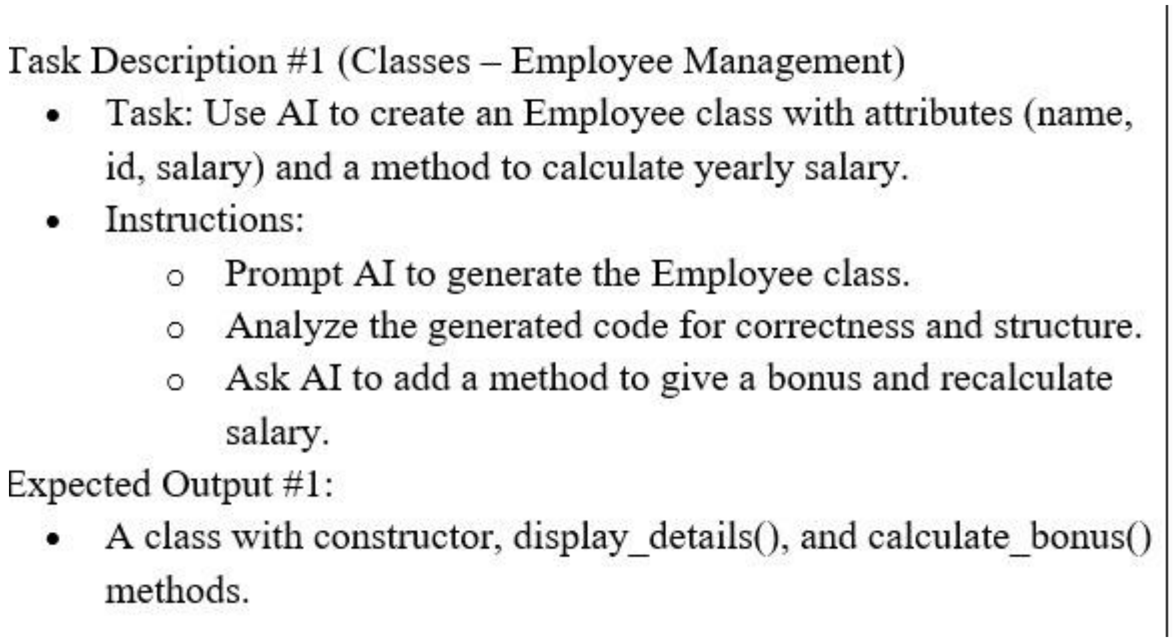
AI ASSIGNEMNT-6.1

2403A51269

A.DHANALAXMI

TASK-1



Task Description #1 (Classes – Employee Management)

- Task: 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:

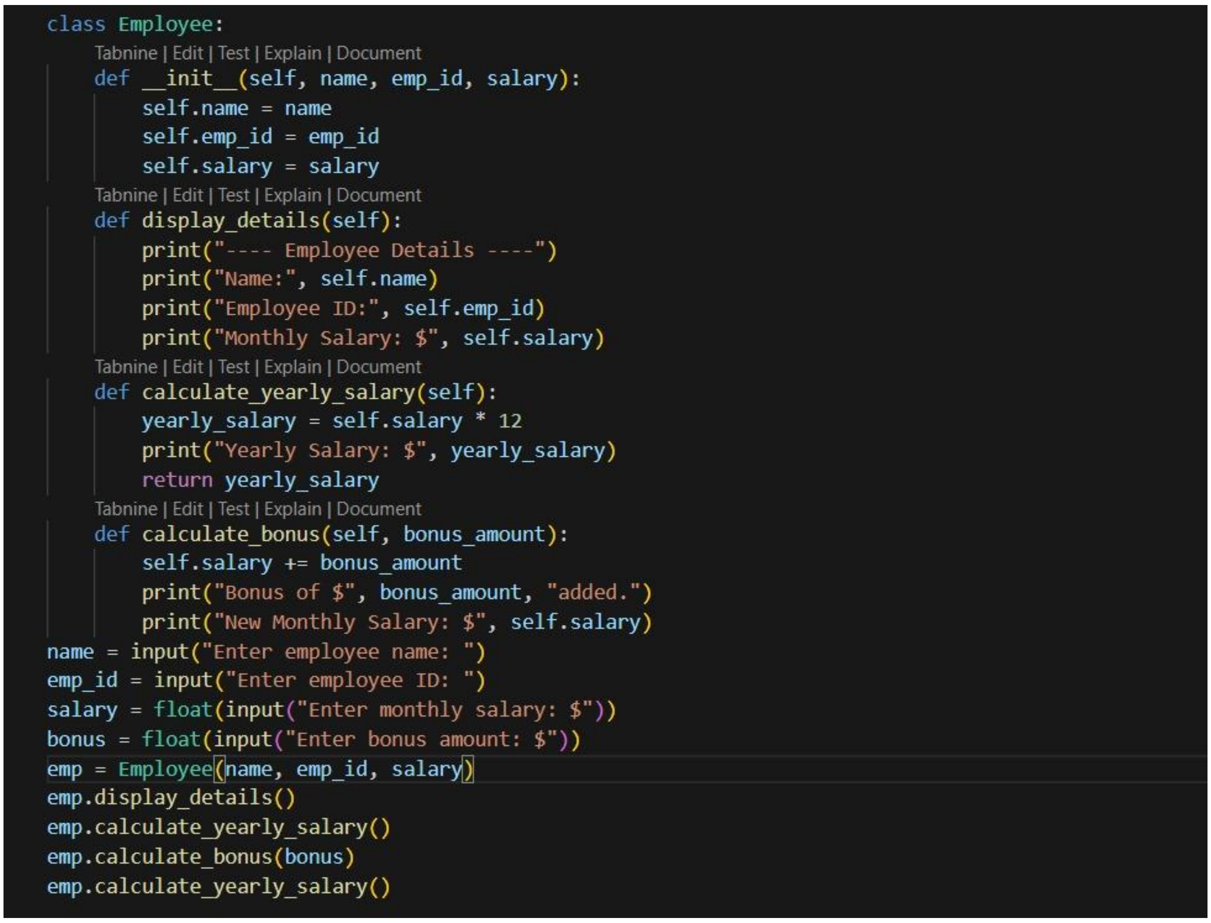
- A class with constructor, display_details(), and calculate_bonus() methods.

