# AI ASSISTED CODING

ASSIGNMENT-6.1

NAME:S.VRINDHA REDDY

HALL NUMBER:2403A51255

BATCH:11

**Task-1: (Classes – Employee Management)**

Use AI to create an employee class with attributes (name, id, salary) and a method to calculate yearly salary.

* Instructions:
  + Prompt AI to generate the Employee class.
  + Analyze the generated code for correctness and structure.
  + Ask AI to add a method to give a bonus and recalculate salary.

**Expected Output -1:**

* A class with constructor, display\_details(), and calculate\_bonus () methods.

**Prompt-1**: Create an employee class with attributes name, id, salary, and a method to calculate yearly salary. Show example object creation and method call.

**Code:**

A screen shot of a computer

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Task-2: (Loops – Automorphic Numbers in a Range)**

Prompt AI to generate a function that displays all Automorphic numbers between 1 and 1000 using a for loop.

* Instructions:
  + Get AI-generated code to list Automorphic numbers using a for loop.
  + Analyze the correctness and efficiency of the generated logic.
  + Ask AI to regenerate using a while loop and compare both implementations.

**Expected Output -2:**

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

**Prompt-2:** Write a program with a function that finds and displays all Automorphic numbers between 1 and 1000 using a for loop. Show the output.

Explanation:

**Using a for loop:**

The find\_automorphic\_for function iterates through each number in the specified range (1 to 1000 inclusive) using a for loop. For each number, it calculates its square and then checks if the string representation of the square ends with the string representation of the original number. If it does, the number is Automorphic and is printed. This approach is concise and directly iterates over the desired range.

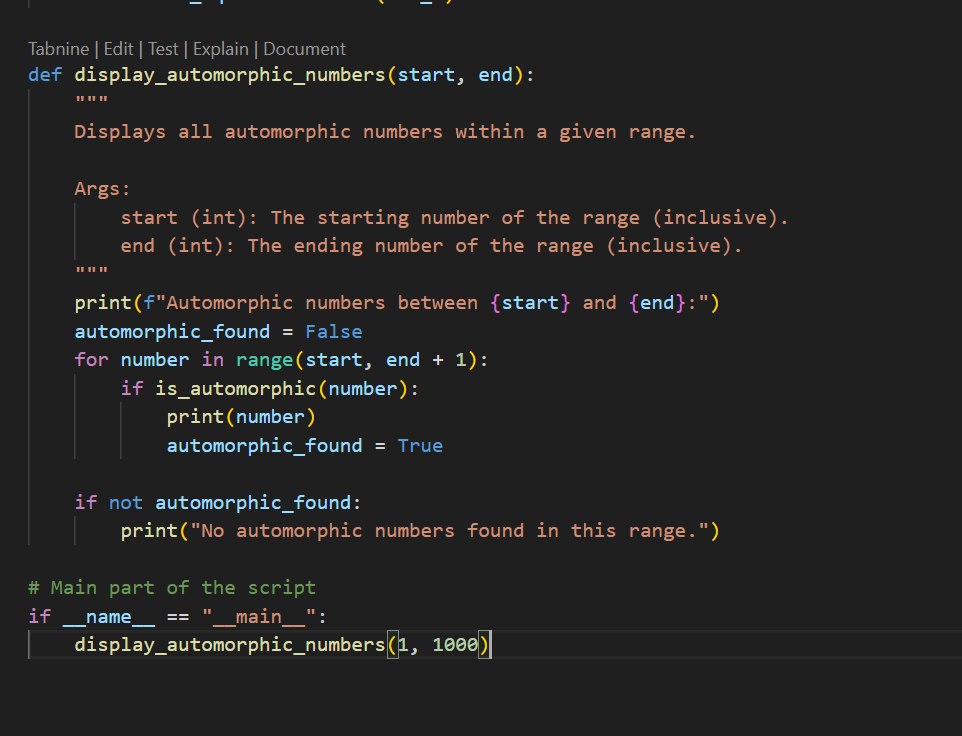
**Using a while loop:**

The find\_automorphic\_while function achieves the same result using a while loop. It initializes a variable num to the starting number of the range. The while loop continues as long as num is less than or equal to the ending number. Inside the loop, it performs the same calculation and check for Automorphic numbers as the for-loop version. After each iteration, num is incremented to move to the next number in the range. This approach provides more explicit control over the loop's continuation condition and the iteration variable.

