

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>Lab 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals</p> <p><u>Lab Assignment 1: Intelligent Code Completion for Object-Oriented Programming</u></p> <p>Objective: 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 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.</p> <p>Tasks to be completed are as below</p> <p>1. Setup AI Coding Tool:</p> <ul style="list-style-type: none"> Install and configure GitHub Copilot or Kite with VS Code or JetBrains IDE. Enable real-time code suggestions. <p>Setting Up and Using GitHub Copilot in VS Code:</p> <ol style="list-style-type: none"> Launched VS Code and opened the Extensions view. Searched for and installed the GitHub Copilot extension. Logged in with a GitHub account. 	15.08.2025 EOD	

3. Created a new file and started writing code.
 4. While coding, Copilot shows inline hints press tab to accept a suggestion.
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

1. Create a Python program to define a Product class with the following features :Name, price and quantity. A method `calculate_value()` that returns the total value of the product (`price × quantity`).
2. In the main code, take product details as input from the user.

Code

The screenshot shows a Jupyter Notebook titled 'Untitled15.ipynb'. The code cell contains the following Python 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

# Taking inputs from user
name = input("Enter product name: ")
price = float(input("Enter product price: "))
quantity = int(input("Enter product quantity: "))

product = Product(name, price, quantity)
print(f"Total value of {product.name}: ${product.calculate_value():.2f}")
```

The output of the code is displayed below the code cell:

```
Enter product name: laptop
Enter product price: 1000
Enter product quantity: 10
Total value of laptop: $10000.00
```

Observations:

1. User provides product details (name, price, quantity).
2. The method `calculate_value()` computes total cost.
3. Output clearly shows product name and total value

Code Explanation:

1. `class Product:` → Defines the product
2. `__init__` → Initializes name, price, and quantity.
3. `calculate_value()` → Returns `price * quantity`.
4. `input()` → Takes product details from user.
5. Displays product name and total value.

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:

1. Create a Python program with Product and Warehouse classes to store multiple products, take user input, and display the most valuable product

Code:

```
colab.researchn.google.com/drive/1gyss5ijyga4/TVKkvXSXZ/56Z3KqphUW#scrollto=e4tzacuc

Untitled15.ipynb
File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

class Warehouse:
    def __init__(self):
        self.products = []
    def add_product(self, product):
        if isinstance(product, Product):
            self.products.append(product)
        else:
            print("Invalid product type. Please add a Product object.")
    def find_most_valuable_product(self):
        if not self.products:
            return None
        most_valuable = None
        max_value = -1
        for product in self.products:
            product_value = product.calculate_value()
            if product_value > max_value:
                max_value = product_value
                most_valuable = product
        return most_valuable
warehouse = Warehouse()
while True:
    name = input("Enter product name (or 'done' to finish): ")
    if name.lower() == 'done':
        break
    try:
        price = float(input("Enter product price: "))
        quantity = int(input("Enter product quantity: "))
        product = Product(name, price, quantity)
        warehouse.add_product(product)
    except ValueError:
        print("Invalid input. Please enter valid numbers for price and quantity.")
most_valuable_product = warehouse.find_most_valuable_product()
if most_valuable_product:
    print(f"The most valuable product in the warehouse is: {most_valuable_product.name}")
```

```
colab.researchn.google.com/drive/1gyss5ijyga4/TVKkvXSXZ/56Z3KqphUW#scrollto=e4tzacuc

Untitled15.ipynb
File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

if most_valuable_product:
    print(f"\nThe most valuable product in the warehouse is: {most_valuable_product.name}")
    print(f"\nIts total value is: ${most_valuable_product.calculate_value():.2f}")
else:
    print("\nThe warehouse is empty.")

Enter product name (or 'done' to finish): mouse
Enter product price: 200
Enter product quantity: 10
Enter product name (or 'done' to finish): keyboard
Enter product price: 2000
Enter product quantity: 10
Enter product name (or 'done' to finish): monitor
Enter product price: 7000
Enter product quantity: 10
Enter product name (or 'done' to finish): done

The most valuable product in the warehouse is: monitor
Its total value is: $70000.00
```

Observations:

1. Warehouse can store multiple Product objects in a list.
2. User inputs product details dynamically (name, price, quantity).
3. `calculate_value()` computes total value for each product.
4. `most_valuable_product()` identifies the product with the highest total value.

Code Explanation:

1. `class Warehouse:` → Holds multiple Product objects and related methods
2. `__init__` → Initializes name, price, quantity for each product object
3. `calculate_value()` → Returns $\text{price} \times \text{quantity}$.
4. `most_valuable_product()` → Uses `max()` with a key function to find highest value.

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.

Reflection on AI Coding Tool:

The AI contributed to most of the code by suggesting the class design, constructors, and methods, while I implemented the user input, loops for adding multiple products, and the output display. Its recommendations were largely accurate and time-saving, though slight modifications were needed to the program to my requirements.

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