#Task 1

#Write a python code which will count the number of unique letters and their frequency.

#Test the code with the word ‘Anaconda’. (Note: Remove Case Sensitivity)

#Input: Anaconda

#Output 1: Number of Unique Letters = 5.

#Output 2: {a: 3, c: 1, d: 1, n: 2, o: 1}

word = "Anaconda"

# Convert the word to lowercase to remove case sensitivity

word = word.lower()

# Create an empty dictionary to store letter frequencies

letter\_count = {}

# Iterate through the characters in the word

for char in word:

if char.isalpha():

if char in letter\_count:

letter\_count[char] += 1

else:

letter\_count[char] = 1

# Calculate the number of unique letters

num\_unique\_letters = len(letter\_count)

# Print the results

print("Output 1: Number of Unique Letters =", num\_unique\_letters)

print("Output 2:", letter\_count)

# Task 2

#A student’s evaluation is done based on 4 components: Class\_Test (10%), Mid\_Term (20%), Project (30%) & End\_Term (40%).

#Write a python code to generate a random score between 10 & 90 (use python library: random) and get the evaluation bifurcation (Round off to Nearest Integer).

#Example

#Input = 80 (marks)

#Class\_Test = 8 (marks)

#Mid\_Term = 16 (marks)

#Project = 24 (marks)

#End\_Term = 32 (marks)

import random

# Generate a random score between 10 and 90

total\_score = random.randint(10, 90)

# Calculate scores for each component based on weights

class\_test\_weight = 0.10

mid\_term\_weight = 0.20

project\_weight = 0.30

end\_term\_weight = 0.40

class\_test\_score = round(total\_score \* class\_test\_weight)

mid\_term\_score = round(total\_score \* mid\_term\_weight)

project\_score = round(total\_score \* project\_weight)

end\_term\_score = round(total\_score \* end\_term\_weight)

# Print the results

print("Class\_Test =", class\_test\_score, "(marks)")

print("Mid\_Term =", mid\_term\_score, "(marks)")

print("Project =", project\_score, "(marks)")

print("End\_Term =", end\_term\_score, "(marks)")

#3 Write a python code which will count the number of unique letters and their frequency.

#Test the code with the word ‘Anaconda’. (Note: Remove Case Sensitivity)

#Input: Anaconda

#Output 1: Number of Unique Letters = 5.

#Output 2: {a: 3, c: 1, d: 1, n: 2, o: 1}

def common\_letters(w1, w2):

w1 = w1.lower()

w2= w2.lower()

common\_letters = set(w1) & set(w2) #checking common letter

return common\_letters

w1 = 'Python' # assignment of string "python" in variable w1

w2 = 'Anaconda' # assignment of string "Anaconda" in variable w2

common\_letters = common\_letters(w1, w2) #calling of a function common\_letters have two parameter w1,w2 and return the content into common\_letters

print("Common letters between " ,w1 ,"and", w2)

for letter in common\_letters: # read letter in common\_letter

print(letter)

#task4

# Write a python code to generate a random score between 0 & 100 (use python library: random). Print the following as output:

#1. The score

#2. ‘Grade F’ : If the score is less than 40

#3. ‘Grade C’ : If the score is between 40 & 59

#4. ‘Grade B’ : If the score is between 60 & 84

#5. ‘Grade A’ : If the score is between 85 & 100

# code

#importing a module name random as alias name as rn

import random as rn

#taking a random value from randint which is a function in a module random

grade= rn.randint(0,100)

#printing the grade value

print("The Grade is : ",grade)

print("The score ")

#‘Grade F’ : If the score is less than 40

if(grade<40):

print("Grade F")

#‘Grade C’ : If the score is between 40 & 59

elif (grade>=40 and grade<=59):

print("Grade C")

#Grade B’ : If the score is between 60 & 84

elif(grade>=60 and grade<=84):

print("Grade B")

# 'Grade A’ : If the score is between 85 & 100

elif(grade>=85 and grade<=100):

print("Grade A")

#5

#Write a python code to generate a random number between 1 & 99 (use python library: random). Print the following as output:

#1. The random number [say, 9]

#2. List of even numbers up to the random number [2, 4, 6, 8]

#3. List of odd numbers up to the random number [1, 3, 5, 7, 9]

#4. List of prime numbers up to the random number [2, 3, 5, 7]

# code

#creating a function is\_prime which print prime no. up to num

def is\_prime(upto):

# loop from 2 to upto+1

for num in range(2, upto + 1):

i = 2

#loop of i in range 2 to num

for i in range(2, num):

if(num % i == 0):

i = num #if num is divisible by i that means it is not a prime no.

break; #break the loop

# If the number is prime then print it.

if(i != num):

print(num, end=" ")

