ASSIGNMENT-6.1

2303A51635

Batch:23

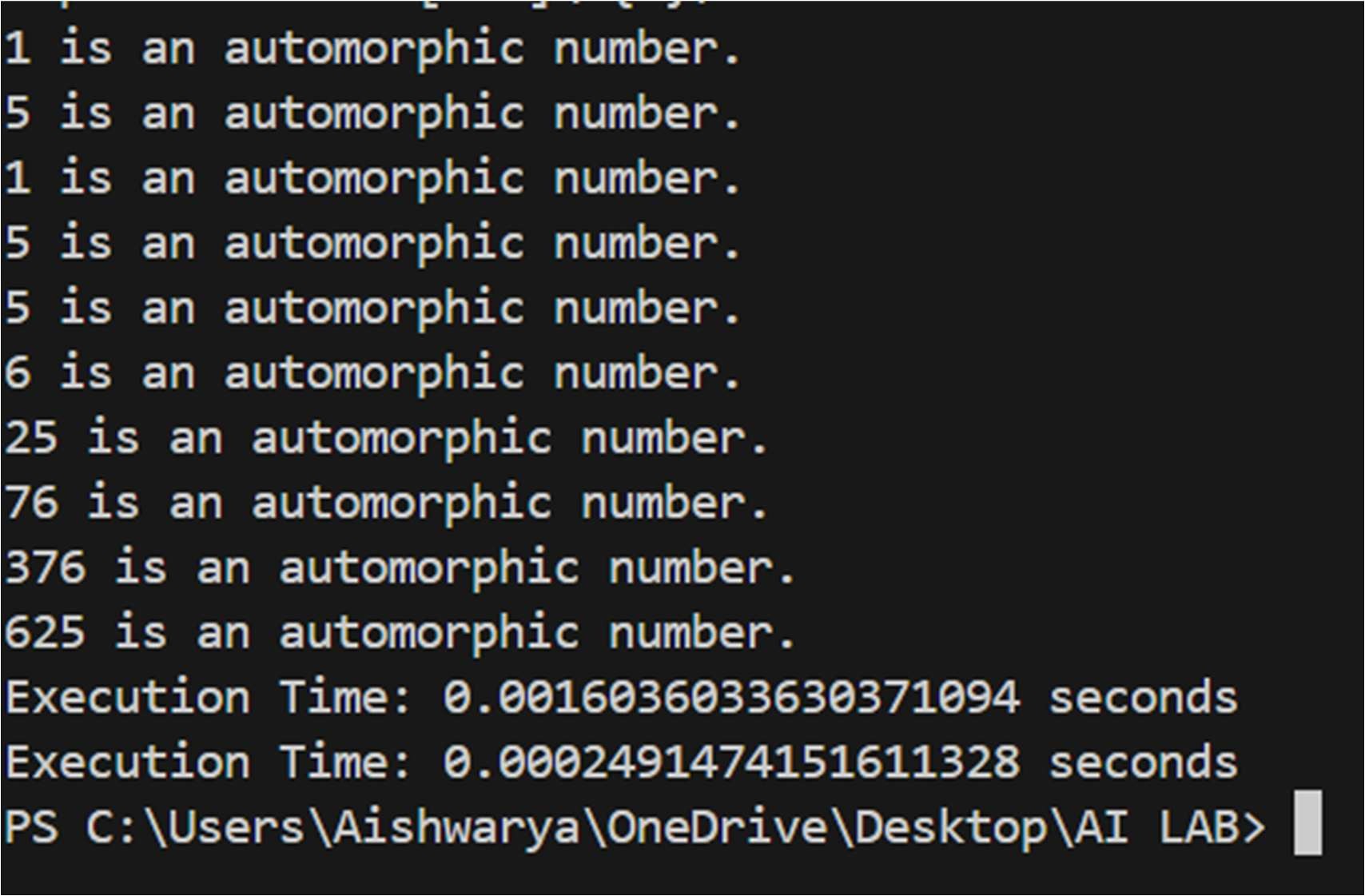
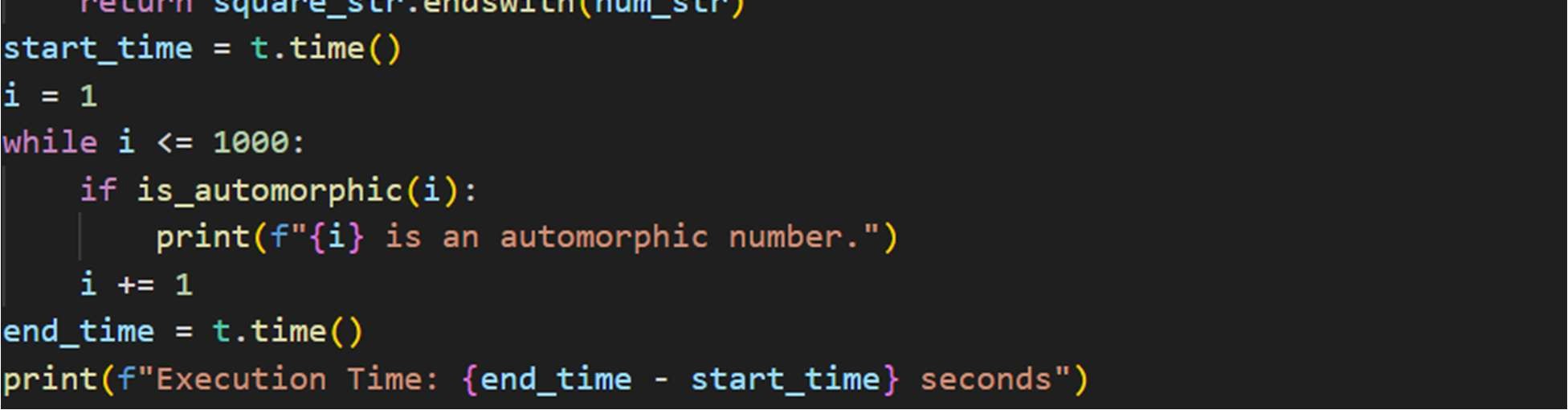
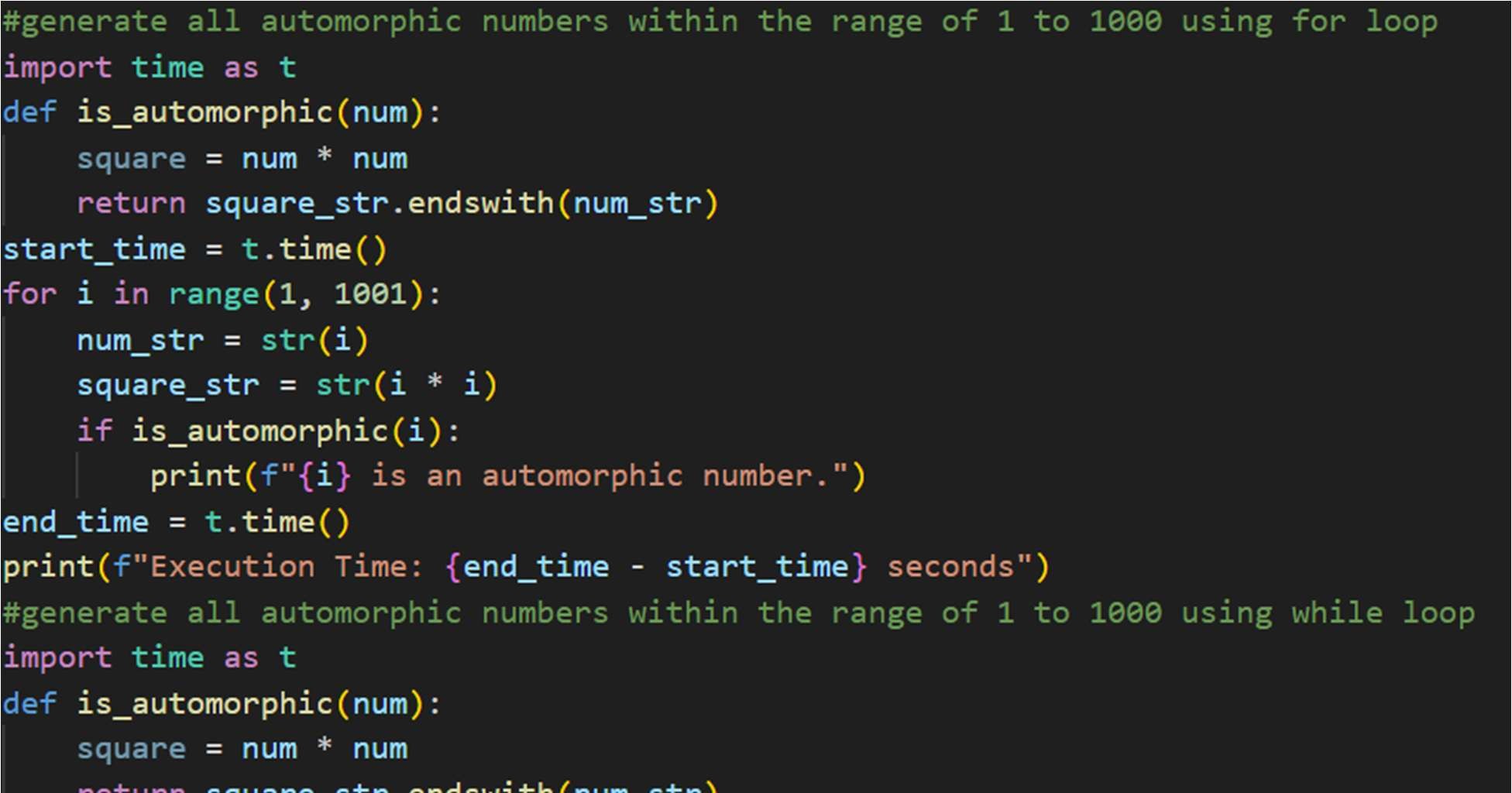
Task Description #1 (Loops – Automorphic Numbers in a Range) • Task: Prompt AI to generate a function that displays all Automorphic numbers between 1 and 1000 using a for loop.

