

AI-Based Code Completion - Classes, Loops, and Conditionals

NAME:CHENCHU RAJU

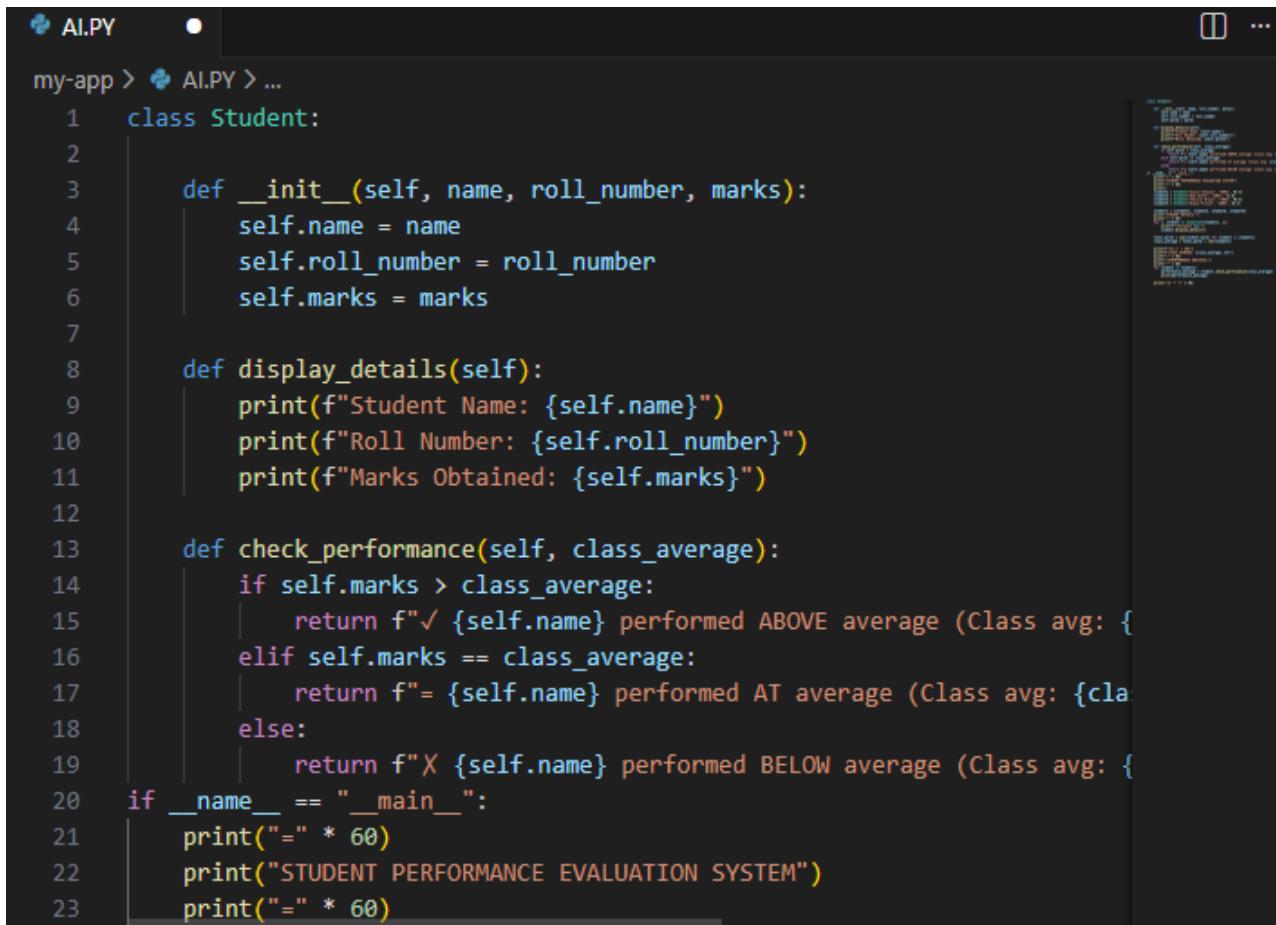
H.T.NO:2303A52206

BATCH:43

AI Assisted Coding

Assignment_6.4

Task 1: Student Performance Evaluation System



The screenshot shows a code editor window with the file 'AI.PY' open. The code defines a class 'Student' with methods for initializing student details, displaying them, and checking performance against a class average. The code uses f-strings for printing and includes a main block at the end.

```
AI.PY
my-app > AI.PY > ...
1  class Student:
2
3      def __init__(self, name, roll_number, marks):
4          self.name = name
5          self.roll_number = roll_number
6          self.marks = marks
7
8      def display_details(self):
9          print(f"Student Name: {self.name}")
10         print(f"Roll Number: {self.roll_number}")
11         print(f"Marks Obtained: {self.marks}")
12
13     def check_performance(self, class_average):
14         if self.marks > class_average:
15             return f"✓ {self.name} performed ABOVE average (Class avg: {"
16         elif self.marks == class_average:
17             return f"= {self.name} performed AT average (Class avg: {cla
18         else:
19             return f"✗ {self.name} performed BELOW average (Class avg: {"
20
21 if __name__ == "__main__":
22     print("=" * 60)
23     print("STUDENT PERFORMANCE EVALUATION SYSTEM")
24     print("=" * 60)
```

```
◆ AI.PY ●
my-app > ◆ AI.PY > ... == "__main__":
20     if __name__ == "__main__":
21         print("=" * 60)
22         print("STUDENT PERFORMANCE EVALUATION SYSTEM")
23         print("=" * 60)
24         print()
25         student1 = Student("Alice Johnson", "S001", 85.5)
26         student2 = Student("Bob Smith", "S002", 72.0)
27         student3 = Student("Charlie Brown", "S003", 90.0)
