Project 12-1: Game Stats

Create a program that allows you to view the statistics for a player of a game.

Console

```
Game Stats program

ALL PLAYERS:
Elizabeth
Joel
Mike

Enter a player name: elizabeth
Wins:   41
Losses: 3
Ties:   22

Continue? (y/n): y

Enter a player name: john
There is no player named John.

Continue? (y/n): y

Enter a player name: joel
Wins:   32
Losses: 14
Ties:   17

Continue? (y/n): y

Enter a player name: mike
Wins:   8
Losses: 19
Ties:   11

Continue? (y/n): n

Bye!
```

Specifications

- The program should use a dictionary of dictionaries to store the stats (wins, losses, and ties) for each player. You can code this dictionary of dictionaries at the beginning of the program using any names and statistics that you want. Make sure to provide stats for at least three players.
- The program should begin by displaying an alphabetical list of the names of the players.
- The program should allow the user to view the stats for the specified player.

Student Guide: Python Game Statistics Tracker

Introduction

This guide now includes explanation sections to help you understand the Python concepts applied in the code. These explanations will provide insights into why certain code structures are used and how they contribute to the functionality of the program.

Step 1: Writing and Testing `display_names`

Task

Write a function named `display_names` that takes a dictionary of players and displays their names in alphabetical order.

Explanation

This function demonstrates:

1. **Function Definition**: `def display_names(players):` defines a new function with one parameter, `players`.
2. **Dictionary Handling**: `players` is expected to be a dictionary. Dictionaries in Python are key-value pairs.
3. **List Operations**: `list(players.keys())` converts the dictionary keys (player names) into a list.
4. **Sorting**: `names.sort()` sorts the list alphabetically.
5. **Looping**: The `for` loop iterates over each name in the sorted list.
6. **Printing**: `print(name)` displays each name.

Instructions

1. **Code**: Type the following function in your Python file:

```python
def display_names(players):
    names = list(players.keys())
    names.sort()
    print("ALL PLAYERS: ")
    for name in names:
        print(name)
    print()
```

2. **Test**: Create a test dictionary and call the `display_names` function, inside `main()`.

```python
def main():
    # Test code for display_names
    players_test = {"Alice": {}, "Bob": {}, "Charlie": {}}
    display_names(players_test)
if __name__ == "__main__":
    main()
```

3. **Verify**: Ensure that the player names are printed in alphabetical order.

Step 2: Writing and Testing `display_stats`

Task

Develop a function named `display_stats` that asks for a player's name and displays their statistics.

Explanation

This function demonstrates:

1. **Input Handling**: `input("Enter a player name: ").title()` gets user input and capitalizes the first letter of each word.
2. **Conditional Statements**: `if name in players.keys():` checks if the entered name is a key in the dictionary.
3. **Accessing Dictionary Values**: `player = players[name]` accesses the value (another dictionary) associated with the key `name`.
4. **String Formatting**: `print(f"There is no player named {name}.")` uses an f-string for formatted output.

Instructions

1. **Code**: Add the `display_stats` function to your file.

```python
def display_stats(players):
    name = input("Enter a player name: ").title()
    if name in players.keys():
        player = players[name]
        print("Wins:   ", player["wins"])
```

```
            print("Losses: ", player["losses"])
            print("Ties:   ", player["ties"])
        else:
            print(f"There is no player named {name}.")
        print()
```

2. **Test**: Use a dictionary with player statistics and test the function.
3. **Verify**: Check that the stats are correctly displayed for existing players and that entering an unknown player shows an error message.

Step 3: Implementing and Testing `main`

Task

Implement the `main` function to initialize player data and control the flow of the program.

Explanation

This function demonstrates:

1. **Program Entry Point**: `if __name__ == "__main__":` ensures that the code runs only if the script is executed as the main program.
2. **Infinite Loop with Exit Condition**: `while choice.lower() == "y":` creates a loop that continues until the user decides to exit.
3. **Function Calls**: `display_names(players)` and `display_stats(players)` call the functions defined earlier.

Instructions

1. **Code**: Complete your program with the `main` function.
2. **Test**: Run the entire program and interact with it.
3. **Verify**: Ensure that all functionalities work as expected, including displaying player names and stats, and the ability to repeatedly ask the user for input.

Conclusion

With these added explanations, you should have a better understanding of the Python concepts used in this program. Practice implementing these concepts in different scenarios to enhance your coding skills.

Source Code - p12-1_game_stats.py

python

```python
#!/usr/bin/env python3

def display_names(players):
    names = list(players.keys())
    names.sort()
    print("ALL PLAYERS: ")
    for name in names:
        print(name)
    print()


def display_stats(players):
    name = input("Enter a player name: ").title()
    if name in players.keys():
        player = players[name]
        print("Wins:   ", player["wins"])
        print("Losses: ", player["losses"])
        print("Ties:   ", player["ties"])
    else:
        print(f"There is no player named {name}.")
    print()

def main():
    print("Game Stats program")
    print()

    players = {
        "Joel": {"wins": 32, "losses": 14, "ties": 17},
        "Elizabeth": {"wins": 41, "losses": 3, "ties": 22},
        "Mike": {"wins": 8, "losses": 19, "ties": 11}
    }

    display_names(players)

    choice = "y"
    while choice.lower() == "y":
        display_stats(players)

        # ask if user wants to continue
        choice = input("Continue? (y/n): ")
        print()

    print("Bye!")

if __name__ == "__main__":
    main()
```