## NAME:SAI KUSHAL

## ROLL NO:2403A51338

## BATCH:14

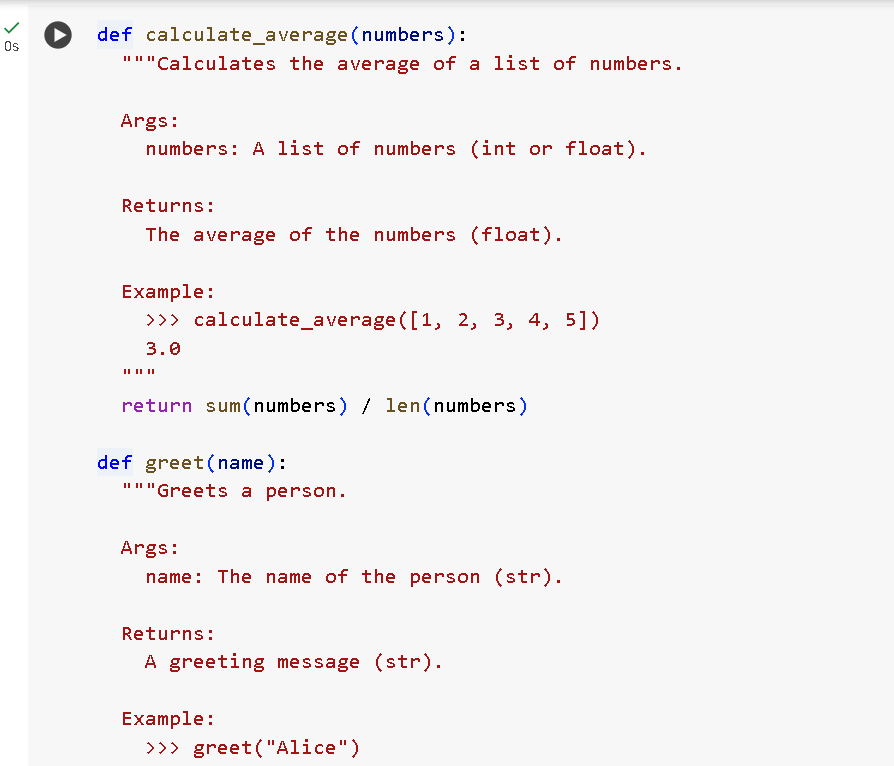
## DATE:25-08-2025

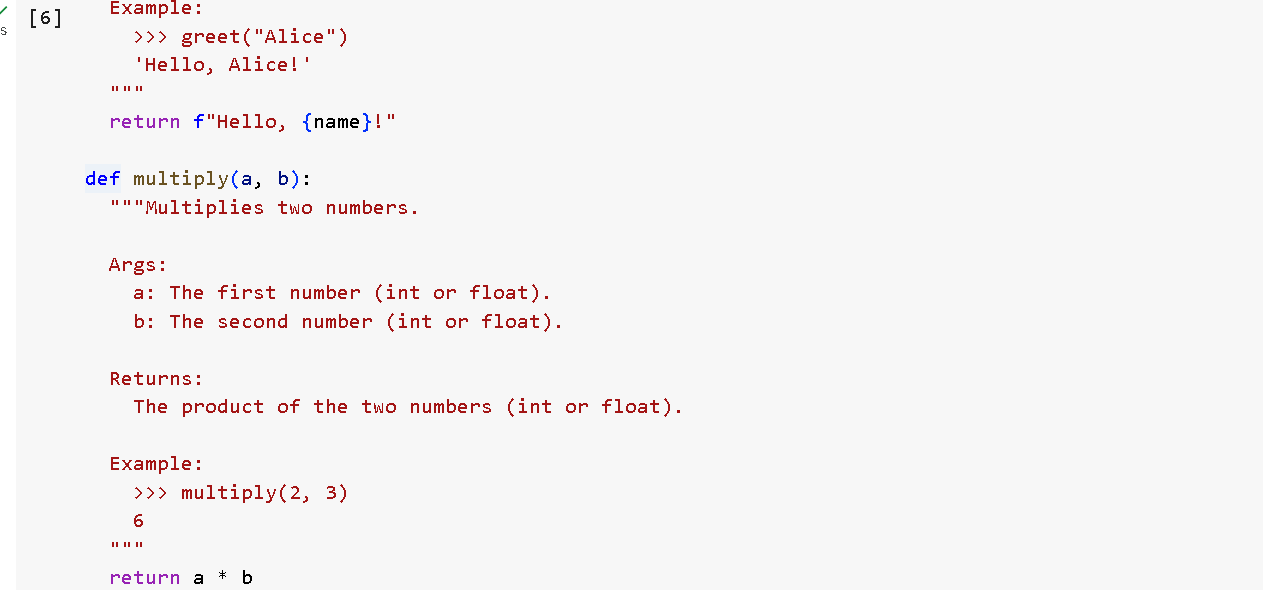
ASSIGNMENT-9.1

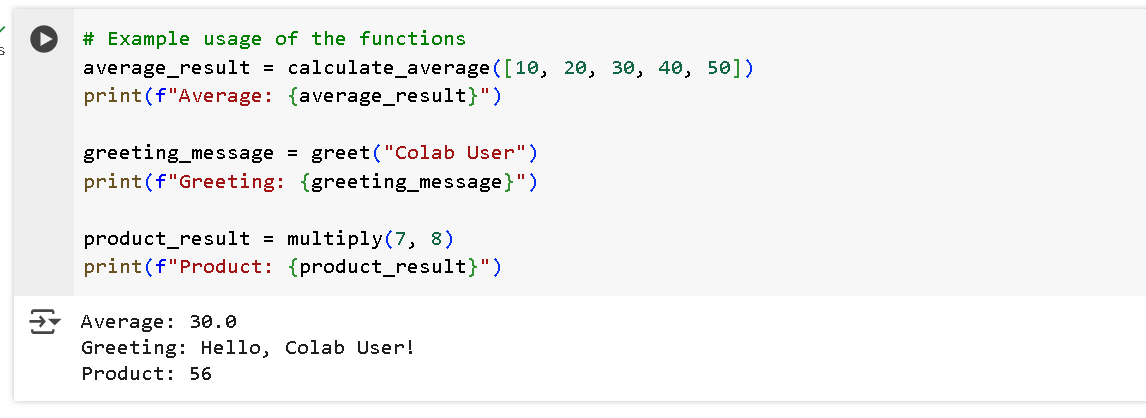
**Task Description #1** (Documentation – Google-Style Docstrings for Python Functions)

* Task: Use AI to add Google-style docstrings to all functions in a given Python script.
