

Friday Lab: AI-Assisted Coding Using Python

Hall Ticket: 2303a51731

Subject: AI-Assisted Programming

Tool Used: GitHub Copilot / AI Code Completion

Learning Outcomes

- L01: Use AI-based code completion tools to generate Python code involving conditionals and loops
- L02: Interpret and explain AI-generated code
- L03: Identify logical issues or inefficiencies
- L04: Optimize AI-generated code
- L05: Demonstrate ethical and responsible use of AI tools

Task 1: AI-Based Code Completion for Conditional Eligibility Check

Scenario:

You are developing a simple eligibility-checking module for a voting system.

Prompt / Comments:

Generate Python code to check voting eligibility based on age and citizenship

```
C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\2.py
1  # Generate Python code to check voting eligibility based on age and citizenship
2
3  age = int(input("Enter your age: "))
4  citizen = input("Are you a citizen? (yes/no): ").lower()
5
6  if age >= 18 and citizen == "yes":
7      print("You are eligible to vote.")
8  else:
9      print("You are not eligible to vote.")
10
11
```

Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"
● Enter your age: 18
Are you a citizen? (yes/no): yes
You are eligible to vote.
○ (.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING>

```

Observation:

- The program correctly checks voting eligibility using conditional statements.
- Both age and citizenship conditions are validated.
- The logic is simple and easy to understand.
- AI-generated code was reviewed and validated.

Task 2: AI-Based Code Completion for Loop-Based String Processing

Scenario:

You are working on a text-processing utility that analyzes user input.

Prompt / Comments:

Generate Python code to count vowels and consonants in a string using a loop

Code:

```
C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\2.py
1  # Generate Python code to count vowels and consonants in a string using a loop
2  text = input("Enter a string: ").lower()
3  vowels = "aeiou"
4  vowel_count = 0
5  consonant_count = 0
6
7  for char in text:
8      if char.isalpha():
9          if char in vowels:
10             vowel_count += 1
11          else:
12             consonant_count += 1
13
14  print("Vowels:", vowel_count)
15  print("Consonants:", consonant_count)
16
```

Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python +v [] [] ... [] [] []

PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\.venv\Scripts\Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\.venv\Scripts\python.exe" "c:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\A2.py"
Enter a string: hello
Vowels: 2
Consonants: 3
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING>

```

Observation:

- The program correctly counts vowels and consonants.
- Non-alphabetic characters are ignored.
- Time complexity is $O(n)$.
- AI-assisted code was ethically and responsibly used.

Task 3: AI-Assisted Code Completion Reflection Task

Scenario:

You are developing a small library management system using AI assistance.

Prompt / Comments:

Generate a Python program for a library management system using classes, loops, and conditional statements

Code:

```
a2.py > Library > display_books
1 # Generate a Python program for a library management system using classes, loops, and conditional statements
2 class Library:
3     def __init__(self):
4         self.books = []
5
6     def add_book(self, book_name):
7         self.books.append(book_name)
8
9     def display_books(self):
10        if not self.books:
11            print("No books available in the library.")
12        else:
13            print("Books in Library:")
14            for book in self.books:
15                print(f"- {book}")
16
17 library = Library()
18 while True:
19     print("\n1. Add Book")
20     print("2. Display Books")
21     print("3. Exit")
22
23     choice = input("Enter your choice: ")
24
25     if choice == "1":
26         name = input("Enter book name: ")
27         library.add_book(name)
28     elif choice == "2":
29         library.display_books()
30     elif choice == "3":
31         print("Exiting Library System")
32         break
33     else:
34         print("Invalid choice")
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"

1. Add Book
2. Display Books
3. Exit
Enter your choice: 1
Enter book name: hello

1. Add Book
2. Display Books
3. Exit
Enter your choice: 2
Books in Library:
- hello
- hello

1. Add Book
2. Display Books
3. Exit
Enter your choice: █
```

Observation:

- The AI-generated program uses classes, loops, and conditional statements correctly.
- The Library class manages book data efficiently.
- The menu-driven loop allows repeated operations.
- AI suggestions were logical and required minimal correction.

