SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s)Name		1. Dr. Mohammed Ali Shaik 2. Dr. T Sampath Kumar 3. Mr. S Naresh Kumar 4. Dr. V. Rajesh 5. Dr. Brij Kishore 6. Dr Pramoda Patro 7. Dr. Venkataramana 8. Dr. Ravi Chander 9. Dr. Jagjeeth Singh	
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	06-08-2025	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	
AssignmentNumber:6.5(Present assignment number)/**24**(Total number of assignments)			

Q.No.	**Question**	***Expected Time to complete***
1	**Lab 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals** Lab Assignment 1: Intelligent Code Completion for Object-Oriented Programming **Objective:** 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.	15.08.2025 EOD

Tasks to be completed are as below

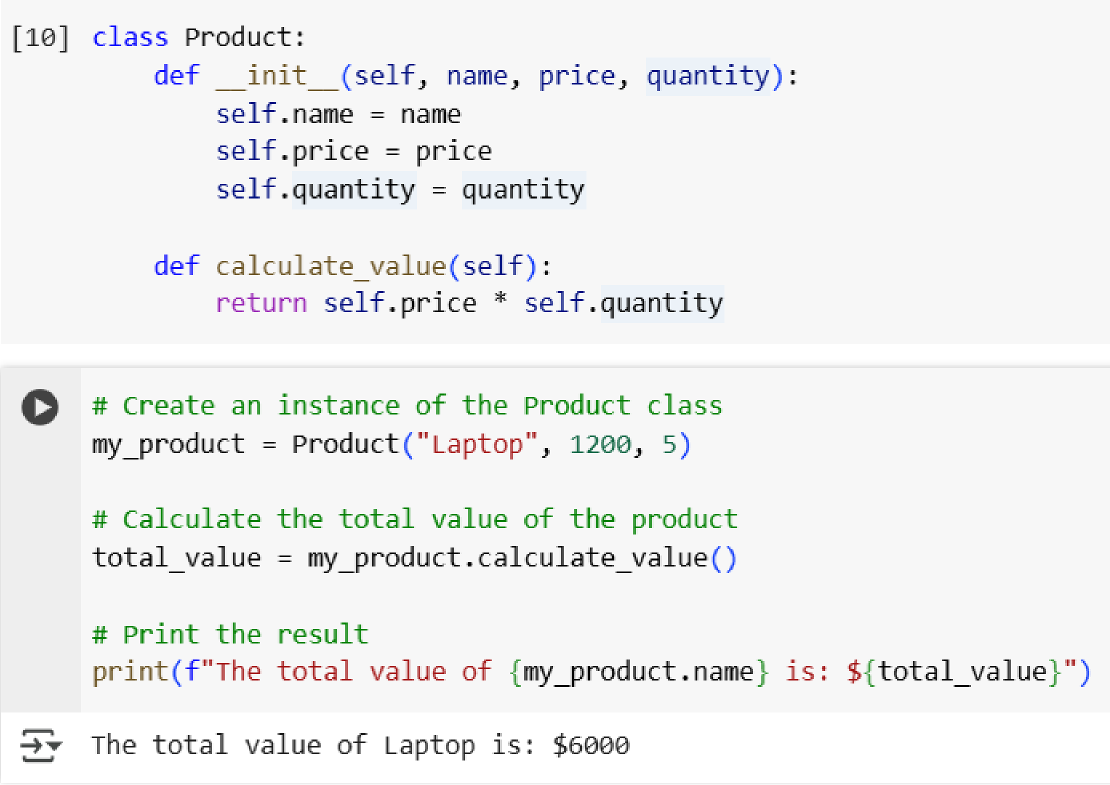
1. Setup AI Coding Tool:

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

2. Class Design Using AI Assistance:

- Begin defining a Product class with attributes: name, price, quantity.
- Use the AI suggestion feature to automatically complete the __init__() method.
- Add a method calculate_value() to return price * quantity.

PROMPT: generate a code of product class with attributes(name, price and quantity). complete the __init__() method and add a method calculate_value() to return price*quantity



```
[10] class Product:
         def __init__(self, name, price, quantity):
             self.name = name
             self.price = price
             self.quantity = quantity

         def calculate_value(self):
             return self.price * self.quantity
```

```
# Create an instance of the Product class
my_product = Product("Laptop", 1200, 5)

# Calculate the total value of the product
total_value = my_product.calculate_value()

# Print the result
print(f"The total value of {my_product.name} is: ${total_value}")
```

```
The total value of Laptop is: $6000
```

