Python Basics - Assignments

Section 1: Python Basics Assignments

These assignments will help students develop **logical thinking** while learning Python basics like **data types, loops, functions, and conditionals**.

---

1. String Manipulation & Methods

Task:
Write a Python program that takes a sentence as input and: - Counts the number of words. - Reverses the order of words. - Converts the first letter of each word to uppercase. - Replaces all occurrences of "a" with "@".

Example Input:
`"python is an amazing language"`

Example Output:

```
Word Count: 5
Reversed Sentence: "language amazing an is python"
Title Case: "Python Is An Amazing Language"
Replaced Sentence: "python is @n @m@zing l@ngu@ge"
```

---

2. Number Guessing Game (Logical Thinking)

Task:
Create a **number guessing game** where: - The program randomly selects a number between **1 and 100**. - The user gets **7 attempts** to guess the correct number. - After each guess, the program should inform the user if the guess is **too high**, **too low**, or **correct**. - If the user fails in 7 attempts, display the correct number.

Example Output:

```
Guess the number (1-100): 50
Too high! Try again.
Guess the number (1-100): 30
Too low! Try again.
...
Correct! You guessed it in 4 attempts.
```

Hint: Use the `random` module.

---

3. List Processing & Operations

Task:
Write a program that: 1. Takes a list of numbers as input. 2. Removes duplicates. 3. Sorts the list in ascending order. 4. Finds the maximum and minimum number. 5. Finds the sum of all even numbers.

Example Input:

```
numbers = [5, 3, 8, 3, 10, 2, 8, 5]
```

Example Output:

```
Unique List: [2, 3, 5, 8, 10]
Sorted List: [2, 3, 5, 8, 10]
Max: 10, Min: 2
Sum of Even Numbers: 20
```

---

4. Dictionary Frequency Counter

Task:
Write a program that counts the occurrence of each word in a given string using a dictionary.

Example Input:
`"hello world hello python world"`

Example Output:

```
{'hello': 2, 'world': 2, 'python': 1}
```

Hint: Use `.split()` and dictionary methods.

---

5. Fibonacci Series (Recursion)

Task:
Write a function that prints the **Fibonacci sequence** up to `n` terms.

Example Input:

```
fibonacci(6)
```

Example Output:

```
0, 1, 1, 2, 3, 5
```

Hint: Use recursion.

---

6. File Handling Challenge

Task:
1. Ask the user to enter their name and age. 2. Save this information to a text file called `user_data.txt`. 3. Read and display the content of the file.

Example Interaction:

```
Enter your name: John Doe
Enter your age: 25
Data saved successfully!

Reading File...
John Doe, Age: 25
```

Hint: Use `open()` with `"w"` and `"r"` modes.

---

7. Rock-Paper-Scissors Game

Task:
Write a Python program that allows the user to play **Rock, Paper, Scissors** against the computer.

- The user inputs "rock", "paper", or "scissors".
- The computer randomly picks one.
- Display who wins.

Example Interaction:

```
Enter your choice (rock, paper, scissors): rock
Computer chose: scissors
You win!
```

Hint: Use the `random.choice()` function.

---

8. Palindrome Checker

Task:
Write a program that checks if a given word is a **palindrome** (reads the same forward and backward).

Example Input:

```
is_palindrome("madam")
```

Example Output:

```
True
```

Hint: Use string slicing.

---

9. Prime Number Checker

Task:
Write a Python function that checks if a number is **prime**.

Example Input:

```
is_prime(11)
```

Example Output:

```
True
```

Hint: A prime number is only divisible by 1 and itself.

---

10. Leap Year Checker

Task:
Write a Python program that determines if a given year is a **leap year**.

Example Input:

```
is_leap_year(2024)
```

Example Output:

```
True
```

Hint: A leap year is divisible by **4**, but not by **100** unless also divisible by **400**.

Bonus Challenge - Tic-Tac-Toe Game

Write a Python program that allows **two players** to play **Tic-Tac-Toe** in the console. The board should be displayed after every move, and the program should detect when a player wins.

Conclusion

These assignments will strengthen students' **problem-solving skills** while ensuring they practice **data structures, loops, functions, and control flow** effectively. Happy coding!