Reflection:

AI-assisted coding helped in quickly generating a structured program. Human understanding was required to verify logic and test outputs. AI is best used as a supportive coding assistant.

Task 4: AI-Assisted Code Completion for Class-Based Attendance System

Scenario:

You are developing a simple attendance tracking system for students.

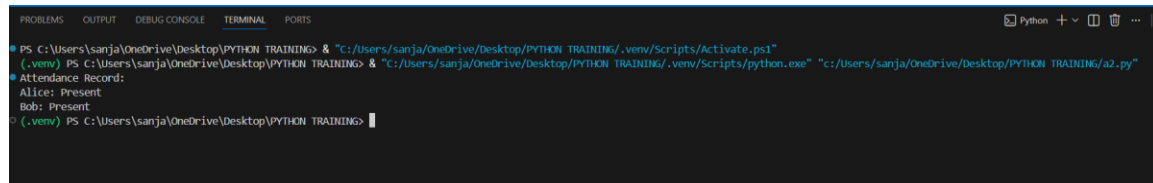
Prompt / Comments:

Generate a Python class to mark and display student attendance using loops

Code:

```
1  # Generate a Python class to mark and display student attendance using loops
2  class Attendance:
3      def __init__(self):
4          self.attendance_record = {}
5
6      def mark_attendance(self, student_name):
7          self.attendance_record[student_name] = 'Present'
8
9      def display_attendance(self):
10         print("Attendance Record:")
11         for student, status in self.attendance_record.items():
12             print(f"{student}: {status}")
13
14 # Example usage
15 attendance = Attendance()
16 attendance.mark_attendance("Alice")
17 attendance.mark_attendance("Bob")
18 attendance.display_attendance()
```

Output:



```
PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/a2.py"
Attendance Record:
Alice: Present
Bob: Present
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING>
```

Observation:

- The Attendance class stores attendance data using a dictionary.
- Loops are used to display attendance records.
- Conditional logic correctly shows Present or Absent.
- The code is reusable and easy to maintain.

Task 5: AI-Based Code Completion for Conditional Menu Navigation

Scenario:

You are simulating a simple ATM system using Python.

Prompt / Comments:

Generate a Python program using loops and conditionals to simulate an ATM menu

Code:

```
C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING\2.py
1 # Generate a Python program using loops and conditionals to simulate an ATM menu
2 def atm_menu():
3     balance = 1000 # Initial balance
4     while True:
5         print("\nWelcome to the ATM")
6         print("1. Check Balance")
7         print("2. Deposit Money")
8         print("3. Withdraw Money")
9         print("4. Exit")
10
11         choice = input("Please select an option (1-4): ")
12
13         if choice == '1':
14             print(f"Your current balance is: ${balance}")
15
16         elif choice == '2':
17             deposit = float(input("Enter amount to deposit: $"))
18             if deposit > 0:
19                 balance += deposit
20                 print(f"${deposit} deposited successfully.")
21             else:
22                 print("Invalid amount. Please enter a positive number.")
23
24         elif choice == '3':
25             withdraw = float(input("Enter amount to withdraw: $"))
26             if 0 < withdraw <= balance:
27                 balance -= withdraw
28                 print(f"${withdraw} withdrawn successfully.")
29             else:
30                 print("Invalid amount. Please check your balance and try again.")
31
32         elif choice == '4':
33             print("Thank you for using the ATM. Goodbye!")
34             break
35
36         else:
37             print("Invalid selection. Please choose a valid option.")
38 # Run the ATM menu
39 atm_menu()
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\sanja\OneDrive\Desktop\PYTHON TRAINING> & "C:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/.venv/Scripts/python.exe" "c:/Users/sanja/OneDrive/Desktop/PYTHON TRAINING/2.py"

Welcome to the ATM
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 1
Your current balance is: $1000

Welcome to the ATM
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4):
```

Observation:

- The program correctly simulates an ATM menu.

- User options are handled using conditional statements.
- The loop allows continuous transactions.
- AI-generated logic was verified and tested.