#2. Basic of Python Programming

Roll Number: CB.EN.P2EBS24021
Date of Submission: 27/08/2024
Aim:

Write Python programs for the following requirements:

i. Prompt two numbers from user along with the option of arithmetic operation. Use appropriate data type to print the results. Run the program continuously until the user chooses an option for exit.
[Note: Use conditional and looping statements]

ii. Check a number within the given range belongs to the Fibonacci Series

iii. Prompt the user for a string and print the output without any punctuation marks from the given string.

iv. Prompt the user for the names and ages of 5 members, then sort the members based on their age.

v. Get the information of name, date of birth and blood group from the user and store it as key value pairs for 5 persons. Compute the age from the date of birth information and add as another key value pair. Combine them as a data frame and store it in a csv file.

vi. Automate a task of backing up files based on time in the Windows environment.

Tools Required:

IDE suitable with a suitable Python interpreter.

Experiment:

i) **Pseudo Code**

Code

```
# Prompt two numbers from user along with the option of arithmetic operation. Use
appropriate data type to print the results. Run the program continuously until the
user chooses an option for exit.
# NAME : Hariharan P
# ROLL NO : CB.EN.P2EBS24021
# DATE : 28.08.2024
BLACK_BOLD = "\033[1m"
RED = "\033[31m"
RESET = "\033[0m"
# Initialize a loop condition to keep the program running until the user decides to
quit.
while True:
    # Start an infinite loop that continues until the user enters 'q' to quit.
    try:
        print('INPUT'.center(100,'-'))
        # Try block to handle potential errors, such as invalid input types.
        operator = input("\nEnter the operator (+, -, *, /) or 'q' to quit:
").strip()
        # Prompt the user to enter the operator. Strip any surrounding whitespace.
        result = None
        # Initialize the result variable to None, which will be updated with the
result of the operation.
        if operator in ['q','Q','QUIT','Quit','quit']:
            # If the user enters 'q' (in any case), set condition to False to exit
the loop.
            print("\nExiting the program.".center(60,'-'))
            # Inform the user that the program is exiting.
            break  # Break out of the loop to end the program.
        if operator not in ['*','/','+','-']:
            # If the user enters 'q' (in any case), set condition to False to exit
the loop.
            print(f"\n{RED}Enter the proper Operator.{RESET}".center(60,'-'))
            # Inform the user that the program is exiting.
            continue
        try:
            num1 = eval(input("\nEnter the first number: "))
            # Prompt the user to enter the first number and convert it to a float.
            num2 = eval(input("Enter the second number: "))
            # Prompt the user to enter the second number and convert it to a float.
        except:
            print(f"\n{RED}Enter Valid Input Values{RESET}")
            # Prompt the user to enter valid input.
            continue


        match operator:
            # Use the match statement to handle different operators.
            case '+':
```

```
                result = num1 + num2  # Return the sum of num1 and num2 if the
operator is '+'.
            case '-':
                result = num1 - num2  # Return the difference between num1 and num2
if the operator is '-'.
            case '*':
                result = num1 * num2  # Return the product of num1 and num2 if the
operator is '*'.
            case '/':
                try:
                    result = num1 / num2
                except ZeroDivisionError as z:
                    print(f"Error {z}")  # Return the quotient of num1 divided by
num2 if the operator is '/'.
            case _:
                # Handle any other operator as invalid.
                print(f"{RED}Invalid operator!{RESET}\n")
                pass
        if result is not None:
            print('\n')
            print('OUTPUT'.center(100,'-'))
            print(f"The result of{BLACK_BOLD} {num1} {operator} {num2} =
{result}{RESET}")
            print('\n')
            print('END'.center(100,'-'))
            print('\n')
            # If the function returns a valid result (not None), print the result.

    except ValueError:
        # Catch a ValueError if the user enters non-numeric input.
        print(f"{RED}Invalid input! Please enter numeric values.{RESET}")
```

