Programming Lab Assignment-4

Samarth Motka

202311023

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

Q 1: Implement a custom exception named "CustomOutOfRangeError." Write a Python program that asks the user to enter an integer within a specified range (e.g., 10 to 50). If the input is outside this range, raise the "CustomOutOfRangeError" with a meaningful error message. Handle this exception and ask the user for input again until a valid input is provided.

```python
class CustomOutOfRangeError(Exception):

    def __init__(self, message="Input is out of the specified 
range."):
        self.message = message
        super().__init__(self.message)

def get_input():
    while True:
        try:
            n = int(input("Enter an integer within the range (10 to 
50): "))
            if 10 <= n <= 50:
                return n
            else:
                raise CustomOutOfRangeError()
        except CustomOutOfRangeError as e:
            print(e)
        except ValueError:
            print("Invalid input. Please enter an integer.")


try:
    valid_input = get_input()
    print(f"You entered a valid integer within the range: 
{valid_input}")
except KeyboardInterrupt:
    print("\nProgram terminated by the user.")
```

```
Enter an integer within the range (10 to 50): 1
Input is out of the specified range.
Enter an integer within the range (10 to 50): 1
Input is out of the specified range.
```

```
Enter an integer within the range (10 to 50): 1
Input is out of the specified range.
Enter an integer within the range (10 to 50): 55
Input is out of the specified range.
Enter an integer within the range (10 to 50): 4
Input is out of the specified range.
Enter an integer within the range (10 to 50): 43
You entered a valid integer within the range: 43
```

Q 2: Write a Python program that reads data from a CSV file. Implement exception handling to manage potential errors such as file not found, invalid data format, or missing columns. Create a custom exception "CSVDataError" to handle issues related to data format. Display appropriate error messages for different exceptions.

```
class CSVDataError(Exception):
    def __init__(self, message):
        super().__init__(message)

def read_csv_file():
    try:
      df = pd.read_csv('some.csv')
    except FileNotFoundError as CSVDataError  :
      print("Invalid Name or file is not there")
      print("\n")

    try:
      dataset = pd.read_csv('smoking_driking_dataset_Ver01.csv',
index_col=['Date'],
          parse_dates=['Date'])
    except ValueError  as CSVDataError  :
      print("Missing column provided")
      print("\n")

    try:
      dataset = pd.read_csv('smoking_driking_dataset_Ver01.csv')
      for i in dataset['DRK_YN']:
        print(int(i))
    except ValueError  as CSVDataError  :
      print("Invalid Data Format")
      print("\n")



read_csv_file()


Invalid Name or file is not there


Missing column provided
```

```
Invalid Data Format
```

Q 3: Create a Python module that defines a function to calculate the nth Fibonacci number. Import this module into another script and use the function to calculate the Fibonacci sequence for the first 15 numbers. Display the results.

```
import fibo as fb

x = int(input("Enter the value of N : "))

fb.fibonacci(x)

Enter the value of N : 15
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
```

Q 4: Develop a Python module that provides functionalities for matrix operations, including addition, subtraction, and multiplication using NumPy. Use this module to perform matrix multiplication on two randomly generated matrices.

```
import matrix_operation as mxo
import numpy as np

matrix_A = np.random.rand(3, 3)
matrix_B = np.random.rand(3, 3)

print(matrix_A)
print("\n")
print(matrix_B)

print("\n")
print("The addition of Matrix : ")
print(mxo.addition(matrix_A,matrix_B))

print("\n")
print("The subtraction of Matrix : ")

print(mxo.subtraction(matrix_A,matrix_B))

print("\n")
print("The multiplication of Matrix : ")

print(mxo.multiplication(matrix_A,matrix_B))
```

```
[[0.3160994  0.82833988 0.06900721]
 [0.24027949 0.08424787 0.70384552]
 [0.20684776 0.02419881 0.68999503]]


[[0.96480022 0.4565416  0.75931422]
 [0.1603123  0.84769538 0.05466333]
 [0.78427405 0.50786174 0.34098988]]


The addition of Matrix :
[[1.28089962 1.28488149 0.82832143]
 [0.40059178 0.93194324 0.75850885]
 [0.99112181 0.53206055 1.03098492]]


The subtraction of Matrix :
[[-0.64870082  0.37179828 -0.69030702]
 [ 0.07996719 -0.76344751  0.64918219]
 [-0.57742629 -0.48366294  0.34900515]]


The multiplication of Matrix :
[[0.4918864  0.88153853 0.30882935]
 [0.79733544 0.53857032 0.4270571 ]
 [0.74459133 0.46536991 0.39366656]]
```

Q 5.Write a Python program that generates a 10x10 numpy array with random integers between 1 and 100. Find and display the indices of the top 5 maximum values in the array.

```
import numpy as np

random_array = np.random.randint(1, 100, (10, 10))

# Find the indices of the top 5 maximum values using np.argpartition
top_indices = np.argpartition(-random_array.flatten(), 5)[:5]

# Convert the flattened indices to 2D indices
top_indices_2d = np.unravel_index(top_indices, random_array.shape)
```

```
print(top_indices_2d)
print(top_indices)
print("\n")
# Display the original array
print("Random Array:")
print(random_array)
print("\n")

# Display the top 5 maximum values and their indices
for i, index in enumerate(top_indices):
    value = random_array.flatten()[index]
    row, col = top_indices_2d[0][i], top_indices_2d[1][i]
    print(f"Top {i+1}: Value={value}, Index=({row}, {col})")
```

```
(array([2, 0, 9, 3, 4], dtype=int64), array([4, 0, 6, 2, 4],
dtype=int64))
[24  0 96 32 44]


Random Array:
[[99 18 92 63  8  1 11 88 25  8]
 [78  6 10 35 78  8 62 23 85 29]
 [22 91 21 66 99 68 59 44 91  4]
 [13 79 92  9 57 57 57  2 20 59]
 [37  1 55 18 95 85  7 87 44 54]
 [52 11 63 44 27  8 46 85 35  7]
 [53 70 86 36 27 84 19  4 17 85]
 [53  7 11 72 27 37 35  8 29 71]
 [69  4 50 56  2 55 74 14 86 67]
 [13 58 79 40 85 55 96 16 55 44]]


Top 1: Value=99, Index=(2, 4)
Top 2: Value=99, Index=(0, 0)
Top 3: Value=96, Index=(9, 6)
Top 4: Value=92, Index=(3, 2)
Top 5: Value=95, Index=(4, 4)
```