**AI ASSISTED CODING**

**ASSIGMENT-6.1**

B.Ranhitha

Batch—11

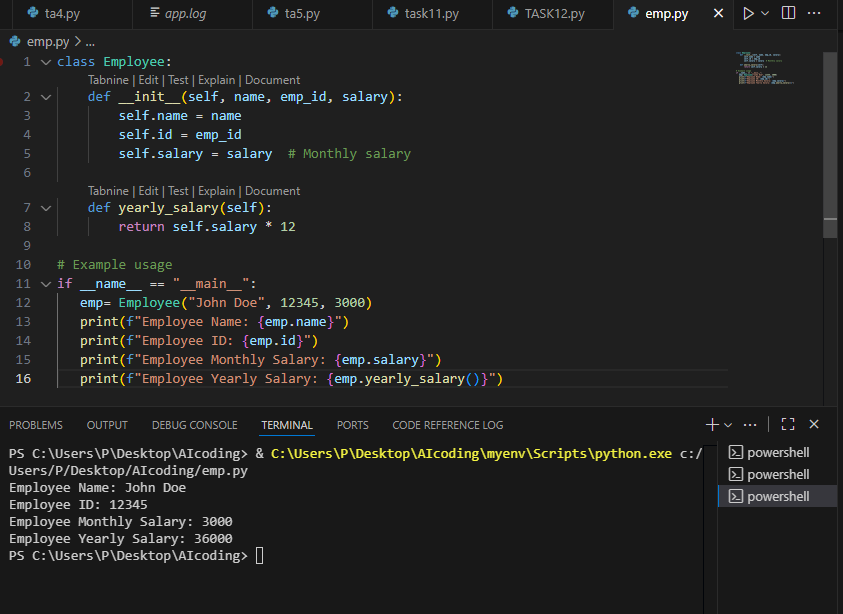
2403A51247

**TASK—1:**

**Prompt:**

"Write a Python class Employee with attributes (name, id, salary) and a method to calculate yearly salary."

**CODE AND OUTPUT:**

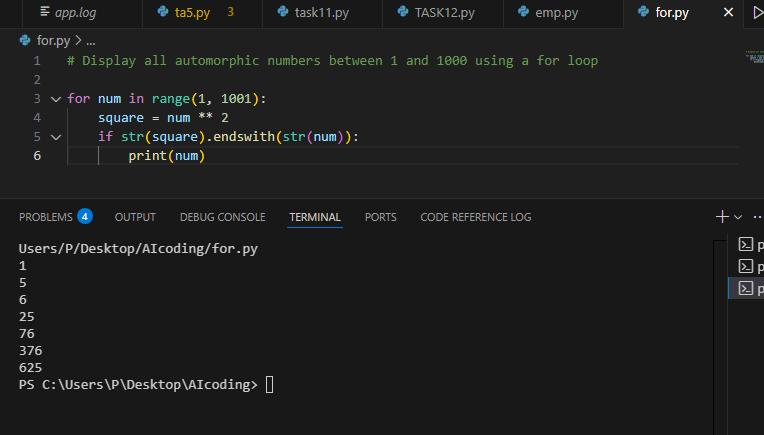


**TASK—2:**

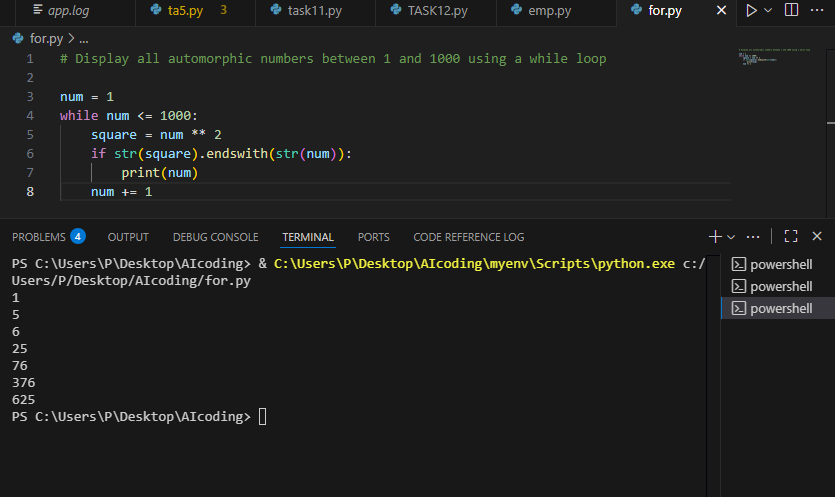
**Prompt**: Generate Python code to display all automorphic numbers between 1 and 1000 using a for loop.