• Instructions:

o Get AI-generated code to list Automorphic numbers using a for loop. o Analyze the correctness and e iciency of the generated logic. o Ask AI to regenerate using a while loop and compare both implementations.

Expected Output #1:

* Correct implementation that lists Automorphic numbers using both loop types, with explanation



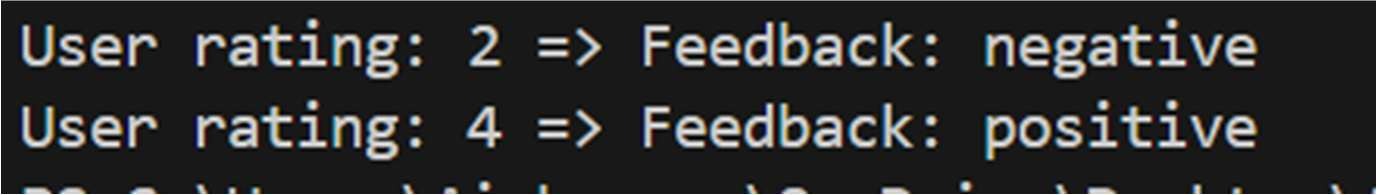
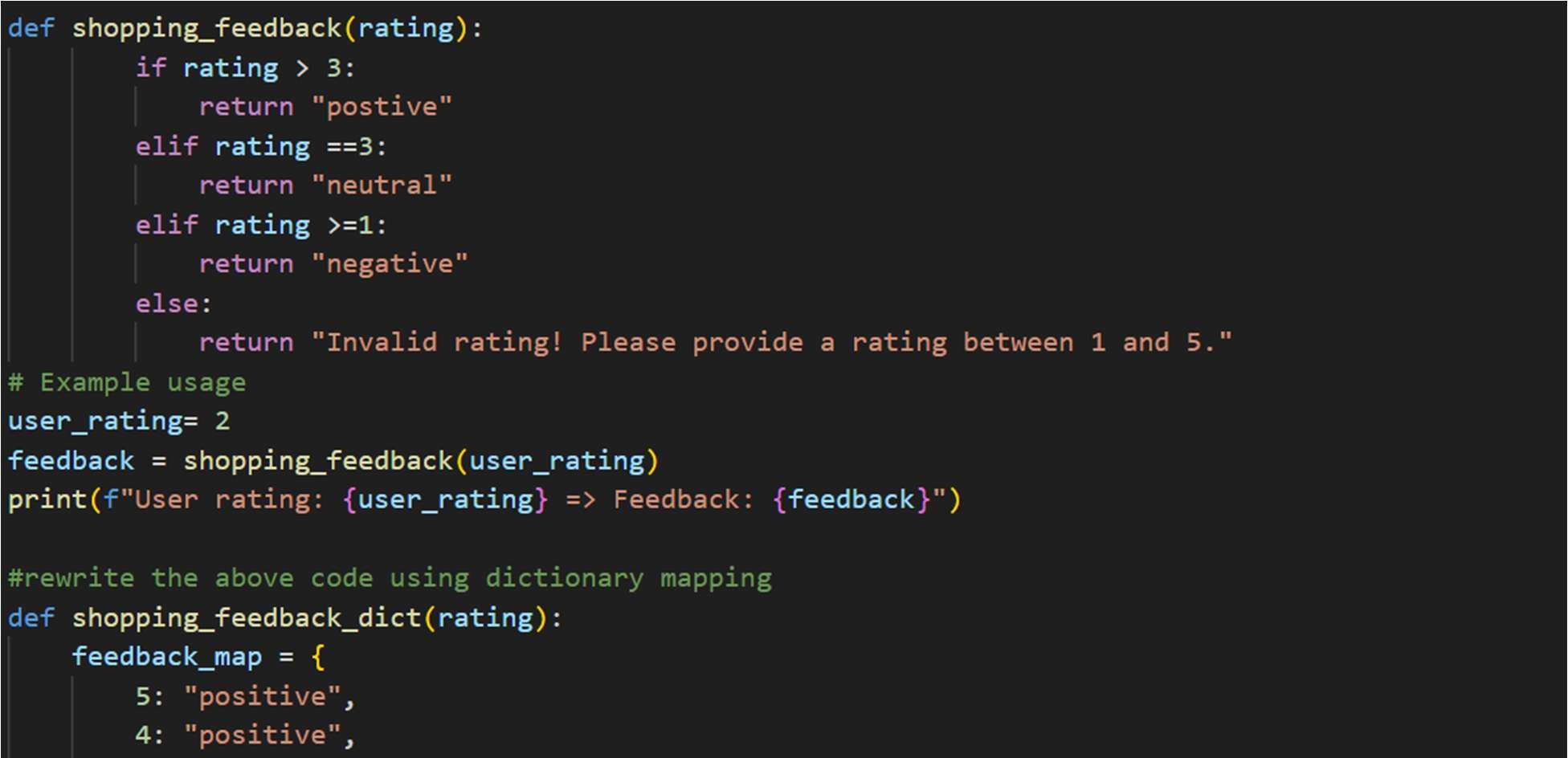
Task Description #2 (Conditional Statements – Online Shopping Feedback

Classification)

* Task: Ask AI to write nested if-elif-else conditions to classify online shopping feedback as Positive, Neutral, or Negative based on a numerical rating (1–5).
* Instructions:
* Generate initial code using nested if-elif-else.
* Analyze correctness and readability.
* Ask AI to rewrite using dictionary-based or match-case structure.

Expected Output #2:

* Feedback classification function with explanation and an alternative approach.

