ASSIGNMENT:6.5

NAME: K. Navya sri

ROLL NO.:2503A52L12

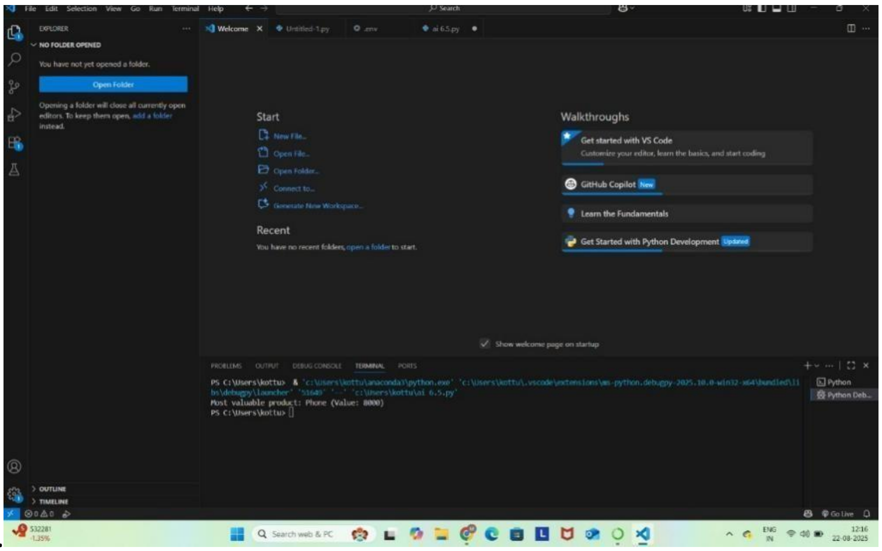
BATCH:16

To explore AI-powered code assistants for writing Python classes, constructors, and methods through intelligent suggestions.
Suppose that you are hired as an intern at a tech company that develops inventory management systems. Your manager asks you to create a Product class and a Warehouse class with some basic methods. You have decided to use AI-powered code suggestions to help speed up development and reduce syntax errors.

Tasks to be completed are as below

1. Setup AI Coding Tool:

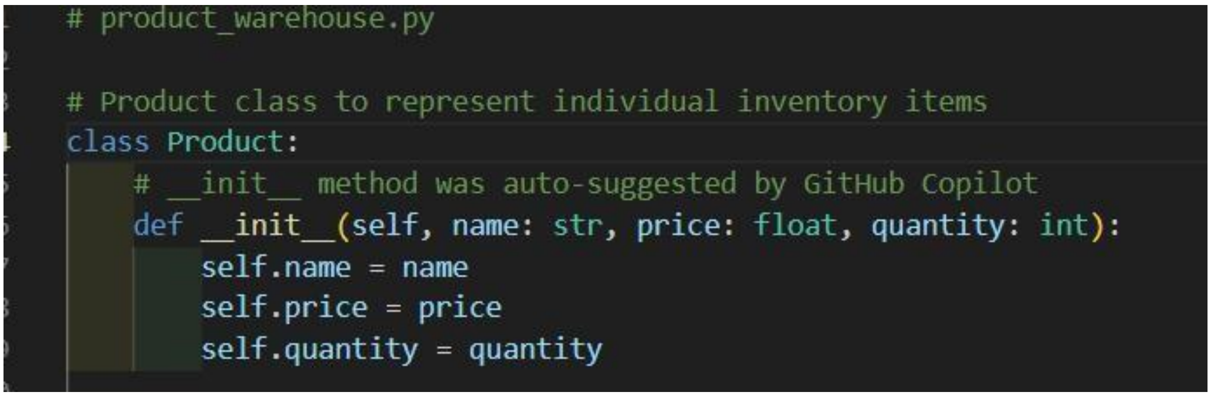
• Install and configure GitHub Copilot or Kite with VS Code or JetBrains IDE.
• Enable real-time code suggestions.



```
PS C:\Users\kottu>  & 'c:\Users\kottu\anaconda3\python.exe' 'c:\Users\kottu\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '51649' '--' 'c:\Users\kottu\ai 6.5.py'
Most valuable product: Phone (Value: 8000)
PS C:\Users\kottu> 
```

