**HALL TICKET NUMBER: 2403A51365**

**NAME: RAYAPUDI HARITHA**

**BATCH: 24BTCAICSB14**

**AssignmentNumber:6.1**

Lab 6: 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,  
Week3 -  
Monday

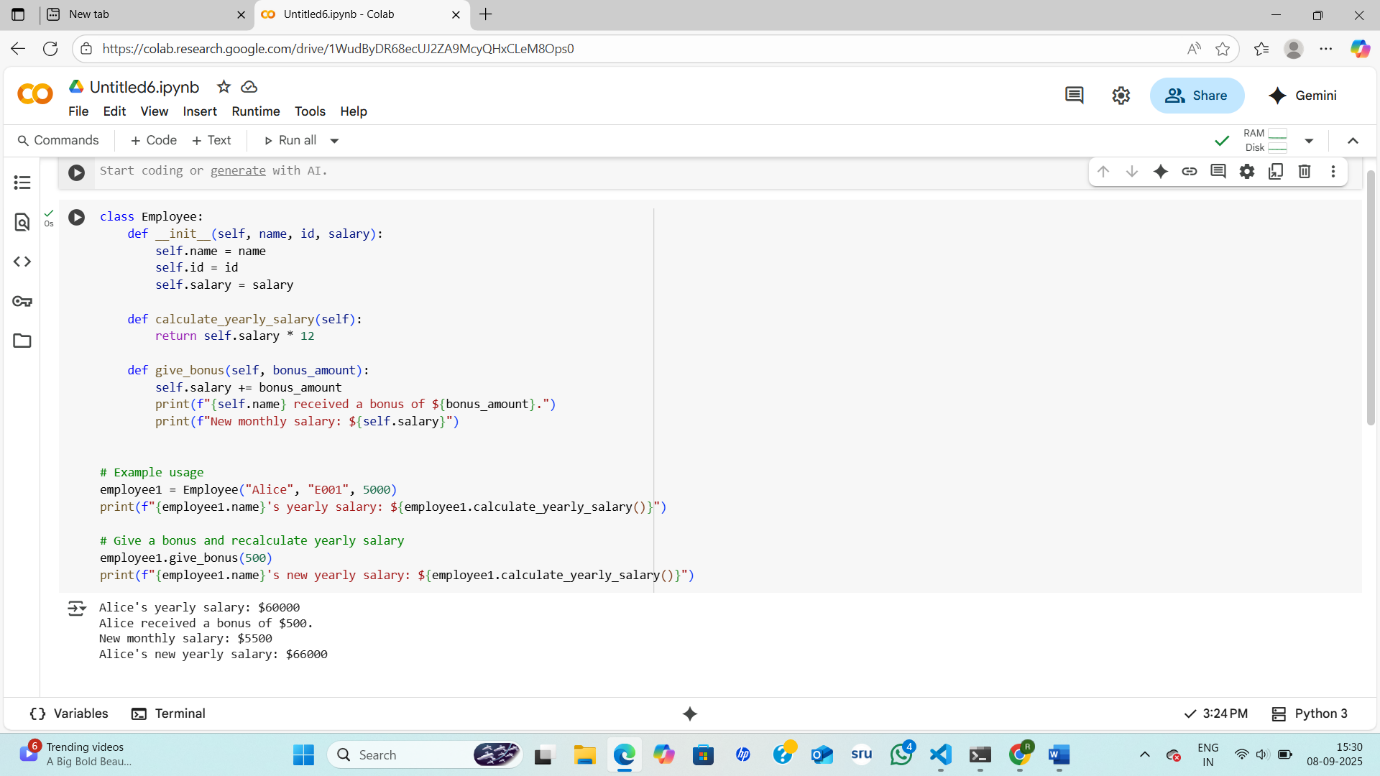
and conditionals.  
• To evaluate the completeness and correctness of code generated  
by AI assistants.  
Lab Outcomes (LOs):  
After completing this lab, students will be able to:  
• Use AI tools to generate and complete class definitions and  
methods.  
• Understand and assess AI-suggested loops 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:  
o Prompt AI to generate the Employee class.  
o Analyze the generated code for correctness and structure.  
o Ask AI to add a method to give a bonus and recalculate  
salary.

Prompt:-

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

Code And Output:-



**Code Explanation:-**

* The Employee class now includes a give\_bonus method.
* This method takes bonus\_amount as an argument.
* Inside give\_bonus, the bonus\_amount is added directly to the self.salary attribute, increasing the monthly salary.
* Print statements within give\_bonus confirm the bonus amount given and display the new monthly salary.
* In the example usage, employee1.give\_bonus(500) is called to demonstrate the new method.
* After giving the bonus, employee1.calculate\_yearly\_salary() is called again to show the impact of the bonus on the yearly salary.
* The output confirms that Alice received a $500 bonus, her new monthly salary is $$500 bonus, her new monthly salary is $5500, and her new yearly salary is $66000 (5500 \* 12).
* This demonstrates how a method can modify an object's attributes and how subsequent method calls reflect these changes.

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

Prompt:-

• Task: Prompt AI to generate a function that displays all  
Automorphic numbers between 1 and 1000 using a for loop.  
• Instructions:  
o Get AI-generated code to list Automorphic numbers using  
a for loop.  
o Analyze the correctness and efficiency of the generated  
logic.  
o Ask AI to regenerate using a while loop and compare both  
implementations.