28         student4 = Student("Diana Prince", "S004", 65.5)
29
30         students = [student1, student2, student3, student4]
31         print("STUDENT DETAILS:")
32         print("-" * 60)
33         for i, student in enumerate(students, 1):
34             print(f"\nStudent {i}:")
35             student.display_details()
36
37         total_marks = sum(student.marks for student in students)
38         class_average = total_marks / len(students)
39
40         print("\n" + "=" * 60)
41         print(f"CLASS AVERAGE: {class_average:.2f}")
42         print("=" * 60)
43         print("\nPERFORMANCE ANALYSIS:")
44         print("-" * 60)
45         for student in students:
46             performance_message = student.check_performance(class_average)
47             print(performance_message)
48
49         print("\n" + "=" * 60)
```

OUTPUT:

```
=====
STUDENT PERFORMANCE EVALUATION SYSTEM
=====

STUDENT DETAILS:
-----
Student 1:
Student Name: Alice Johnson
Roll Number: S001
Marks Obtained: 85.5

Student 2:
Student Name: Bob Smith
Roll Number: S002
Marks Obtained: 72.0

Student 3:
Student Name: Charlie Brown
Roll Number: S003
Marks Obtained: 90.0

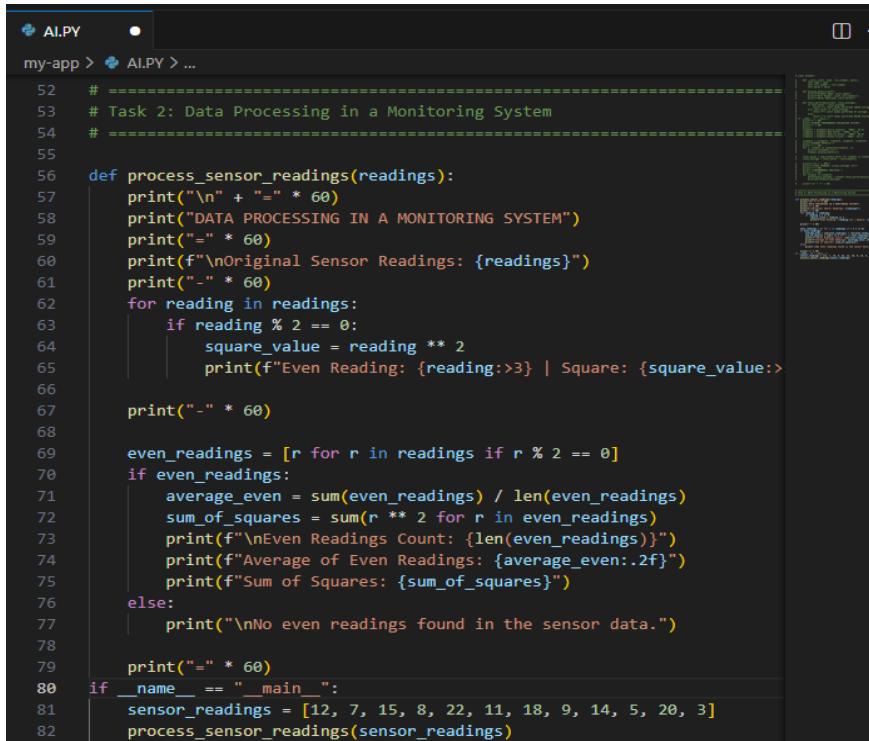
Student 4:
Student Name: Diana Prince
Roll Number: S004
Marks Obtained: 65.5

=====
CLASS AVERAGE: 78.25
=====

PERFORMANCE ANALYSIS:
-----
✓ Alice Johnson performed ABOVE average (Class avg: 78.25, Student marks: 85.5)
✗ Bob Smith performed BELOW average (Class avg: 78.25, Student marks: 72.0)
✓ Charlie Brown performed ABOVE average (Class avg: 78.25, Student marks: 90.0)
-----
✓ Alice Johnson performed ABOVE average (Class avg: 78.25, Student marks: 85.5)
✗ Bob Smith performed BELOW average (Class avg: 78.25, Student marks: 72.0)
-----
✓ Alice Johnson performed ABOVE average (Class avg: 78.25, Student marks: 85.5)
```