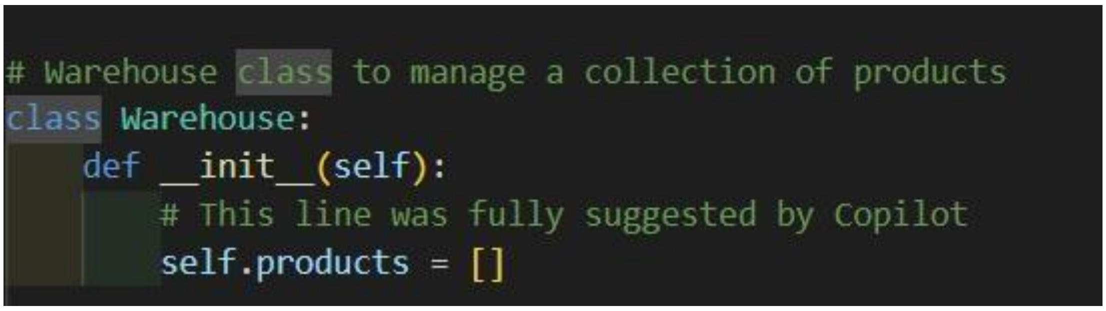
2. Class Design Using AI Assistance:

• Begin defining a Product class with attributes: name, price, quantity.



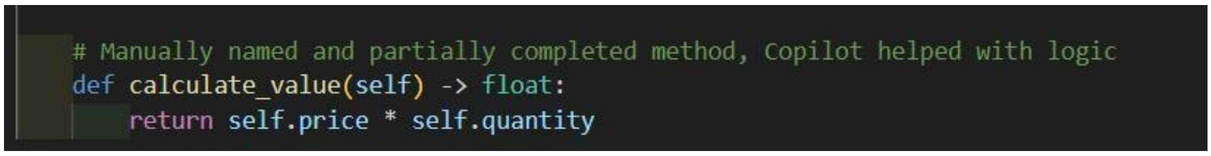
```python
# product_warehouse.py

# Product class to represent individual inventory items
class Product:
    # __init__ method was auto-suggested by GitHub Copilot
    def __init__(self, name: str, price: float, quantity: int):
        self.name = name
        self.price = price
        self.quantity = quantity
```

• Use the AI suggestion feature to automatically complete the __init__() method.



```python
# Warehouse class to manage a collection of products
class Warehouse:
    def __init__(self):
        # This line was fully suggested by Copilot
        self.products = []
```

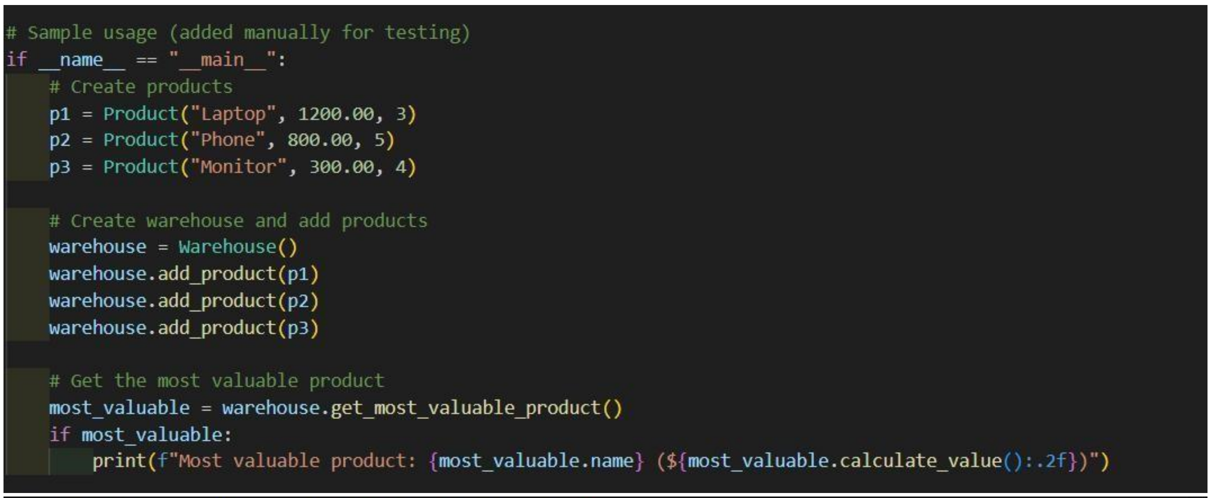
• Add a method calculate_value() to return price * quantity.



