Assignment 1

Python program to calculate the simple interest.

```
def calculate_simple_interest(principal, rate, time):
    return (principal * rate * time) / 100
# Main program loop
while True:
    try:
        # Get user inputs
        principal = float(input("Enter the principal amount: "))
        rate = float(input("Enter the rate of interest (in %): "))
        time = float(input("Enter the time period (in years): "))
        # Calculate simple interest
        interest = calculate_simple_interest(principal, rate, time)
        print(f"The simple interest is: {interest}")
        # Ask the user if they want to calculate again
        again = input("Do you want to calculate again? (yes/no): ").strip().lower()
        if again != "yes":
            print("Thank you for using the Simple Interest Calculator!")
            break
    except ValueError:
        print("Invalid input. Please enter numeric values for principal, rate, and time.")
```

```
Enter the principal amount: 10000
Enter the rate of interest (in %): 10
Enter the time period (in years): 2
The simple interest is: 2000.0
Do you want to calculate again? (yes/no): no
Thank you for using the Simple Interest Calculator!
```

Write a Python program that:

1. Explores a given directory and lists all the files inside it.

2. Asks the user to input a word they want to search for in the files in that directory.

3. Searches for the specified word in each file and counts the occurrences of the word in each file.

4. Displays the filename(s) where the word is found and how many times it appears.

```
import os

# Function to search for a word in a file and count occurrences

def search_word_in_file(filename, word):

    try:

        with open(filename, 'r', encoding='utf-8') as file:

            content = file.read()

            return content.lower().count(word.lower())

    except Exception as e:

        print(f"Error reading file {filename}: {e}")

        return 0

# Function to explore the directory and list all files

def list_files_in_directory(directory):

    files = []

    try:

        for root, dirs, files_in_dir in os.walk(directory):

            for file in files_in_dir:

                files.append(os.path.join(root, file))

    except Exception as e:

        print(f"Error accessing directory {directory}: {e}")

    return files

# Main program

def main():

    # Ask the user for the directory to explore

    directory = input("Enter the directory path to explore: ").strip()
```

```
    # List all files in the directory
    files = list_files_in_directory(directory)
    if not files:
        print("No files found in the directory.")
        return
    print(f"Files found in the directory: {len(files)}")
    for file in files:
      print(file)
    # Ask the user for the word to search
    word = input("Enter the word you want to search for: ").strip()
    # Search for the word in each file and count occurrences
    for file in files:
        count = search_word_in_file(file, word)
        if count > 0:
            print(f"The word '{word}' was found {count} time(s) in the file: {file}")
if __name__ == "__main__":
    main()
```

two.txt

```
Hello world
Hello Python
Hello Java
Hello C
```

```
PS D:\Assignment> python main.py
Enter the directory path to explore: D:\Assignment
Files found in the directory: 2
D:\Assignment\main.py
D:\Assignment\two.txt
Enter the word you want to search for: Hello
The word 'Hello' was found 4 time(s) in the file: D:\Assignment\two.txt
```

Python program that implements a circular queue using a fixed-size list.

```
class CircularQueue:
    def __init__(self, size):
        self.size = size
        self.queue = [None] * size  # Initialize the queue with None values
        self.front = -1  # Points to the front of the queue
        self.rear = -1   # Points to the rear of the queue
    def is_empty(self):
        return self.front == -1
    def is_full(self):
        return (self.rear + 1) % self.size == self.front
    def enqueue(self, element):
        if self.is_full():
            print("Queue is full. Cannot enqueue.")
        else:
            if self.front == -1:  # If the queue is empty, set both front and rear to 0
                self.front = 0
            self.rear = (self.rear + 1) % self.size  # Circular increment
            self.queue[self.rear] = element
            print(f"Enqueued: {element}")
            self.display_queue()
    def dequeue(self):
        if self.is_empty():
            print("Queue is empty. Cannot dequeue.")
        else:
            dequeued_element = self.queue[self.front]
            if self.front == self.rear:  # Only one element in the queue
                self.front = self.rear = -1
            else:
                self.front = (self.front + 1) % self.size  # Circular increment
            print(f"Dequeued: {dequeued_element}")
```

```
        self.display_queue()
    def peek(self):
        if self.is_empty():
            print("Queue is empty. No front element.")
        else:
            print(f"Front element: {self.queue[self.front]}")
    def display_queue(self):
        if self.is_empty():
            print("Queue is empty.")
        else:
            idx = self.front
            elements = []
            while idx != self.rear:
                elements.append(self.queue[idx])
                idx = (idx + 1) % self.size
            elements.append(self.queue[self.rear])  # Add the rear element
            print("Current queue:", elements)

# Main program
def main():
    size = int(input("Enter the size of the queue: "))
    queue = CircularQueue(size)
    while True:
        print("\nOperations:")
        print("1. Enqueue")
        print("2. Dequeue")
        print("3. Peek")
        print("4. Check if Queue is Empty")
        print("5. Check if Queue is Full")
        print("6. Exit")
```

```
        choice = input("Enter your choice: ").strip()
        if choice == "1":
            element = input("Enter the element to enqueue: ").strip()
            queue.enqueue(element)
        elif choice == "2":
            queue.dequeue()
        elif choice == "3":
            queue.peek()
        elif choice == "4":
            if queue.is_empty():
                print("Queue is empty.")
            else:
                print("Queue is not empty.")
        elif choice == "5":
            if queue.is_full():
                print("Queue is full.")
            else:
                print("Queue is not full.")
        elif choice == "6":
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a valid option.")

if __name__ == "__main__":
    main()
```

```
Enter the size of the queue: 2

Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 1
Enter the element to enqueue: 10
Enqueued: 10
Current queue: ['10']
```

```
Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 4
Queue is not empty.
```

```
Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 5
Queue is not full.
```

```
Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 1
Enter the element to enqueue: 20
Enqueued: 20
Current queue: ['10', '20']
```

```
Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 2
Dequeued: 10
Current queue: ['20']
```

```
Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Queue is Empty
5. Check if Queue is Full
6. Exit
Enter your choice: 6
Exiting the program. Goodbye!
```

