INDIRA GANDHI DELHI TECHNICAL UNIVERSITY

COMPUTER VISION ASSIGNMENT 1

Name : Siya Pathak
Enrollment No:06601032021
Course : BTech
Branch :IT-1

Q1. Write a python program to find the sum of first N natural numbers, take N as input from the user. For example if N=4, then output is 10, computed as 1+2+3+4=10.

Code:

```
def sum_of_natural_numbers(n):

    sum_natural_numbers = (n * (n + 1)) // 2
    return sum_natural_numbers

n = int(input("Enter a positive integer (n): "))
result = sum_of_natural_numbers(n)

print(f"The sum of the first {n} natural numbers is: {result}")
```

Output:

```
Enter a positive integer (n): 4
The sum of the first 4 natural numbers is: 10
```

Q2. Write a python program which defines a function myMerge (list1, list2) which merges two input sorted linked lists list1 and list2. First also check whether the lists are sorted or not, with function isSorted (list).

Code:

```
class Node:
    def __init__(self, data=None):
        self.data = data
        self.next = None

def isSorted(head):
    current = head
    while current and current.next:
        if current.data > current.next.data:
            return False
        current = current.next
    return True

def myMerge(list1, list2):
    if not isSorted(list1) or not isSorted(list2):
        return "Input lists are not sorted."

    dummy = Node()
    current = dummy

    while list1 and list2:
        if list1.data < list2.data:
            current.next = list1
            list1 = list1.next
        else:
            current.next = list2
            list2 = list2.next
        current = current.next

    current.next = list1 or list2

    return dummy.next

def printList(head):
    current = head
    while current:
```

```
        current = current.next

    current.next = list1 or list2

    return dummy.next

def printList(head):
    current = head
    while current:
        print(current.data, end=" ")
        current = current.next
    print()

# Example usage:
# Creating sorted linked lists
list1 = Node(1)
list1.next = Node(3)
list1.next.next = Node(5)

list2 = Node(2)
list2.next = Node(4)
list2.next.next = Node(6)

merged_list = myMerge(list1, list2)

print("Merged Sorted Linked List:")
printList(merged_list)
```

Output:

```
Merged Sorted Linked List:
1 2 3 4 5 6
```

Q3. Write a python program which defines a isStrongPassword function (myString) which takes an input string myString and returns True or Fals depending on whether the input string is a strong password. A password is strong if it follows the following conditions:
a. It contains at least one lowercase and one uppercase English character.
b. It contains at least one special character (allowed special characters are: !, $, *, @, #.
c. Its length is at least 8.
d. It contains at least one digit.

Code:

```python
def isStrongPassword(myString):
    # Condition (a)
    if not any(char.islower() for char in myString):
        return False
    if not any(char.isupper() for char in myString):
        return False
    # Condition (b)
    if not any(char in "!$*@#" for char in myString):
        return False

    # Condition (c)
    if len(myString) < 8:
        return False

    # Condition (d)
    if not any(char.isdigit() for char in myString):
        return False

    # If all conditions are met
    return True

# Example usage:
password = input("Enter your password: ")

if isStrongPassword(password):
    print("Strong password!")
else:
    print("Weak password. Make sure it follows the specified conditions.")
```

Output:

```
Enter your password: Siya@1407
Strong password!
```

```
Enter your password: siya
Weak password. Make sure it follows the specified conditions.
```

Q4.

Write a python program that reads text from an input file (use your own input text file, and refer this to learn how to read a file in google drive from google colab) and counts the number of times each alphabet is appearing in it, and displays the frequency of the occurrence of each alphabet in decreasing order of their frequency. (Use data-structure to dictionary solve it, create an input file as given below)

Sample input file: My name is (write your name). I have done my schooling from (write your school name). My hobbies are (mention some).