**For loop:**

**Output and code:**



**While loop:**

****

**Difference between for loop and while loop in Python:**

**for loop**

* Used when you know in advance how many times you want to iterate.
* Iterates over a sequence (like a list, range, or string).

Example:

**for i in range(5):**

**print(i)**

**while loop**

* Used when you want to repeat an action until a condition changes (not necessarily a fixed number of times).
* Continues as long as the condition is True.

Example:

**i = 0**

**while i < 5:**

**print(i)**

**i += 1**

**Summary:**

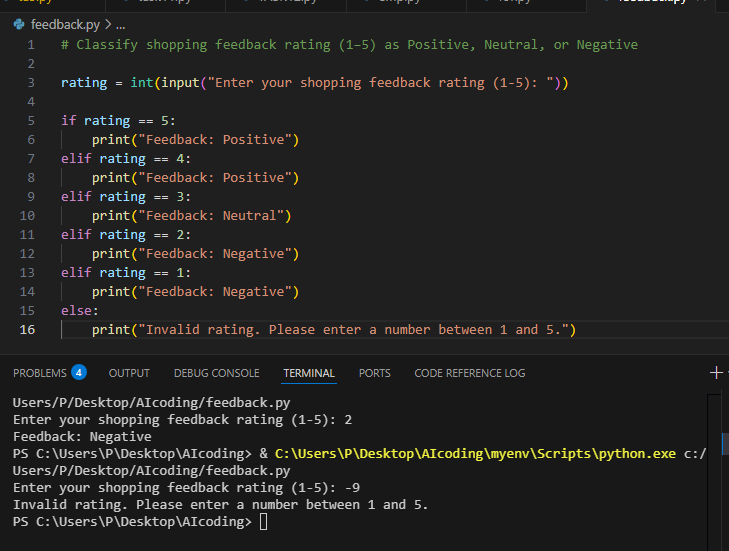
* Use for when the number of iterations is known or you are iterating over a collection.
* Use while when you want to loop until a condition is no longer true

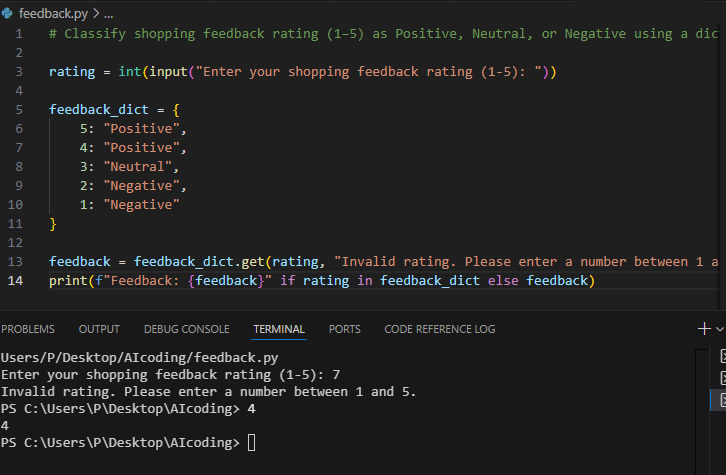
**TASK—3:**

**Prompt:**

Write Python nested if-elif-else to classify shopping feedback rating (1–5) as Positive, Neutral, or Negative

**Using nested if else:**



* rewrite using dictionary-based
* 

Using a dictionary-based approach (as in your code) is generally more readable and efficient than nested if-elif-else statements for this use case:

**Readability:**The dictionary clearly maps each rating to its feedback, making it easy to see and update the classification logic. There’s less code and no repeated print statements.

