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**	***ExpectedTime 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. 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. **Implementation of GitHub Copilot in VS Code** 1. Installed GitHub Copilot extension in VS Code. 2. Logged in with GitHub and enabled real-time code suggestions. 3. Tested it by writing a small function, and Copilot suggested the code automatically.	15.08.2025 EOD

4. Now, while typing code, suggestions appear, and I can press Tab to accept.

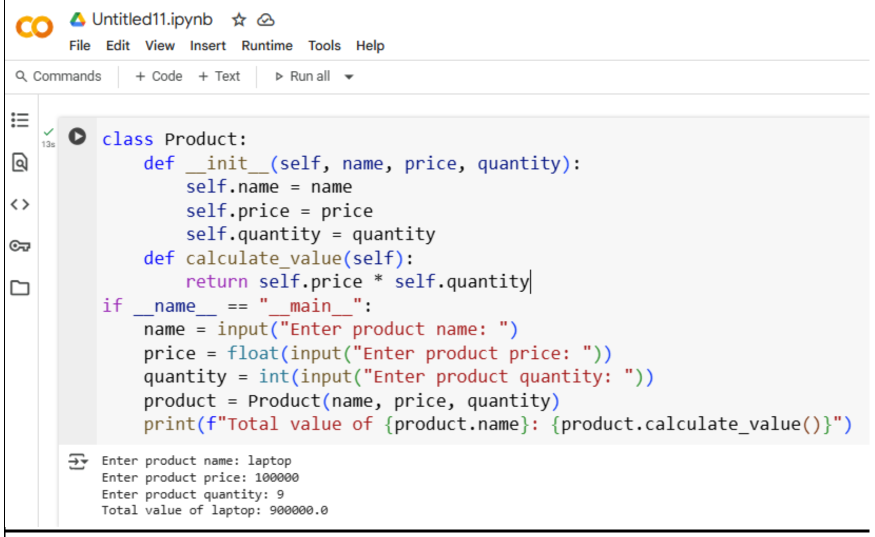
2. Class Design Using AI Assistance:

- 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 with a Product class. The class should have attributes name, price, and quantity. Add a method calculate_value() that returns the total value . Then write code to take input from the user, create a Product object, and print the total value

Code


Untitled11.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```python
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
if __name__ == "__main__":
    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()}")
```

```
Enter product name: laptop
Enter product price: 100000
Enter product quantity: 9
Total value of laptop: 900000.0
```

Observations

1. The program correctly calculates the total value of a product.

·Example : For product laptop with price 100000 and quantity 9, the output is 900000.0.

2.The code is simple, readable, and reusable for any product

Code Explanation

1. A class Product is created with attributes: name, price, and quantity.
2. The constructor __init__ initializes these attributes when a new product is created.
3. The method calculate_value() multiplies price × quantity and returns the total value.

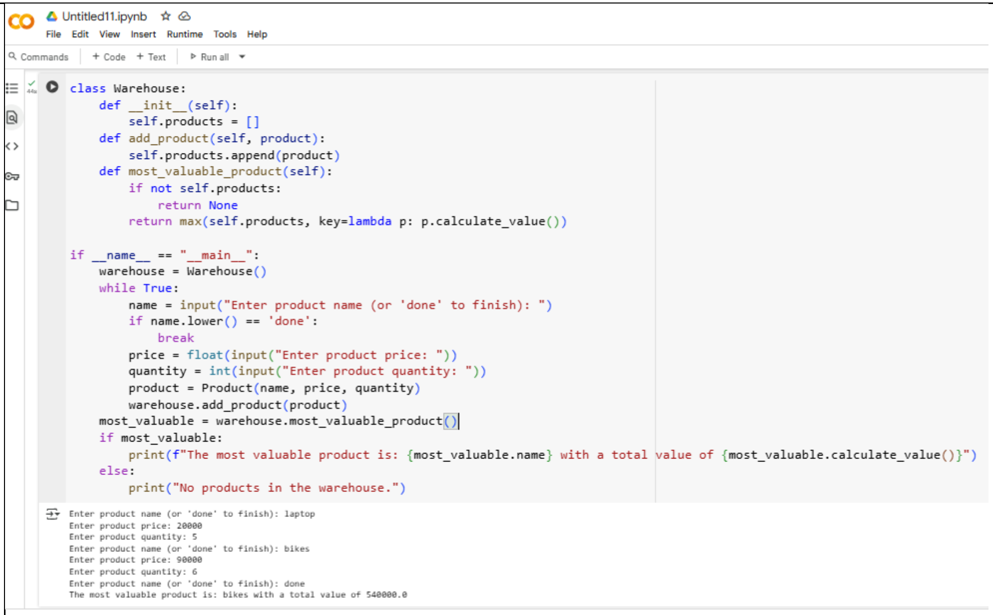
3. Create Another Class:

- 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

_Write a Python program with a Warehouse class that stores multiple Product objects,and add methods to a product to the warehouse and display the most valuable product with user input

Code


Untitled11.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```python
class Warehouse:
    def __init__(self):
        self.products = []
    def add_product(self, product):
        self.products.append(product)
    def most_valuable_product(self):
        if not self.products:
            return None
        return max(self.products, key=lambda p: p.calculate_value())

if __name__ == "__main__":
    warehouse = Warehouse()
    while True:
        name = input("Enter product name (or 'done' to finish): ")
        if name.lower() == 'done':
            break
        price = float(input("Enter product price: "))
        quantity = int(input("Enter product quantity: "))
        product = Product(name, price, quantity)
        warehouse.add_product(product)
    most_valuable = warehouse.most_valuable_product()
    if most_valuable:
        print(f"The most valuable product is: {most_valuable.name} with a total value of {most_valuable.calculate_value()}")
    else:
        print("No products in the warehouse.")
```

```
Enter product name (or 'done' to finish): laptop
Enter product price: 20000
Enter product quantity: 5
Enter product name (or 'done' to finish): bikes
Enter product price: 90000
Enter product quantity: 6
Enter product name (or 'done' to finish): done
The most valuable product is: bikes with a total value of 540000.0
```

Observations:

1. The program correctly stores multiple products in the warehouse.
2. It calculates the total value of each product using the formula: price×quantity.
3. It then compares values and prints the most valuable product.

Code Explanation

1. Creates a Warehouse object.
2. Uses a loop to take product details (name, price, quantity) from the user until they type ‘done’.
3. Each product is stored in the warehouse.
4. At the end, it finds and prints the most valuable product with its total 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 Usage

Most of the code was written with the help of AI . It suggested the class structure, constructor, and methods.
I had to edit and add parts like input prompts, the loop for entering many products, and the final print statements.
The AI suggestions were useful and mostly correct. They saved time, but I still needed to make small changes to fit the program’s needs.

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.

.