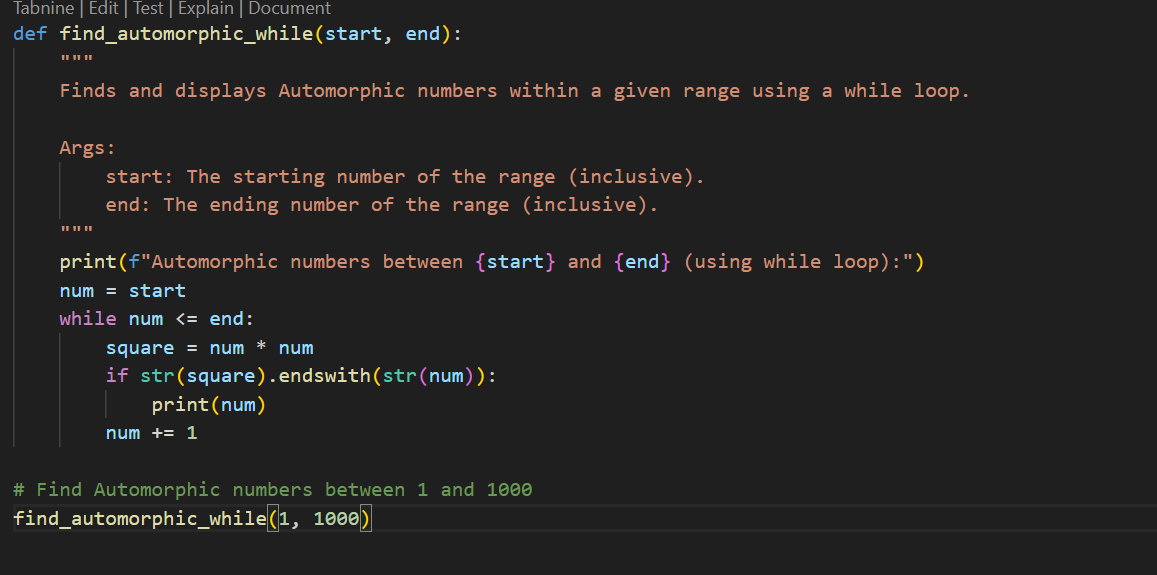
Both implementations correctly identify and print the Automorphic numbers within the given range. The choice between a for loop and a while loop often depends on the specific requirements and personal preference; a for loop is typically preferred when iterating over a known sequence or range, while a while loop is more suitable when the loop's termination depends on a condition that may not be directly tied to a sequence.

**Code:**

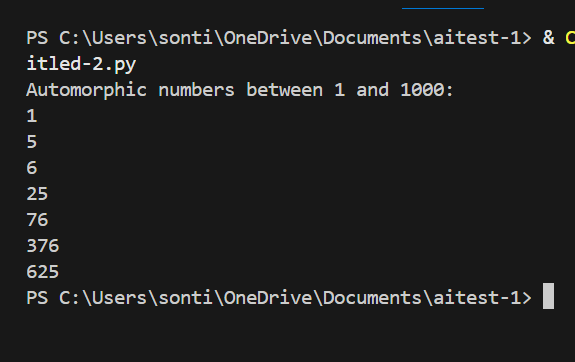
**Using for loop:**



**Using while loop:**



**Output:**



**Task-3:(Conditional Statements-Online Shopping Feedback Classification)**

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).

