**Create a JUPYTER NOTEBOOK or GOOGLE COLAB File with your Enrollment Number [Ex. 12345.ipynb] and UPLOAD the File on the Shared Folder DEV-I\_TEST-I.**

**Total Marks : 50 Duration: 01 Hr**

**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}

def count\_unique\_letters(word):

    word = word.lower()  # Convert the word to lowercase to remove case sensitivity

    # Initialize an empty dictionary to store letter frequencies

    letter\_count = {}

    # Loop through each letter in the word

    for letter in word:

        if letter.isalpha():  # Check if the character is a letter

            if letter in letter\_count:  # If the letter is already in the dictionary

                letter\_count[letter] += 1  # Increment its count

            else:

                letter\_count[letter] = 1  # If the letter is not in the dictionary, add it

    # Count the number of unique letters

    unique\_count = len(letter\_count)

    return unique\_count, letter\_count

# Input word

input\_word = 'Anaconda'

# Call the function and store the results in variables

num\_unique\_letters, letter\_frequency = count\_unique\_letters(input\_word)

# Display the results

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

print("Letter Frequencies:", letter\_frequency)

output

Number of Unique Letters = 5

Letter Frequencies: {'a': 3, 'n': 2, 'c': 1, 'o': 1, 'd': 1}

**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).

import random

# Generate random scores between 10 and 90 for each component

class\_test\_score = random.randint(10, 90)

mid\_term\_score = random.randint(10, 90)

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

end\_term\_score = random.randint(10, 90)

# Calculate the weighted scores for each component

class\_test\_weighted = round(class\_test\_score \* 0.10)

mid\_term\_weighted = round(mid\_term\_score \* 0.20)

project\_weighted = round(project\_score \* 0.30)

end\_term\_weighted = round(end\_term\_score \* 0.40)

# Calculate the total evaluation score

total\_evaluation = class\_test\_weighted + mid\_term\_weighted + project\_weighted + end\_term\_weighted

# Display the scores and evaluation bifurcation

print("Class Test Score:", class\_test\_score)

print("Mid Term Score:", mid\_term\_score)

print("Project Score:", project\_score)

print("End Term Score:", end\_term\_score)

print("\nEvaluation Bifurcation:")

print("Class Test Weighted:", class\_test\_weighted)

print("Mid Term Weighted:", mid\_term\_weighted)

print("Project Weighted:", project\_weighted)

print("End Term Weighted:", end\_term\_weighted)

print("Total Evaluation Score:", total\_evaluation)

output

Class Test Score: 45

Mid Term Score: 79

Project Score: 29

End Term Score: 29

Evaluation Bifurcation:

Class Test Weighted: 4

Mid Term Weighted: 16

Project Weighted: 9

End Term Weighted: 12

Total Evaluation Score: 41

**Example**

Input = 80 (marks)

Class\_Test = 8 (marks)

Mid\_Term = 16 (marks)

Project = 24 (marks)

End\_Term = 32 (marks)

**Task 3**

Write a python code to display the letters occurring commonly in 2 words.

Test the code with the words ‘Python’ & ‘Anaconda’. (Note: Remove Case Sensitivity)

Input 1: Python

Input 2: Anaconda

Output: [o, n]

# Input two words from the user

word1 = input("Input 1: ").lower()

word2 = input("Input 2: ").lower()

# Initialize empty lists to store unique letters from each word

unique\_letters\_word1 = []

unique\_letters\_word2 = []

# Populate the lists with unique letters from each word

for letter in word1:

    if letter.isalpha() and letter not in unique\_letters\_word1:

        unique\_letters\_word1.append(letter)

for letter in word2:

    if letter.isalpha() and letter not in unique\_letters\_word2:

        unique\_letters\_word2.append(letter)

# Initialize an empty list to store common letters

common\_letters = []

# Check for common letters and add them to the list

for letter in unique\_letters\_word1:

    if letter in unique\_letters\_word2:

        common\_letters.append(letter)

# Display the common letters

print("Output:", common\_letters)

output

Input 1: anaconda

Input 2: python

Output: ['n', 'o']

**Task 4**

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

import random

# Generate a random score between 0 and 100

score = random.randint(0, 100)

# Print the generated score

print("1. The score:", score)

# Assign grades based on score ranges and print the appropriate grade

if score < 40:

    print("2. 'Grade F'")

elif 40 <= score <= 59:

    print("3. 'Grade C'")

elif 60 <= score <= 84:

    print("4. 'Grade B'")

else:

    print("5. 'Grade A'")

ouput

1. The score: 61

4. 'Grade B'

**Task 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]

import random

# Generate a random number between 1 and 99

random\_number = random.randint(1, 99)