Result

```
q1.py
----------------------------------------------INPUT----------------------------------------------

Enter the operator (+, -, *, /) or 'q' to quit: 20
------------
Enter the proper Operator.------------
----------------------------------------------INPUT----------------------------------------------

Enter the operator (+, -, *, /) or 'q' to quit: /

Enter the first number: 20
Enter the second number: 0
Error division by zero
----------------------------------------------INPUT----------------------------------------------

Enter the operator (+, -, *, /) or 'q' to quit: /

Enter the first number: 10
Enter the second number: 20


----------------------------------------------OUTPUT---------------------------------------------
The result of 10 / 20 = 0.5


----------------------------------------------END------------------------------------------------


----------------------------------------------INPUT----------------------------------------------

Enter the operator (+, -, *, /) or 'q' to quit: q
--------------------
Exiting the program.--------------------
(.conda) (base) hari@Hariharans-MacBook-Pro Python %
```

ii) Pseudo Code

Code

```
# Check a number within the given range belongs to the Fibonacci Series
# NAME : Hariharan P
# ROLL NO : CB.EN.P2EBS24021
# DATE : 28.08.2024

def is_fibonacci(n):
    # A helper function that checks if a given number n is a Fibonacci number.

    # A number is a Fibonacci number if one or both of (5*n^2 + 4) or (5*n^2 - 4)
is a perfect square.
    def is_perfect_square(x):
        # Check if x is a perfect square.
        s = int(x**0.5)  # Calculate the square root of x and convert it to an
integer.
        return s * s == x  # Check if the square of the integer is equal to x.

    # Return True if n is a Fibonacci number, otherwise return False.
    return is_perfect_square(5 * n * n + 4) or is_perfect_square(5 * n * n - 4)


def check_fibonacci_in_range(start, end):
    # Check all numbers in the given range and print those that belong to the
Fibonacci series.
    fib_list,non_fib_list=list(),list()
    if start<end:
        start,end=end,start
    # assigning the lists to store save fib numbers and non-fib numbers
    for num in range(start, end + 1):
        try:
            # Try to check if the current number num is in the Fibonacci series.
            if is_fibonacci(num):
                fib_list.append(num)
                #print(f"{num} belongs to the Fibonacci series")
            else:
                non_fib_list.append(num)
                #print(f"{num} does not belong to the Fibonacci series")
        except Exception as e:
            # If any error occurs during the check, print an error message.
            print(f"An error occurred while checking {num}: {e}")
    return fib_list,non_fib_list
    # return the fib and non-fib numbers


# Example usage:
try:
    print('INPUT'.center(100,'-'))
    # ANSI escape code for red text
    RED = "\033[31m"
    # ANSI escape codes for green and bold text
    GREEN_BOLD = "\033[1;32m"
```

```
    BLACK_BOLD = "\033[1m"
    RESET = "\033[0m"

    # Prompt the user to input the start of the range in red color
    start = int(input(f"{RED}\nEnter the start of the range: {RESET}{GREEN_BOLD}"))

    # Prompt the user to input the end of the range in red color
    end = int(input(f"{RESET}{RED}Enter the end of the range:
{RESET}{GREEN_BOLD}"))


    # Check and print whether the numbers in the given range belong to the
Fibonacci series.
    fib,non_fib = check_fibonacci_in_range(start, end)
    print(f'{RESET}\n')
    print('OUTPUT'.center(100,'-'))
    print(f"\n{GREEN_BOLD}Fibonacci numbers are {RESET}{RED}{fib}{RESET}")
    print('\n')
    print('END'.center(100,'-'))

except ValueError:
    # Catch a ValueError if the input is not a valid integer and print an error
message.
    print(f"{RED}Invalid input! Please enter valid integer values for the
range.{RESET}")

except Exception as e:
    # Catch any other exceptions that may occur and print an error message.
    print(f"{RED}An unexpected error occurred: {e}{RESET}")
```