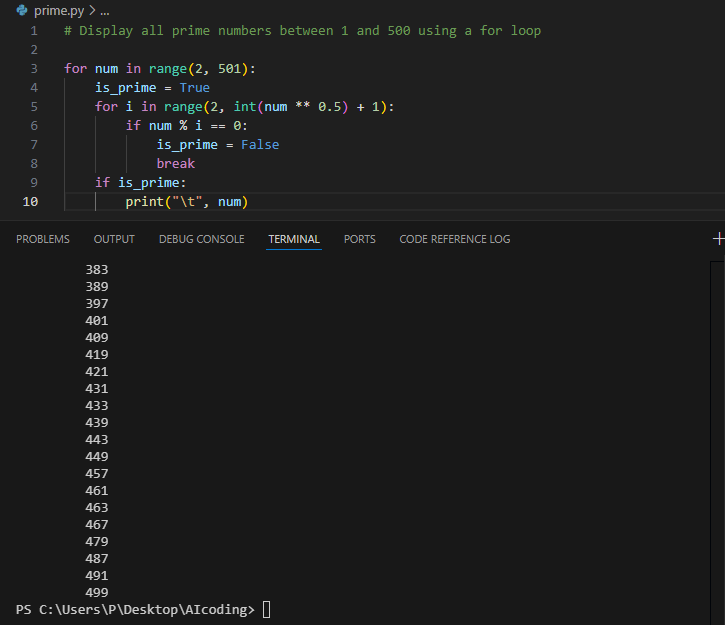
**Efficiency:**Dictionary lookups are fast (constant time, O(1)), while nested if-elif-else statements require checking each condition in sequence (worst case O(n)).

