🧠 Week 6 Student Guide – Statistical Basketball Analysis with NumPy

This guide will help you complete the **Week 6 Assignment** by focusing on how to use the **NumPy** library for vector-based math, descriptive statistics, and correlation analysis of basketball data.

💻 Part 1: NumPy and Foundational Statistics Review

NumPy (Numerical Python) is the core library for scientific computing in Python. It provides high-performance array objects and tools for working with these arrays (also known as vectors) efficiently.

1. Vector Creation and Broadcasting

In NumPy, data is stored in N-dimensional arrays. This structure enables a powerful feature called **Broadcasting**, which allows you to perform arithmetic operations (like addition, multiplication, or division) on entire arrays at once, without needing a `for` loop.

Concept	Python Code Example	Purpose
Array Creation	`points = np.array([27, 22, 15, 18, 8])`	Converts a standard Python list into a high-speed NumPy array.
Broadcasting	`efficiency = points * 0.5 + rebounds * 1.2 + assists * 0.8`	Multiplies and adds entire arrays element-wise with scalar values. This is how you compute custom efficiency formulas quickly.

2. Descriptive Statistics

NumPy provides simple functions for calculating key statistical measures for a single array.

Statistic	NumPy Function	Description	Python Code Example
Mean (Average)	`np.mean()`	The sum of values divided by the count.	`np.mean(points)`
Standard Deviation	`np.std()`	Measures the amount of variation or dispersion from the average. A **higher** value means the data points are more spread out.	`np.std(points)`
Median	`np.median()`	The middle value when the data is sorted.	`np.median(rebounds)`

3. Correlation Analysis

Correlation measures the strength and direction of a linear relationship between two numerical variables. The result is the **Correlation Coefficient (r)**, which ranges from -1.0 to +1.0.

Correlation (r)	Relationship
$r \approx 1.0$	**Strong Positive:** As one stat goes up, the other tends to go up (e.g., Points and Field Goal Attempts).
$r \approx -1.0$	**Strong Negative:** As one stat goes up, the other tends to go down (e.g., Minutes Played and Fatigue Score).
$r \approx 0.0$	**Weak/No Correlation:** No clear linear relationship.

- **Key Function:** np.corrcoef(array1, array2)
    - This function returns a correlation matrix. You access the correlation coefficient between the two arrays using the index **[0, 1]**.

🏀 Part 2: Week 6 Assignment Walkthrough

Your assignment involves creating a small box score dataset and applying the NumPy tools reviewed above.

1. Create Your Dataset (Code Cell 2)

Define your arrays using np.array().

Python
```
players = np.array(["Player A", "Player B", "Player C", "Player D", "Player E"])
points = np.array([25, 19, 22, 15, 10])
rebounds = np.array([9, 6, 11, 8, 5])
assists = np.array([7, 5, 6, 4, 3])
```

2. Calculate Key Metrics (Code Cell 3)

Use NumPy's functions to calculate the required descriptive statistics and the correlation coefficient.

Python
```
# Calculate and store the metrics
avg_points = np.mean(points)
std_points = np.std(points)
median_rebounds = np.median(rebounds)

# Calculate correlation between points and rebounds
# The [0, 1] index extracts the correlation coefficient
```

```
corr_pr = np.corrcoef(points, rebounds)[0, 1]
```

3. Compute Player Efficiency (Code Cell 5)

Use **Broadcasting** to calculate a custom efficiency metric and compile the results into a Pandas DataFrame.

- **Formula:** $\text{Efficiency} = (\text{Points} \times 0.5) + (\text{Rebounds} \times 1.2) + (\text{Assists} \times 0.8)$

Python

```
# Calculate efficiency using broadcasting
efficiency = points * 0.5 + rebounds * 1.2 + assists * 0.8

# Create a final DataFrame
df = pd.DataFrame({
    "Player": players,
    "Points": points,
    "Rebounds": rebounds,
    "Assists": assists,
    "Efficiency": efficiency
})
# The DataFrame output shows Player C has the highest efficiency score (29.0).
```

4. Visualize Your Findings (Code Cell 6)

Create a bar chart comparing the calculated Efficiency score for each player.

Python

```
plt.figure(figsize=(8,5))
plt.bar(df["Player"], df["Efficiency"], color="purple")
plt.title("Player Efficiency Comparison")
plt.ylabel("Efficiency Score")
plt.show()
```

5. Reflection (Code Cell 7)

Use the output from your code (especially the DataFrame and the correlation) to answer the reflection questions.

- **Which player had the best efficiency?** Check the Efficiency column of your DataFrame (It is **Player C** with a score of 29.0).
- **How strongly are rebounds related to points?** Refer to the **Correlation** output ($r \approx 0.73$), which indicates a **strong positive correlation**.
- **What might improve the team's performance overall?** Consider the range of scores (standard deviation) and the correlation.