Result

```
----------------------------------------------INPUT----------------------------------------------

Enter the start of the range: 10
Enter the end of the range: 50


----------------------------------------------OUTPUT---------------------------------------------

Fibonacci numbers are [13, 21, 34]


----------------------------------------------END------------------------------------------------
```

iii) Pseudo Code

Code

```
# Prompt the user for a string and print the output without any punctuation marks
from the given string.
# NAME : Hariharan P
# ROLL NO : CB.EN.P2EBS24021
# DATE : 28.08.2024

import string  # Import the string module for easy access to punctuation marks.

try:
    BLACK_BOLD = "\033[1m"
    RESET = "\033[0m"
    # Prompt the user to enter a string.
    print('INPUT'.center(100,'-'))
    user_input = input("\nEnter a string: ")

    # Use a list comprehension to remove any characters that are punctuation marks.
    # This iterates over each character in the input string and only includes those
not in string.punctuation.
    output = ''.join([char for char in user_input if char not in
string.punctuation])
    print('\n')
    print('OUTPUT'.center(100,'-'))
    # Print the resulting string without punctuation.
    print("String without punctuation: {} {} {}".format(BLACK_BOLD,output,RESET))
    print('\n')
    print('END'.center(100,'-'))
    print('\n')
except Exception as e:
    # Catch any unexpected exceptions and print an error message.
    print(f"An error occurred: {e}")
```

Result

```
----------------------------------------------INPUT-----------------------------------------------

Enter a string: Hello World!, How are You?


----------------------------------------------OUTPUT----------------------------------------------
String without punctuation:  Hello World How are You


-----------------------------------------------END------------------------------------------------
```

iv) Pseudo Code

Code

```
try:
    BLACK_BOLD = "\033[1m"
    RESET = "\033[0m"
    # Prompt the user for the number of members.
    print('INPUT'.center(100,'-'))
    num_members = int(input("Enter the number of members: "))

    # Initialize an empty list to store the members' information (name and age).
    members = []

    # Prompt the user for the names and ages of the specified number of members.
    for i in range(num_members):
        name = input(f"Enter the name of member {i+1}: ")  # Prompt for the name.
        age = int(input(f"Enter the age of {name}: "))  # Prompt for the age and
convert to integer.

        # Append the tuple (name, age) to the members list.
        members.append((name, age))

    # Sort the list of members by age.
    members.sort(key=lambda member: member[1])
    print('OUTPUT'.center(100,'-'))
    # Print the sorted list of members.
    print("\nMembers sorted by age:")
    for name, age in members:
        print(f"{BLACK_BOLD}{name}: {age}{RESET} years old")
    print('END'.center(100,'-'))

except ValueError:
    # Catch a ValueError if the input is not a valid integer.
    print("Invalid input! Please enter a valid number for the age or number of
members.")

except Exception as e:
    # Catch any other unexpected exceptions and print an error message.
    print(f"An error occurred: {e}")
```

Result

```
----------------------------------------------INPUT----------------------------------------------
Enter the number of members: 5
Enter the name of member 1: Hari
Enter the age of Hari: 24
Enter the name of member 2: Jainil
Enter the age of Jainil: 69
Enter the name of member 3: Mukunth
Enter the age of Mukunth: 25
Enter the name of member 4: Anand
Enter the age of Anand: 22
Enter the name of member 5: Nidhish
Enter the age of Nidhish: 22
----------------------------------------------OUTPUT---------------------------------------------

Members sorted by age:
Anand: 22 years old
Nidhish: 22 years old
Hari: 24 years old
Mukunth: 25 years old
Jainil: 69 years old
----------------------------------------------END-----------------------------------------------
```