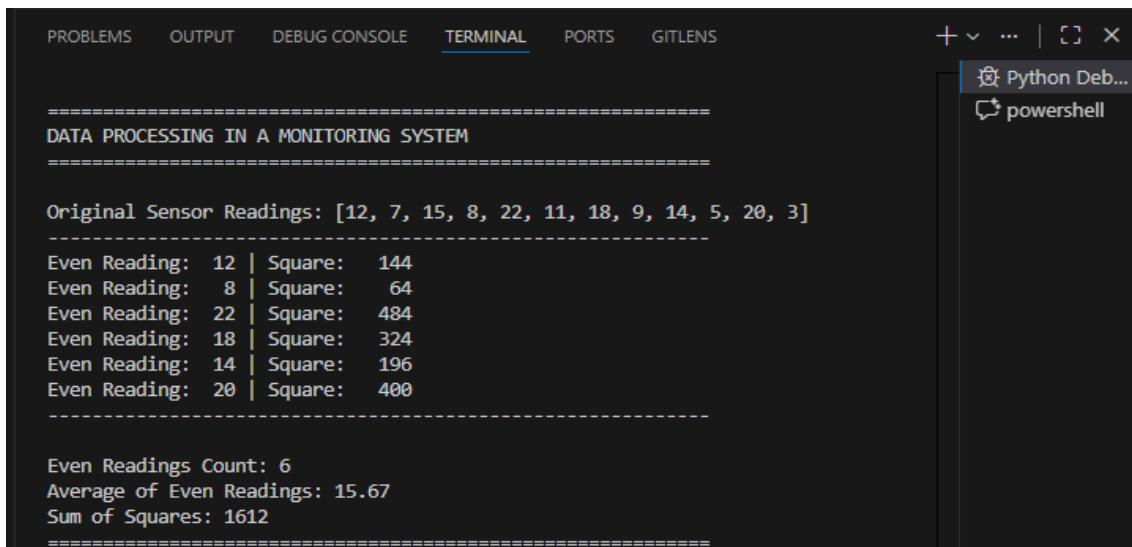
Summary: a *Student Performance Evaluation System* was created using a Python class. AI-generated methods helped display student details and evaluate performance using conditional statements. This demonstrated how AI can quickly complete class methods and reduce manual coding effort

Task 2: Data Processing in a Monitoring System



```
AI.PY
my-app > AI.PY > ...
52  # -----
53  # Task 2: Data Processing in a Monitoring System
54  # -----
55
56  def process_sensor_readings(readings):
57      print("\n" + "=" * 60)
58      print("DATA PROCESSING IN A MONITORING SYSTEM")
59      print("=" * 60)
60      print(f"\nOriginal Sensor Readings: {readings}")
61      print("-" * 60)
62      for reading in readings:
63          if reading % 2 == 0:
64              square_value = reading ** 2
65              print(f"Even Reading: {reading:>3} | Square: {square_value:>3}")
66
67      print("-" * 60)
68
69      even_readings = [r for r in readings if r % 2 == 0]
70      if even_readings:
71          average_even = sum(even_readings) / len(even_readings)
72          sum_of_squares = sum(r ** 2 for r in even_readings)
73          print(f"\nEven Readings Count: {len(even_readings)}")
74          print(f"Average of Even Readings: {average_even:.2f}")
75          print(f"Sum of Squares: {sum_of_squares}")
76      else:
77          print("\nNo even readings found in the sensor data.")
78
79      print("=" * 60)
80  if __name__ == "__main__":
81      sensor_readings = [12, 7, 15, 8, 22, 11, 18, 9, 14, 5, 20, 3]
82      process_sensor_readings(sensor_readings)
```