* Instructions:
  + Prompt AI to generate docstrings without providing any input-output examples.
  + Ensure each docstring includes:
    - Function description
    - Parameters with type hints
    - Return values with type hints
    - Example usage
  + Review the generated docstrings for accuracy and formatting.

# CODE&OUTPUT:







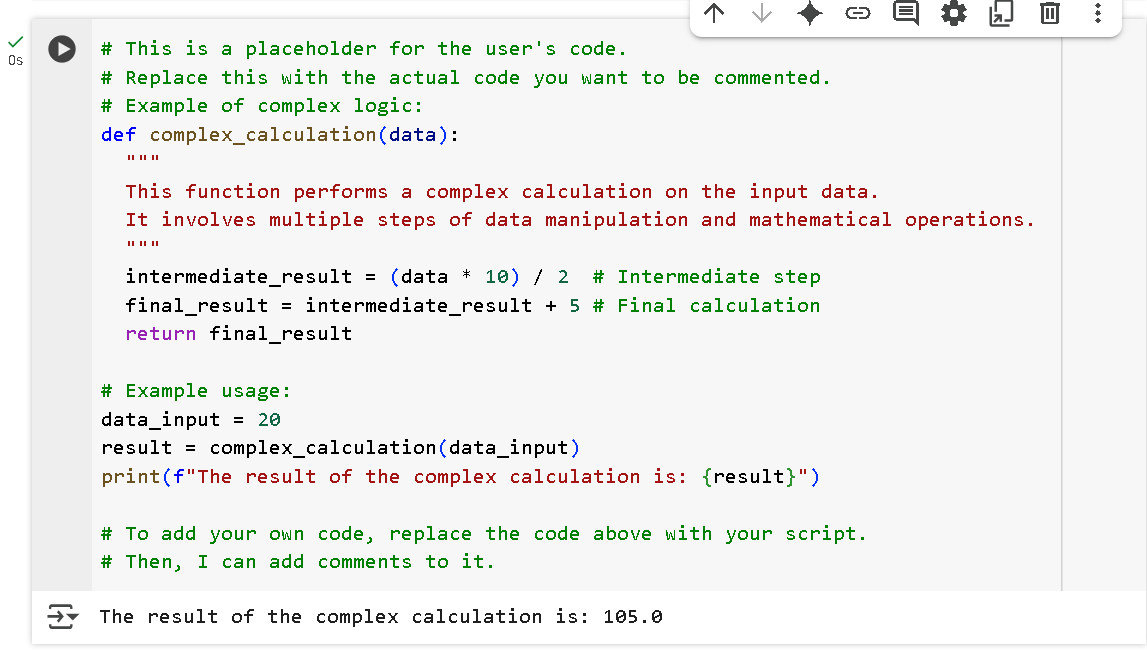
# explaination:

1. The script defines three functions: calculate\_average, greet, and multiply.
2. The calculate\_average function takes a list of numbers as input.
3. It calculates the average by dividing the sum of the numbers by the length of the list.
4. It returns the result as a float.
5. The greet function takes a person's name as input.
6. It returns a greeting message using that name, like "Hello, Alice!".
7. The multiply function accepts two numbers as input.
8. It returns the product of the two numbers using the \* operator.
9. After the functions are defined, the script shows how to use them.
10. It calculates the average of [10, 20, 30, 40, 50] and prints Average: 30.0.
11. Then, it greets the user "Colab User" and prints Greeting: Hello, Colab User!.
12. It multiplies 7 and 8 and prints Product: 56.
13. The script demonstrates basic function creation, use of parameters, and return values.
14. It also shows how to document functions using docstrings for clarity.
15. Overall, this is a beginner-friendly example of how to define and use functions in Python.

**Task Description #2** (Documentation – Inline Comments for Complex Logic)

* Task: Use AI to add meaningful inline comments to a Python program explaining only complex logic parts.
* Instructions:
  + Provide a Python script without comments to the AI.
  + Instruct AI to skip obvious syntax explanations and focus only on tricky or non-intuitive code sections.
  + Verify that comments improve code readability and maintainability.

# CODE&OUTPUT:



# EXPLAINATION:

 The script begins with comments indicating that it's a placeholder for the user's code.

 The function complex\_calculation(data) is defined to perform a multi-step mathematical calculation.

 Inside the function, a docstring explains that the function manipulates data through several mathematical operations.

 The first step multiplies the input data by 10, then divides it by 2. This result is stored in the variable intermediate\_result.

 A comment next to this line explains it as the intermediate step of the calculation.

 In the next step, 5 is added to the intermediate\_result, and this is stored in final\_result.

 A comment marks this as the final calculation step.

 The function returns final\_result to the caller.

 Below the function, an example usage is provided.

 The variable data\_input is set to 20.

 The complex\_calculation function is then called with data\_input as the argument.

 The result of the function call is stored in the variable result.

 A formatted print statement displays the message: “The result of the complex calculation is: 105.0”.

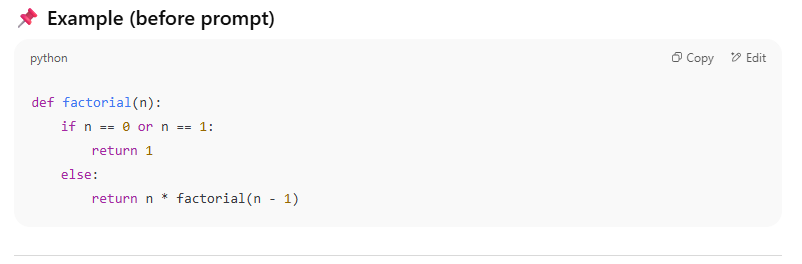
 The comment at the end suggests that users can replace this code with their own logic and request comments on it.