**Summary:**For simple mappings like this, the dictionary approach is preferred for both readability and efficiency.

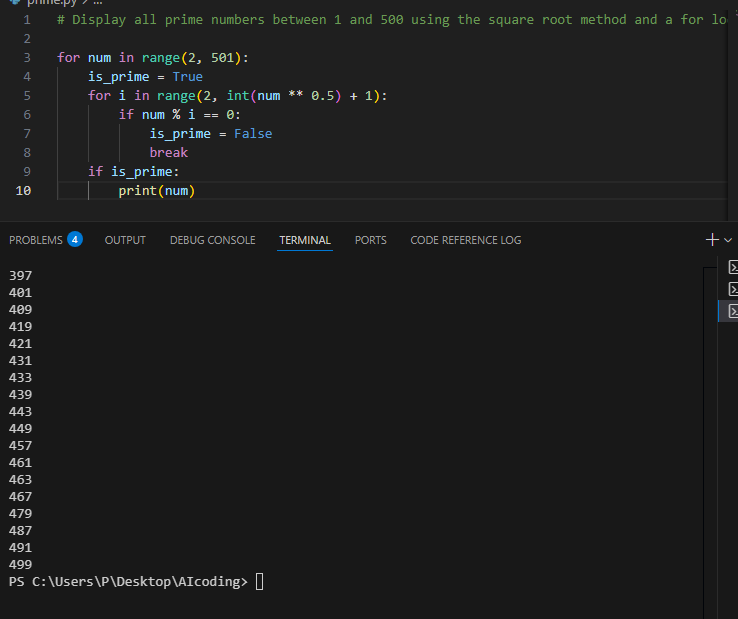
**TASK—4:**

**Prompt:** Write Python code to display all prime numbers between 1 and 500 using a for loop.

**Code and output:**

****

**Prompt 2:** Optimized Version (Square Root Method)

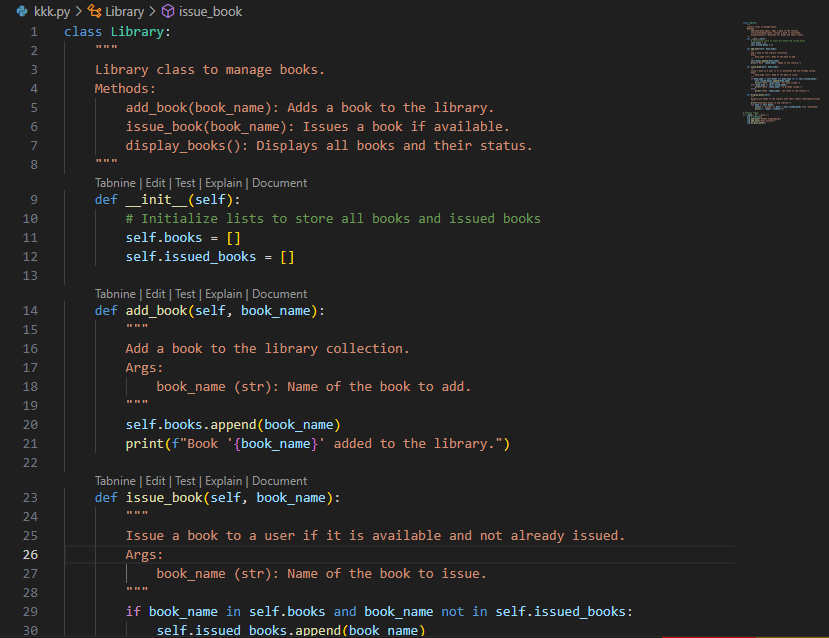
****

**Comparison**

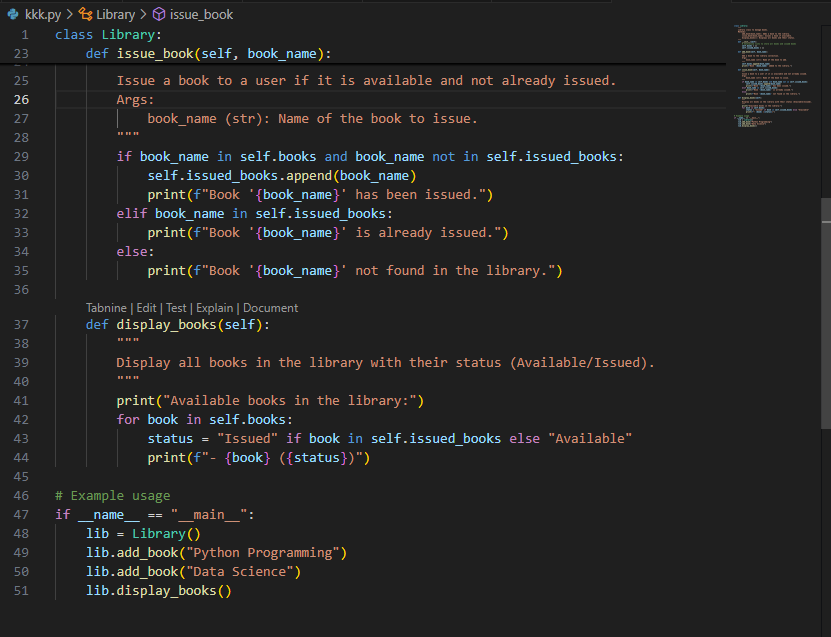
* First version is correct but inefficient (O(n²)).
* Optimized version reduces iterations to √n, making it much faster**.**

**TASK—5:**

**Prompt**: Write Python class Library with methods add\_book(), issue\_book(), and display\_books().

****

* **add comments/documentation**

****

**Final Conclusion**

This lab showed how AI tools can help us write Python code faster for classes, loops, and conditionals. AI generated working solutions for tasks like employee management, automorphic numbers, feedback classification, prime numbers, and a library system. At the same time, we saw that AI code still needs checking, especially for efficiency and edge cases.

Overall, the lab proved that AI is a useful assistant, but human understanding is necessary to improve, explain, and validate the code.