Title: Data Science with Python Tasks Submission

Your Name: Vishal Pandey
GitHub Profile: https://github.com/vishalP-08/Zillion.git
Course/Assignment Name: Data Science with Python, Task for Week 2
Date: 10 March 2025

Task 1: Build a simple command-line to-do list application in python

Problem:
The goal of this project was to develop a simple **To-Do List program** using Python. The program allows users to manage their tasks efficiently through a command-line interface (CLI). Users can **view, add, remove**, and **exit** the to-do list.

2. Problem Statement

A to-do list is essential for managing tasks effectively. The requirement was to create a **CLI-based to-do list application** that enables users to:

1. **Show Tasks** - Display all tasks currently in the list.
2. **Add Task** - Add a new task to the list.
3. **Remove Task** - Remove a specific task from the list.
4. **Exit** - Close the application.

The program should handle user input correctly and ensure smooth task management.

3. Source Code

```
def show_tasks(tasks):
    if not tasks:
        print("No tasks available.")
    else:
        print("\nTo-Do List:")
        for idx, task in enumerate(tasks, start=1):
            print(f"{idx}. {task}")
    print()

def add_task(tasks):
    task = input("Enter a new task: ")
    tasks.append(task)
    print("Task added successfully!\n")

def remove_task(tasks):
    show_tasks(tasks)
    if tasks:
        try:
            task_num = int(input("Enter the task number to remove: "))
            if 1 <= task_num <= len(tasks):
                removed_task = tasks.pop(task_num - 1)
```

4. Expected Output

```
To-Do List Menu:
1. Show Tasks
2. Add Task
3. Remove Task
4. Exit
Enter your choice: 2

Enter a new task: To make a list.
Task added successfully!

To-Do List Menu:
1. Show Tasks
2. Add Task
3. Remove Task
4. Exit
Enter your choice: 1

To-Do List:
1. To make a list.

To-Do List Menu:
1. Show Tasks
2. Add Task
3. Remove Task
4. Exit
Enter your choice: 3

To-Do List:
1. To make a list.
```

5. Explanation of the Code

1. **Data Structure Used**: The program stores tasks in a simple **list** (todo_list).
2. **Functions Implemented**:
    - show_tasks(): Displays the tasks in a numbered format.
    - add_task(): Takes user input and appends the task to the list.
    - remove_task(): Prompts the user for the task number and removes it from the list.
3. **Menu-Driven Approach**:
    - The program runs in a loop, presenting a menu to the user.
    - Based on the user's input, corresponding functions are executed.
    - The program continues until the user chooses to exit.
4. **Error Handling**: The program checks for invalid inputs (e.g., non-numeric input for task removal) and handles them gracefully.

Task 2: Random Password Generator

Task Overview: The objective of this project is to develop a Random Password Generator using Python. The program generates strong passwords consisting of letters, numbers, and special characters. The user specifies the desired password length, and the generated password is displayed in a user-friendly graphical interface (GUI) built with Tkinter.

Problem Explanation: With the increasing number of online accounts, having strong and secure passwords is crucial for security. Remembering manually created passwords can be difficult, and weak passwords are vulnerable to attacks. This program automates the process by generating a secure password instantly.

Code:

```
import tkinter as tk
from tkinter import messagebox
import random
import string

generated_passwords = []  # List to store the last 5 passwords

def generate_password():
    try:
        length = int(entry_length.get())
        if length < 1:
            messagebox.showerror("Error", "Password length must be at least 1")
            return

        characters = string.ascii_letters + string.digits + string.punctuation
        password = ''.join(random.choice(characters) for _ in range(length))

        # Enable, update password field, then disable again
        entry_password.config(state=tk.NORMAL)
        entry_password.delete(0, tk.END)
        entry_password.insert(0, password)
        entry_password.config(state=tk.DISABLED)

        # Save the last 5 passwords
        generated_passwords.append(password)
        if len(generated_passwords) > 5:
            generated_passwords.pop(0)

        # Update the password history display
        update_password_history()

    except ValueError:
        messagebox.showerror("Error", "Please enter a valid number")

def copy_password():
    root.clipboard_clear()
    root.clipboard_append(entry_password.get())
    root.update()
    messagebox.showinfo("Copied", "Password copied to clipboard!")

def update_password_history():
    history_text.config(state=tk.NORMAL)
```

Output Explanation:

1. The user enters the desired password length in the input box.
2. Upon clicking the "Generate Password" button, a secure password is generated.
3. The password consists of a mix of uppercase and lowercase letters, numbers, and special characters.
4. The password is displayed in a read-only text box to prevent accidental modification.

Enhancements Made:

- Implemented a read-only password field for security.
- Center-aligned the generated password for better readability.
- Stored the last 5 generated passwords for reference.
- Ensured the generated password can be copied but not edited.