OUTPUT:

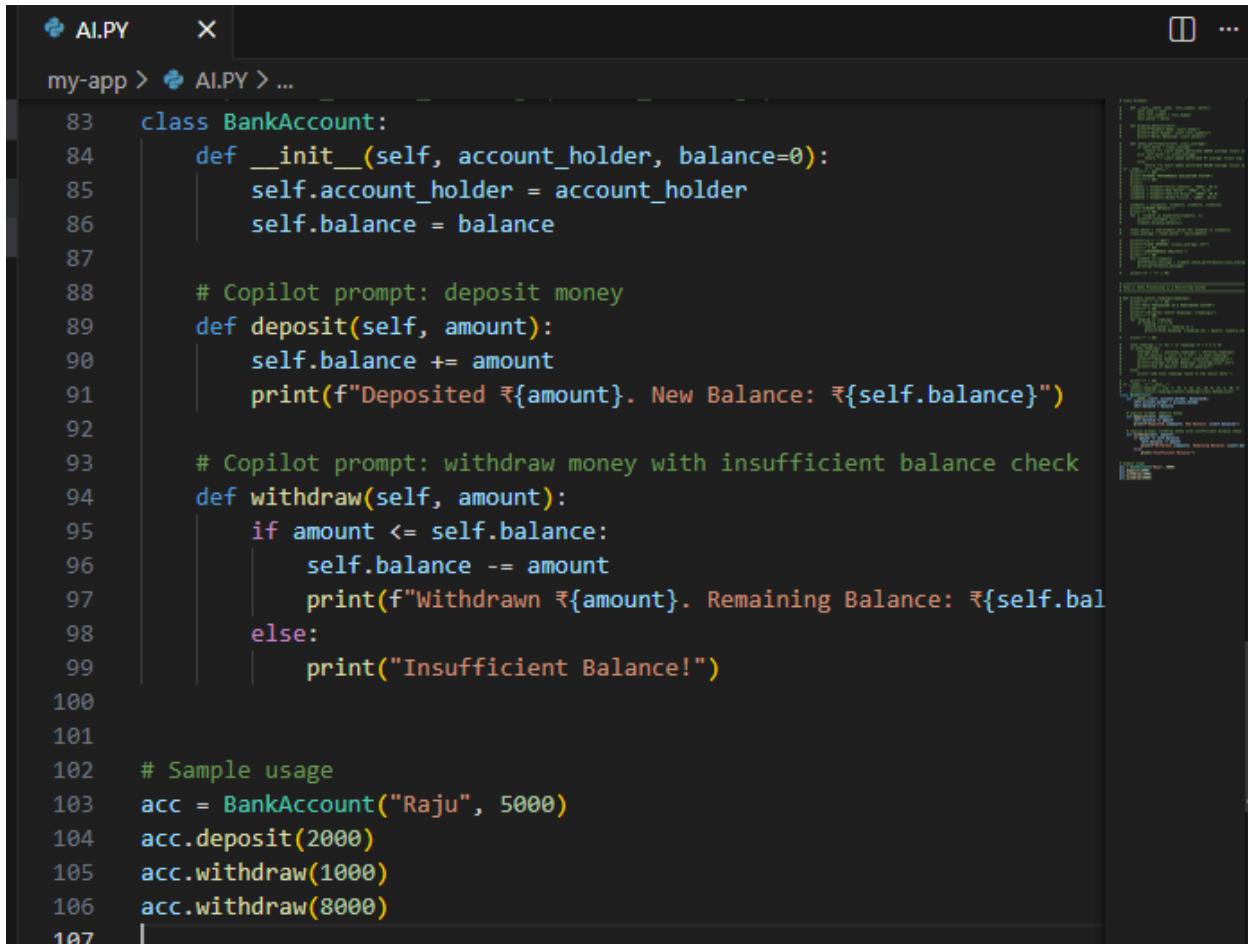


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS + v ... | [] X
=====
DATA PROCESSING IN A MONITORING SYSTEM
=====