PROMPT:

Create an Employee class with attributes name, id, and salary. Add methods to display details, calculate yearly salary, and apply a bonus to update the salary.

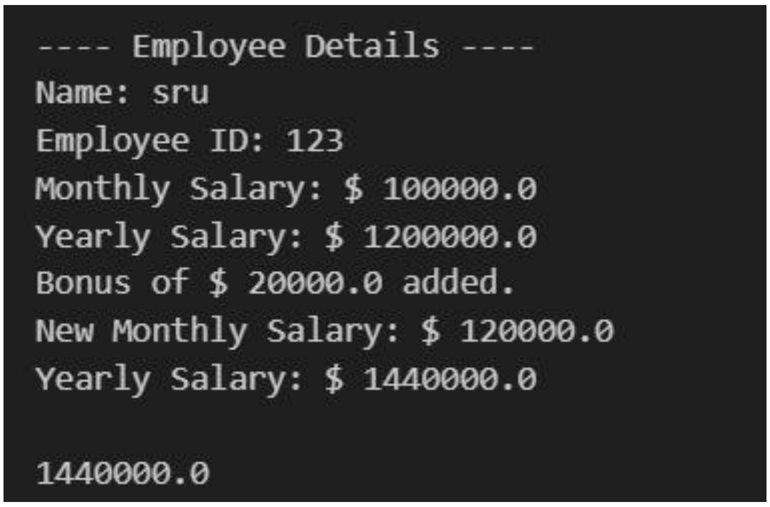
CODE:

TASK-2



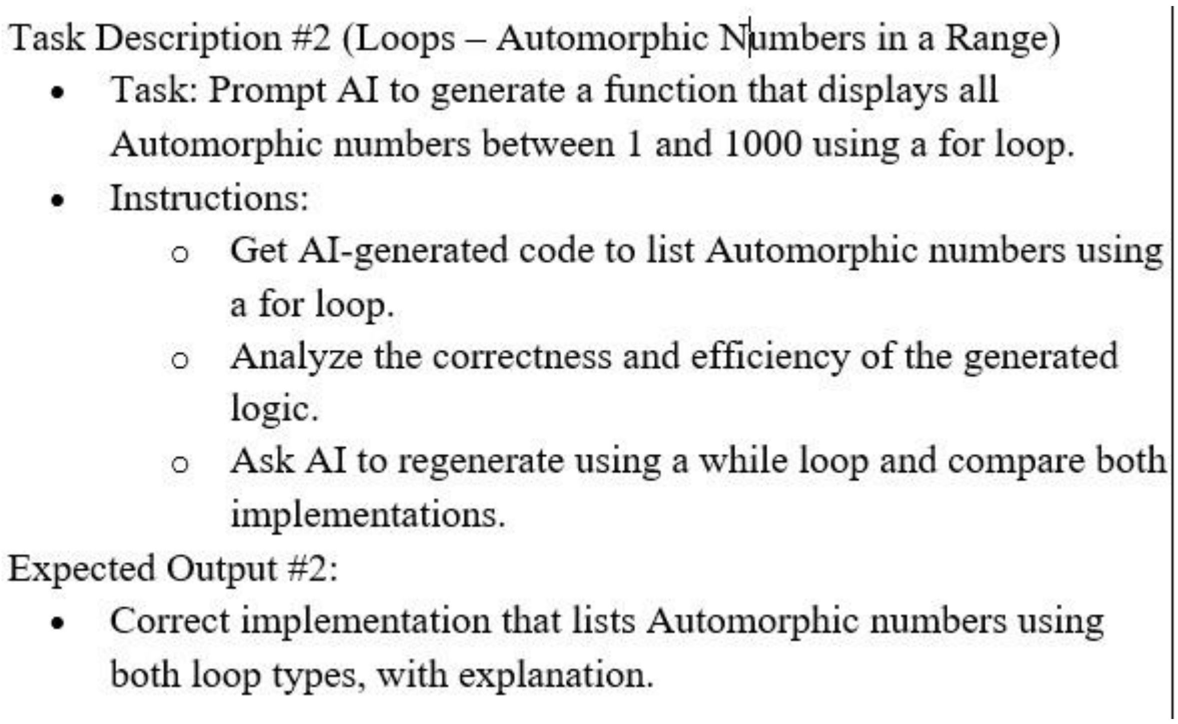
```
class Employee:
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, name, emp_id, salary):
        self.name = name
        self.emp_id = emp_id
        self.salary = salary
    Tabnine | Edit | Test | Explain | Document
    def display_details(self):
        print("---- Employee Details ----")
        print("Name:", self.name)
        print("Employee ID:", self.emp_id)
        print("Monthly Salary: $", self.salary)
    Tabnine | Edit | Test | Explain | Document
    def calculate_yearly_salary(self):
        yearly_salary = self.salary * 12
        print("Yearly Salary: $", yearly_salary)
        return yearly_salary
    Tabnine | Edit | Test | Explain | Document
    def calculate_bonus(self, bonus_amount):
        self.salary += bonus_amount
        print("Bonus of $", bonus_amount, "added.")
        print("New Monthly Salary: $", self.salary)
name = input("Enter employee name: ")
emp_id = input("Enter employee ID: ")
salary = float(input("Enter monthly salary: $"))
bonus = float(input("Enter bonus amount: $"))
emp = Employee(name, emp_id, salary)
emp.display_details()
emp.calculate_yearly_salary()
emp.calculate_bonus(bonus)
emp.calculate_yearly_salary()
```