Python program to calculate the area and perimeter of different shapes,

```
import math
# Function to calculate area and perimeter of a rectangle
def calculate_rectangle():
    length = float(input("Enter the length of the rectangle: "))
    width = float(input("Enter the width of the rectangle: "))
    area = length * width
    perimeter = 2 * (length + width)
    print(f"Rectangle - Area: {area}, Perimeter: {perimeter}")
# Function to calculate area and perimeter of a square
def calculate_square():
    side = float(input("Enter the side length of the square: "))
    area = side ** 2
    perimeter = 4 * side
    print(f"Square - Area: {area}, Perimeter: {perimeter}")
# Function to calculate area and circumference of a circle
def calculate_circle():
    radius = float(input("Enter the radius of the circle: "))
    area = math.pi * radius ** 2
    circumference = 2 * math.pi * radius
    print(f"Circle - Area: {area:.2f}, Circumference: {circumference:.2f}")
# Main program
def main():
    while True:
        print("\nChoose a shape to calculate its area and perimeter/circumference:")
        print("1. Rectangle")
        print("2. Square")
        print("3. Circle")
        print("4. Exit")
        choice = input("Enter your choice (1/2/3/4): ").strip()
```

```
        if choice == "1":
            calculate_rectangle()
        elif choice == "2":
            calculate_square()
        elif choice == "3":
            calculate_circle()
        elif choice == "4":
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a valid option.")


if __name__ == "__main__":
    main()
```

```
Choose a shape to calculate its area and perimeter/circumference:
1. Rectangle
2. Square
3. Circle
4. Exit
Enter your choice (1/2/3/4): 1
Enter the length of the rectangle: 20
Enter the width of the rectangle: 10
Rectangle - Area: 200.0, Perimeter: 60.0

Choose a shape to calculate its area and perimeter/circumference:
1. Rectangle
2. Square
3. Circle
4. Exit
Enter your choice (1/2/3/4): 4
Exiting the program. Goodbye!
```

KBC (Kaun Banega Crorepati)-style Quiz

```
def load_questions(filename="questions.txt"):
    questions = []
    try:
        with open(filename, "r") as file:
            lines = file.readlines()
            i = 0
            while i < len(lines):
                # Skip any empty lines
                if lines[i].strip() == "":
                    i += 1
                    continue
                question = lines[i].strip()
                if i + 5 >= len(lines):
                    print(f"Warning: Missing options or answer for question at line {i+1}")
                    break
                options = {
                    'a': lines[i+1].strip(),
                    'b': lines[i+2].strip(),
                    'c': lines[i+3].strip(),
                    'd': lines[i+4].strip()
                }
                correct_answer = lines[i+5].strip()
                # Ensure that the correct answer is one of the options
                if correct_answer not in ['a', 'b', 'c', 'd']:
                    print(f"Warning: Invalid correct answer for question at line {i+1}. Expected one of 'a', 'b', 'c', or 'd'.")
                    break
                questions.append((question, options, correct_answer))
                i += 6  # Move to the next question
    except FileNotFoundError:
        print(f"Error: The file '{filename}' was not found.")
```

```
        return []
    return questions
def ask_question(question, options, correct_answer):
    print("\n" + question)
    for option, answer in options.items():
        print(f"{option}) {answer}")
    user_answer = input("Enter your answer (a, b, c, d): ").strip().lower()
    if user_answer == correct_answer:
        print("Correct!")
        return True
    else:
        print(f"Wrong! The correct answer was {correct_answer}.")
        return False
def play_game():
    questions = load_questions()
    if not questions:
        print("No questions to play. Exiting.")
        return
    score = 0
    for question, options, correct_answer in questions:
        if ask_question(question, options, correct_answer):
            score += 1
        else:
            break
    print(f"\nGame Over! Your score is {score}/{len(questions)}.")


if __name__ == "__main__":
    play_game()
```

Questions.txt:

What is the capital of France?

a) Berlin

b) Madrid

c) Paris

d) Rome

c

Who invented Python programming language?

a) Bjarne Stroustrup

b) Dennis Ritchie

c) Guido van Rossum

d) James Gosling

c

Which planet is known as the Red Planet?

a) Earth

b) Mars

c) Jupiter

d) Saturn

b

```
What is the capital of France?
a) a) Berlin
b) b) Madrid
c) c) Paris
d) d) Rome
Enter your answer (a, b, c, d): c
Correct!

Who invented Python programming language?
a) a) Bjarne Stroustrup
b) b) Dennis Ritchie
c) c) Guido van Rossum
d) d) James Gosling
Enter your answer (a, b, c, d): c
Correct!

Which planet is known as the Red Planet?
a) a) Earth
b) b) Mars
c) c) Jupiter
d) d) Saturn
Enter your answer (a, b, c, d): d
Wrong! The correct answer was b.

Game Over! Your score is 2/3.
```