Assignment 5

- Define a Python class called `Shape` with the following specifications:
  - - Attributes: `name` (string), `color` (string).
  - - Methods: `__init__` (constructor to initialize attributes), `display_info` (prints the name and color).
  - Create an instance of the `Shape` class and demonstrate its usage.
- Define a Python class called `Rectangle` that inherits from the `Shape` class. Add the following specifications:
  - Additional attribute: `length` (float), `width` (float).
  - Additional method: `calculate_area` (calculates and sets the area of the rectangle).
  Create an instance of the `Rectangle` class, set its attributes, and demonstrate the usage of both `Shape` and `Rectangle` methods.

- Create a Python class called `Student` with the following specifications:
  Attributes: `name` (string), `age` (integer), `grades` (list of integers).
  Methods: `__init__` (constructor to initialize attributes), `average_grade` (calculates and returns the average grade).
  Create instances of the `Student` class for three different students and demonstrate the usage of the methods.

- Define a Python class called `BankAccount` with the following specifications:
  - Attributes: `account_number` (integer), `balance` (float).
  - Methods: `__init__` (constructor to initialize attributes), `deposit` (adds money to the balance), `withdraw` (subtracts money from the balance), `display_balance` (prints the current balance).
  Create an instance of the `BankAccount` class, perform several deposits and withdrawals, and demonstrate the usage of all methods.
- Create a Python class called `Car` with the following specifications:
  - Attributes: `make` (string), `model` (string), `year` (integer), `mileage` (float).
  - Methods: `__init__` (constructor to initialize attributes), `drive` (simulates driving by increasing the mileage), `display_info` (prints the make, model, year, and mileage).
  Create an instance of the `Car` class, perform a few drives, and demonstrate the usage of all methods

Given the list numbers = [1, 4, 7, 2, 5], write Python code to:

- Append the number 10 to the end of the list.
- Remove the element with the value 4.
- Print the length of the list.

- Create a Python dictionary representing a book with keys for title, author, and year. Print the author of the book.

- Given two sets: set1 = {1, 2, 3, 4} and set2 = {3, 4, 5, 6}, write Python code to find and print the union and intersection of the two sets.

TINY PROJECT

Consider a simple phone directory system.
Create a Python class named Contact to represent a contact with attributes name and phone_number. Then, create a PhoneDirectory class that stores a list of contacts.

Implement the Contact class with an __init__ method to initialize the name and phone number of a contact.

Implement the PhoneDirectory class with the following methods:

- __init__(self): Initializes an empty list to store contacts.
- add_contact(self, name, phone_number): Adds a new contact to the directory.
- get_contact(self, name): Retrieves the phone number of a contact by name. If the contact is not found, return "Contact not found."
- display_contacts(self): Displays the list of contacts in the directory.

Demonstrate the usage of these classes by creating an instance of PhoneDirectory, **adding** a few contacts, and **displaying** the contacts.

Book Class:

- The Book class should have attributes for title, author, and availability (whether the book is available for borrowing).
- Implement an __init__ method to initialize a book with the given title, author, and set availability to True by default.
- Include a method borrow_book that changes the availability status to False when the book is borrowed.
- Include a method return_book that changes the availability status to True when the book is returned.

Library Class:

- The Library class should have a list to store instances of the Book class.
- Implement an __init__ method to initialize an empty library.
- Include a method add_book that adds a new book to the library.
- Include a method display_books that displays all the books in the library, including their title, author, and availability.
- Include a method borrow_book that allows a user to borrow a book by changing its availability status.
- Include a method return_book that allows a user to return a book by changing its availability status.

Demonstrate Usage:

- Create an instance of the Library class.
- Add a few books using the add_book method.
- Display the list of books using the display_books method.
- Borrow a book using the borrow_book method.
- Display the list of books again to show the updated availability.
- Return a book using the return_book method.
- Display the list of books once more to show the updated availability.

The End