v) Pseudo Code

Code

```
# Get the information of name, date of birth and blood group from the user and
store it as key value pairs for 5 persons. Compute the age from the date of birth
information and add as another key value pair. Combine them as a data frame and
store it in a csv file.
# NAME : Hariharan P
# ROLL NO : CB.EN.P2EBS24021
# DATE : 28.08.2024

import pandas as pd  # Import pandas for DataFrame creation and CSV handling.
from datetime import datetime  # Import datetime to work with date of birth and
calculate age.

def calculate_age(dob):
    # Function to calculate age given a date of birth.
    today = datetime.today()  # Get today's date.
    age = today.year - dob.year - ((today.month, today.day) < (dob.month, dob.day))
    print((today.month, today.day) < (dob.month, dob.day))
    return age

try:
    # Get the number of persons from the user.
    num_persons = int(input("Enter the number of persons: "))

    # Initialize an empty list to store the information for each person.
    persons = []

    # Loop to gather information for the specified number of persons.
    for i in range(num_persons):
        # Get the name, date of birth, and blood group from the user.
        name = input(f"Enter the name of person {i+1}: ")
        dob_input = input(f"Enter the date of birth of {name} (format DD-MM-YYYY):
")
        blood_group = input(f"Enter the blood group of {name}: ")

        # Try to convert the date of birth string into a datetime object.
        try:
            dob = datetime.strptime(dob_input, "%d-%m-%Y")
        except ValueError:
            # Handle the case where the date of birth format is incorrect.
            print(f"Invalid date format for {name}. Please enter the date in YYYY-
MM-DD format.")
            continue  # Skip this entry and ask again.

        # Calculate the age based on the date of birth.
        age = calculate_age(dob)

        # Store the information in a dictionary.
        person_info = {
            "Name": name,
```

```
            "Date of Birth(dd-mm-yyyy)": dob_input,
            "Blood Group": blood_group,
            "Age": age
        }

        # Append the dictionary to the persons list.
        persons.append(person_info)

    # Convert the list of dictionaries into a DataFrame.
    df = pd.DataFrame(persons)

    # Save the DataFrame to a CSV file.
    df.to_csv("persons_info.csv", index=False)

    print("\nData successfully saved to persons_info.csv")
    print(df)  # Optionally print the DataFrame to see the result.

except ValueError:
    # Catch a ValueError if the input is not a valid integer.
    print("Invalid input! Please enter a valid number for the number of persons.")

except Exception as e:
    # Catch any other unexpected errors and print an error message.
    print(f"An error occurred: {e}")
```

Result

```
-----------------------------------------------INPUT-----------------------------------------------
Enter the number of persons: 5
Enter the name of person 1: Hari
Enter the date of birth of Hari (format DD-MM-YYYY): 30-07-2000
Enter the blood group of Hari: O+ve
Enter the name of person 2: Mukunth
Enter the date of birth of Mukunth (format DD-MM-YYYY): 11-10-1999
Enter the blood group of Mukunth: O+ve
Enter the name of person 3: Jainil
Enter the date of birth of Jainil (format DD-MM-YYYY): 20-05-2002
Enter the blood group of Jainil: AB+ve
Enter the name of person 4: Shravan
Enter the date of birth of Shravan (format DD-MM-YYYY): 10-05-1997
Enter the blood group of Shravan: A+ve
Enter the name of person 5: Nidhish
Enter the date of birth of Nidhish (format DD-MM-YYYY): 14-10-2001
Enter the blood group of Nidhish: B+ve
----------------------------------------------OUTPUT----------------------------------------------

Data successfully saved to persons_info.csv
      Name Date of Birth(dd-mm-yyyy) Blood Group  Age
0     Hari                30-07-2000        O+ve   24
1  Mukunth                11-10-1999        O+ve   24
2   Jainil                20-05-2002       AB+ve   22
3  Shravan                10-05-1997        A+ve   27
4  Nidhish                14-10-2001        B+ve   22
-----------------------------------------------END-----------------------------------------------
```