Task 3: Simple Calculator

Problem Statement:

Develop a simple GUI-based calculator using Python and Tkinter. The calculator should perform basic arithmetic operations and have a properly aligned UI with no extra spacing between buttons.

Code:

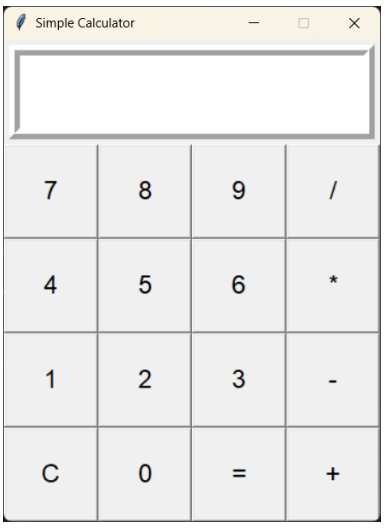
```
import tkinter as tk

def on_click(button_text):
    current_text = entry_var.get()
    if button_text == "=":
        try:
            result = str(eval(current_text))
            entry_var.set(result)
        except:
            entry_var.set("Error")
    elif button_text == "C":
        entry_var.set("")
    else:
        entry_var.set(current_text + button_text)

# Create main window
root = tk.Tk()
root.title("Simple Calculator")
root.geometry("350x450")  # Adjusted for compact layout
root.resizable(False, False)

entry_var = tk.StringVar()
entry = tk.Entry(root, textvariable=entry_var, font=("Arial", 20), bd=10, relief="ridge",
justify="right")
entry.grid(row=0, column=0, columnspan=4, ipadx=10, ipady=10, pady=5, padx=5, sticky="nsew")
```

Output:



Explanation:

This program creates a GUI-based calculator using Tkinter. The on_click function handles input and evaluation of expressions.

Task 4: Computer Knowledge Quiz Game

Problem Statement:

Write a command-line program in python to make Quiz game where you can ask user some questions and user can give input and for every correct answer they will get +**2 marks** for wrong answer they get **-1 marks**. And at the end show the total score.

Code:

```
# Basic Computer Knowledge Quiz Game

questions = [
    {
        "question": "What does CPU stand for?",
        "options": ["Central Processing Unit", "Computer Personal Unit", "Central Processor Utility",
"Central Performance Unit"],
        "answer": "A"
    },
    {
        "question": "Which of the following is an output device?",
        "options": ["Keyboard", "Mouse", "Monitor", "Scanner"],
        "answer": "C"
    },
    {
        "question": "What is the full form of RAM?",
        "options": ["Random Access Memory", "Read Access Memory", "Rapid Action Machine",
"Random Allocated Memory"],
        "answer": "A"
    },
    {
        "question": "Which of these is a web browser?",
        "options": ["Google", "Firefox", "Windows", "Linux"],
        "answer": "B"
    },
    {
        "question": "Which language is known as the 'Father of all programming languages'?",
        "options": ["Python", "C", "Assembly", "FORTRAN"],
        "answer": "D"
    },
    {
        "question": "What is the brain of the computer?",
        "options": ["RAM", "Hard Drive", "CPU", "GPU"],
        "answer": "C"
    },
    {
        "question": "Which one is an operating system?",
        "options": ["MS Word", "Windows", "Google Chrome", "Intel"],
        "answer": "B"
    },
    {
        "question": "What does HTTP stand for?",
        "options": ["Hyper Text Transfer Protocol", "High Transfer Text Protocol", "Hyperlink Transfer
Text Protocol", "Hyper Tool Text Processing"],
        "answer": "A"
    },
    {
        "question": "Which company developed the Windows operating system?"
```

Output:

A quiz game that asks 10 questions and calculates the total score based on correct and incorrect answers.

```
Welcome to the Basic Computer Knowledge Quiz!

Q1: What does CPU stand for?
A) Central Processing Unit
B) Computer Personal Unit
C) Central Processor Utility
D) Central Performance Unit

Your answer (A/B/C/D): a
• Correct! +2 points

Q2: Which of the following is an output device?
A) Keyboard
B) Mouse
C) Monitor
D) Scanner

Your answer (A/B/C/D): b
• Wrong! The correct answer was C. -1 point

Q3: What is the full form of RAM?
A) Random Access Memory
B) Read Access Memory
C) Rapid Action Machine
D) Random Allocated Memory

Your answer (A/B/C/D): a
• Correct! +2 points

Q4: Which of these is a web browser?
A) Google
B) Firefox
C) Windows
D) Linux
```

Explanation:

The program maintains a list of 10 questions with multiple-choice answers. The user selects an option, and the program evaluates correctness, awarding **+2 points** for correct answers and **-1 point** for wrong ones. At the end, feedback is provided based on the total score.