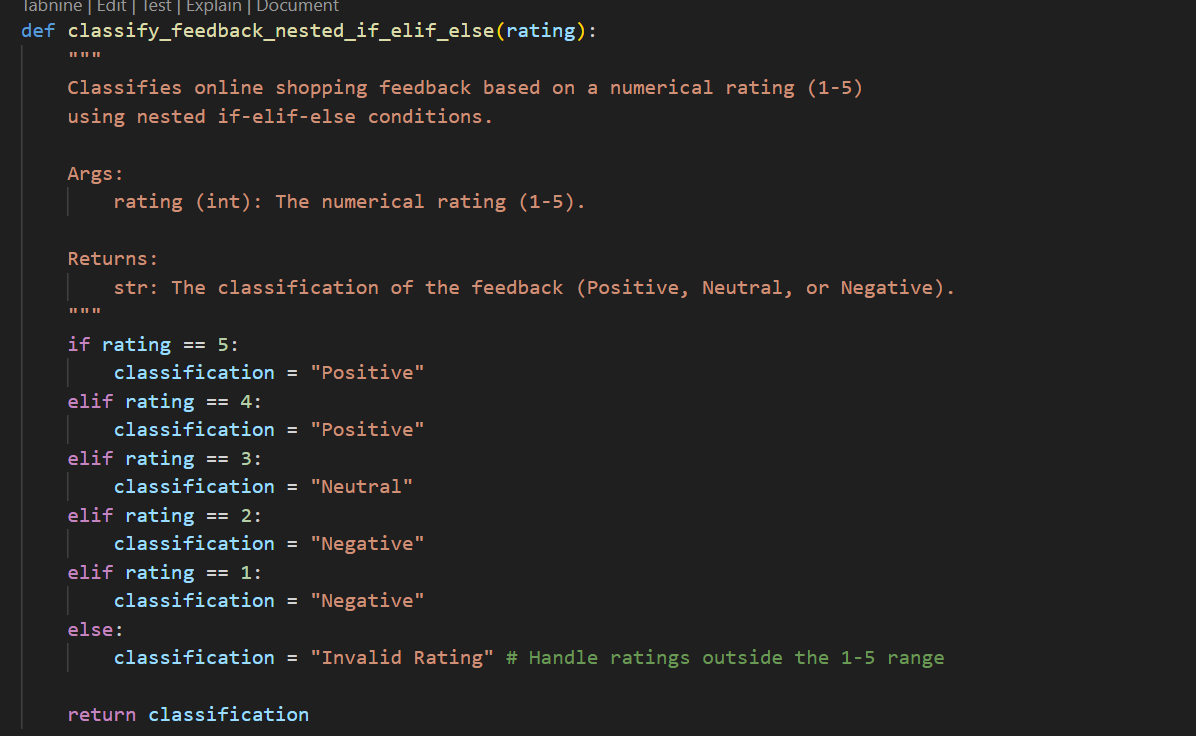
* Task: Ask AI to write nested if-elif-else conditions to classify online shopping feedback as Positive, Neutral, or Negative based on a numerical rating (1–5).
* Instructions:
  + Generate initial code using nested if-elif-else.
  + Analyze correctness and readability.
  + Ask AI to rewrite using dictionary-based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

**Expected Output -3:**

* Feedback classification function with explanation and an alternative approach.

**Prompt-3:** Write a program using nested if-elif-else to classify online shopping feedback as Positive, Neutral, or Negative based on a rating from 1 to 5.

**Code:**



A screen shot of a computer program

AI-generated content may be incorrect.

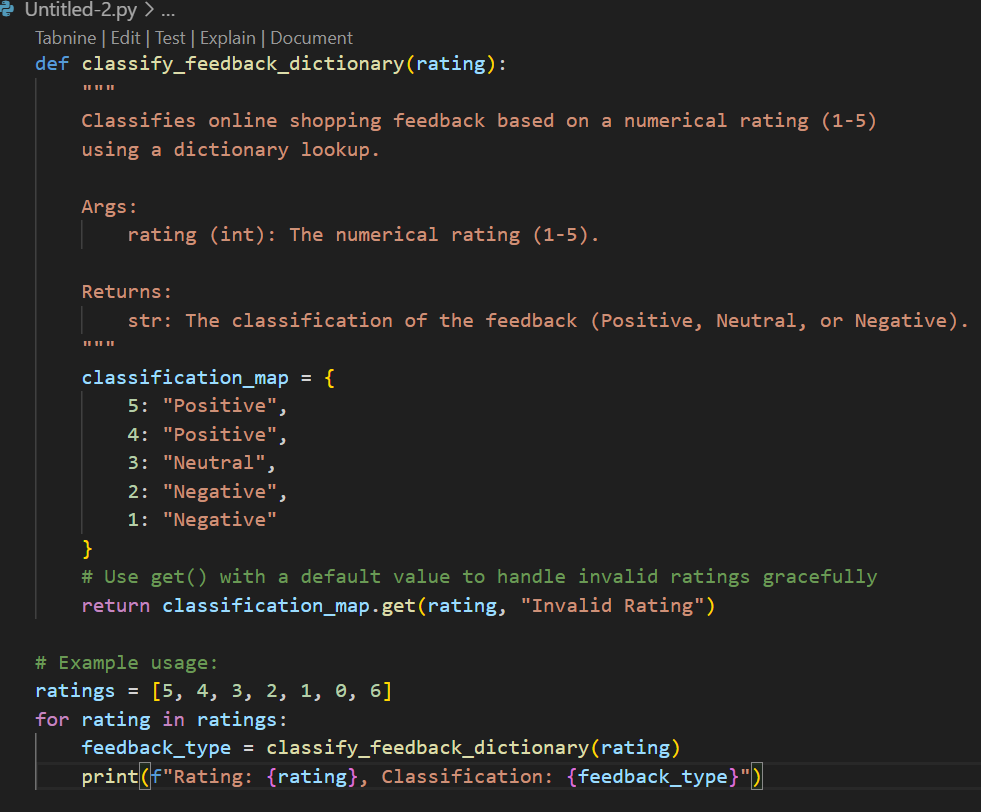
**Explanation:** The classify feedback \_nested\_if\_elif\_else function takes a numerical rating (from 1 to 5) as input. It uses a series of if and elif conditions to check the value of the rating and assign a corresponding classification:

* If the rating is 5 or 4, it's classified as "Positive".
* If the rating is 3, it's classified as "Neutral"
* If the rating is 2 or 1, it's classified as "Negative".

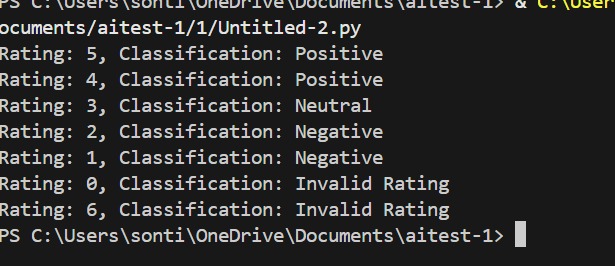
. If the rating is anything else, it's classified as "Invalid Rating".

This approach is straightforward and easy to understand for a small number of conditions. However, for a larger number of possible ratings or more complex classification logic, it could become lengthy and less readable

**Alternative approach.:**



**Output:**



**Task-4: (Loops – Prime Numbers in a Range)**

Generate a function using AI that displays all prime numbers within a user-specified range (e.g., 1 to 500).

* Instructions:
  + Get AI-generated code to list all primes using a for loop.
  + Analyze the correctness and efficiency of the prime-checking logic.
  + Ask AI to regenerate an optimized version (e.g., using the square root method).

**Expected Output -4:**

* Python program that lists all prime numbers within a given range, with an optimized version and explanation.

**Prompt-4**: Generate a function using AI that displays all prime numbers within a user-specified range (e.g., 1 to 500).

**Explanation:** This function uses a more efficient method to check for primality. It incorporates a few optimizations:

1. **Handles small numbers:** It quickly returns False for numbers less than or equal to 1 and True for 2 and 3.
2. **Checks divisibility by 2 and 3**: It checks if the number is divisible by 2 or 3 first, as these are common factors.
3. **Checks divisibility up to the square root**: The key optimization is that it only checks for divisors up to the square root of num. If a number num has a divisor i greater than its square root, then num/i must be a divisor smaller than its square root. Therefore, we only need to check for divisors up to the square root.
4. **Steps by 6:** After checking 2 and 3, all prime numbers greater than 3 can be expressed in the form of 6k ± 1. The loop checks divisibility by numbers of the form 6k - 1 and 6k + 1, skipping many composite numbers and further improving efficiency.
5. **Correctness:** This method is also correct due to the mathematical property that if a number has a divisor, it must have one less than or equal to its square root.
6. **Efficiency**: This method is significantly more efficient than the basic approach, especially for larger numbers, because it drastically reduces the number of divisions required.

**Code:**

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Task-5: (Classes – Library System)**

* Task: Use AI to build a Library class with methods to add\_book(), issue\_book(), and display\_books().
* Instructions:
  + Generate Library class code using AI.
  + Analyse if methods handle edge cases (e.g., issuing unavailable books).
  + Ask AI to add comments and documentation.

**Expected Output -5:**

* Library class with all methods, inline comments, and explanation.

**Prompt**: Write a Library class with methods add\_book (), issue\_book (), and display\_books (). Show how to create a Library object, add books, issue a book, and display the remaining books.

**Explanation:**

* Initializes an empty list book to store all available books.

**add\_book(book):**

* Appends a new book name into the books list.
* Prints confirmation message.