OUTPUT:



```
---- Employee Details ----
Name: sru
Employee ID: 123
Monthly Salary: $ 100000.0
Yearly Salary: $ 1200000.0
Bonus of $ 20000.0 added.
New Monthly Salary: $ 120000.0
Yearly Salary: $ 1440000.0

1440000.0
```

TASK-2



Task Description #2 (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:
  - 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.

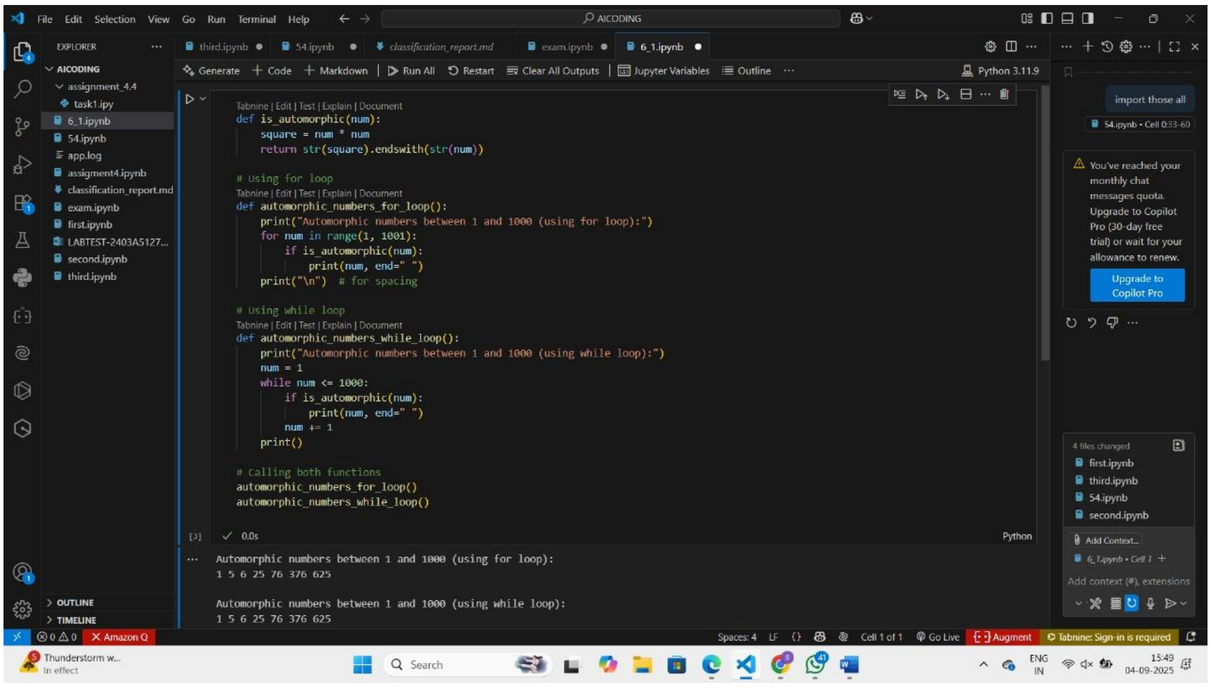
Expected Output #2:

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

PROMPT:

Write a function to print all Automorphic numbers between 1 and 1000 using a for loop, and then rewrite it using a while loop.