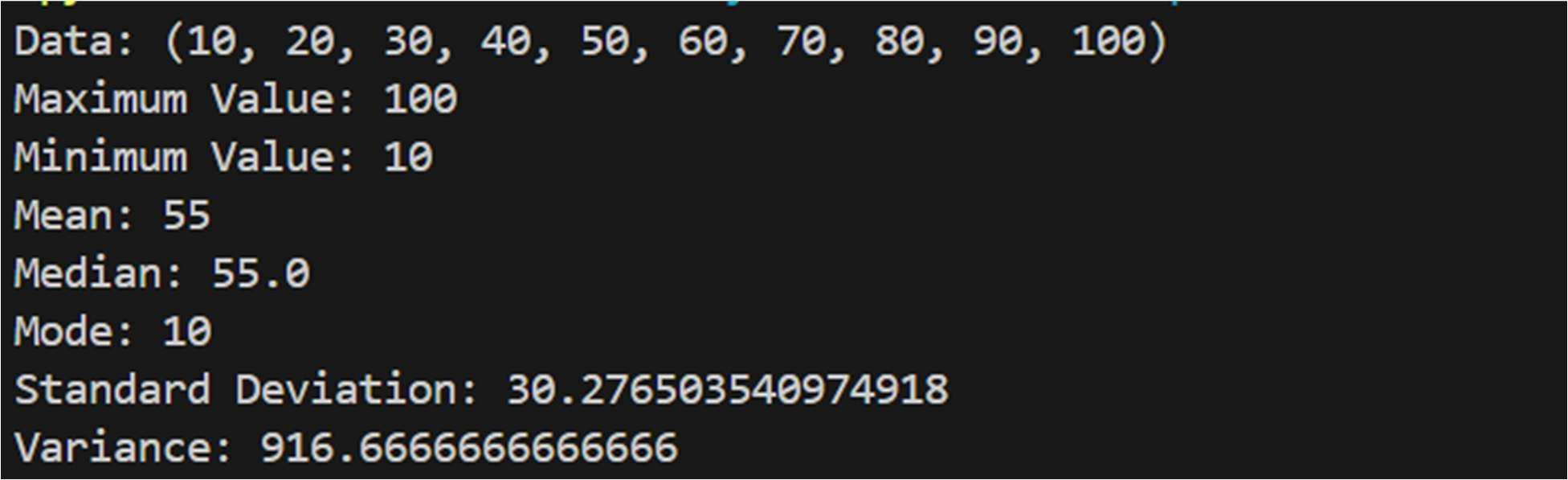
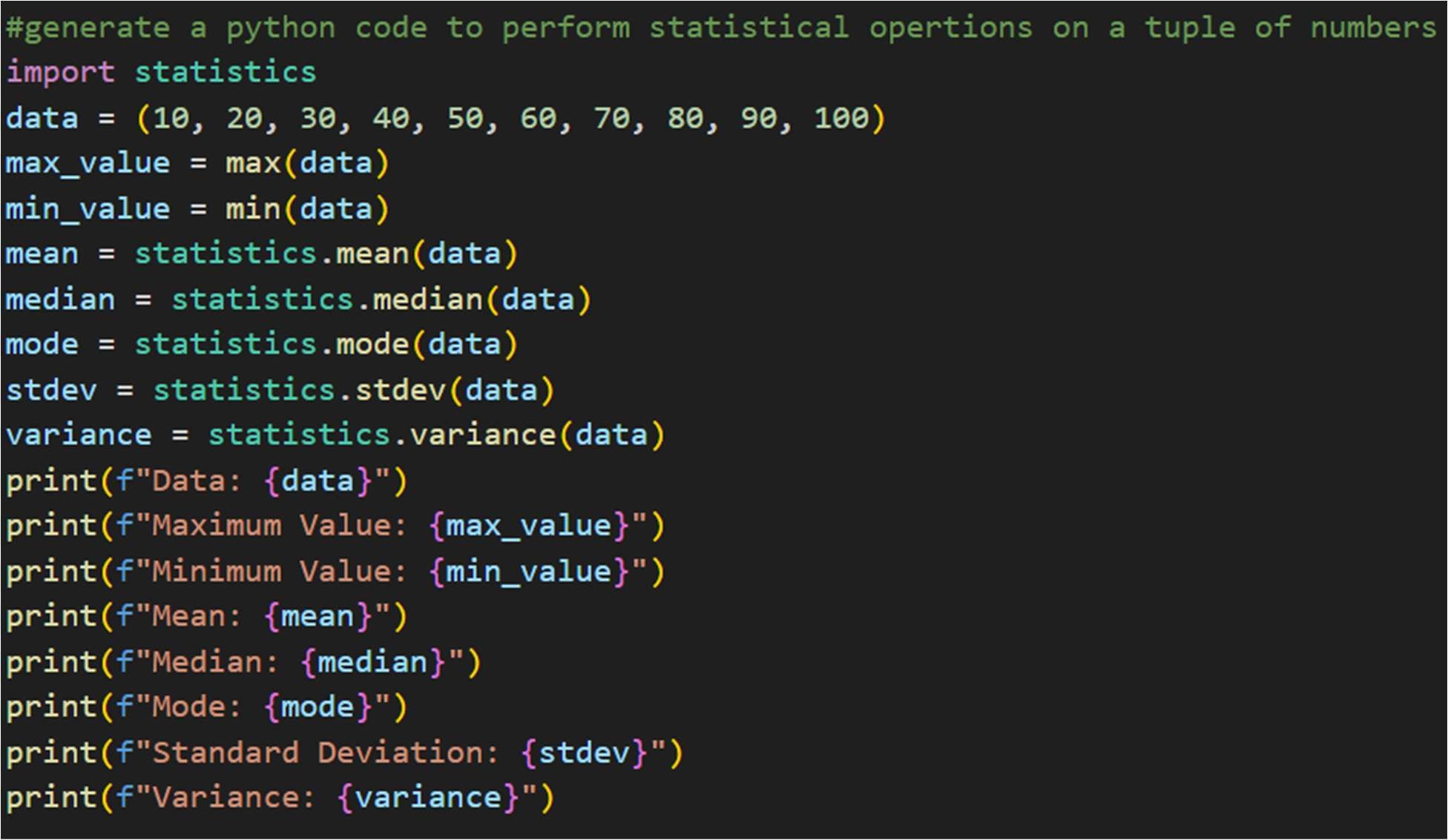


Task 3: Statistical\_operations

Define a function named statistical\_operations(tuple\_num) that performs the following statistical operations on a tuple of numbers:

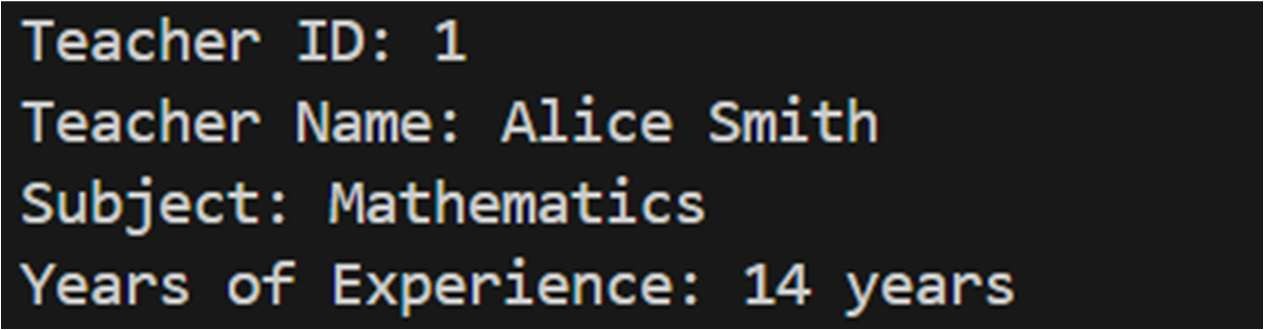
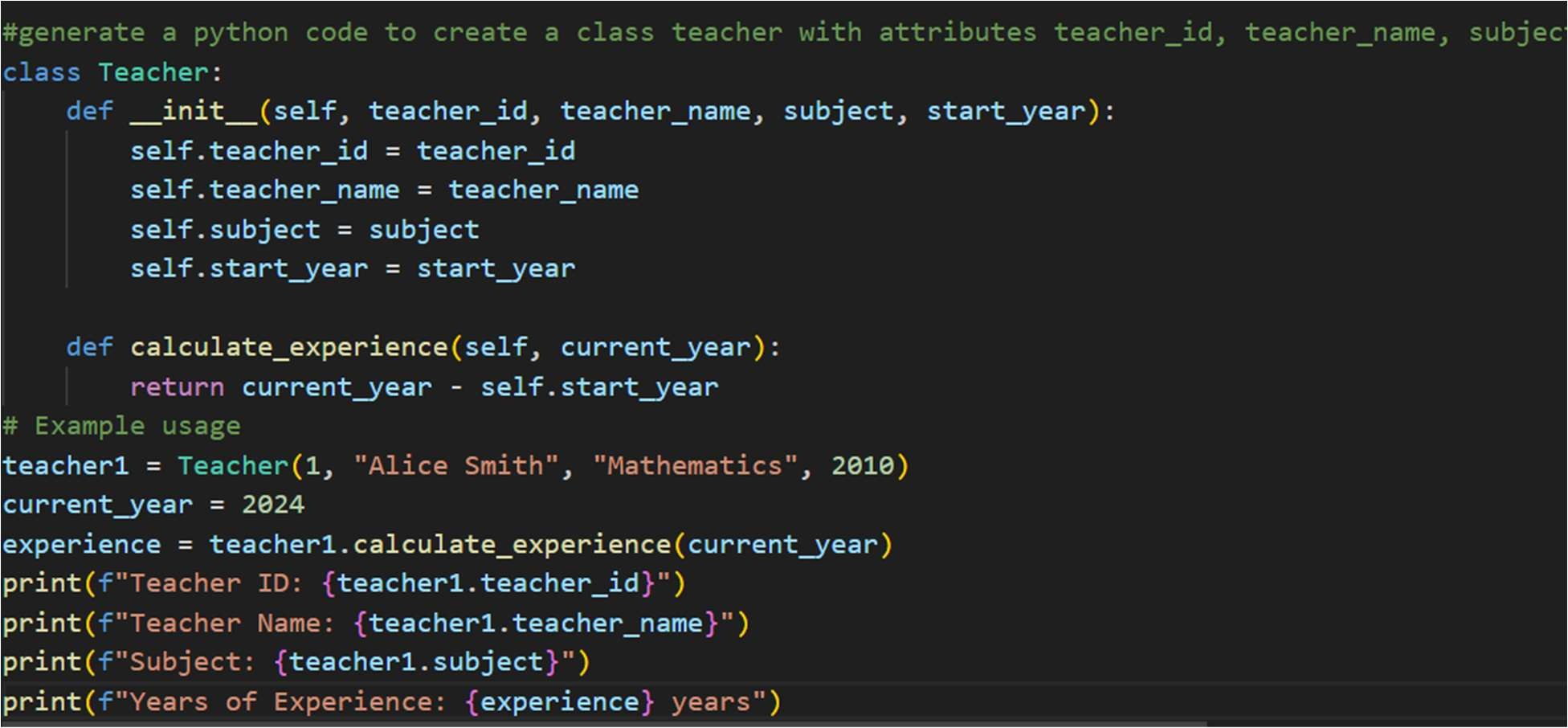
* Minimum, Maximum
* Mean, Median, Mode
* Variance, Standard Deviation

While writing the function, observe the code suggestions provided by GitHub Copilot.Make decisions to accept, reject, or modify the suggestions based on their relevance and correctness



Task 4: Teacher Profile

* Prompt: Create a class Teacher with attributes teacher\_id, name, subject, and experience. Add a method to display teacher details.
* Expected Output: Class with initializer, method, and object creation.



Task #5 – Zero-Shot Prompting with Conditional Validation

Use zero-shot prompting to instruct an AI tool to generate a function that validates an Indian mobile number.

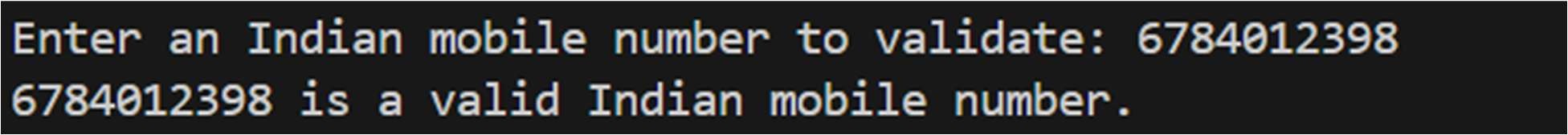
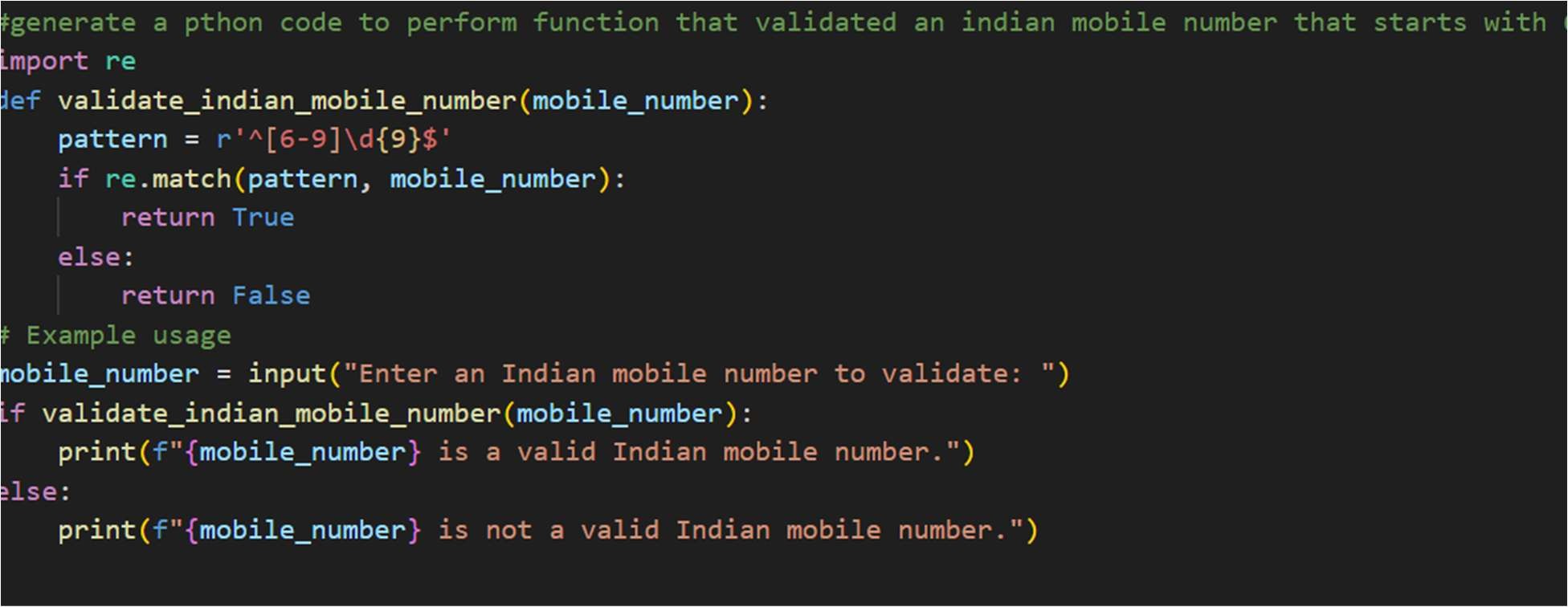
Requirements