**issue\_book(book):**

* Checks if the requested book exists in the library.
* If yes → removes it.
* If not → prints warning (edge case handling).

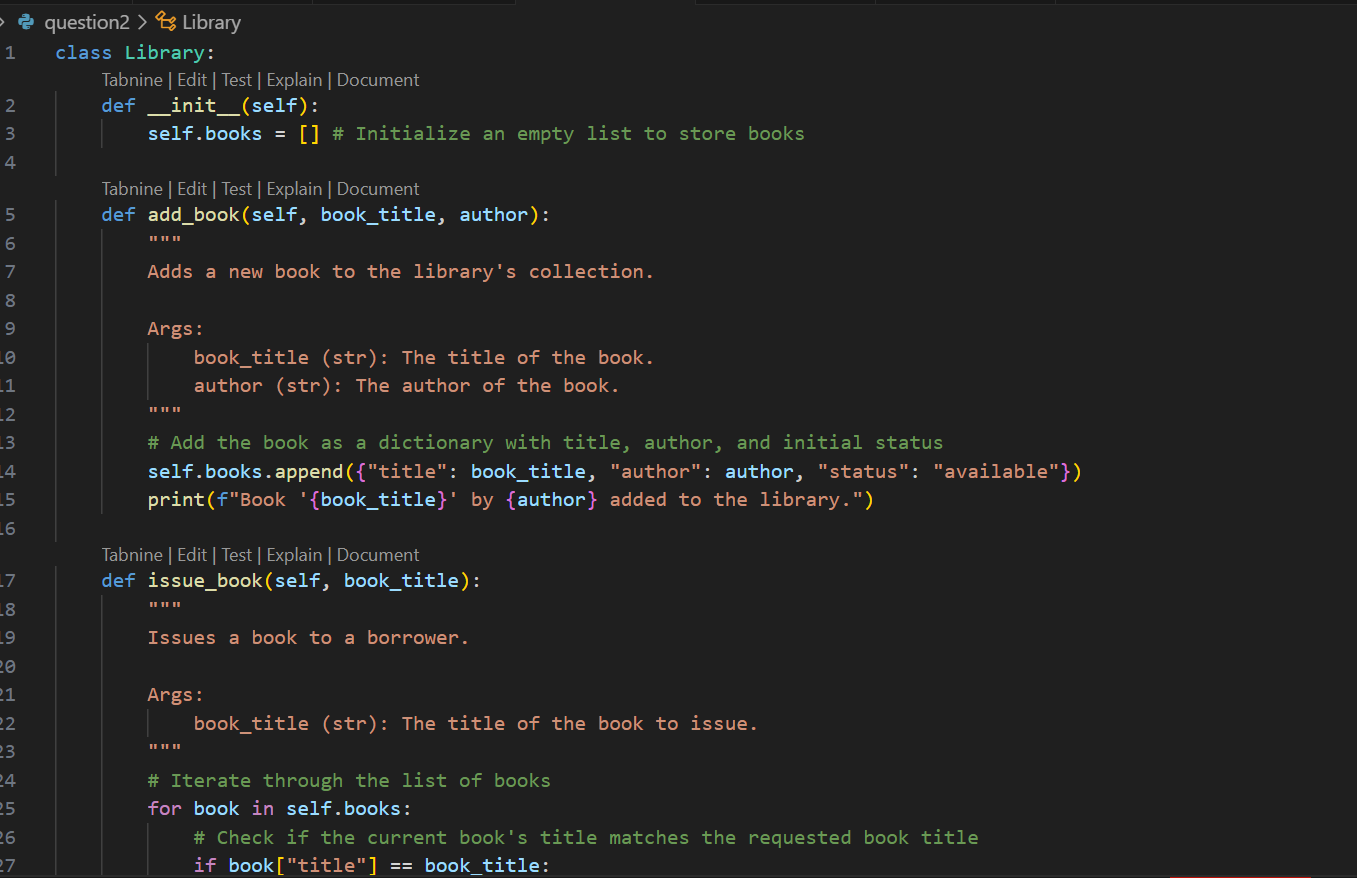
**display\_books ():**

* Loops through books and prints them with numbering.
* If no books exist → prints warning.

**Example Usage:**

* Demonstrates adding books, issuing available/unavailable books, and displaying the collection before & after.

**Code:**



A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Output:**

A computer screen with white text

AI-generated content may be incorrect.