Matrix Slicing - An Interactive Python Slicing Game

Welcome to "Matrix Slicing," a Jupyter Notebook designed to enhance your understanding of Python slicing through an interactive game. In this notebook, we will engage with a dynamic matrix slicing game that challenges you to apply Python slicing techniques in varied scenarios. This hands-on approach aims to solidify the understanding of list and array manipulations using Python's powerful slicing syntax.

Sample Output:

```
C:\Users\PC\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\PC\PycharmProjects\pythonProject\.venv\Scripts\Slicing_game.py

New game started:

Matrix with highlighted section:
29 48 **49**
53 55 **75**
96 52 68
Enter the slice indices to extract from column 2 (e.g., 0:2 for first to second rows): 0:2
Correct! Well done.
Do you want to play again? (yes/no): yes

New game started:

Matrix with highlighted section:
**53** **62** 25
67 94 20
94 16 54
Enter the slice indices to extract from row 0 (e.g., 1:3 for second to third elements): 0:2
Correct! Well done.
Do you want to play again? (yes/no): yes

New game started:

Matrix with highlighted section:
56 36 41
51 53 89
**10** 26 65
Enter the slice indices to extract from column 0 (e.g., 0:2 for first to second rows): 2:3
Correct! Well done.
Do you want to play again? (yes/no): yes
```

```
New game started:

Matrix with highlighted section:
40 60 **37**
12 20 12
15 5 38
Enter the slice indices to extract from row 0 (e.g., 1:3 for second to third elements): 1:2
Incorrect slice. Try again. Expected was: [37]
Enter the slice indices to extract from row 0 (e.g., 1:3 for second to third elements): 1:2
Incorrect slice. Try again. Expected was: [37]
Enter the slice indices to extract from row 0 (e.g., 1:3 for second to third elements): 1:3
Incorrect slice. Try again. Expected was: [37]
No more attempts left. The correct answer was: 'matrixmatrix[0][2:3]'.
Do you want to play again? (yes/no): no
Thanks for playing our game! Goodbye.
```

Learning Objectives

By the end of this notebook, you will be able to:

- Understand and apply Python slicing syntax effectively.
- Manipulate matrices using slicing to extract specific data.
- Enhance problem-solving skills in Python with real-time feedback.

How to Use This Notebook

- Follow the instructions in each section to interact with the game.
- Run the Python cells to generate matrices and input your slices based on the game's challenges.
- Learn from feedback provided by the game to improve your slicing strategies.

Let's dive into the world of matrices and discover the versatility of slicing in Python!

Setup.

- Importing libraries
- Generate required matrix

In [5]:
```python
# Import necessary libraries
import random

# Function to generate a 3x3 matrix of random integers
def generate_matrix():
    return [[random.randint(1, 100) for _ in range(3)] for _ in range(3)]

# Example of a generated matrix
generate_matrix()
```

Out[5]:
```
[[18, 17, 23], [88, 5, 4], [24, 62, 65]]
```

Displaying the Matrix

To help players understand which part of the matrix they are supposed to slice, the game highlights the relevant rows or columns. Here's how we implement the display functionality:

In [8]:
```python
def print_matrix_with_highlight(matrix, slice_type, index, start, length):
    print("\nMatrix with highlighted section:")
    for i, row in enumerate(matrix):
        for j, val in enumerate(row):
            highlight = ((slice_type == 'row' and i == index and start <= j < start +
                          (slice_type == 'column' and j == index and start <= i < start
            print(f"**{val}**" if highlight else val, end=' ')
        print()
```

`print_matrix_with_highlight`

This function prints the matrix with specific sections highlighted to guide the player on which part to slice. It iterates through each element of the matrix:

- It checks if the current element falls within the specified slice range based on the `slice_type` ('row' or 'column').
- If the conditions are met (i.e., the element is within the highlighted slice), it prints the element surrounded by double asterisks (`**value**`), otherwise, it prints the element normally. This visualization helps players see exactly which part of the matrix they are working with.

