PYTHON DISCUSSION QUESTIONS

Loops

1. **Basic**: Write a Python program that prints all even numbers between 1 and 20 using a `for` loop.
2. **Intermediate**: Use a `while` loop to ask the user to input a number until they provide a number greater than 10.
3. **Advanced**: Write a program that prints the multiplication table (from 1 to 10) for numbers from 1 to 5 using nested `for` loops.
4. **Challenge**: Given a list of integers, `[4, 7, 2, 9, 12, 15]`, write a program using a `for` loop to find the sum of all odd numbers and print the result.

Lists

1. **Basic**: Create a list of 5 fruits and print each fruit on a new line using a `for` loop.
2. **Intermediate**: Write a function that takes a list of numbers and returns a new list with each number squared. Example: `[1, 2, 3]` should become `[1, 4, 9]`.
3. **Advanced**: Write a Python program that takes two lists, `list1 = [1, 2, 3]` and `list2 = [4, 5, 6]`, and combines them into a single list, `combined = [1, 4, 2, 5, 3, 6]`.
4. **Challenge**: Given a list of numbers, `[3, 1, 4, 1, 5, 9, 2]`, write a program to find and print the two largest numbers in the list without using the `max()` function.

Dictionaries

1. **Basic**: Create a dictionary with three key-value pairs representing a student's information: `name`, `age`, and `grade`. Print each key-value pair on a new line.

2. **Intermediate**: Write a function that takes a dictionary of people's names and their ages, `{'Alice': 24, 'Bob': 19, 'Charlie': 30}`, and returns a list of names of people who are 21 or older.
3. **Advanced**: Given a dictionary representing items in a store with their prices, `{'apple': 0.5, 'banana': 0.3, 'orange': 0.7}`, write a program that takes a list of items bought, `['apple', 'orange', 'banana', 'banana']`, and calculates the total cost.
4. **Challenge**: Write a program that counts the occurrences of each word in a given sentence. Example: for the sentence "hello world hello," the output should be `{'hello': 2, 'world': 1}`.

Object-Oriented Programming (OOP)

1. **Basic**: Create a class called `Car` with attributes `brand` and `color`. Instantiate an object of the `Car` class and print its attributes.
2. **Intermediate**: Add a method called `start_engine` to the `Car` class that prints a message saying the engine of the car has started. Create an instance of `Car` and call the method.
3. **Advanced**: Create a class called `BankAccount` with attributes `account_number` and `balance`. Add methods to:
    - Deposit an amount.
    - Withdraw an amount (only if sufficient balance exists).
    - Print the account balance.
4. **Challenge**: Implement a class called `Library` that manages a collection of books. Each book has a `title`, `author`, and `available` status. The `Library` class should have methods to:
    - Add a book to the library.
    - Remove a book from the library.
    - Check if a book is available by title.
    - Borrow a book (mark it as unavailable if it’s available).
    - Return a book (mark it as available again if it was borrowed).