Nowadays, I am observing (think, think, and think, mention one issue that you see around you which you think can be solved if you have)

Code:

```python
from collections import Counter

def count_alphabet_frequency(file_path):
    # Read text from the input file
    with open(file_path, 'r') as file:
        text = file.read()

    text = text.lower()

    alphabet_counts = Counter(char for char in text if char.isalpha())

    sorted_alphabet_counts = dict(sorted(alphabet_counts.items(), key=lambda item: item[1], reverse=True))

    for alphabet, frequency in sorted_alphabet_counts.items():
        print(f"{alphabet}: {frequency} times")

input_file_path = 'Siya.txt'
count_alphabet_frequency(input_file_path)
```

Output:

```
a: 13 times
i: 13 times
s: 12 times
n: 10 times
o: 10 times
e: 9 times
m: 8 times
h: 6 times
y: 5 times
t: 5 times
v: 4 times
d: 4 times
g: 4 times
r: 4 times
f: 3 times
b: 3 times
c: 2 times
l: 2 times
k: 1 times
u: 1 times
w: 1 times
```

Q5. Python is an object oriented language. Even if you have not studied object oriented languages like C++ or Java, it is a good idea to understand some basic concepts of object oriented programming languages. Refer to any tutorial, say this or this. Create a suitable class in python to represent the mathematical concept of 'vector' (use list data structure to represent vector). Create appropriate member variables and member functions of this class to perform operations: Length of vector, Cosine similarity between two vectors, Euclidean distance between two vectors.

Code:

```python
import math
class Vector:
    def __init__(self, components):
        self.components = components

    def length(self):
        return math.sqrt(sum(component ** 2 for component in self.components))

    def cosine_similarity(self, other_vector):
        dot_product = sum(x * y for x, y in zip(self.components, other_vector.components))
        magnitude_self = self.length()
        magnitude_other = other_vector.length()

        if magnitude_self == 0 or magnitude_other == 0:
            raise ValueError("Cannot calculate cosine similarity with a zero vector.")

        return dot_product / (magnitude_self * magnitude_other)

    def euclidean_distance(self, other_vector):
        if len(self.components) != len(other_vector.components):
            raise ValueError("Vectors must have the same dimension for Euclidean distance calculation.")

        squared_diff = sum((x - y) ** 2 for x, y in zip(self.components, other_vector.components))
        return math.sqrt(squared_diff)

vector1 = Vector([1, 2, 3])
vector2 = Vector([4, 5, 6])

print("Length of vector1:", vector1.length())
print("Cosine Similarity between vector1 and vector2:", vector1.cosine_similarity(vector2))
print("Euclidean Distance between vector1 and vector2:", vector1.euclidean_distance(vector2))
```

Output:

```
Length of vector1: 3.7416573867739413
Cosine Similarity between vector1 and vector2: 0.9746318461970762
Euclidean Distance between vector1 and vector2: 5.196152422706632
```

Q6. Download the dataset and solve questions that follow.

a. Load the csv file of the dataset into a dataframe in Python program.

0s

```python
import pandas as pd
import numpy as np

df = pd.read_csv('/content/KCLT.csv')
```

