Python for Sports Science: Advanced Stats Calculator Walkthrough

This guide covers the core concepts of **Control Flow** and **Functions** in Python, using them to build an advanced basketball shooting efficiency calculator.

Section 1: Core Python Concepts

Concept	Description	Basketball Example
Function (def)	A named block of reusable code used to perform a specific task or encapsulate a formula.	Defining a function to calculate **eFG%** (Effective Field Goal Percentage) or **TS%** (True Shooting Percentage) given player stats.
Control Flow (if/elif/else)	Conditional logic that allows the program to make decisions and execute different code blocks based on whether a condition is True or False.	Printing a message like "Elite shooting night!" if a player's eFG% is above a certain threshold.
Loop (for)	Iteration used to repeat a block of code for every item in a sequence (like a list of players).	Calculating and printing stats for a **list of multiple players** without having to write the code repeatedly.

Section 2: Building the Advanced Stats Functions

The first step is to define functions that represent the mathematical formulas for advanced efficiency. These are found in the Week2_Assignment_AdvancedStatsCalculator.ipynb notebook (Step 1).

2.1 Effective Field Goal Percentage (eFG%)

This statistic adjusts for the added value of a three-point shot.

$$\text{eFG%} = \frac{\text{FGM} + 0.5 \times \text{3PM}}{\text{FGA}}$$

Python Function Code:

Python

```
def effective_fg(FGM, threePM, FGA):
    """Calculates Effective Field Goal Percentage (eFG%)."""
    return (FGM + 0.5 * threePM) / FGA
```

2.2 True Shooting Percentage (TS%)

This is the most comprehensive efficiency metric, including field goals and free throws.

$$\text{TS%} = \frac{\text{PTS}}{2 \times (\text{FGA} + 0.44 \times \text{FTA})}$$
Python Function Code:

```Python
def true_shooting(PTS, FGA, FTA):
    """Calculates True Shooting Percentage (TS%)."""
    return PTS / (2 * (FGA + 0.44 * FTA))
```

Section 3: Applying Control Flow for Analysis

Once the functions are defined, you can use them and then use an `if/else` statement to provide analytical feedback (Step 3).

3.1 Efficiency Message Script

This script uses an `if/elif/else` structure to evaluate the calculated eFG% and print a message about the player's performance.

- **Goal:** Use the **eFG%** output from the function to determine a performance category.

Python Control Flow Code:

```Python
# Assume 'efg' has been calculated by the effective_fg function
efg = 0.575  # Example value

if efg > 0.6:
    print("Elite shooting night!")
elif efg >= 0.5:
    print("Above average efficiency.")
else:
    print("Below average efficiency.")

# Output for efg=0.575: "Above average efficiency."
```

Section 4: Extra Challenge – Analyzing Multiple Players

The **`for` loop** is essential for analyzing entire datasets quickly (Extra Challenge).

4.1 Looping and Comparing Players

This script iterates through a list of player statistics (stored as dictionaries), calculates the TS% for each, and uses a comparison (`if`) inside the loop to track the **most efficient player**.

Python Loop Code:

Python

```
players = [
    {"name": "Player A", "FGM": 10, "FGA": 20, "3PM": 3, "FTM": 6, "FTA": 8, "PTS": 29},
    {"name": "Player B", "FGM": 5, "FGA": 12, "3PM": 1, "FTM": 2, "FTA": 2, "PTS": 13},
    {"name": "Player C", "FGM": 8, "FGA": 15, "3PM": 2, "FTM": 4, "FTA": 5, "PTS": 22},
]

best_ts = -1  # Initialize to a value lower than any possible TS%
most_efficient_player = ""

for player in players:
    # 1. Calculate the True Shooting Percentage for the current player
    ts_player = true_shooting(player["PTS"], player["FGA"], player["FTA"])

    # 2. Check if this player is the most efficient found so far
    if ts_player > best_ts:
        best_ts = ts_player
        most_efficient_player = player['name']

    # Print the current player's result
    print(f"{player['name']}: TS% = {round(ts_player, 3)}")


print(f"\nMOST EFFICIENT PLAYER: {most_efficient_player} (TS% of {round(best_ts, 3)})")
```