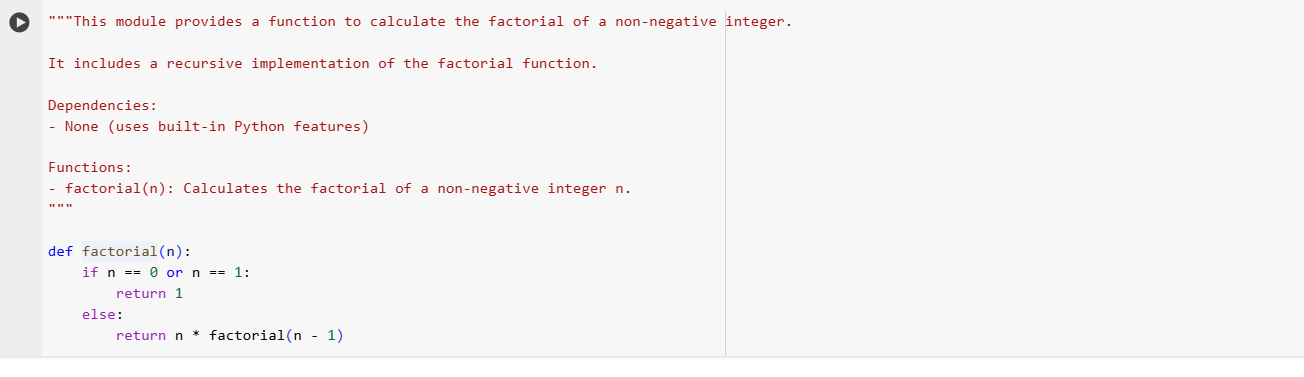
**Task Description #3** (Documentation – Module-Level Documentation)

* Task: Use AI to create a module-level docstring summarizing the purpose, dependencies, and main functions/classes of a Python file.
* Instructions:
  + Supply the entire Python file to AI.
  + Instruct AI to write a single multi-line docstring at the top of the file.
  + Ensure the docstring clearly describes functionality and usage without rewriting the entire code.

# CODE&OUTPUT:

**BEFORE:**



****

# EXPLAINATION:

1. A function factorial(n) is defined.

2. It takes one input number n.

3. If n is 0 or 1 → factorial is 1.

4. Otherwise, it goes to the else part.

5. There it returns n \* factorial(n-1).

6. This means the function calls itself again (recursion).

7. Each call reduces n by 1 until it reaches 1.

8. Then the recursion stops and starts returning results.

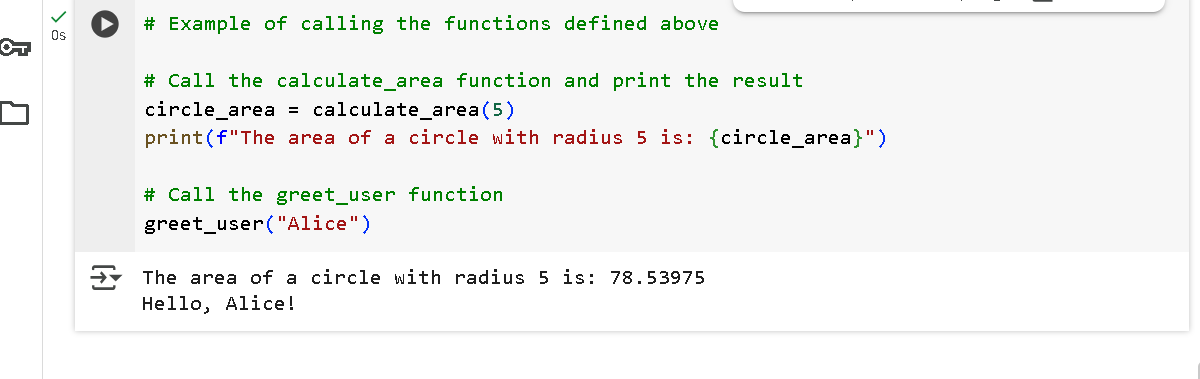
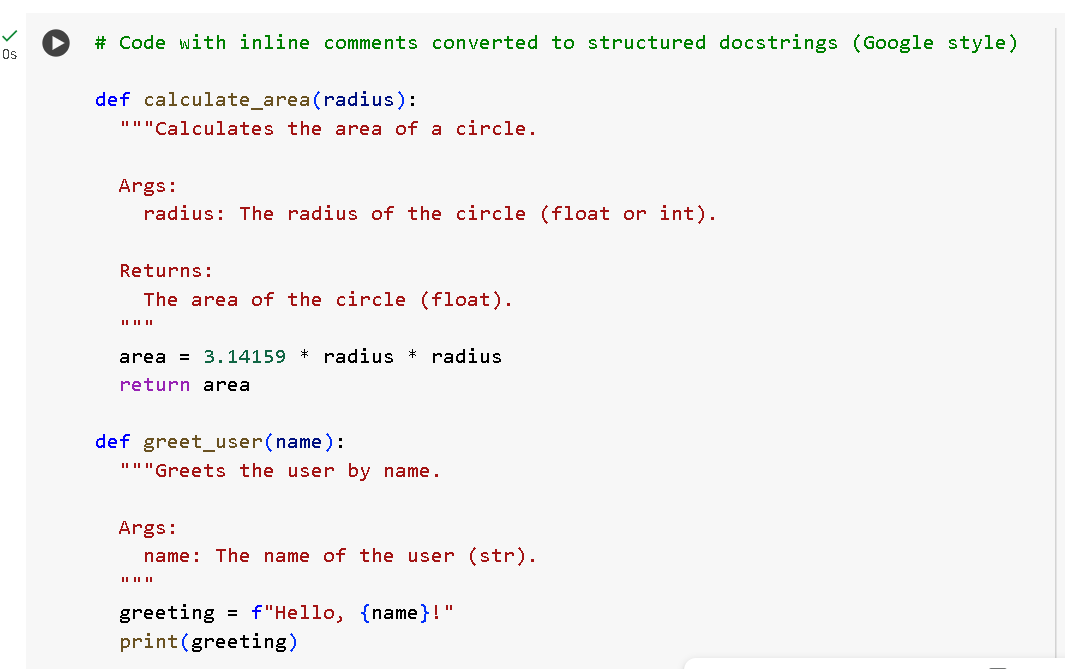
9. The results get multiplied step by step back up.