CODE & OUTPUT:



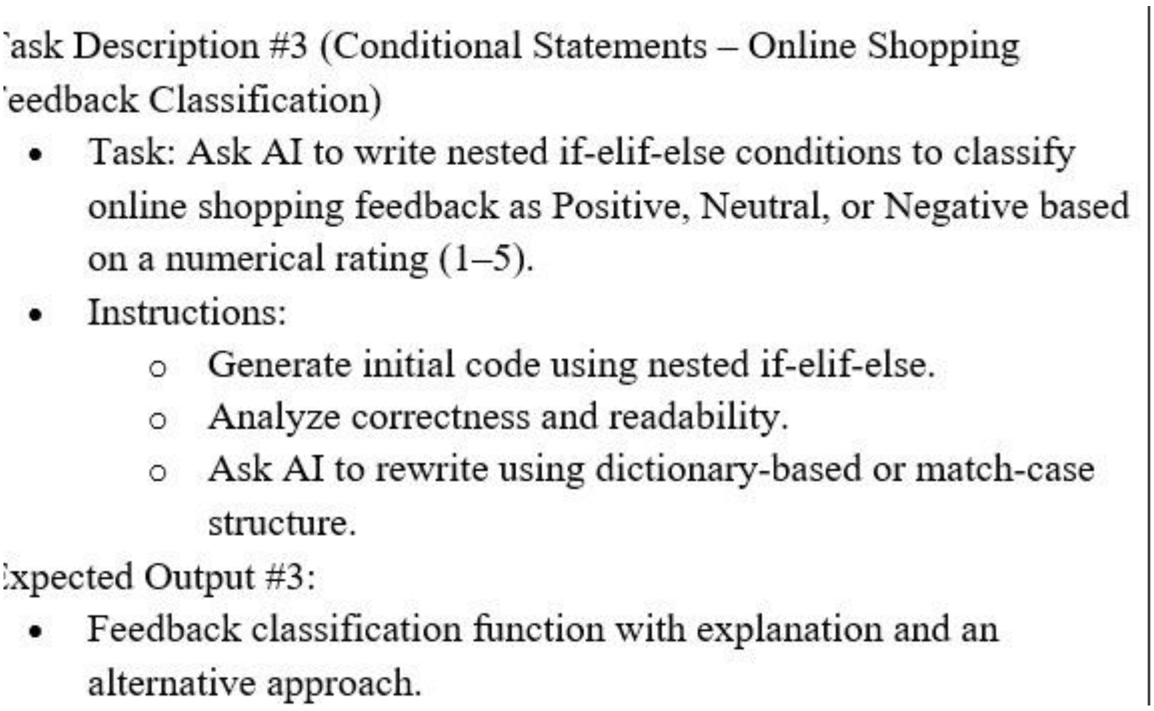
```python
def is_automorphic(num):
    square = num * num
    return str(square).endswith(str(num))

# Using for loop
def automorphic_numbers_for_loop():
    print("Automorphic numbers between 1 and 1000 (using for loop):")
    for num in range(1, 1001):
        if is_automorphic(num):
            print(num, end=" ")
    print("\n")  # for spacing

# Using while loop
def automorphic_numbers_while_loop():
    print("Automorphic numbers between 1 and 1000 (using while loop):")
    num = 1
    while num <= 1000:
        if is_automorphic(num):
            print(num, end=" ")
        num += 1
    print()

# Calling both functions
automorphic_numbers_for_loop()
automorphic_numbers_while_loop()
```

```
Automorphic numbers between 1 and 1000 (using for loop):
1 5 6 25 76 376 625

Automorphic numbers between 1 and 1000 (using while loop):
1 5 6 25 76 376 625
```

import those all

You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

TASK-3:



ask Description #3 (Conditional Statements – Online Shopping
eedback 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.