b. Find the mean, median, mode, min and max for all numeric attributes.

```
import pandas as pd
import numpy as np
df = pd.read_csv('/content/KCLT.csv')
string_list = ('actual_mean_temp', 'actual_min_temp', 'actual_max_temp',
               'average_min_temp', 'average_max_temp', 'record_min_temp',
               'record_max_temp', 'actual_precipitation', 'average_precipitation',
               'record_precipitation')

for i, column_name in enumerate(string_list):
    mean_value = df[column_name].mean()
    median_value = df[column_name].median()
    mode_value = df[column_name].mode()[0]  # mode() returns a Series, use [0] to get the first mode
    min_value = df[column_name].min()
    max_value = df[column_name].max()
    # Display the results
    print(f"Column: {column_name}")
    print(f"Mean: {mean_value}")
    print(f"Median: {median_value}")
    print(f"Mode: {mode_value}")
    print(f"Min: {min_value}")
    print(f"Max: {max_value}")
```

```
Column: actual_mean_temp
Mean: 61.04931506849315
Median: 63.0
Mode: 78
Min: 18
Max: 88
Column: actual_min_temp
Mean: 49.95890410958904
Median: 52.0
Mode: 67
Min: 7
Max: 75
Column: actual_max_temp
Mean: 71.63013698630137
Median: 73.0
Mode: 84
Min: 26
Max: 100
Column: average_min_temp
Mean: 48.81917808219178
Median: 48.0
Mode: 68
Min: 29
Max: 68
Column: average_max_temp
Mean: 70.98356164383561
Median: 72.0
Mode: 89
Min: 50
Max: 89
Column: record_min_temp
Mean: 31.465753424657535
Median: 30.0
Mode: 53
Min: -5
Max: 62
Column: record_max_temp
Mean: 88.72876712328767
Median: 90.0
Mode: 100
Min: 69
Max: 104
Column: actual_precipitation
Mean: 0.10241095890410958
Median: 0.0
Mode: 0.0
Min: 0.0
Max: 2.65
Column: average_precipitation
Mean: 0.1140821917808219
Median: 0.11
Mode: 0.11
Min: 0.09
Max: 0.15
```

c. Print the top 20% of rows showing only the first four columns.

```
num_rows_to_display = int(len(df) * 0.20)

# Display the top 20% of rows showing only the first four columns
result = df.iloc[:num_rows_to_display, :4]
print(result)
```

```
         date  actual_mean_temp  actual_min_temp  actual_max_temp
0    2014-7-1                81               70               91
1    2014-7-2                85               74               95
2    2014-7-3                82               71               93
3    2014-7-4                75               64               86
4    2014-7-5                72               60               84
..        ...               ...              ...              ...
68   2014-9-7                79               70               88
69   2014-9-8                70               67               73
70   2014-9-9                72               66               77
71  2014-9-10                72               65               79
72  2014-9-11                77               64               89

[73 rows x 4 columns]
```

d. Create a new column (call it ‘newColumn’), it should have the same values as the column ‘actual_mean_temp’. Print head of dataframe.

```
import pandas as pd

# Assuming you already have a DataFrame named 'df'

# Create the new column 'newColumn'
df['newColumn'] = df['actual_mean_temp']

# Print the head of the DataFrame
print(df.head())
```

```
       date  actual_mean_temp  actual_min_temp  actual_max_temp  \
0  2014-7-1                81               70               91
1  2014-7-2                85               74               95
2  2014-7-3                82               71               93
3  2014-7-4                75               64               86
4  2014-7-5                72               60               84

   average_min_temp  average_max_temp  record_min_temp  record_max_temp  \
0                67                89               56              104
1                68                89               56              101
2                68                89               56               99
3                68                89               55               99
4                68                89               57              100

   record_min_temp_year  record_max_temp_year  actual_precipitation  \
0                  1919                  2012                  0.00
1                  2008                  1931                  0.00
2                  2010                  1931                  0.14
3                  1933                  1955                  0.00
4                  1967                  1954                  0.00

   average_precipitation  record_precipitation  newColumn
0                   0.10                  5.91        81
1                   0.10                  1.53        85
2                   0.11                  2.50        82
3                   0.10                  2.63        75
4                   0.10                  1.65        72
```

e. Remove the new column that you have created above. Print head of dataframe.

```
df = df.drop('newColumn', axis=1)
print(df.head())
```

```
       date  actual_mean_temp  actual_min_temp  actual_max_temp  \
0  2014-7-1                81               70               91
1  2014-7-2                85               74               95
2  2014-7-3                82               71               93
3  2014-7-4                75               64               86
4  2014-7-5                72               60               84

   average_min_temp  average_max_temp  record_min_temp  record_max_temp  \
0                67                89               56              104
1                68                89               56              101
2                68                89               56               99
3                68                89               55               99
4                68                89               57              100

   record_min_temp_year  record_max_temp_year  actual_precipitation  \
0                  1919                  2012                  0.00
1                  2008                  1931                  0.00
2                  2010                  1931                  0.14
3                  1933                  1955                  0.00
4                  1967                  1954                  0.00

   average_precipitation  record_precipitation
0                   0.10                  5.91
1                   0.10                  1.53
2                   0.11                  2.50
3                   0.10                  2.63
4                   0.10                  1.65
```

