2403A51266

yashwanth

Batch-12

AI ASSISTED CODING

Question:

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

Lab Objectives:

- To explore AI-powered auto-completion features for core Python constructs.
- To analyze how AI suggests logic for class definitions, loops, and conditionals.
- To evaluate the completeness and correctness of code generated by AI assistants.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Use AI tools to generate and complete class definitions and methods.
- Understand and assess AI-suggested loops for iterative tasks.
- Generate conditional statements through prompt-driven suggestions.

- Critically evaluate AI-assisted code for correctness and clarity.

Task Description #1 (Classes – Employee Management)

- Task: Use AI to create an Employee class with attributes (name, id, salary) and a method to calculate yearly salary.
- Instructions:
  - Prompt AI to generate the Employee class.
  - Analyze the generated code for correctness and structure.
  - Ask AI to add a method to give a bonus and recalculate salary.

Expected Output #1:

- A class with constructor, display_details(), and calculate_bonus() methods.

Task Description #2 (Loops – Automorphic Numbers in a Range)

- Task: Prompt AI to generate a function that displays all Automorphic numbers between 1 and 1000 using a for loop.
- Instructions:

- Get AI-generated code to list Automorphic numbers using a for loop.
- Analyze the correctness and efficiency of the generated logic.
- Ask AI to regenerate using a while loop and compare both implementations.

Expected Output #2:

- Correct implementation that lists Automorphic numbers using both loop types, with explanation.

Task Description #3 (Conditional Statements – Online Shopping Feedback Classification)

- Task: Ask AI to write nested if-elif-else conditions to classify online shopping feedback as Positive, Neutral, or Negative based on a numerical rating (1–5).
- Instructions:
  - Generate initial code using nested if-elif-else.
  - Analyze correctness and readability.
  - Ask AI to rewrite using dictionary-based or match-case structure.

Expected Output #3:

- Feedback classification function with explanation and an alternative approach.

Task Description #4 (Loops – Prime Numbers in a Range)

- Task: Generate a function using AI that displays all prime numbers within a user-specified range (e.g., 1 to 500).
- Instructions:
    - Get AI-generated code to list all primes using a for loop.
    - Analyze the correctness and efficiency of the prime-checking logic.
    - Ask AI to regenerate an optimized version (e.g., using the square root method).

Expected Output #4:

- Python program that lists all prime numbers within a given range, with an optimized version and explanation.

Task Description #5 (Classes – Library System)

- Task: Use AI to build a Library class with methods to add_book(), issue_book(), and display_books().
- Instructions:
    - Generate Library class code using AI.
    - Analyze if methods handle edge cases (e.g., issuing unavailable books).

- Ask AI to add comments and documentation.

Expected Output #5:

- Library class with all methods, inline comments, and explanation.

Task 1 – Classes (Employee Management)

Prompt :
"Write a Python class Employee with attributes (name, id, salary). Add methods to display details, calculate yearly salary, and give a bonus."

Code:

```
class Employee:
    def __init__(self, name, emp_id, salary):
        self.name = name
        self.emp_id = emp_id
        self.salary = salary

    def display_details(self):
        print(f"Employee Name: {self.name}")
        print(f"Employee ID: {self.emp_id}")
        print(f"Monthly Salary: {self.salary}")

    def yearly_salary(self):
```

```
        return self.salary * 12

    def calculate_bonus(self, bonus_percent):
        bonus = (self.salary * bonus_percent) / 100
        self.salary += bonus
        return self.salary
emp = Employee("John", 101, 50000)
emp.display_details()
print("Yearly Salary:", emp.yearly_salary())
print("Updated Salary after 10% bonus:",
emp.calculate_bonus(10))
```

output:

```
Employee Name: John
Employee ID: 101
Monthly Salary: 50000
Yearly Salary: 600000
Updated Salary after 10% bonus: 55000.0
```

Task 2 – Loops (Automorphic Numbers)

Prompt:
"Generate a function in Python that prints all Automorphic numbers between 1 and 1000 using a for loop."

Code:

```
def automorphic_for():
    print("Automorphic numbers between 1 and 1000 (for loop):")
    for num in range(1, 1001):
        sq = num * num
        if str(sq).endswith(str(num)):
            print(num, end=" ")
automorphic_for()
automorphic_while()
```

output:

Automorphic numbers between 1 and 1000 (for loop):

1 5 6 25 76 376 625

Automorphic numbers between 1 and 1000 (while loop):

1 5 6 25 76 376 625

Task 3 – Conditional Statements (Feedback Classification)

Prompt:
"Write a Python function that classifies online shopping feedback (1–5) using nested if-elif-else: Positive, Neutral, or Negative."

Code:

```
def feedback_classification(rating):
```

```
    if rating == 5:
        return "Positive Feedback"
    elif rating == 4:
        return "Positive Feedback"
    elif rating == 3:
        return "Neutral Feedback"
    elif rating == 2:
        return "Negative Feedback"
    elif rating == 1:
        return "Negative Feedback"
    else:
        return "Invalid Rating"
print("Rating: 5 →", feedback_classification(5))
print("Rating: 3 →", feedback_classification(3))
print("Rating: 1 →", feedback_dict(1))
print("Rating: 7 →", feedback_dict(7))
```

output:

```
Rating: 5 → Positive Feedback
Rating: 3 → Neutral Feedback
Rating: 1 → Negative Feedback
Rating: 7 → Invalid Rating
```

Task 4 – Loops (Prime Numbers)

Prompt:

"Write a Python function that prints all prime numbers between 1 and 500 using a for loop."

Code:

```
def primes_in_range(start, end):
    print(f"Prime numbers between {start} and {end}:")
    for num in range(start, end + 1):
        if num > 1:
            for i in range(2, num):
                if num % i == 0:
                    break
            else:
                print(num, end=" ")
primes_in_range(1, 50)
optimized_primes(1, 50)
```

output:

```
Prime numbers between 1 and 50:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

Optimized prime numbers between 1 and 50:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

Task 5 – Classes (Library System)

Prompt:

"Create a Python class Library with methods add_book(), issue_book(), and display_books(). Include comments and handle cases where a book is not available."

Code:

```
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book_name):
        """Adds a book to the library collection"""
        self.books.append(book_name)
        print(f'"{book_name}" has been added to the library.')

    def display_books(self):
        """Displays all books in the library"""
        if self.books:
            print("Books available in the library:")
            for book in self.books:
                print(f"- {book}")
        else:
```

```
            print("No books available in the library.")


    def issue_book(self, book_name):
        """Issues a book if available, else shows an error"""
        if book_name in self.books:
            self.books.remove(book_name)
            print(f'"{book_name}" has been issued.')
        else:
            print(f'Sorry, "{book_name}" is not available.')
lib = Library()
lib.add_book("Python Programming")
lib.add_book("Data Science")
lib.display_books()
lib.issue_book("Python Programming")
lib.issue_book("Machine Learning")
lib.display_books()
```

output:

"Python Programming" has been added to the library.

"Data Science" has been added to the library.

Books available in the library:

- Python Programming

- Data Science

"Python Programming" has been issued.

Sorry, "Machine Learning" is not available.

Books available in the library:

- Data Science