vi) Pseudo Code

Code

```
# Automate a task of backing up files based on time in the Windows environment.
# NAME : Hariharan P
# ROLL NO : CB.EN.P2EBS24021
# DATE : 28.08.2024

import os
import shutil
import time
from datetime import datetime

def backup_files(src_dir, backup_dir, days):
    try:
        # Calculate the cutoff time for file modification.
        cutoff_time = time.time() - days * 86400  # Convert days to seconds.
        print('OUTPUT'.center(100, '-'))

        # Ensure the backup directory exists.
        if not os.path.exists(backup_dir):
            os.makedirs(backup_dir)
            print(f"Backup directory {backup_dir} created.")

        # Loop through all files in the source directory.
        for foldername, subfolders, filenames in os.walk(src_dir):
            for filename in filenames:
                # Get the full file path.
                file_path = os.path.join(foldername, filename)

                # Determine the backup file path.
                backup_file_path = os.path.join(backup_dir, filename)

                # Ensure the source file and backup file are not the same
                if os.path.abspath(file_path) == os.path.abspath(backup_file_path):
                    print(f"Skipping copy for {file_path} as it is the same as the
backup path.")
                    continue

                # Check if the file was modified within the last 'days' days.
                if os.path.getmtime(file_path) >= cutoff_time:
                    # Copy the file to the backup directory.
                    shutil.copy2(file_path, backup_file_path)
                    print(f"Copied: {file_path} to {backup_file_path}")

        print("Backup process completed successfully.")
        print('END'.center(100, '-'))

    except Exception as e:
        # Handle any unexpected errors during the backup process.
        print(f"An error occurred during the backup process: {e}")
```

```
try:
    # Define source and backup directories.
    print('INPUT'.center(100, '-'))
    src_directory = input("Enter the source directory path: ")
    backup_directory = input("Enter the backup directory path: ")

    # Get the number of days from the user to determine which files to back up.
    days_to_backup = int(input("Enter the number of days to look back for file
modifications: "))

    # Perform the backup.
    backup_files(src_directory, backup_directory, days_to_backup)

except ValueError:
    # Handle invalid input for the number of days.
    print("Invalid input! Please enter a valid number of days.")

except Exception as e:
    # Handle any unexpected errors.
    print(f"An error occurred: {e}")
```

Result

```
q6.py"
----------------------------------------------INPUT-----------------------------------------------
Enter the source directory path: ./
Enter the backup directory path: ../backup
Enter the number of days to look back for file modifications: 2
----------------------------------------------OUTPUT----------------------------------------------
Backup directory ../backup created.
Copied: ./python_q1.py to ../backup/python_q1.py
Copied: ./python_q5.py to ../backup/python_q5.py
Copied: ./.DS_Store to ../backup/.DS_Store
Copied: ./python_q4.py to ../backup/python_q4.py
Copied: ./persons_info.csv to ../backup/persons_info.csv
Copied: ./python_q3.py to ../backup/python_q3.py
Copied: ./test.ipynb to ../backup/test.ipynb
Copied: ./python_q2 to ../backup/python_q2
Copied: ./python_q6.py to ../backup/python_q6.py
Backup process completed successfully.
----------------------------------------------END-------------------------------------------------
```

Inference and Result:

i. Arithmetic Operations with User Input

The program continuously prompts the user for two numbers and an arithmetic operation, performing the calculation and displaying the result until the user chooses to exit. It ensures that all inputs are valid and handles division by zero gracefully.

ii. Check if a Number Belongs to the Fibonacci Series

The program verifies if a given number is part of the Fibonacci sequence by generating Fibonacci numbers up to that value and checking for membership, thus determining if the number is a Fibonacci number.

iii. Remove Punctuation from a String

The program processes a user-provided string to remove all punctuation marks, outputting a cleaned version of the string. This is achieved by using Python's string translation capabilities to filter out punctuation.

iv. Sort Members by Age

The program collects names and ages of five individuals, then sorts and displays them in ascending order of age. This involves storing data in a list of tuples and using Python's sorting functionality based on the age field.

v. Store Personal Information in CSV File

The program gathers personal information (name, date of birth, and blood group) from five individuals, computes their ages, and saves this data in a CSV file using a DataFrame. This allows for organized storage and easy retrieval of the data.

vi. Automate File Backup Based on Time

The script automates the backup of files from a source directory to a backup directory at regular intervals (e.g., daily) by using a loop and time-based sleep function, or it can be scheduled to run periodically via Windows Task Scheduler.