* The function must ensure the mobile number:

o Starts with 6, 7, 8, or 9 o Contains exactly 10 digits

Expected Output

* A valid Python function that performs all required validations without using any input-output examples in the prompt.



Task Description #6 (Loops – Armstrong Numbers in a Range)

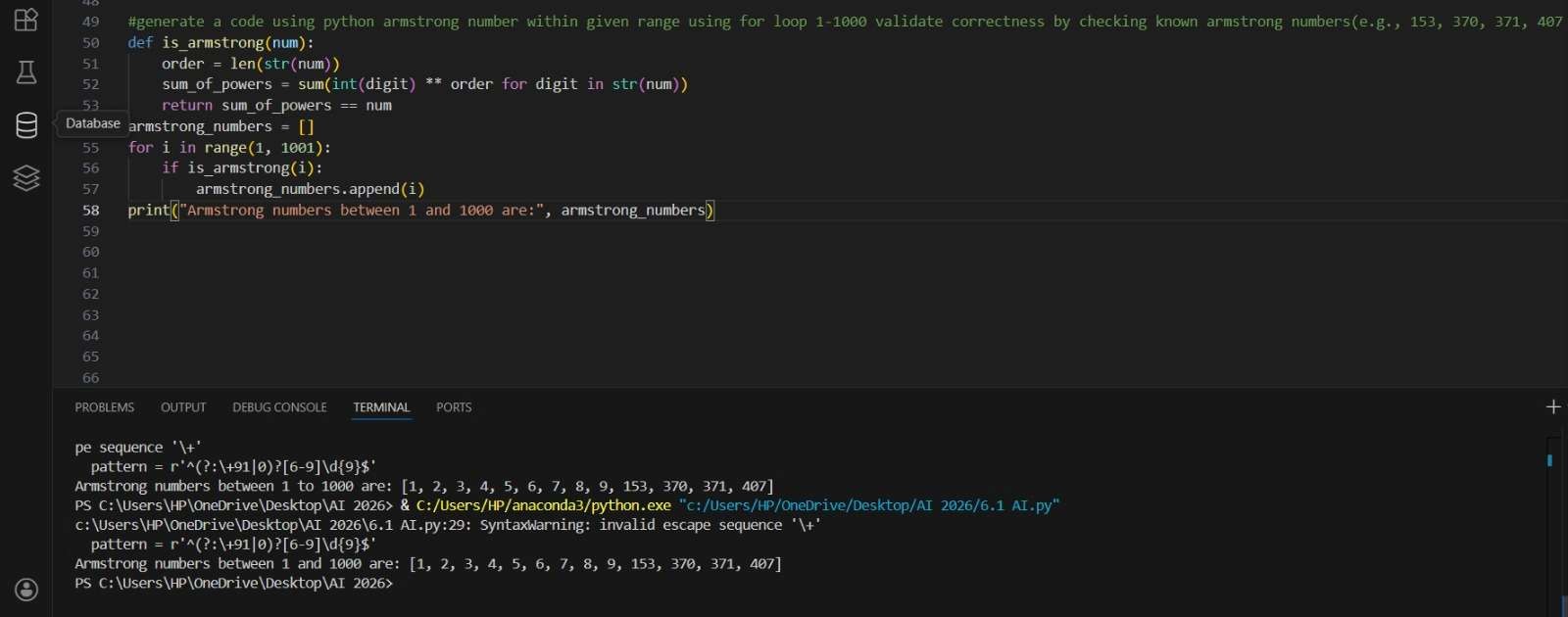
Task: Write a function using AI that finds all Armstrong numbers in a user- specified range (e.g., 1 to 1000).

Instructions:

* Use a for loop and digit power logic.
* Validate correctness by checking known Armstrong numbers (153, 370, etc.).
* Ask AI to regenerate an optimized version (using list comprehensions).

Expected Output #7:

* Python program listing Armstrong numbers in the range.
* Optimized version with explanation.



Task Description #7 (Loops – Happy Numbers in a Range)