Original Sensor Readings: [12, 7, 15, 8, 22, 11, 18, 9, 14, 5, 20, 3]
-----
Even Reading: 12 | Square: 144
Even Reading: 8 | Square: 64
Even Reading: 22 | Square: 484
Even Reading: 18 | Square: 324
Even Reading: 14 | Square: 196
Even Reading: 20 | Square: 400
-----
Even Readings Count: 6
Average of Even Readings: 15.67
Sum of Squares: 1612
=====
```

Summary: a *Monitoring System* processed sensor readings using a loop. AI completed the logic to identify even numbers and calculate their squares. This showed how AI effectively handles repetitive tasks and loop-based operations.

Task 3: Banking Transaction Simulation



```
AI.PY
my-app > AI.PY > ...
83     class BankAccount:
84         def __init__(self, account_holder, balance=0):
85             self.account_holder = account_holder
86             self.balance = balance
87
88         # Copilot prompt: deposit money
89         def deposit(self, amount):
90             self.balance += amount
91             print(f"Deposited ₹{amount}. New Balance: ₹{self.balance}")
92
93         # Copilot prompt: withdraw money with insufficient balance check
94         def withdraw(self, amount):
95             if amount <= self.balance:
96                 self.balance -= amount
97                 print(f"Withdrawn ₹{amount}. Remaining Balance: ₹{self.balance}")
98             else:
99                 print("Insufficient Balance!")
100
101
102     # Sample usage
103     acc = BankAccount("Raju", 5000)
104     acc.deposit(2000)
105     acc.withdraw(1000)
106     acc.withdraw(8000)
107 
```

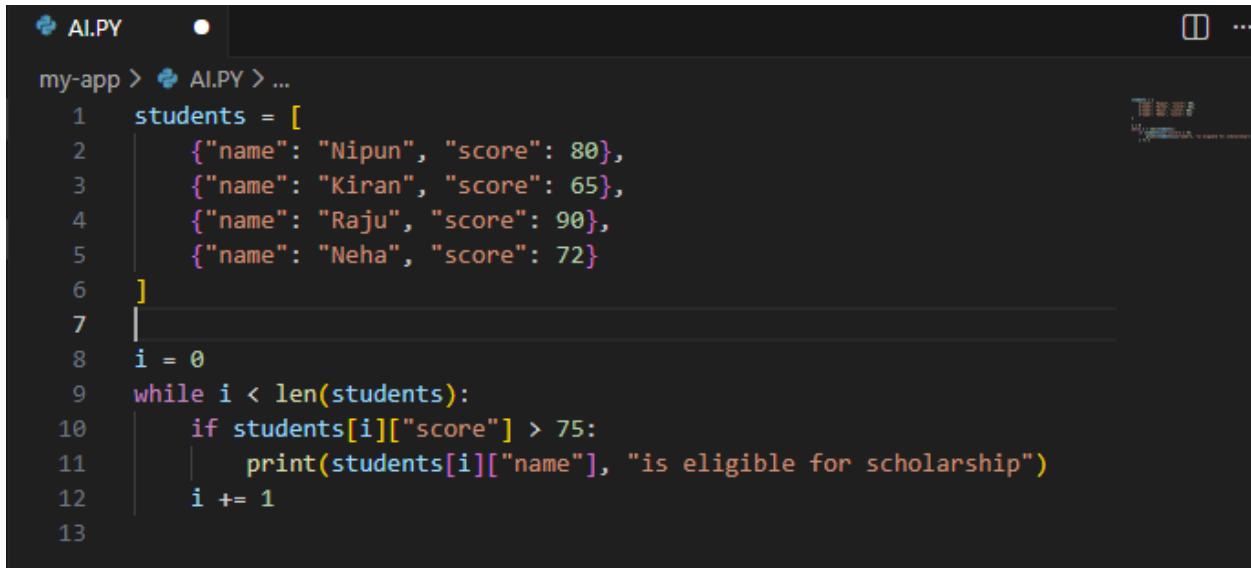
Output:

```
documents\Desktop\Desktop\full_stack\my-app\AI.PY
Deposited ₹2000. New Balance: ₹7000
Withdrawn ₹1000. Remaining Balance: ₹6000
Insufficient Balance!
```

Summary: A *Bank Account Simulation* was implemented with deposit and withdrawal features. AI-generated methods included balance updates and

validation checks for insufficient funds. This highlighted the usefulness of AI in writing secure and logical conditional statements.

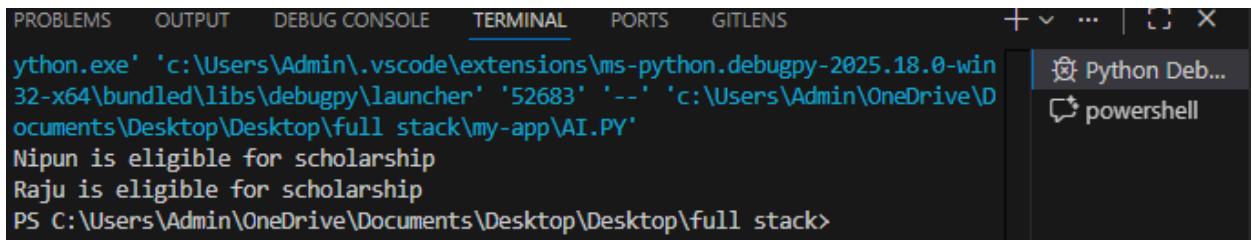
Task 4: Student Scholarship Eligibility Check



The screenshot shows a code editor window in VS Code with a dark theme. The file is named 'AI.PY'. The code defines a list of students and uses a while loop to print the names of students whose scores are greater than 75.

```
AI.PY
my-app > AI.PY > ...
1  students = [
2      {"name": "Nipun", "score": 80},
3      {"name": "Kiran", "score": 65},
4      {"name": "Raju", "score": 90},
5      {"name": "Neha", "score": 72}
6 ]
7
8 i = 0
9 while i < len(students):
10     if students[i]["score"] > 75:
11         print(students[i]["name"], "is eligible for scholarship")
12     i += 1
13
```

Output:



The screenshot shows the terminal tab in VS Code. The command 'python AI.PY' is run, and the output shows that Nipun and Raju are eligible for scholarship.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS + ... X
python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52683' '--' 'c:\Users\Admin\OneDrive\Documents\Desktop\Desktop\full stack\my-app\AI.PY'
Nipun is eligible for scholarship
Raju is eligible for scholarship
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\full stack>
```

Summary: A *Scholarship Eligibility Checker* used a while loop to iterate through student records and filter eligible candidates. This task demonstrated AI assistance in list traversal and condition-based filtering.

Task 5: Online Shopping Cart Module

```
my-app > AI.PY > ShoppingCart
13     class ShoppingCart:
14         def __init__(self):
15             self.items = [] # each item: {name, price, quantity}
16
17         # Copilot prompt: add items
18         def add_item(self, name, price, quantity):
19             self.items.append({"name": name, "price": price, "quantity": q
20                 print(f"{name} added to cart")
21
22         # Copilot prompt: remove items
23         def remove_item(self, name):
24             for item in self.items:
25                 if item["name"] == name:
26                     self.items.remove(item)
27                     print(f"{name} removed")
28                     return
29             print("Item not found")
30         # Copilot prompt: calculate total using loop + apply discount
31         def calculate_total(self):
32             total = 0
33             for item in self.items:
34                 total += item["price"] * item["quantity"]
35
36             def calculate_total(self):
37                 total = 0
38                 for item in self.items:
39                     total += item["price"] * item["quantity"]
40
41                 if total > 2000:
42                     discount = total * 0.10
43                     total -= discount
44                     print("10% discount applied")
45
46                     return total
47         cart = ShoppingCart()
48         cart.add_item("Laptop Mouse", 500, 2)
49         cart.add_item("Keyboard", 1500, 1)
50         print("Total Bill:", cart.calculate_total())
51
```

Output:

```
Laptop Mouse added to cart
python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2025.18.0-win
32-x64\bundled\libs\debugpy\launcher' '55597' '--' 'c:\Users\Admin\OneDrive\D
ocuments\Desktop\Desktop\full stack\my-app\AI.PY'
Laptop Mouse added to cart
documents\Desktop\Desktop\full stack\my-app\AI.PY'
Laptop Mouse added to cart
Laptop Mouse added to cart
Keyboard added to cart
10% discount applied
10% discount applied
Total Bill: 2250.0
```

Summary: A *Shopping Cart System* was developed where AI helped generate methods to add/remove items, compute totals using loops, and apply discounts using conditions. This showed how AI can help build complete real-world modules efficiently.