#importing a module name random as alias name as rn

import random as rn

#taking a random value from randint which is a function in a module random

list\_even=[] # created an empty list which store even no

list\_odd=[] # created an empty list which store odd no

#taking a random value in variable num

num= rn.randint(1,100)

print("The random number is : ",num)

#loop that will read all numbers till num

for i in range(1,num+1):

if(i%2==0): #if divisible by 2 it is even

list\_even.append(i) #even no. append to the list

else:

list\_odd.append(i) #odd no. append to list

print("List of even numbers up to the random number",list\_even)

print("List of odd numbers up to the random number",list\_odd)

print("List of prime numbers up to the random number")

is\_prime(num) # calling of function created above

#6

import random

# Generate a random 4-digit number between 1000 and 9999

random\_number = random.randint(1000, 9999)

print(f"Random 4-digit number: {random\_number}")

# Extract and print each digit with its place value

thousands\_digit = random\_number // 1000

hundreds\_digit = (random\_number // 100) % 10

tens\_digit = (random\_number // 10) % 10

ones\_digit = random\_number % 10

print(f"Thousands digit: {thousands\_digit}")

print(f"Hundreds digit: {hundreds\_digit}")

print(f"Tens digit: {tens\_digit}")

print(f"Ones digit: {ones\_digit}")

#Task 7

#Write a python code to generate 5 random numbers between -9 & +9 (use python library: random). Print the list of 5 random numbers and their sum.

import random as rn

random\_numbers = [rn.randint(-9, 9) for \_ in range(5)] #importing random number from -9 to 9 in random\_number

sum\_of\_numbers = sum(random\_numbers) #calculating sum

print("List of 5 random numbers:", random\_numbers)

print("Sum of the random numbers:", sum\_of\_numbers)

#task8 Task 8

#Write a python code to calculate the number of Years, Months & Days, with respect to Today, given a Date.

#Example

#Input: 15-08-1947 or August 15, 1947

#Output: {Years: 77, Months: 01, Days: 01} (As on Aug 16, 2023)

from datetime import datetime # Import the 'datetime' module

# Define a function to calculate age given a date

def calculate\_age(date):

today = datetime.now().date() # Get today's date

age = today - date # Calculate the difference between today's date and the given date

# Calculate years, months, and days from the age in days

years = age.days // 365

remaining\_days = age.days % 365

months = remaining\_days // 30

days = remaining\_days % 30

return years, months, days

# Input date in YYYY-MM-DD format

input\_date\_str = input("Enter a date (YYYY-MM-DD): ") # Ask the user for input

input\_date = datetime.strptime(input\_date\_str, '%Y-%m-%d').date() # Convert the input string to a date

# Call the calculate\_age function to get years, months, and days

years, months, days = calculate\_age(input\_date)

# Print the calculated age

print(f"Years: {years}, Months: {months}, Days: {days}")

**#Task 9**

**#Write a python code to generate the following ‘diamond’ pattern.**

**# Define the number of rows (half of the diamond height)**

**n = 5**

**# Upper part of the diamond**

**for i in range(n):**

**spaces = " " \* (n - i - 1)**

**stars = "\*" \* (2 \* i + 1)**

**print(spaces + stars)**

**# Lower part of the diamond**

**for i in range(n - 2, -1, -1):**

**spaces = " " \* (n - i - 1)**

**stars = "\*" \* (2 \* i + 1)**

**print(spaces + stars)**

**# task 10 Write a python code to create a list of 10 random letters (say List A). Create a copy of List A (say List B).**

**#Shuffle both List A & B. Treat each element of List A (say a) as Source and the corresponding element of List B (say b) as Destination. It is assumed that a from list A is connected to b from List B. Map the network of all connected elements such that an inquiry of any element will show the connected paths to & from the element.**

**import random**

**# Create List A with 10 random lowercase letters**

**list\_a = [chr(random.randint(97, 122)) for \_ in range(10)]**

**# Create List B as a shuffled copy of List A**

**list\_b = random.sample(list\_a, len(list\_a))**

**print(list\_a)**

**print(list\_b)**

**# Create a dictionary to represent the network connections**

**network = {a: b for a, b in zip(list\_a, list\_b)}**

**# Function to find connected paths to and from an element**

**def find\_connected\_paths(element):**

**paths = [f"{element} -> {b}" for a, b in network.items() if a != element and b != element]**

**return paths**

**# Test the network by querying an element (replace 'x' with the element you want to inquire about)**

**inquiry\_element = 'x' # Replace 'x' with the element you want to inquire about**

**connected\_paths = find\_connected\_paths(inquiry\_element)**

**# Print the results**

**print(f"Connections for '{inquiry\_element}':")**

**for path in connected\_paths:**

**print(path)**