🧪 Simulated Jupyter Notebook: Interactive Python

This notebook demonstrates the core structure of a Jupyter file, showing how **Markdown cells** (like this one) are mixed with **Code cells** and their **Output** to create a clear, documented workflow.

Cell 1: Defining a Simple Function

This is our first **Code Cell**. We're going to define a basic Python function that takes a number and squares it, just like you would in a regular script. Note that when we run this cell, it doesn't produce output yet; it just stores the function definition in memory.

```
# Code Cell 1: Define a function
def square_number(x):
    """Returns the square of a given number."""
    return x * x
```

Cell 2: Executing the Function

Now, in this second **Code Cell**, we will call the function we defined above and print the result. In a real notebook, the output appears immediately below this cell.

```
# Code Cell 2: Execute the function
number_to_test = 7
result = square_number(number_to_test)
print(f"The square of {number_to_test} is:")
print(result)
```

Output:

```
The square of 7 is:
49
```

Cell 3: Analysis of Results (Markdown Cell)

This **Markdown Cell** lets us document what just happened. The code was executed, and we confirmed that the `square_number` function works correctly, returning 49 for the input 7. This commentary makes the notebook easy to follow.

Cell 4: Using a Library for a Small Task

Jupyter Notebooks are fantastic for working with data. Here is another **Code Cell** where we import a built-in Python library (`random`) and use it to generate a short list of simulated weekly activity scores.

```
# Code Cell 4: Import a library and generate data
import random

# Generate 5 random "activity scores" between 50 and 100
scores = [random.randint(50, 100) for _ in range(5)]

print("Generated Weekly Scores:")
print(scores)
```

Output:

```
Generated Weekly Scores:
[88, 62, 95, 78, 51]
```

(Note: The numbers you see would change every time you run the cell!)

Cell 5: Simple Data Calculation

Finally, we use the list of scores generated in the previous cell to calculate the total score, demonstrating how variables persist across different Code Cells.

```
# Code Cell 5: Calculate the sum
total_score = sum(scores)
print(f"Total points earned this week: {total_score}")
```

Output:

```
Total points earned this week: 374
```

(The output number will correspond to the sum of the scores printed above.)