TASK-4 ROCK-PAPER-SCISSOR

This Python code implements a classic Rock-Paper-Scissors game with a graphical user interface (GUI) using the Tkinter library. The game allows the user to choose Rock, Paper, or Scissors, and the computer makes a random choice. The code determines the winner based on the standard game rules and updates the score accordingly. The GUI displays the user's and computer's choices, the game result, and the current scores. A reset button allows the user to clear the scores and start a new game. The game logic is encapsulated in functions for determining the winner, playing a round, and resetting scores, making the code modular and readable. The use of Unicode characters for the choices adds a visually appealing touch to the game.

This Python code creates a Rock-Paper-Scissors game with a graphical user interface (GUI) using the Tkinter library. Let's break down the code in detail:

1. Imports:
    - `import tkinter as tk`: Imports the Tkinter library for creating the GUI. `tk` is a common alias.
    - `from tkinter import messagebox`: Imports the `messagebox` module from Tkinter, although it's not actually used in this specific code. It's likely a leftover from an earlier version or intended for future use (e.g., to display a final game over message).
    - `import random`: Imports the `random` module for generating the computer's random choices.
    - `import json`: Imports the `json` module. This is also not used in the current code, suggesting it might be intended for saving/loading scores in a future version.
2. `determine_winner(user_choice, computer_choice)` Function:
    - Takes the user's and computer's choices (strings like "✊ Rock") as input.
    - Uses global variables `user_score`, `computer_score`, `wins`, `losses`, and `ties` to keep track of game statistics. Using `global` is necessary to modify these variables within the function.
    - Implements the game logic:

    - If choices are the same, it's a tie.
    - Uses a series of `or` conditions to check the winning combinations (Rock beats Scissors, Scissors beats Paper, Paper beats Rock).
    - Increments the appropriate score and win/loss/tie counters.
  - Returns a string indicating the result of the round ("It's a tie!", "You win!", or "You lose!").

3. `play_game(user_choice)` Function:
   - Takes the user's choice as input.
   - `computer_choice = random.choice(["👊 Rock", "📄 Paper", "✂️ Scissors"])`: Randomly selects the computer's choice.
   - Calls `determine_winner()` to get the result of the round.
   - `result_label.config(...)`: Updates the `result_label` on the GUI to display both the user's and computer's choices, and the final result message.
   - `score_label.config(...)`: Updates the `score_label` to display the current scores.
4. `reset_scores()` Function:
   - Resets all the score variables (`user_score`, `computer_score`, `wins`, `losses`, `ties`) to 0.
   - Updates the `score_label` and `result_label` to reflect the reset.
5. GUI Setup:
   - `root = tk.Tk()`: Creates the main application window.
   - `root.title(...)`: Sets the title of the window.
   - `root.geometry(...)`: Sets the size of the window.
6. Global Variables:
   - Initializes the score variables (`user_score`, `computer_score`, `wins`, `losses`, `ties`) to 0. These are declared outside any function so they can be accessed and modified by multiple functions.
7. GUI Elements:
   - `instructions = tk.Label(...)`: Creates a label to display instructions.
   - `button_frame = tk.Frame(...)`: Creates a frame to hold the buttons. This helps organize the layout.

- `rock_button`, `paper_button`, `scissors_button`: Creates the buttons for the user's choices. The `command` argument is crucial; it specifies the function to be called when the button is clicked. `lambda: play_game(...)` creates an anonymous function that calls `play_game` with the corresponding choice.
- `result_label`: Creates a label to display the game result.
- `score_label`: Creates a label to display the scores.
- `reset_button`: Creates a button to reset the scores.

8. Layout:
    - `pack()` and `grid()`: These are layout managers used to arrange the GUI elements within the window. `pack()` is used for simpler layouts, while `grid()` allows for more complex, grid-based arrangements.
9. `root.mainloop()`:
    - Starts the Tkinter event loop. This is essential; it makes the GUI interactive and responsive to user input. The program execution will stay in the `mainloop` until the window is closed.

In summary: The code creates a functional Rock-Paper-Scissors game with a simple GUI. While the `json` and `messagebox` imports aren't used, they suggest potential future enhancements like saving scores or adding more complex game logic. The code is well-structured with functions for different parts of the game, making it reasonably easy to understand and modify.