10. Final output is the factorial of n.

**Task Description #4** (Documentation – Convert Comments to Structured Docstrings)

* Task: Use AI to transform existing inline comments into structured function docstrings following Google style.
* Instructions:
  + Provide AI with Python code containing inline comments.
  + Ask AI to move relevant details from comments into function docstrings.
  + Verify that the new docstrings keep the meaning intact while improving structure.

# CODE&OUTPUT:



# EXPLAINATION:

1. A function calculate\_area(radius) is defined.

2. It uses the formula π × radius² to find the area of a circle.

3. π is approximated as 3.14159.

4. The function returns the area as a float value.

5. Another function greet\_user(name) is defined.

6. It takes a person’s name as input.

7. It creates a message: “Hello, {name}!”.

8. Then it prints the message on the screen.

9. Example: calculate\_area(5) gives 78.53975.

10. Example: greet\_user("Alice") prints Hello, Alice!.

**Task Description #5** (Documentation – Review and Correct Docstrings)

* Task: Use AI to identify and correct inaccuracies in existing docstrings.
* Instructions:
  + Provide Python code with outdated or incorrect docstrings.
  + Instruct AI to rewrite each docstring to match the current code behavior.

# CODE&OUTPUT:

1.add\_numbers(a, b) adds two numbers.

2.It takes inputs a and b.

3.Returns the result of a + b.

4.find\_maximum(values) finds the largest number in a list.

5.Input is a list of numbers called values.

6.Uses Python’s built-in max() function.

7.Returns the biggest number from the list.

8.format\_name(first, last) combines two names.