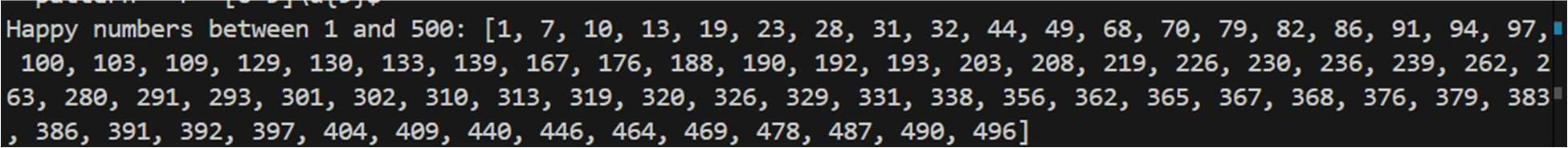
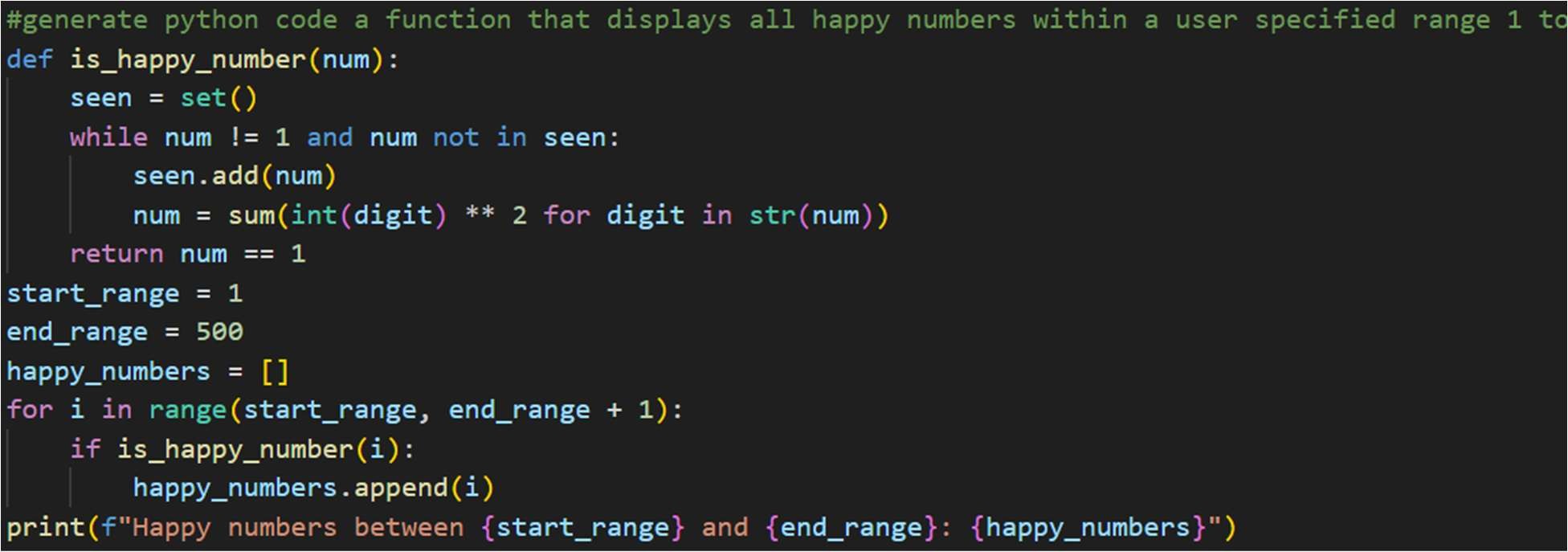
Task: Generate a function using AI that displays all Happy Numbers within a user-specified range (e.g., 1 to 500).

Instructions:

* Implement the logic using a loop: repeatedly replace a number with the sum of the squares of its digits until the result is either 1 (Happy Number) or enters a cycle (Not Happy).
* Validate correctness by checking known Happy Numbers (e.g., 1, 7, 10, 13, 19, 23, 28…).
* Ask AI to regenerate an optimized version (e.g., by using a set to detect cycles instead of infinite loops).

Expected Output #8:

* Python program that prints all Happy Numbers within a range.
* Optimized version using cycle detection with explanation.