User Input for Slicing

In [9]:
```python
def get_user_input(slice_type, index):
    slice_type_description = 'row' if slice_type == 'row' else 'column'
    example = '1:3 for second to third elements' if slice_type == 'row' else '0:2 for
    prompt = f"Enter the slice indices to extract from {slice_type_description} {index
    return input(prompt)
```

`get_user_input`

This function prompts the user to input the indices for the slice they want to make. Depending on whether the current game slice is a row or a column:

- It adjusts the prompt to specify the correct slice type and provides an example format for clarity.
- The function captures the user input, which should specify the start and end points of the slice. This interactive step is crucial for engaging the player in the game and testing their understanding of slicing syntax.

Main Game Loop

In [10]:
```python
# Assuming additional code for parse_slice and correct implementation

def main():
    while True:
        matrix = generate_matrix()
        slice_type, index, start, length = get_slice(matrix)  # Sample setup
        print_matrix_with_highlight(matrix, slice_type, index, start, length)

        attempts = 3
        correct = False
        while not correct and attempts > 0:
            slice_part = get_user_input(slice_type, index)
            correct, message = parse_slice(slice_part, index, matrix, (slice_type, inc
            print(message)
            attempts -= 1
            if not correct and attempts == 0:
                print("No more attempts left. Ending game.")

        if input("Do you want to play again? (yes/no): ").strip().lower() != "yes":
            print("Thanks for playing our game! Goodbye.")
            break
```

`main`

The `main` function orchestrates the entire game:

- It repeatedly generates a new matrix and determines a random slice to challenge the player.
- The matrix is displayed with the slice highlighted.
- Players are given multiple attempts to input the correct slice.
- Feedback is provided after each attempt. If the player fails all attempts, the game ends, or they can choose to play again. This loop continues until the player decides to stop, making the game an ongoing interactive session.

Sample Output:

```
New game started:

Matrix with highlighted section:
56 36 41
51 53 89
**10** 26 65
Enter the slice indices to extract from column 0 (e.g., 0:2 for first to second rows): 2:3
Correct! Well done.
Do you want to play again? (yes/no): yes

New game started:

Matrix with highlighted section:
40 60 **37**
12 20 12
15 5 38
Enter the slice indices to extract from row 0 (e.g., 1:3 for second to third elements): 1:2
Incorrect slice. Try again. Expected was: [37]
Enter the slice indices to extract from row 0 (e.g., 1:3 for second to third elements): 1:2
Incorrect slice. Try again. Expected was: [37]
Enter the slice indices to extract from row 0 (e.g., 1:3 for second to third elements): 1:3
Incorrect slice. Try again. Expected was: [37]
No more attempts left. The correct answer was: 'matrixmatrix[0][2:3]'.
Do you want to play again? (yes/no): no
Thanks for playing our game! Goodbye.
```

Fully functional code is available in my GitHub repository.

- GitHub: https://github.com/RajasekharUmapathy/100DaysofDatascience

Summary and Key Takeaways

Here are a few key points we've covered:

- **Python Slicing Syntax**: You've learned how to use slicing to access and manipulate data within Python lists and arrays.

- **Interactive Learning**: Through real-time feedback, this game provided an opportunity to experiment with and refine your slicing techniques.
- **Practical Application**: The skills you've developed here are applicable in various Python data manipulation tasks, enhancing your coding toolkit.

Further Exploration

To continue improving your Python skills, consider the following:

- **Modify the Game**: Add complexity by increasing matrix sizes or introducing non-linear slicing challenges.
- **Apply Skills in Projects**: Use slicing in your data science or analysis projects to handle datasets efficiently.
- **Explore More Python Features**: Dive deeper into Python's libraries like NumPy and pandas for more sophisticated data manipulation capabilities.

Thank you for playing "Matrix Slicing"! Keep slicing and dicing your way through data with Python!