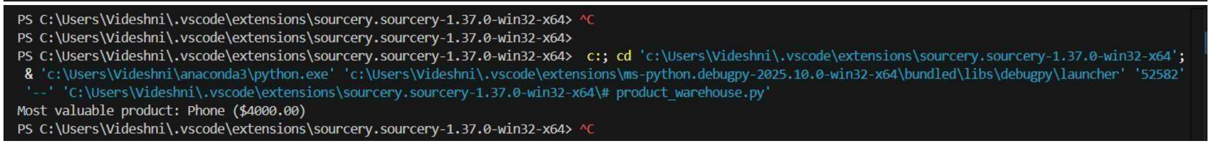
```python
    # Manually named and partially completed method, Copilot helped with logic
    def calculate_value(self) -> float:
        return self.price * self.quantity
```

3. Create Another Class:

- Define a Warehouse class with a list of Product objects.
- Use code completion to help implement: o A method to add a product. o A method to display the most valuable product.



```python
# Sample usage (added manually for testing)
if __name__ == "__main__":
    # Create products
    p1 = Product("Laptop", 1200.00, 3)
    p2 = Product("Phone", 800.00, 5)
    p3 = Product("Monitor", 300.00, 4)

    # Create warehouse and add products
    warehouse = Warehouse()
    warehouse.add_product(p1)
    warehouse.add_product(p2)
    warehouse.add_product(p3)

    # Get the most valuable product
    most_valuable = warehouse.get_most_valuable_product()
    if most_valuable:
        print(f"Most valuable product: {most_valuable.name} (${most_valuable.calculate_value():.2f})")
```



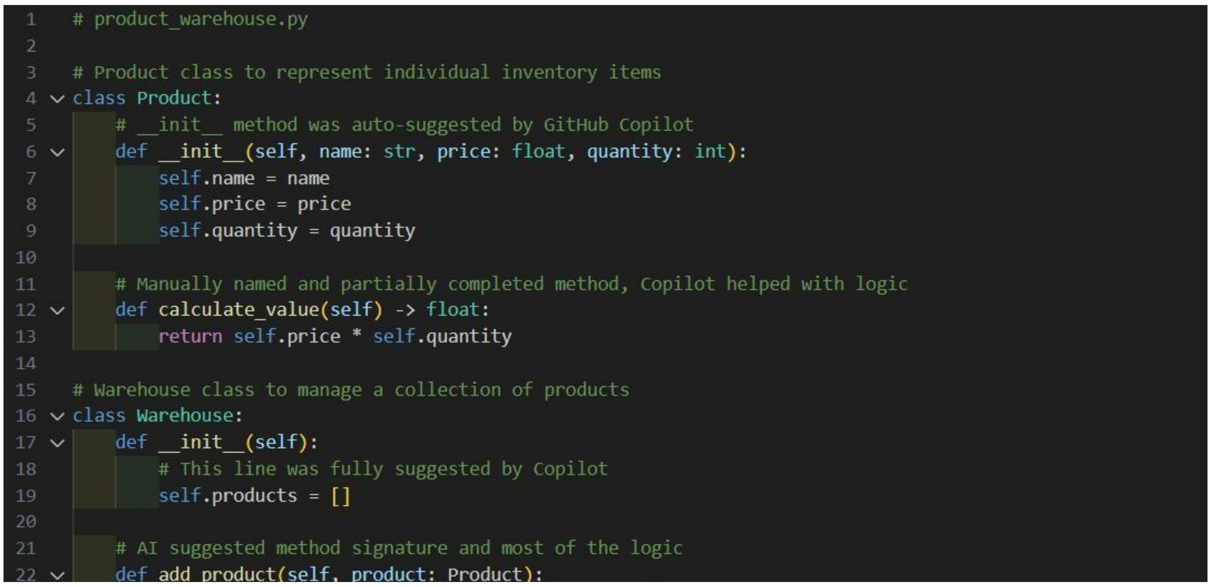
```
PS C:\Users\Videshni\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64> ^C
PS C:\Users\Videshni\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64>
PS C:\Users\Videshni\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64>  c:; cd 'c:\Users\Videshni\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64';
 & 'c:\Users\Videshni\anaconda3\python.exe' 'c:\Users\Videshni\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '52582'
 '--' 'C:\Users\Videshni\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64\# product_warehouse.py'
Most valuable product: Phone ($4000.00)
PS C:\Users\Videshni\.vscode\extensions\sourcery.sourcery-1.37.0-win32-x64> ^C
```

