Lab 1

1. Prime Numbers Dictionary

Goal

Create a Python program that:

1. Generates all prime numbers up to a user-specified limit.
2. Stores them in a dictionary with their **order** as the key.
3. Allows the user to view all primes or look up a prime by its position.

Features

1. **Check if a number is prime** using a function.
2. **Generate primes** up to a given limit using a function that returns a dictionary:

   ```
   {1: 2, 2: 3, 3: 5, 4: 7, ...}
   ```

3. **Display all primes** in the dictionary.
4. **Look up a prime** by its position.
5. **Menu loop** for interaction until the user exits.

Example Run

```
Enter upper limit for primes: 20
Prime dictionary created with 8 entries.

Menu:
1. View all primes
2. Get prime by position
3. Exit

Choose: 1
{1: 2, 2: 3, 3: 5, 4: 7, 5: 11, 6: 13, 7: 17, 8: 19}

Choose: 2
Enter position: 4
Prime at position 4 is 7
```

Extra Challenges

- Allow the user to search for a prime number and see its position.
- Save/Load the dictionary to/from a json file.
- Display primes in a formatted table instead of plain dictionary output.

2. Rock-Paper-Scissors Game

Goal

Create a simple Rock-Paper-Scissors game where the user plays against the computer.

Features

1. The user chooses: rock, paper, or scissors.
2. The computer chooses randomly.
3. Determine the winner:
   - Rock beats Scissors
   - Scissors beats Paper
   - Paper beats Rock
4. Keep score over multiple rounds until the user quits.

Suggested Functions

- `get_user_choice()` → Read and validate user input.
- `get_computer_choice()` → Randomly choose rock/paper/scissors.
- `determine_winner(user, computer)` → Return result message.
- `play_game()` → Game loop.

Example Run

```
Rock, Paper, or Scissors? rock
Computer chose: scissors
You win!

Score: You 1 - Computer 0
Play again? (y/n): y
...
```

Extra Challenges

- Allow best-of-N rounds.
- Save score history to a file.

Note: Search for Python's `random` module to learn how to generate random choices.

3. Multiple-Choice Quiz Game (Bonus)

Goal

Create a quiz game with multiple-choice questions.

Features

1. Store questions, options, and correct answers in a **list of dictionaries**:

    ```
    quiz = [
        {
            "question": "What is the capital of France?",
            "options": ["A) Paris", "B) London", "C) Rome", "D) Berlin"],
            "answer": "A"
        },
        ...
    ]
    ```

2. Ask each question and display the options.
3. Read the user's answer and check if it matches the correct answer.
4. Keep score and display the final result.

Suggested Functions

- `ask_question(q_data)` → Display question and get answer.
- `run_quiz(quiz_data)` → Loop through all questions.
- `show_score(score, total)` → Print final score.

Example Run

```
Q1: What is the capital of France?
A) Paris
B) London
C) Rome
D) Berlin
Your answer: A
Correct!

...
Your final score: 4/5
```

Extra Challenges

- Shuffle questions and/or answer options.
- Allow the quiz to load from an external json file.