# **💼 Challenge 1: The Financial Analyst’s Dilemma 💰**

⏳ **Time: 30 minutes**  
📝 **Concepts Covered:** NumPy array creation, statistics (mean, max), normalization, boolean masking

### **📌 Scenario:**

You are a financial analyst at a top investment firm. Your boss hands you a dataset containing **stock prices of 5 companies** over a period of **30 days**. Your task is to analyze the data and extract meaningful insights!

### **🛠 Task:**

1. **Create a (30, 5) NumPy array** with random stock prices between **$100 and $500**, where:
   * **Rows (0-29)** represent **days** (30 days of stock prices).
   * **Columns (0-4)** represent **5 different companies**.
2. Compute the **average stock price** for each company over the 30-day period.
3. Identify **the highest stock price recorded** in the dataset and the **day and company** it belongs to.
4. **Normalize the stock prices** between **0 and 1** using the formula:  
   [ X\_{} = ]
5. Flag days where any stock **drops below $200** as a “⚠️ Risky Investment”.

### **🖥 Sample Input & Output:**

#### **🔹 Sample Input (Generated Randomly)**

stock\_prices = np.array([  
 [120, 340, 450, 220, 310],  
 [330, 420, 380, 250, 410],  
 [150, 280, 290, 230, 200],  
 # ... (27 more rows)  
])

#### **🔹 Expected Output**

Average stock prices: [230.5 350.2 400.1 260.9 320.8]  
Highest price recorded: 480 at Day 14, Company 3  
Normalized prices:  
[[0.1 0.7 0.85 0.3 0.6 ]  
 [0.6 0.8 0.72 0.35 0.75 ]  
 ...  
]  
Risky Investment Days:  
Day 1: [120, 150]  
Day 10: [180, 195]

# **🚀 Challenge 2: The Space Mission Resource Allocation 🌌**

⏳ **Time: 30 minutes**  
📝 **Concepts Covered:** Array creation, sum, max, standard deviation, transposition

### **📌 Scenario:**

NASA is preparing a **Mars mission** 🚀 and needs to efficiently allocate **oxygen, water, and food** for astronauts over **6 months**. Your job as a space engineer is to analyze and optimize resource consumption!

### **🛠 Task:**

1. **Create a (6, 3) NumPy array**, where:
   * **Rows (0-5)** represent **months**.
   * **Columns (0-2)** represent **oxygen, water, and food** consumption (in tons).
2. Compute **total resources required** for the entire mission.
3. Identify **which resource had the highest consumption** in any given month and in total.
4. Compute the **standard deviation** of each resource to measure consistency in consumption.
5. **Transpose** the matrix to display the resource-wise breakdown by month.

### **🖥 Sample Input & Output:**

#### **🔹 Sample Input**

resources = np.array([  
 [15, 40, 32],  
 [20, 42, 35],  
 [25, 38, 30],  
 [18, 50, 40],  
 [22, 37, 36],  
 [28, 45, 33]  
])

#### **🔹 Expected Output**

Total resources needed (tons): Oxygen: 128, Water: 252, Food: 206  
Highest consumption in a month: Water (50 tons in month 4)  
Standard deviation of consumption: Oxygen: 4.6, Water: 4.5, Food: 3.8  
Transposed matrix:  
[[15 20 25 18 22 28] # Oxygen  
 [40 42 38 50 37 45] # Water  
 [32 35 30 40 36 33]] # Food

# **🏀 Challenge 3: Basketball Performance Analyzer 📊**

⏳ **Time: 30 minutes**  
📝 **Concepts Covered:** Indexing, filtering, sorting, statistics

### **📌 Scenario:**

You are a **sports data analyst** for an NBA team! You need to analyze the performance of **5 players** over **10 games** to determine their strengths and weaknesses.

### **🛠 Task:**

1. **Create a (5, 10) NumPy array**, where:
   * **Rows (0-4)** represent **players**.
   * **Columns (0-9)** represent **points scored in 10 games**.
2. Compute **each player’s average points per game**.
3. Identify the **best-performing** and **worst-performing players** based on total points.
4. Create a **mask to filter out games where a player scored above 30 points** (highlighting exceptional performances).
5. **Sort players based on their total points in descending order**.

### **🖥 Sample Input & Output:**

#### **🔹 Sample Input**

points = np.array([  
 [10, 25, 30, 22, 12, 18, 26, 32, 24, 29],  
 [20, 15, 12, 28, 24, 26, 30, 18, 22, 25],  
 [35, 30, 32, 40, 28, 34, 29, 31, 26, 37],  
 [12, 18, 20, 15, 22, 14, 19, 21, 23, 17],  
 [28, 26, 27, 25, 30, 29, 35, 32, 31, 38]  
])

#### **🔹 Expected Output**

Average points per game:  
[22.8, 22.0, 32.2, 18.1, 30.1]  
  
Best-performing player: Player 3 (Total: 322 points)  
Worst-performing player: Player 4 (Total: 181 points)  
  
Games with scores above 30:  
Player 1: Games [8]  
Player 3: Games [1, 3, 4, 7, 10]  
Player 5: Games [6, 9, 10]  
  
Sorted Players by Total Points:  
1. Player 3 - 322 points  
2. Player 5 - 301 points  
3. Player 1 - 228 points  
4. Player 2 - 220 points  
5. Player 4 - 181 points