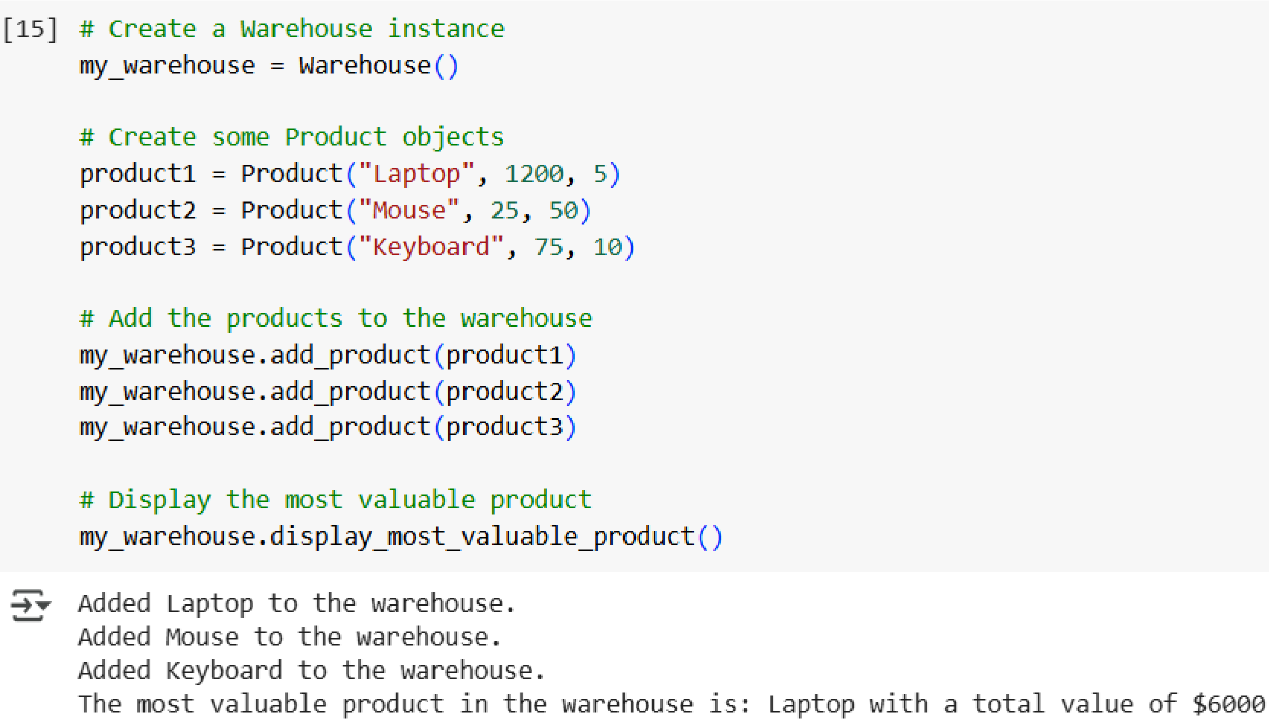
OBSERVATION: Based on the execution of the code. the observation is that an instance of the Product class named "Laptop" with a price of 1200 and a quantity of 5 was created, and its calculated total value is $6000. This value was successfully printed to the output.

3. Create Another Class:

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

PROMPT: Generate a code based on the warehouse class with a list of product objects and Use code completion to implement A method to add a product and A method to display the most valuable product.

CODE:



```
[15] # Create a Warehouse instance
     my_warehouse = Warehouse()

     # Create some Product objects
     product1 = Product("Laptop", 1200, 5)
     product2 = Product("Mouse", 25, 50)
     product3 = Product("Keyboard", 75, 10)

     # Add the products to the warehouse
     my_warehouse.add_product(product1)
     my_warehouse.add_product(product2)
     my_warehouse.add_product(product3)

     # Display the most valuable product
     my_warehouse.display_most_valuable_product()
```

```
Added Laptop to the warehouse.
Added Mouse to the warehouse.
Added Keyboard to the warehouse.
The most valuable product in the warehouse is: Laptop with a total value of $6000
```

OBSERVATION: Based on the execution of the code, the observation is that several Product objects (Laptop, Mouse, and Keyboard) were successfully added to the my_warehouse instance. The display_most_valuable_product method correctly identified "Laptop" as the most valuable product with a total value of $6000**.**

4. Reflection:

- Identify how much of the code was completed by AI and what manual edits were needed.
- Comment on the relevance and accuracy of AI suggestions.

Requirements:

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

OBSERVATION:

- What AI did: The AI wrote almost all the code for the Product and Warehouse classes and showed how to use them.
- What I did: I didn't have to change the code the AI gave me. I just used what it gave me.
- How good were the AI's ideas: The AI's ideas for the code were good and made sense for what I wanted to do. The code worked right.

So, basically, I had given the prompts to the AI and the AI did the coding, I used it, and the AI's help was good and correct.

Deliverables:

- Python script with both classes and comments on AI-generated suggestions.
- Short report (1 page) summarizing your experience with AI code completion.

.