pen_color)
t.fillcolor(fill_color)

# Draw polygon
```

```
angle = 360 / num_sides
t.begin_fill()
for i in range(num_sides):
    t.forward(side_length)
    t.right(angle)
t.end_fill()

# Keep turtle window open until user clicks to close
turtle.done()
```

5.a)WAP to implement merge sort

```
import turtle

# Ask user for number of sides, length of side, color, and fill color
num_sides = int(input("Enter the number of sides: "))
side_length = int(input("Enter the length of each side: "))
pen_color = input("Enter the pen color: ")
fill_color = input("Enter the fill color: ")

# Set up turtle
t = turtle.Turtle()
t.color(pen_color)
t.fillcolor(fill_color)

# Draw polygon
```

```
angle = 360 / num_sides
t.begin_fill()
for i in range(num_sides):
    t.forward(side_length)
    t.right(angle)
t.end_fill()

# Keep turtle window open until user clicks to close
turtle.done()
```

5.b) WAP to implement binary search

```
import turtle

# Ask user for number of sides, length of side, color, and fill color
num_sides = int(input("Enter the number of sides: "))
side_length = int(input("Enter the length of each side: "))
pen_color = input("Enter the pen color: ")
fill_color = input("Enter the fill color: ")

# Set up turtle
t = turtle.Turtle()
t.color(pen_color)
t.fillcolor(fill_color)
```

```
# Draw polygon
angle = 360 / num_sides
t.begin_fill()
for i in range(num_sides):
    t.forward(side_length)
    t.right(angle)
t.end_fill()

# Keep turtle window open until user clicks to close
turtle.done()
```

6. Develop a program that takes as input an hourly wage and the number of hours an employee worked in the last week. The program should compute and return the employee's pay. Overtime work is calculated as: any hours beyond 40 but less than or equal 60 should be paid at 1.5 times the regular hourly wage. Any hours beyond 60 should be paid at 2 times the regular hourly wage.

```
def pay(time, wage):
    if time>60:
        return 2*time*wage
    elif time>40:
        return 1.5*time*wage
    else:
        return time*wage
```

```
time = int(input("Enter the hours worked in last week:"))
wage = float(input("Enter wage per hour:"))
print("Your's week pay is:", pay(time, wage))
```

Output: Enter the hours worked in last week:55

Enter wage per hour:2000

Your's week pay is: 165000.0

7.Develop a class BankAccount that supports these methods:

a). init(): Initializes the bank account balance to the value of the input argument, or to 0 if no input argument is given.

b). withdraw(): Takes an amount as input and withdraws it from the balance.

c). deposit(): Takes an amount as input and adds it to the balance.

d). balance(): Returns the balance on the account.

```
#Class
class BankAccount:
  def __init__(self, balance=0):
    self.balances = balance
  def withdraw(self, amount
if self.balances>=amount:
      self.balances-=amount
      print(f"{amount} with draw successfully")
    else:
```

```
      print("Not enough balance")
   defdeposit(self,amount):
     self.balances+=amount
    print(f"{amount} successfully deposited")
   def balance(self):
     print(f"The balance is {self.balances}")
 account = BankAccount(int(input("Enter the opening balance: ")))
loop_runner = True
while loop_runner:
   print("\nBankAccount")
   print("Operations\n 1. Withdraw\n 2. Deposite \n 3. Balance \n 4.
To Exit")
   option = int(input("Choice: "))
   if option == 1:     account.withdraw(int(input("Enter the amount:
")))
   elif option == 2:     account.deposit(int(input("Enter the amount: ")))
   elif option == 3:     account.balance()
   else:     loop_runner = False
```

Output

```
Enter the opening balance: 1000
BankAccount
 Operations
 1. Withdraw
 2. Deposite
```

3. Balance

4. To Exit

Choice: 3 The balance is 1000

BankAccount

Operations

1. Withdraw

2. Deposit

3. Balance

4. To Exit

Choice: 1

Enter the amount: 2000

Not enough balance

BankAccount

Operations 1. Withdraw

2. Deposit

3. Balance

4. To Exit

Choice: 1

Enter the amount: 200

200 with draw successfully

BankAccount

Operations

1. Withdraw

2. Deposite

3. Balance

4. To Exit

Choice: 3

The balance is 800

BankAccount

Operations

1. Withdraw

2. Deposite

3. Balance

4. To Exit

Choice: 2

Enter the amount: 3000

3000 successfully deposited

8. Develop a bike rental system that enables the customer to see available bikes on the shop and hire bikes base their needs.

Rent bikes on hourly basis Rs 100 per hour

Rent bikes on daily basis Rs 500 per day

Rent bikes on weekly basis Rs 2500 per week

Family Rental, a promotional that can include from

(3 to 5 Rentals (of any type) with a discount of 30% of

price.

