Week 10 Assignment

Name: Drishti Durgesh Telgu
Student Id: SM20240093
Unit: ICT_102
Professor: Dr.Duy Nguyen

Part 1: Understanding User Defined Data Types

Create a Python script named user_defined_data_type.py that defines a class to represent a custom data type. This data type should include a constructor, attributes, and methods.

Define a class Book with the following attributes:

title: Title of the book (string)

author: Author of the book (string)

year: Year the book was published (integer)

price: Price of the book (float)

Include a method to display book details.

Code:

```
class Book:
    def __init__(self, title, author, year, price):
        self.title = title
        self.author = author
        self.year = year
        self.price = price

    def display_details(self):
        details = (f"Title: {self.title}\n"
                   f"Author: {self.author}\n"
                   f"Year: {self.year}\n"
                   f"Price: ${self.price:.2f}")
        print(details)
if __name__ == "__main__":
    book = Book("Drishti's Handbook", "George Orwell", 1949, 15.99)
```

```
book.display_details()
```

Output:

C:\Users\61411\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\61411\PycharmProjects\pythonProject\.venv\user_defined_data_type.py

Title: Drishti's Handbook

Author: George Orwell

Year: 1949

Price: $15.99

Process finished with exit code 0

Part 2: Understanding Python Class and Object

Create a Python script named class_and_object.py that demonstrates the creation and use of a class and object.

Define a class Student with the following attributes:

name: Name of the student (string)

roll_number: Roll number of the student (integer)

grades: List of grades (list of floats)

Include methods to:

Add a grade

Calculate the average grade

Display student details

Code:

```
class Student:
    def __init__(self, name, roll_number, grades=None):
        if grades is None:
```

```
            grades = []
        self.name = name
        self.roll_number = roll_number
        self.grades = grades

    def add_grade(self, grade):
        self.grades.append(grade)

    def calculate_average(self):
        if not self.grades:
            return 0.0
        return sum(self.grades) / len(self.grades)

    def display_details(self):
        details = (f"Name: {self.name}\n"
                   f"Roll Number: {self.roll_number}\n"
                   f"Grades: {self.grades}\n"
                   f"Average Grade: {self.calculate_average():.2f}")
        print(details)
if __name__ == "__main__":
    student = Student("Drishti", 123)
    student.add_grade(99.5)
    student.add_grade(92.0)
    student.display_details()
```

Output:

C:\Users\61411\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\61411\PycharmProjects\pythonProject\.venv\calss_and_object.py

Name: Drishti

Roll Number: 123

Grades: [99.5, 92.0]

Average Grade: 95.75

Process finished with exit code 0

Part 3: Understanding Constructor, Attributes, Methods

Create a Python script named constructor_attributes_methods.py that demonstrates the use of a constructor, attributes, and methods within a class.

Define a class Car with the following attributes:

make: Make of the car (string)

model: Model of the car (string)

year: Year the car was manufactured (integer)

mileage: Mileage of the car (integer)

Include methods to:

Drive the car (increase mileage)

Display car details

```python
class Car:
    def __init__(self, make, model, year, mileage):
        self.make = make
        self.model = model
        self.year = year
        self.mileage = mileage

    def drive(self, distance):
        self.mileage += distance

    def display_details(self):
        details = (f"Make: {self.make}\n"
                   f"Model: {self.model}\n"
                   f"Year: {self.year}\n"
                   f"Mileage: {self.mileage} miles")
        print(details)

if __name__ == "__main__":
    car = Car("Toyota", "Camry", 2020, 15000)
    car.drive(100)
    car.display_details()
```

Output:

C:\Users\61411\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\61411\PycharmProjects\pythonProject\.venv\constructor_attribute_methods.py

Make: Toyota

Model: Camry

Year: 2020

Mileage: 15100 miles

Process finished with exit code 0

Part 4: Writing a Program Using Class

Create a Python script named library_management.py that simulates a simple library management system using classes. The system should:

Define a class Library with attributes:

books: List of books (each book represented as a Book object from Part 1)

Include methods to:

Add a book

Remove a book

Display all books

```
from user_defined_data_type import Book


class Library:
    def __init__(self):
        self.books = []
```

```
    def add_book(self, book):
        self.books.append(book)

    def remove_book(self, title):
        self.books = [book for book in self.books if book.title != title]

    def display_all_books(self):
        if not self.books:
            print("No books available in the library.")
        for book in self.books:
            book.display_details()
            print()
if __name__ == "__main__":
    library = Library()
    book1 = Book("1984", "George Orwell", 1949, 15.99)
    book2 = Book("To Kill a Mockingbird", "Harper Lee", 1960, 12.99)
    library.add_book(book1)
    library.add_book(book2)

    print("Library Books:")
    library.display_all_books()

    library.remove_book("1984")
    print("Library Books after removal:")
    library.display_all_books()
```

Output:

C:\Users\61411\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\61411\PycharmProjects\pythonProject\.venv\library_management.py
Library Books:
Title: 1984
Author: George Orwell
Year: 1949
Price: $15.99

Title: To Kill a Mockingbird
Author: Harper Lee
Year: 1960
Price: $12.99

Library Books after removal:
Title: To Kill a Mockingbird
Author: Harper Lee
Year: 1960
Price: $12.99

Process finished with exit code 0