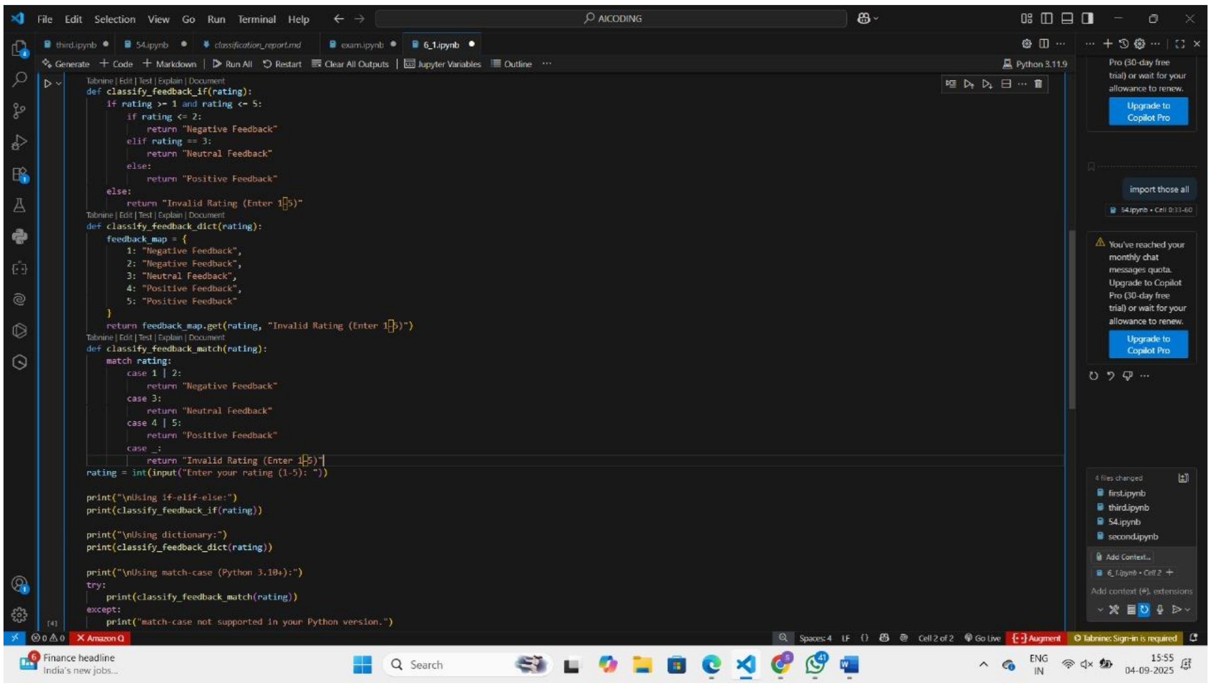
xpected Output #3:

- Feedback classification function with explanation and an alternative approach.

PROMPT:

Write a function using nested if-elif-else to classify ratings (1–5) as Positive, Neutral, or Negative. Then rewrite it using a dictionary or match-case.

CODE:



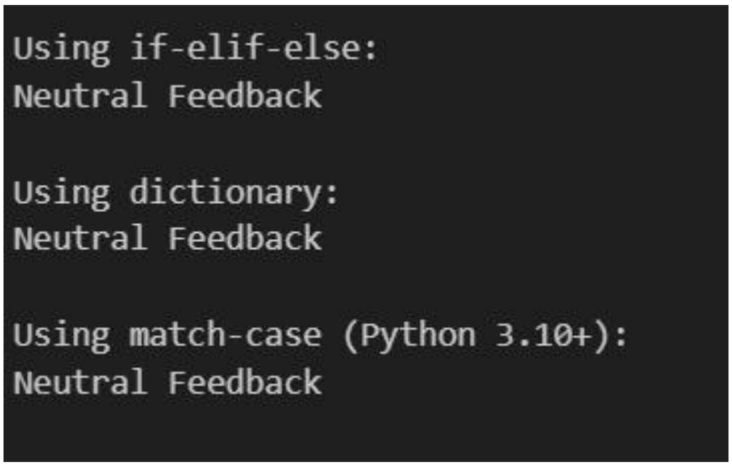
```python
def classify_feedback_if(rating):
    if rating >= 1 and rating <= 5:
        if rating <= 2:
            return "Negative Feedback"
        elif rating == 3:
            return "Neutral Feedback"
        else:
            return "Positive Feedback"
    else:
        return "Invalid Rating (Enter 1–5)"
def classify_feedback_dict(rating):
    feedback_map = {
        1: "Negative Feedback",
        2: "Negative Feedback",
        3: "Neutral Feedback",
        4: "Positive Feedback",
        5: "Positive Feedback"
    }
    return feedback_map.get(rating, "Invalid Rating (Enter 1–5)")
def classify_feedback_match(rating):
    match rating:
        case 1 | 2:
            return "Negative Feedback"
        case 3:
            return "Neutral Feedback"
        case 4 | 5:
            return "Positive Feedback"
        case _:
            return "Invalid Rating (Enter 1–5)"
rating = int(input("Enter your rating (1-5): "))

print("\nUsing if-elif-else:")
print(classify_feedback_if(rating))

print("\nUsing dictionary:")
print(classify_feedback_dict(rating))

print("\nUsing match-case (Python 3.10+):")
try:
    print(classify_feedback_match(rating))
except:
    print("match-case not supported in your Python version.")
```

Pro (30-day free trial) or wait for your allowance to renew.