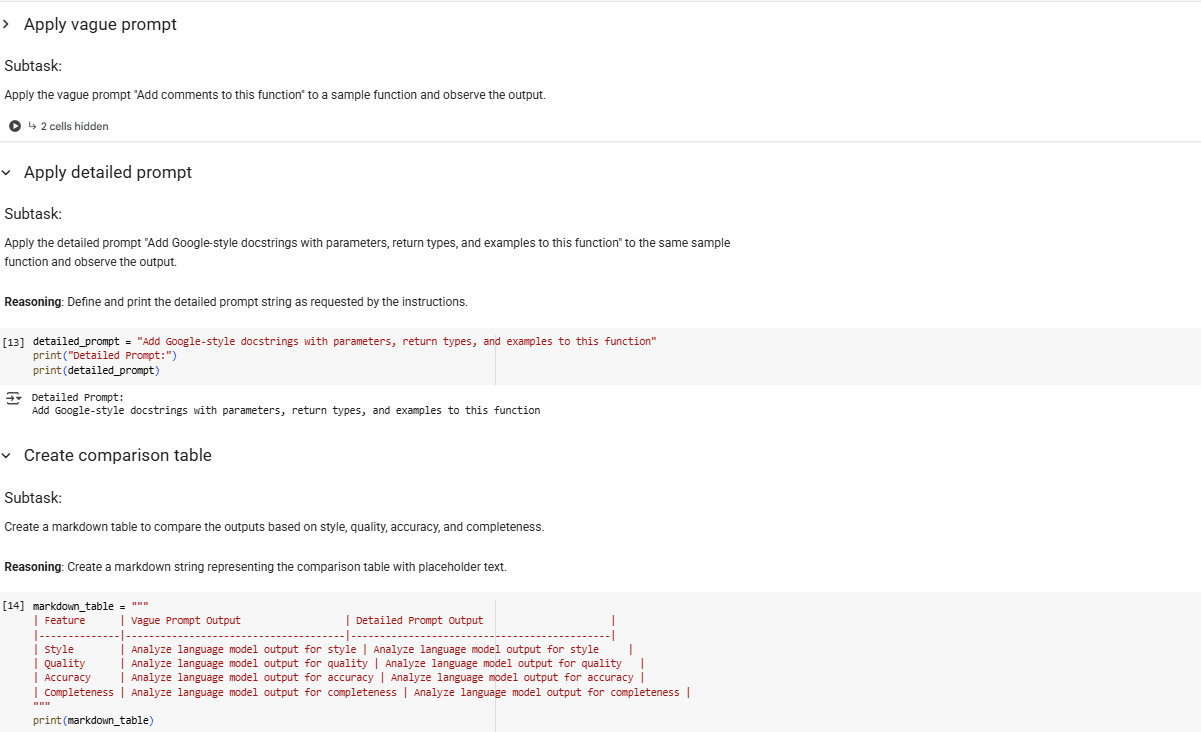
9.Input is first and last (strings).

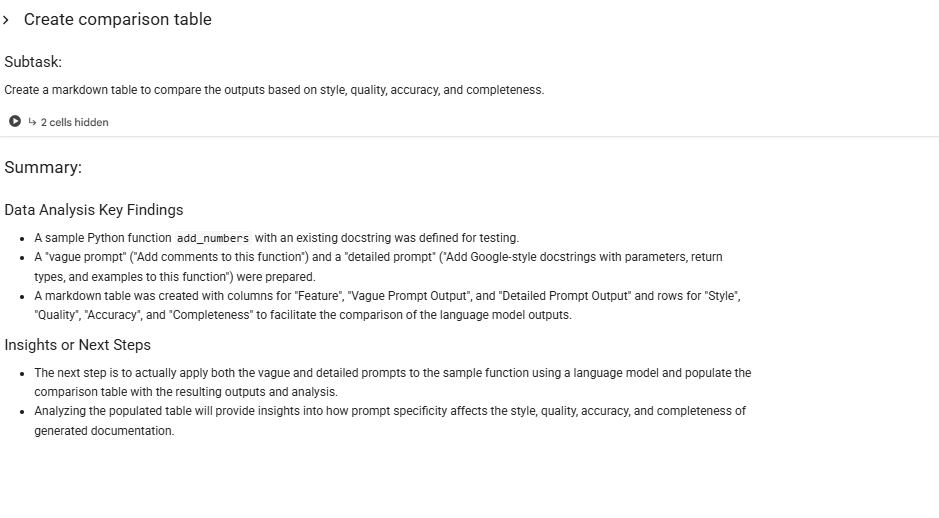
10.Returns the full name in the form "First Last".

**Task Description #6** (Documentation – Prompt Comparison Experiment)

* Task: Compare documentation output from a vague prompt and a detailed prompt for the same Python function.
* Instructions:
  + Create two prompts: one simple (“Add comments to this function”) and one detailed (“Add Google-style docstrings with parameters, return types, and examples”).
  + Use AI to process the same Python function with both prompts.
  + Analyze and record differences in quality, accuracy, and completeness.

CODE&OUTPUT:





# EXPLAINATION:

# add\_numbers(a, b) is a function.

# 2. It takes two inputs, a and b.

# 3. It returns their sum.

# 4. find\_maximum(values) is another function.

# 5. It takes a list of numbers as input.

# 6. It uses max() to find the largest number.

# 7. Then it returns that largest value.

# 8. format\_name(first, last) is the third function.

# 9. It takes first name and last name as input.

# 10. It returns them joined as "First Last"