```
print("Welcome To Bike Shop")
bikes = ["MTB", "Geared", "Non-Geared", "With Training Wheels", "For Trial
Riding"]
```

```
a = 0
net = 0
while (a < 4):
        bill = 0
        print("Chooses any of the following Services\n")
        a = int(input("1: View Bike onsale \n2: View Prices \n3: Place orders
\n4: Exit \n"))
        if a == 1:
                print("The Bikes Avail are\n")
                for i in bikes:
                        print(i)
        elif a == 2:
                print("The prices at our store are: \n1. Hourly----100\n2.
Daily----500\n3. Weekly---2500\n Family pack gets 30% discount on 3-5
bikes\n")
        elif a == 3:
                print("Choose your rental type:\n1. Hourly\n2. Daily\n3.
Weekly\n")
                c = int(input("Enter your option:\n"))
                d = int(input("Enter the number of bikes(put within 3-5 to
avail family pack option):\n"))
                if c == 1:
                        bill += 100*d
                        print("Your actuall Bill is ", bill)
                        print("-----------------------------")
                elif c == 2:
                        bill += 500*d
                        print("Your actuall Bill is ", bill)
                        print("-----------------------------")
                elif c == 3:
                        bill += 2500*d
                        print("Your actuall Bill is ", bill)
                        print("-----------------------------")
                else:
                        print("Enter a valid option")
                        print("-----------------------------")
                if d in range(3,6):
                        print("Do you wanna avail family pack discount?\n")
                        dis = input("y for YES\nn for NO\n")
                        print("-----------------------------")
                        if dis == "y":
                                bill = bill*0.7
                        else:
                                bill = bill
                print("Thanks for purchasing", bill, "is your bill, pay on
checkout")
        else:
                break
```

OUTPUT:

Welcome To Bike Shop

Chooses any of the following Services

1: View Bike onsale

2: View Prices

3: Place orders

4: Exit

2

The prices at our store are:

1. Hourly----100

2. Daily----500

3. Weekly---2500

Family pack gets 30% discount on 3-5 bikes

Chooses any of the following Services

1: View Bike onsale

2: View Prices

3: Place orders

4: Exit

1

The Bikes Avail are

MTB

Geared

Non-Geared

With Training Wheels

For Trial Riding

Chooses any of the following Services

1: View Bike onsale

2: View Prices

3: Place orders

4: Exit

3

Choose your rental type:

1. Hourly

2. Daily

3. Weekly

Enter your option:

1

Enter the number of bikes(put within 3-5 to avail family pack option):

3

Your actuall Bill is 300

Do you wanna avail family pack discount?

y for YES

n for NO

y

Thanks for purchasing 210.0 is your bill, pay on checkout

9. Develop program that takes one input argument the name of a text file. The function should print, on the screen, the number of lines, words, and characters in the file.

```
fname = "File1.txt"
 num_lines = 0
 num_words = 0
num_chars = 0
with open(fname, 'r') as f:
    for line in f:
        words = line.split()
        num_lines += 1
        num_words += len(words)
       num_chars += len(line)
print("The total number of lines in a given file: ", num_lines)
print("The total number of words in a given file: ", num_words)
print("The total number of characters in a given file: ",num_chars)
```

Output:

The total number of lines in a given file: 1

The total number of words in a given file: 19

The total number of characters in a given file: 118

10. WAP to extract and display all image link from Wikipedia.orp/wiki/sachin tendulkar