# Print the generated random number

print("1. The random number:", random\_number)

# Generate a list of even numbers up to the random number

even\_numbers = [num for num in range(2, random\_number + 1) if num % 2 == 0]

# Print the list of even numbers

print("2. List of even numbers up to the random number:", even\_numbers)

# Generate a list of odd numbers up to the random number

odd\_numbers = [num for num in range(1, random\_number + 1) if num % 2 != 0]

# Print the list of odd numbers

print("3. List of odd numbers up to the random number:", odd\_numbers)

# Function to check if a number is prime

def is\_prime(num):

    if num <= 1:

        return False

    for i in range(2, int(num\*\*0.5) + 1):

        if num % i == 0:

            return False

    return True

# Generate a list of prime numbers up to the random number

prime\_numbers = [num for num in range(2, random\_number + 1) if is\_prime(num)]

# Print the list of prime numbers

print("4. List of prime numbers up to the random number:", prime\_numbers)

ouput

1. The random number: 42

2. List of even numbers up to the random number: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42]

3. List of odd numbers up to the random number: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41]

4. List of prime numbers up to the random number: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41]

**Task 6**

Write a python code to generate a 4-digit random number between 1000 & 9999 (use python library: random). Print each digit with its place value.

**Example**

Input: 4321

Output: {4: Thousands, 3: Hundreds, 2: Tens, 1: Ones}

import random

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

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

# Convert the number to a string for easier digit extraction

number\_str = str(random\_number)

# Create a dictionary to map place values

place\_values = {

    0: "Thousands",

    1: "Hundreds",

    2: "Tens",

    3: "Ones"

}

# Print the original random number

print("Input:", random\_number)

# Print each digit along with its place value

for idx, digit\_char in enumerate(number\_str):

    digit = int(digit\_char)

    place\_value = place\_values[idx]

    print(f"{digit}: {place\_value}")

ouput

Input: 1413

1: Thousands

4: Hundreds

1: Tens

3: Ones

**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

# Initialize an empty list to store random numbers

random\_numbers = []

# Generate 5 random numbers between -9 and +9

for \_ in range(5):

    random\_num = random.randint(-9, 9)

    random\_numbers.append(random\_num)

# Calculate the sum of the random numbers

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

# Print the list of random numbers and their sum

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

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

output

List of random numbers: [4, -3, 6, 0, 0]

Sum of the random numbers: 7

**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, timedelta

# Input date in the format 'dd-mm-yyyy' or 'Month day, year'

input\_date = input("Input date: ")

# Convert the input date to a datetime object

try:

    date\_object = datetime.strptime(input\_date, "%d-%m-%Y")

except ValueError:

    date\_object = datetime.strptime(input\_date, "%B %d, %Y")

# Calculate the difference between today's date and the input date

current\_date = datetime.now()

time\_difference = current\_date - date\_object

# Calculate years, months, and days

years = time\_difference.days // 365

remaining\_days = time\_difference.days % 365

months = remaining\_days // 30

days = remaining\_days % 30

# Print the calculated years, months, and days

print("Output: {Years:", years, ", Months:", f"{months:02d}", ", Days:", f"{days:02d}" + "}")

output

Input date: 15-08-1947

Output: {Years: 76 , Months: 01 , Days: 01}

**Task 9**

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

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

# Define the number of rows for the diamond pattern

num\_rows = 5

# Generate the top half of the diamond

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

    spaces = " " \* (num\_rows - i)

    stars = "\*" \* i

    print(spaces + stars)

# Generate the bottom half of the diamond

for i in range(num\_rows - 1, 0, -1):

    spaces = " " \* (num\_rows - i)

    stars = "\*" \* i

    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

# Generate a list of 10 random letters (List A)

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

# Create a copy of List A (List B)

list\_b = list(list\_a)

# Shuffle both List A & List B

random.shuffle(list\_a)

random.shuffle(list\_b)

# Create a dictionary to store the connections

connections = {}

# Map connections from List A to List B

for source, destination in zip(list\_a, list\_b):

    if source not in connections:

        connections[source] = []

    connections[source].append(destination)

# Map connections from List B to List A

for source, destination in zip(list\_b, list\_a):

    if source not in connections:

        connections[source] = []

    connections[source].append(destination)

# Display the network mapping

for element, connected\_elements in connections.items():

    connected\_paths = ', '.join(connected\_elements)

    print(f"{element} is connected to: {connected\_paths}")

output

h is connected to: s, s

l is connected to: t, y

f is connected to: b, t

c is connected to: g, b

y is connected to: l, s

s is connected to: h, y, h, g

b is connected to: c, f

t is connected to: f, l

g is connected to: s, c