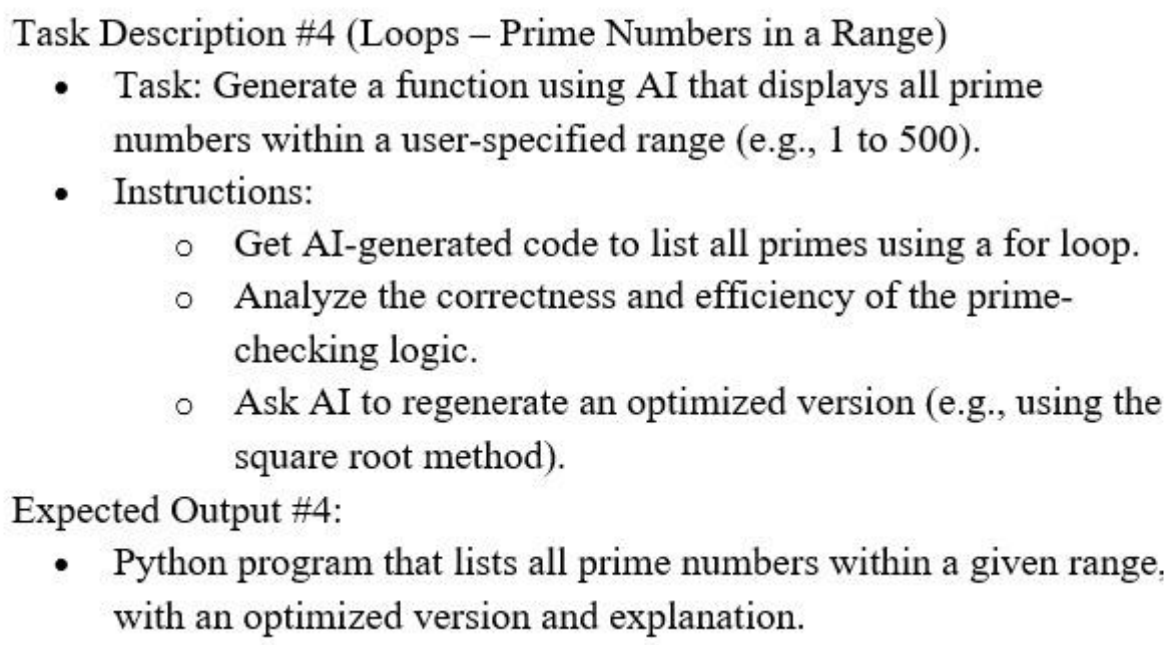
import those all

You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

OUTPUT:


```
Using if-elif-else:
Neutral Feedback

Using dictionary:
Neutral Feedback

Using match-case (Python 3.10+):
Neutral Feedback
```

TASK-4:


Task Description #4 (Loops – Prime Numbers in a Range)

- Task: 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).

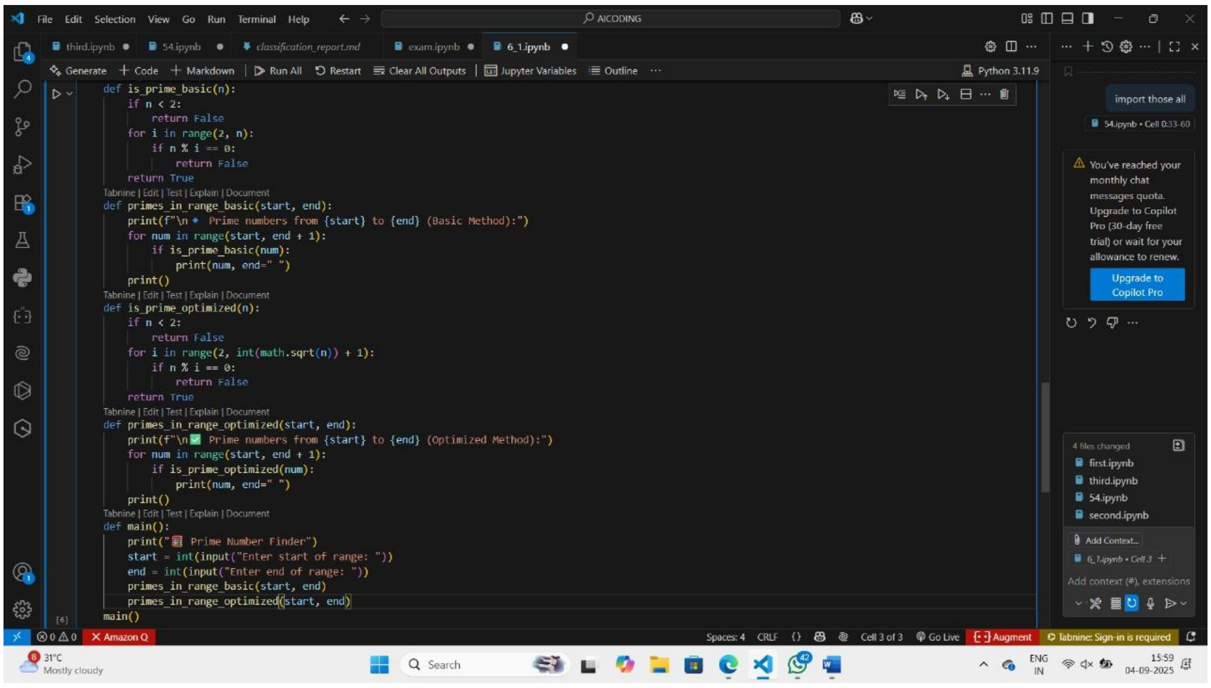
Expected Output #4:

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

PROMPT:

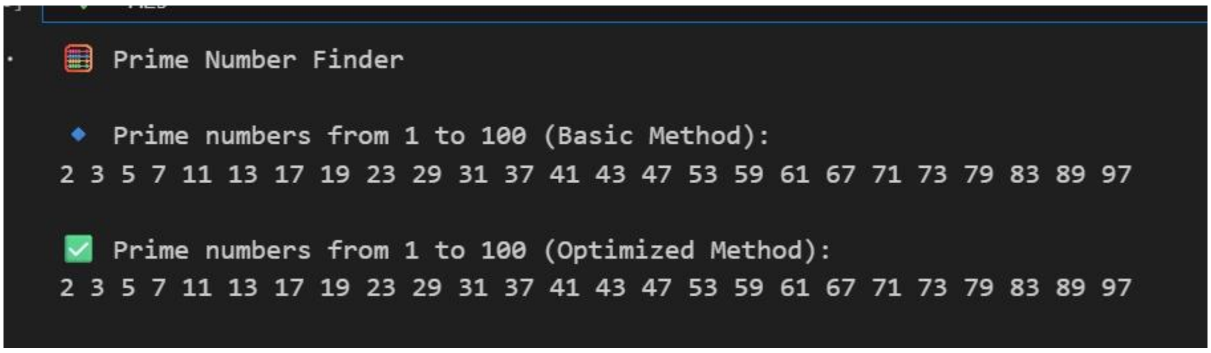
Write a function to display all prime numbers within a given range using a for loop. Then improve it using the square root optimization.

CODE:



```python
def is_prime_basic(n):
    if n < 2:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
def primes_in_range_basic(start, end):
    print(f"\n🔹 Prime numbers from {start} to {end} (Basic Method):")
    for num in range(start, end + 1):
        if is_prime_basic(num):
            print(num, end=" ")
    print()
def is_prime_optimized(n):
    if n < 2:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True
def primes_in_range_optimized(start, end):
    print(f"\n✅ Prime numbers from {start} to {end} (Optimized Method):")
    for num in range(start, end + 1):
        if is_prime_optimized(num):
            print(num, end=" ")
    print()
def main():
    print("🧮 Prime Number Finder")
    start = int(input("Enter start of range: "))
    end = int(input("Enter end of range: "))
    primes_in_range_basic(start, end)
    primes_in_range_optimized(start, end)
main()
```