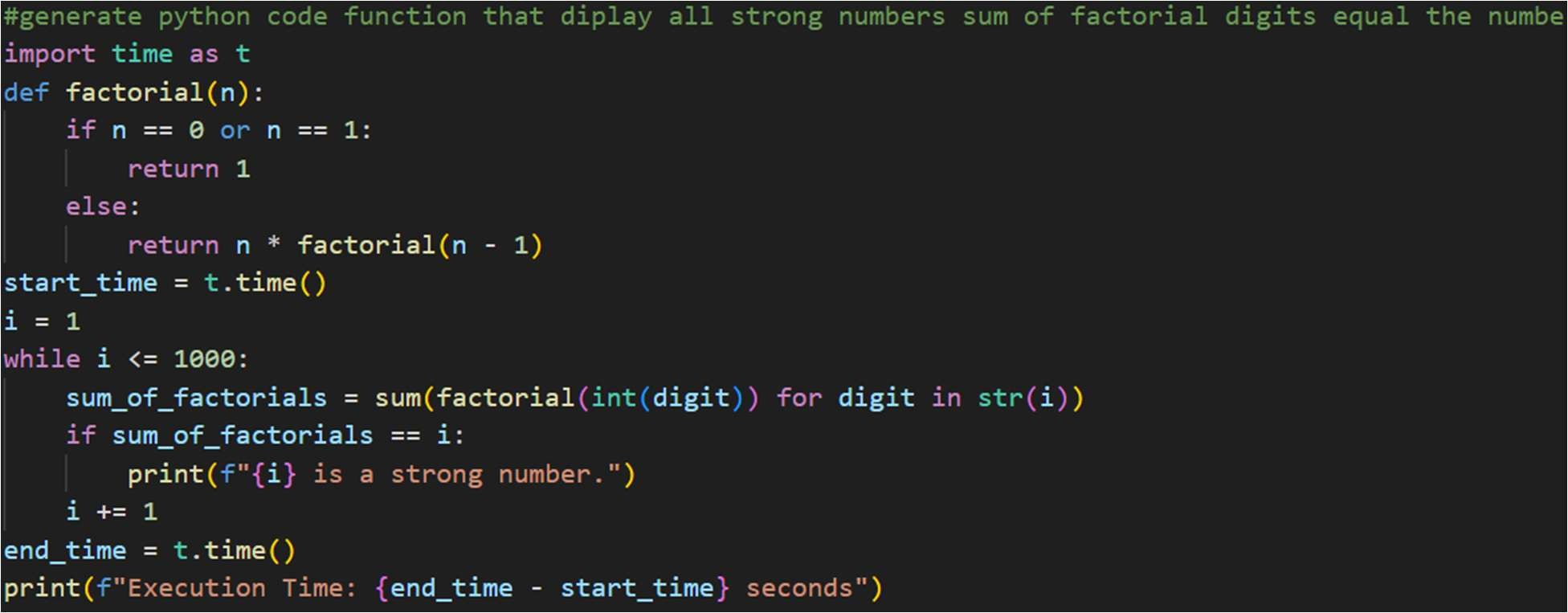


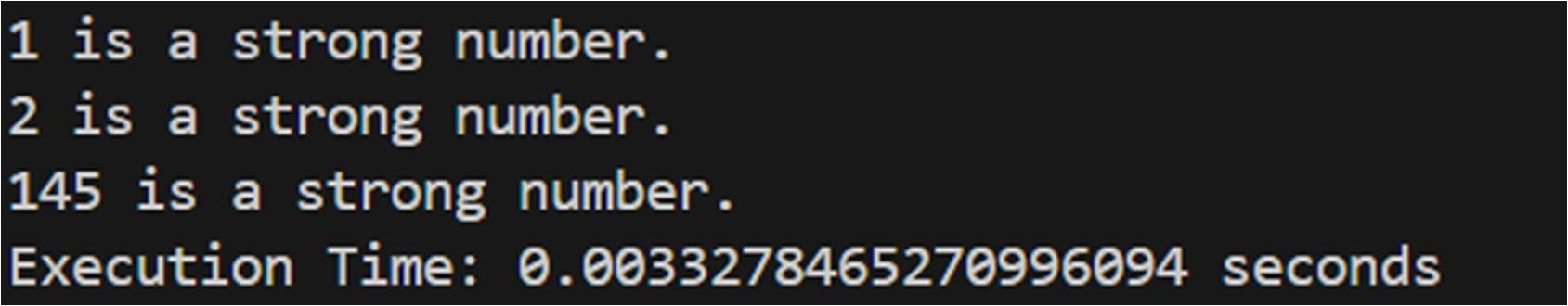
Task Description #8 (Loops – Strong Numbers in a Range)

Task: Generate a function using AI that displays all Strong Numbers (sum of factorial of digits equals the number, e.g., 145 = 1!+4!+5!) within a given range.

Instructions:

* Use loops to extract digits and calculate factorials.
* Validate with examples (1, 2, 145).
* Ask AI to regenerate an optimized version (precompute digit factorials).





Task #9 – Few-Shot Prompting for Nested Dictionary Extraction

Objective

Use few-shot prompting (2–3 examples) to instruct the AI to create a function that parses a nested dictionary representing student information.

Requirements

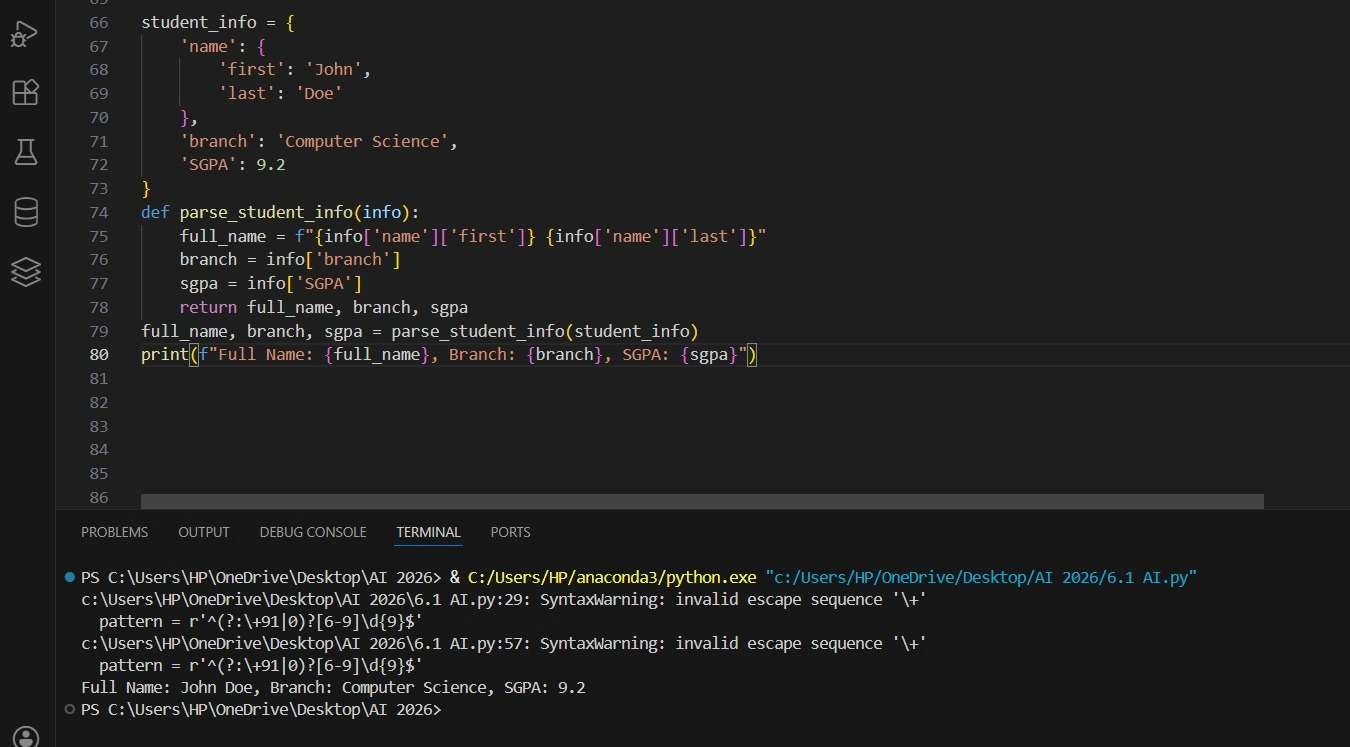
* The function should extract and return:

o Full Name o Branch o SGPA

Expected Output

A reusable Python function that correctly navigates and extracts values from nested dictionaries based on the provided examples

10



Task Description #10 (Loops – Perfect Numbers in a Range)

Task: Generate a function using AI that displays all Perfect Numbers within a user-specified range (e.g., 1 to 1000).

Instructions:

• A Perfect Number is a positive integer equal to the sum of its proper divisors (excluding itself).

o Example: 6 = 1 + 2 + 3, 28 = 1 + 2 + 4 + 7 + 14.

* Use a for loop to find divisors of each number in the range.
* Validate correctness with known Perfect Numbers (6, 28, 496…).
* Ask AI to regenerate an optimized version (using divisor check only up to √n).

Expected Output #12:

* Python program that lists Perfect Numbers in the given range.
* Optimized version with explanation.