```
import requests

from bs4 import BeautifulSoup


url = "https://en.wikipedia.org/wiki/Sachin_Tendulkar"

response = requests.get(url)


soup = BeautifulSoup(response.content, "html.parser")

images = soup.select("img")

for image in images:

    src = image.get("src")

    if src.startswith("//"):

        src = src[2:]

    elif src.startswith("/"):

        src = src[1:]

    print(src)
```

OUTPUT:

static/images/icons/wikipedia.png

static/images/mobile/copyright/wikipedia-wordmark-en.svg

static/images/mobile/copyright/wikipedia-tagline-en.svg

upload.wikimedia.org/wikipedia/en/thumb/1/1b/Semi-protection-shackle.svg/20px-Semi-protection-shackle.svg.png

upload.wikimedia.org/wikipedia/commons/thumb/2/25/Sachin_Tendulkar_at_MRF_Promotion_Event.jpg/220px-Sachin_Tendulkar_at_MRF_Promotion_Event.jpg

upload.wikimedia.org/wikipedia/en/thumb/4/41/Flag_of_India.svg/23px-Flag_of_India.svg.png

upload.wikimedia.org/wikipedia/commons/thumb/8/80/SachinTendulkar_AutographedLetter_%28cropped%29.jpg/128px-SachinTendulkar_AutographedLetter_%28cropped%29.jpg

upload.wikimedia.org/wikipedia/commons/thumb/8/8a/Loudspeaker.svg/11px-Loudspeaker.svg.png

upload.wikimedia.org/wikipedia/commons/thumb/6/65/Sachin_at_the_other_end.jpg/220px-Sachin_at_the_other_end.jpg

upload.wikimedia.org/wikipedia/commons/thumb/6/6e/Tendulkar_closup.jpg/220px-Tendulkar_closup.jpg

upload.wikimedia.org/wikipedia/commons/thumb/5/5b/Tendulkar_goes_to_14%2C000_Test_runs.jpg/600px-Tendulkar_goes_to_14%2C000_Test_runs.jpg

upload.wikimedia.org/wikipedia/commons/thumb/2/28/199Sachin.jpg/220px-199Sachin.jpg

upload.wikimedia.org/wikipedia/commons/thumb/7/7d/Tendulkar_shot.JPG/220px-Tendulkar_shot.JPG

upload.wikimedia.org/wikipedia/commons/thumb/7/72/Master_Blaster_at_work.jpg/220px-Master_Blaster_at_work.jpg

upload.wikimedia.org/wikipedia/commons/thumb/c/cb/Sachin_Tendulkar_bowling_right-arm_leg-spin_26_January_2008.JPG/220px-Sachin_Tendulkar_bowling_right-arm_leg-spin_26_January_2008.JPG

upload.wikimedia.org/wikipedia/commons/thumb/9/95/A_Cricket_fan_at_the_Chepauk_stadium%2C_Chennai.jpg/170px-A_Cricket_fan_at_the_Chepauk_stadium%2C_Chennai.jpg

upload.wikimedia.org/wikipedia/commons/thumb/9/99/Sachin_Ramesh_Tendulkar_Wax_Statue_in_Madame_Tussauds_London.jpg/220px-

11.Develop a program to imput https.imdb url as input and display name,year, brief summary of top10 movies of the year.

```
import requests
from bs4 import BeautifulSoup

url = "https://www.imdb.com/chart/top"
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"}

response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.content, "html.parser")

movies = soup.select(".titleColumn a")
```

```
for movie in movies[:10]:
    link = "https://www.imdb.com" + movie.get("href")
    movie_response = requests.get(link, headers=headers)
    if movie_response.ok:
        soup = BeautifulSoup(movie_response.content, "html.parser")


        # Extract movie name
        try:
            movie_name = soup.select_one('#__next > main > div > section.ipc-page-background.ipc-page-background--base.sc-f9e7f53-0.ifXVtO > section > div:nth-child(4) > section > section > div.sc-ab3b6b3d-3.goiwap > div.sc-2971dade-0.YVoIO > h1 > span').text.strip().split(" (")[0]
        except AttributeError:
            movie_name = "N/A"


        # Extract movie year
        try:
            movie_year = soup.select_one('#__next > main > div > section.ipc-page-background.ipc-page-background--base.sc-f9e7f53-0.ifXVtO > section > div:nth-child(4) > section > section > div.sc-ab3b6b3d-3.goiwap > div.sc-2971dade-0.YVoIO > ul > li:nth-child(1) > a').text.strip()
        except AttributeError:
            movie_year = "N/A"
```

```
        # Extract movie summary
        try:
            movie_summary = soup.select_one('#__next > main > div > section.ipc-page-background.ipc-page-background--base.sc-f9e7f53-0.ifXVtO > section > div:nth-child(4) > section > section > div.sc-ab3b6b3d-4.fzQBuI > div.sc-ab3b6b3d-6.cbFSCE > div.sc-ab3b6b3d-10.kreISa > section > p > span.sc-35061649-0.fjlUgo').text.strip()
        except AttributeError:
            movie_summary = "N/A"

        print(f"Name: {movie_name}")
        print(f"Year: {movie_year}")
        print(f"Summary: {movie_summary}")
        print("--------------------")
```

OUTPUT:

```
Name: The Shawshank Redemption
Year: 1994
Summary: Over the course of several years, two convicts form a friendship, seeking consolation and, eventually, redemption through basic compassion.
--------------------
Name: N/A
```

Year: N/A

Summary: N/A

Name: N/A

Year: N/A

Summary: N/A

Name: The Godfather Part II

Year: 1974

Summary: The early life and career of Vito Corleone in 1920s New York City is portrayed, while his son, Michael, expands and tightens his grip on the family crime syndicate.

Name: 12 Angry Men

Year: 1957

Summary: The jury in a New York City murder trial is frustrated by a single member whose skeptical caution forces them to more carefully consider the evidence before jumping to a hasty verdict.

Name: Schindler's List

Year: 1993

Summary: In German-occupied Poland during World War II, industrialist Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.

Name: The Lord of the Rings: The Return of the King

Year: 2003

Summary: Gandalf and Aragorn lead the World of Men against Sauron's army to draw his gaze from Frodo and Sam as they approach Mount Doom with the One Ring.

Name: N/A

Year: N/A

Summary: N/A

Name: N/A

Year: N/A

Summary: N/A

Name: The Good, the Bad and the Ugly

Year: 1966

Summary: A bounty hunting scam joins two men in an uneasy alliance against a third in a race to find a fortune in gold buried in a remote cemetery.

12. Develop sierpinski triangle with the given details.

```
import turtle


def sierpinski(t, x, y, size, depth):
    if depth == 0:
        t.penup()
```

```
    t.goto(x, y)
    t.pendown()
    for i in range(3):
        t.forward(size)
        t.left(120)
else:
    sierpinski(t, x, y, size/2, depth-1)
    sierpinski(t, x+size/2, y, size/2, depth-1)
    sierpinski(t, x+size/4, y+(size/2)(3*0.5)/2, size/2, depth-1)

    if depth == change_depth:
        t.fillcolor('magenta')
        t.begin_fill()
        sierpinski(t, x+size/4, y+(size/2)(3*0.5)/2, size/2, 0)
        t.end_fill()

        t.fillcolor('red')
        t.begin_fill()
        sierpinski(t, x, y, size/2, 0)
        t.end_fill()

        t.fillcolor('blue')
        t.begin_fill()
        sierpinski(t, x+size/2, y, size/2, 0)
```

```
        t.end_fill()

t = turtle.Turtle()
t.speed(0)

change_depth = 2 # Change this value to specify the depth at which the color changes
sierpinski(t,-200,-200,400,change_depth)

turtle.done()
```

13. WAP to implement Koch fractal recursive program such that it draws Koch snowflake.

```
import turtle

def koch_snowflake(t, x1, y1, x2, y2, depth):
    if depth == 0:
        t.penup()
        t.goto(x1, y1)
        t.pendown()
        t.goto(x2, y2)
    else:
        xa = x1 + (x2 - x1) / 3
        ya = y1 + (y2 - y1) / 3
```

```
        xb = x1 + 2 * (x2 - x1) / 3
        yb = y1 + 2 * (y2 - y1) / 3
        xc = (x1 + x2) / 2 - (y2 - y1) * (3**0.5) / 6
        yc = (y1 + y2) / 2 + (x2 - x1) * (3**0.5) / 6

        koch_snowflake(t, x1, y1, xa, ya, depth-1)
        koch_snowflake(t, xa, ya, xc, yc, depth-1)
        koch_snowflake(t, xc, yc, xb, yb, depth-1)
        koch_snowflake(t, xb, yb, x2, y2, depth-1)

t = turtle.Turtle()
t.speed(0)

depth = 2 # Change this value to specify the depth of recursion
size = 300
x1 = -size / 2
y1 = size * (3**0.5) / 6
x2 = size / 2
y2 = size * (3**0.5) / 6
x3 = 0
y3 = -size * (3**0.5) / 3

koch_snowflake(t,x1,y1,x2,y2,depth)
koch_snowflake(t,x2,y2,x3,y3,depth)
```

```
koch_snowflake(t,x3,y3,x1,y1,depth)