OUTPUT:



```
🧮 Prime Number Finder

🔹 Prime numbers from 1 to 100 (Basic Method):
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

✅ Prime numbers from 1 to 100 (Optimized Method):
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

TASK-5:

Task Description #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.
  - Analyze if methods handle edge cases (e.g., issuing unavailable books).
  - Ask AI to add comments and documentation.

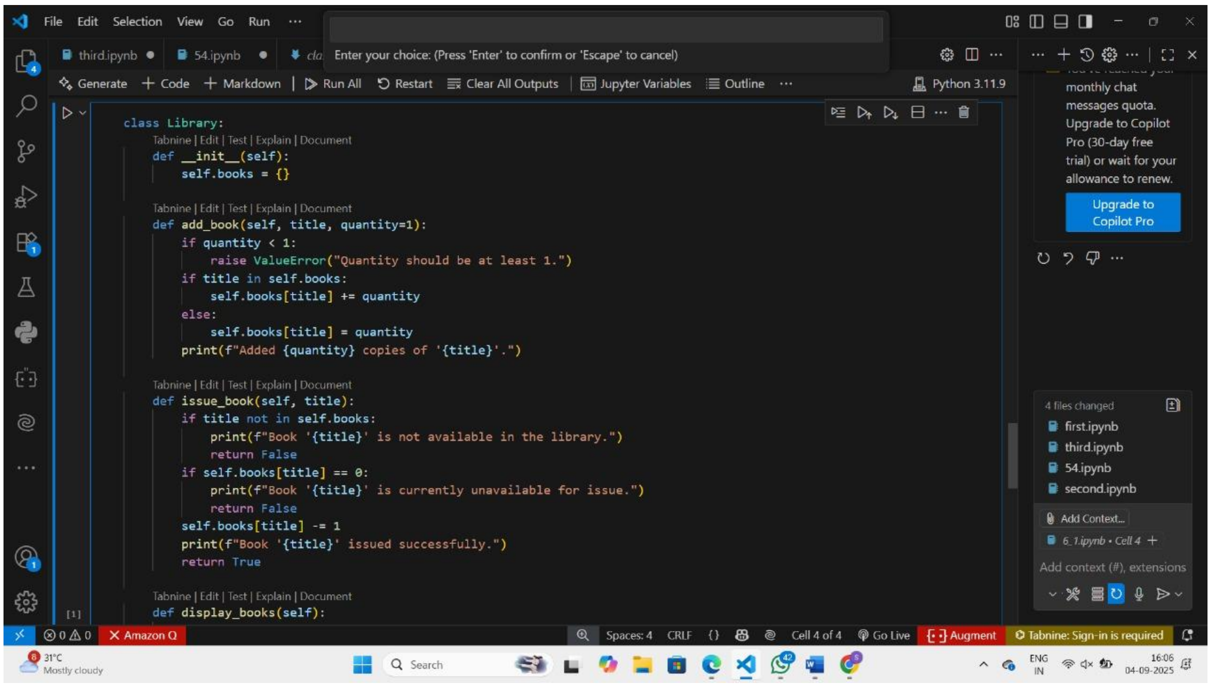
Expected Output #5:

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

PROMPT:

Build a Library class in Python with methods to add books, issue books, and display available books. Make sure to handle unavailable books and add comments.

CODE & OUTPUT:

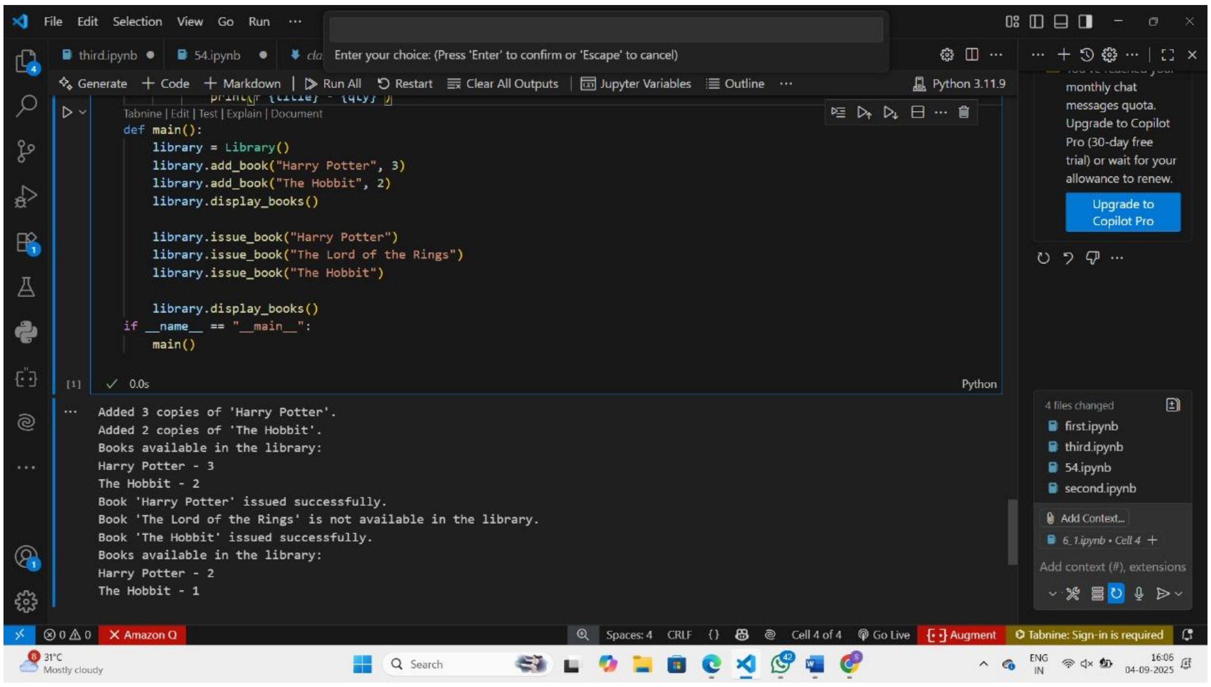


```
class Library:
    def __init__(self):
        self.books = {}

    def add_book(self, title, quantity=1):
        if quantity < 1:
            raise ValueError("Quantity should be at least 1.")
        if title in self.books:
            self.books[title] += quantity
        else:
            self.books[title] = quantity
        print(f"Added {quantity} copies of '{title}'.")

    def issue_book(self, title):
        if title not in self.books:
            print(f"Book '{title}' is not available in the library.")
            return False
        if self.books[title] == 0:
            print(f"Book '{title}' is currently unavailable for issue.")
            return False
        self.books[title] -= 1
        print(f"Book '{title}' issued successfully.")
        return True

    def display_books(self):
```

monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

4 files changed: first.ipynb, third.ipynb, 54.ipynb, second.ipynb

Enter your choice: (Press 'Enter' to confirm or 'Escape' to cancel)



Enter your choice: (Press 'Enter' to confirm or 'Escape' to cancel)

```python
def main():
    library = Library()
    library.add_book("Harry Potter", 3)
    library.add_book("The Hobbit", 2)
    library.display_books()

    library.issue_book("Harry Potter")
    library.issue_book("The Lord of the Rings")
    library.issue_book("The Hobbit")

    library.display_books()
if __name__ == "__main__":
    main()
```

```
Added 3 copies of 'Harry Potter'.
Added 2 copies of 'The Hobbit'.
Books available in the library:
Harry Potter - 3
The Hobbit - 2
Book 'Harry Potter' issued successfully.
Book 'The Lord of the Rings' is not available in the library.
Book 'The Hobbit' issued successfully.
Books available in the library:
Harry Potter - 2
The Hobbit - 1
```