f. Print the first 10 rows, then remove the row containing data of '2014-7-3', save this row in a variable of type series (data structure). Print the first 10 rows after removal of the row.

```
import pandas as pd
print("First 10 rows:")
print(df.head(10))
row_to_remove = df[df['date'] == '2014-7-3'].iloc[0]
df = df.drop(df[df['date'] == '2014-7-3'].index)
print("\nFirst 10 rows after removal:")
print(df.head(10))
print("\nRow removed:")
print(row_to_remove)
```

```
Row removed:
date                     2014-7-3
actual_mean_temp               82
actual_min_temp                71
actual_max_temp                93
average_min_temp               68
average_max_temp               89
record_min_temp                56
record_max_temp                99
record_min_temp_year         2010
record_max_temp_year         1931
actual_precipitation         0.14
average_precipitation        0.11
record_precipitation          2.5
Name: 2, dtype: object
```

g. Add the row that you deleted before. Print the first 10 rows again.

```
import pandas as pd

df = df.append(row_to_remove, ignore_index=True)

print("\nFirst 10 rows after adding the row back:")
print(df.head(10))
```

h. Update the `actual_min_temp` in the data row for date '2014-7-3' to any value. Print the updated row.

```
new_value = 25  # Replace with the new value you want to set
df.loc[df['date'] == '2014-7-3', 'actual_min_temp'] = new_value
```

i. Add 5 to the 'actual_mean_temp' wherever the 'actual_min_temp' is odd. Do this only on the top 10 rows. Print these 10 rows before and after the operation.

```
top_10_rows = df.head(10)

is_odd = top_10_rows['actual_min_temp'] % 2 != 0

top_10_rows.loc[is_odd, 'actual_mean_temp'] += 5

df.iloc[:10] = top_10_rows
```

j. Print only those rows where the absolute difference between the 'record_min_temp_year' and 'record_max_temp_year' is less than 30.

```
condition = abs(df['record_min_temp_year'] - df['record_max_temp_year']) < 30

print(df[condition])
```

```
          date  actual_mean_temp  actual_min_temp  actual_max_temp  \
2     2014-7-4                75               64               86
3     2014-7-5                72               60               84
4     2014-7-6                89               61               87
5     2014-7-7                94               67               91
9    2014-7-11                78               68               87
..         ...               ...              ...              ...
353  2015-6-20                83               71               95
355  2015-6-22                83               65              100
358  2015-6-25                86               74               98
359  2015-6-26                85               70              100
361  2015-6-28                76               66               85

     average_min_temp  average_max_temp  record_min_temp  record_max_temp  \
2                  68                89               55               99
3                  68                89               57              100
4                  68                89               57               99
5                  68                89               55              100
9                  68                89               55              100
..                ...               ...              ...              ...
353                66                87               54              102
355                66                87               53              100
358                67                88               53              102
359                67                88               55              102
361                67                88               53              101

     record_min_temp_year  record_max_temp_year  actual_precipitation  \
2                    1933                  1955                  0.00
3                    1967                  1954                  0.00
4                    1964                  1948                  0.00
5                    1972                  1954                  0.00
9                    1961                  1986                  0.00
..                    ...                   ...                   ...
353                  1879                  1887                  0.01
355                  2003                  2015                  0.00
358                  1889                  1914                  0.00
359                  1979                  1952                  1.21
361                  1968                  1959                  0.00
```