

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	<p><b>Lab 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals</b></p> <p><u>Lab Assignment 1: Intelligent Code Completion for Object-Oriented Programming</u></p> <p><b>Objective:</b> To explore AI-powered code assistants for writing Python classes, constructors, and methods through intelligent suggestions.</p> <p>Suppose that you are hired as an intern at a tech company that develops inventory management systems. Your manager asks you to create a</p>	15.0 8.20 25 EOD	

**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.

### 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:** Write a python program for product class with attributes: name, price, quantity and calculate the return price\*quantity.

### CODE:

```
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
```

```
[18] # Example of how to use the Product class
my_product = Product("Book", 25, 2)
total_value = my_product.calculate_value()
print(f"The total value of {my_product.name} is: ${total_value}")
```

```
➡ The total value of Book is: $50
```

**OBSERVATION:** The code defines a Python class called Product. This class acts as a template for creating objects that represent products. Each

product object has three pieces of information (attributes): its name, its price, and its quantity.

The class also includes a function (method) called `calculate_value`. This method uses the price and quantity of a specific product object to figure out its total value (price multiplied by quantity).

### 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:** Write a python program for warehouse class with a list of product objects and a method to add a product and display the final most valuable product.

### CODE:

```
class Warehouse:
    def __init__(self):
        self.products = []

    def add_product(self, product):
        if isinstance(product, Product):
            self.products.append(product)
            print(f"Added {product.name} to the warehouse.")
        else:
            print("Invalid input. Only Product objects can be added.")

    def display_most_valuable_product(self):
        if not self.products:
            print("The warehouse is empty.")
            return None

        most_valuable = None
        max_value = -1

        for product in self.products:
            current_value = product.calculate_value()
            if current_value > max_value:
                max_value = current_value
                most_valuable = product

        if most_valuable:
            print(f"The most valuable product is: {most_valuable.name} with a total value of ${max_value}")
        return most_valuable
```

```


# Example usage
my_warehouse = Warehouse()

product1 = Product("Laptop", 1200, 1)
product2 = Product("Mouse", 25, 10)
product3 = Product("Keyboard", 75, 5)

my_warehouse.add_product(product1)
my_warehouse.add_product(product2)
my_warehouse.add_product(product3)

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 is: Laptop with a total value of \$1200  
 <\_\_main\_\_.Product at 0x7a5e189528a0>

**OBSERVATION:** The code defines a Python class called Warehouse. Think of it as a digital storage space for your Product objects.

The Warehouse can:

- Hold a list of Product objects.
- `add_product()`: Add new Product objects to its storage. It checks to make sure you're only adding valid product items.
- `display_most_valuable_product()`: Look through all the products it's holding and tell you which one has the highest total value (based on its price and quantity).

#### 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.

**OBSERVATION:** The AI successfully generated all the requested code for both classes and provided accurate examples.

The AI also provided relevant example usage for both classes, demonstrating how to create objects and call the methods.

#### Requirements:

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

	<b>Deliverables:</b>	
--	----------------------	--

- |  |  |  |
|--|--|--|
|  | <ul style="list-style-type: none"><li>• Python script with both classes and comments on AI-generated suggestions.</li><li>• Short report (1 page) summarizing your experience with AI code completion.</li><li>•</li></ul> |  |
|--|--|--|