turtle.done()
```

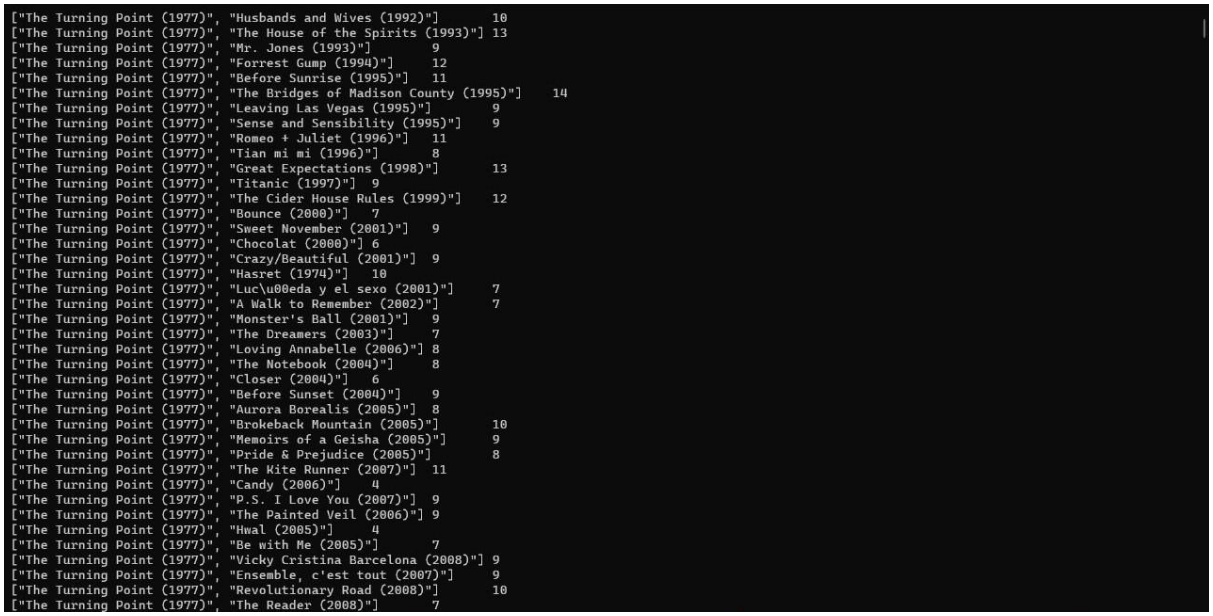
14. Develop movie recommendation using mapreduce framework

```
from mrjob.job import MRJob

class MovieSimilarities(MRJob):

    def mapper(self, _, line):
        twitter_id, movie_name, genre = line.split("::")
        yield genre, movie_name

    def reducer(self, genre, movies):
        movie_list = list(movies)
        for i in range(len(movie_list)):
            for j in range(i+1, len(movie_list)):
                similarity_score = self.calculate_similarity(movie_list[i], movie_list[j])
                yield (movie_list[i], movie_list[j]), similarity_score

    def calculate_similarity(self, movie1, movie2):
        # Convert movie names to lowercase for case-insensitive comparison
```

```
        movie1 = movie1.lower()
        movie2 = movie2.lower()

        # Calculate similarity score based on movie name similarity
        # In this example, we calculate similarity based on the number of common characters
        common_chars = set(movie1) & set(movie2)
        similarity_score = len(common_chars)

        return similarity_score

if _name_ == '_main_':
    MovieSimilarities.run()
```

Movie Recommendation system using map reduce framework

OUTPUT:



15.develop a GUI application to determine BMI.

```
import tkinter as tk
def calculate_bmi():
    weight = float(weight_entry.get())
    height = float(height_entry.get())
    bmi = round(weight / (height ** 2), 2)
    bmi_label.config(text=f"BMI: {bmi}")

root = tk.Tk()
root.title("BMI Calculator")

weight_label = tk.Label(root, text="Weight (kg):")
weight_label.grid(row=0, column=0)
weight_entry = tk.Entry(root)
```

```
weight_entry.grid(row=0, column=1)

height_label = tk.Label(root, text="Height (m):")
height_label.grid(row=1, column=0)
height_entry = tk.Entry(root)
height_entry.grid(row=1, column=1)

calculate_button = tk.Button(root, text="Calculate BMI",
command=calculate_bmi)
calculate_button.grid(row=2, column=0, columnspan=2)

bmi_label = tk.Label(root, text="BMI:")
bmi_label.grid(row=3, column=0, columnspan=2)

root.mainloop()
```