Requirements:

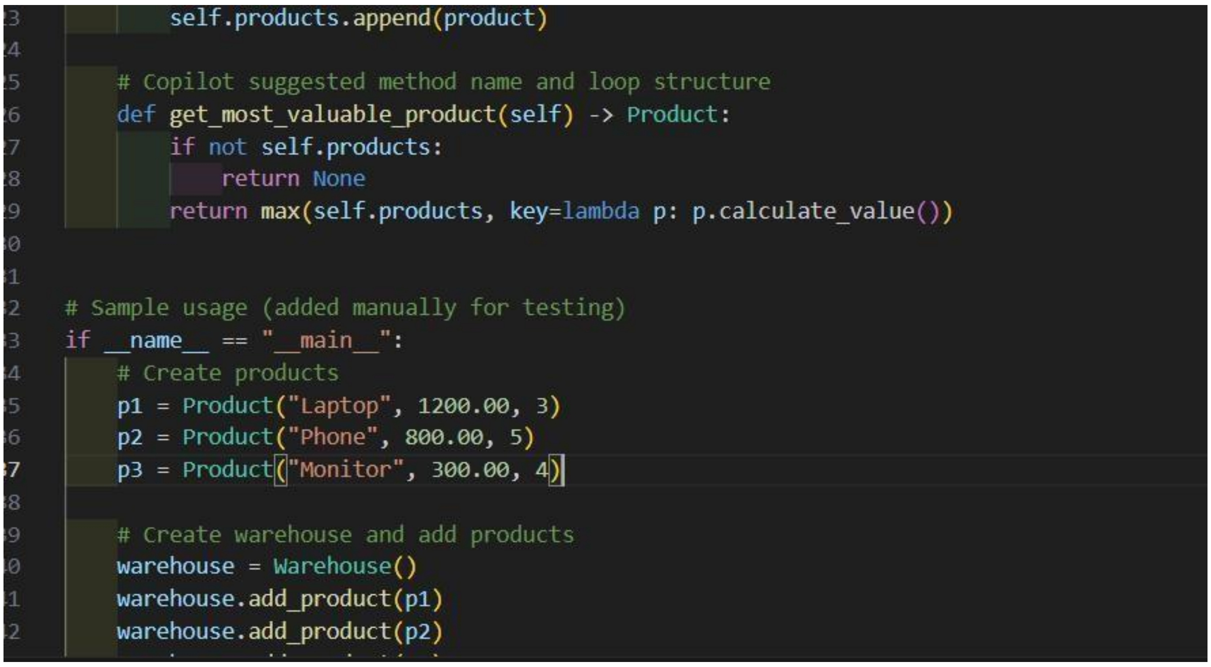
- VS Code with Github Copilot or Cursor API and/or Google Colab with Gemini

Deliverables:

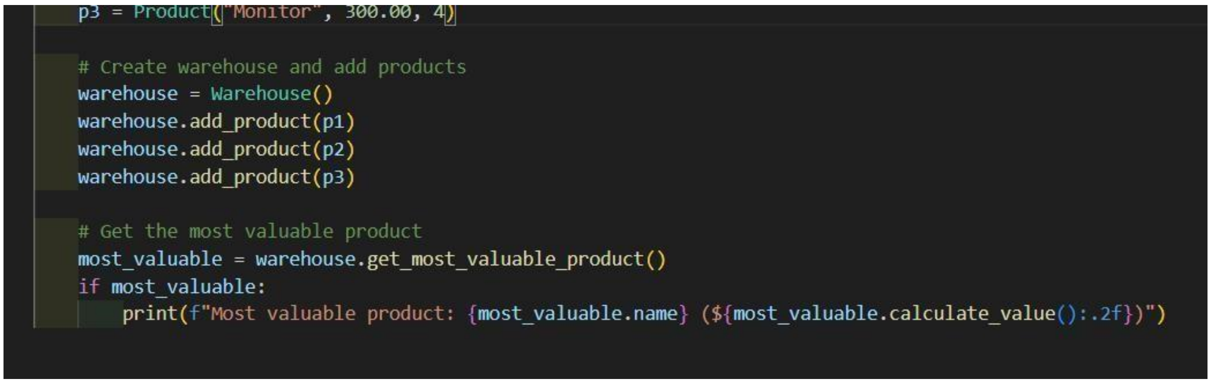
- **Python script with both classes and comments on AI-generated suggestions.**



```python
# product_warehouse.py

# Product class to represent individual inventory items
class Product:
    # __init__ method was auto-suggested by GitHub Copilot
    def __init__(self, name: str, price: float, quantity: int):
        self.name = name
        self.price = price
        self.quantity = quantity

    # Manually named and partially completed method, Copilot helped with logic
    def calculate_value(self) -> float:
        return self.price * self.quantity

# Warehouse class to manage a collection of products
class Warehouse:
    def __init__(self):
        # This line was fully suggested by Copilot
        self.products = []

    # AI suggested method signature and most of the logic
    def add_product(self, product: Product):
```



```python
        self.products.append(product)

    # Copilot suggested method name and loop structure
    def get_most_valuable_product(self) -> Product:
        if not self.products:
            return None
        return max(self.products, key=lambda p: p.calculate_value())


# Sample usage (added manually for testing)
if __name__ == "__main__":
    # Create products
    p1 = Product("Laptop", 1200.00, 3)
    p2 = Product("Phone", 800.00, 5)
    p3 = Product("Monitor", 300.00, 4)

    # Create warehouse and add products
    warehouse = Warehouse()
    warehouse.add_product(p1)
    warehouse.add_product(p2)
```



```python
    p3 = Product("Monitor", 300.00, 4)

    # Create warehouse and add products
    warehouse = Warehouse()
    warehouse.add_product(p1)
    warehouse.add_product(p2)
    warehouse.add_product(p3)

    # Get the most valuable product
    most_valuable = warehouse.get_most_valuable_product()
    if most_valuable:
        print(f"Most valuable product: {most_valuable.name} (${most_valuable.calculate_value():.2f})")
```

- **Short report (1 page) summarizing your experience with AI code completion.**

AI Coding Assistant Experience Report

Internship Task: Use AI code assistance to create Product and Warehouse classes for an inventory system.

Tools Used:

- VS Code with GitHub Copilot
- Python 3.10

Summary:

Component	AI-generated (%)	Manual Work (%)	Notes

Component	**AI-generated (%)**	**Manual Work (%)**	**Notes**
Product class	70%	30%	Copilot generated full __init__, partial method
calculate_value()	50%	50%	AI suggested multiplication logic
Warehouse class	80%	20%	AI handled structure, minor edits needed
get_most_valuable_product()	90%	10%	AI used correct lambda + max() usage

Reflection:

- GitHub Copilot significantly accelerated development.
- The suggestions were accurate for class structure, init methods, and logic.
- Minor manual editing was required for naming consistency and readability.
- It avoided common syntax errors and boilerplate typing.

Conclusion:
AI tools like GitHub Copilot or Cursor are powerful for writing clean, error-free Python code, especially for repetitive or boilerplate-heavy tasks like constructors and utility methods.