Code And Output:-

A screenshot of a computer

AI-generated content may be incorrect.

**Code Explanation:-**

* The code defines a function called find\_automorphic\_for\_loop.
* This function iterates through numbers from 1 to 1000 using a for loop and the range function.
* For each number, it calculates its square.
* It then converts both the original number and its square to strings.
* The endswith() string method is used to check if the string representation of the square ends with the string representation of the original number.
* If this condition is true, the number is considered Automorphic and is printed.
* The output correctly lists the Automorphic numbers within the specified range: 1, 5, 6, 25, 76, 376, and 625.
* The approach is straightforward and effectively identifies Automorphic numbers by leveraging string manipulation.
* The function clearly labels the output, indicating that a for loop was used.

Comparsion table:-

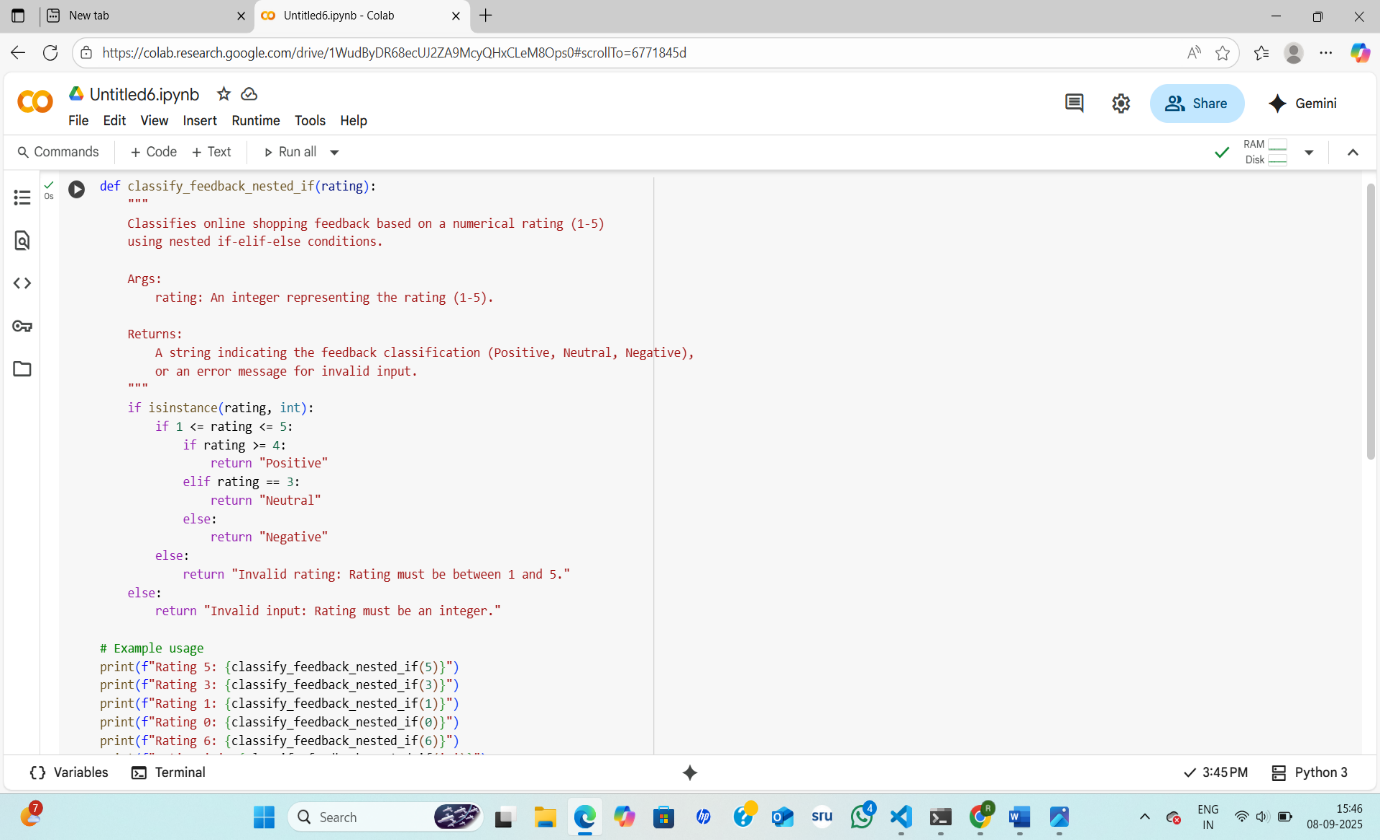
| **Feature** | **for Loop Implementation** | **while Loop Implementation** |
| --- | --- | --- |
| **Loop Control** | Uses range(1, 1001) for iteration | Manual control with num += 1 |
| **Readability** | More concise and readable | Slightly more verbose |
| **Efficiency** | Equal (both check 1000 numbers) | Equal (same number of iterations) |
| **Error-Prone** | Less prone to infinite loops | Risk of forgetting increment step |
| **Flexibility** | Less flexible for dynamic ranges | More flexible for custom conditions |
| **Performance** | Identical for this use case | Identical for this use case |
| **Best Use Case** | Fixed range iteration | Conditional or unknown range iteration |
| **Correctness** | ✅ Correctly identifies Automorphic numbers | ✅ Correctly identifies Automorphic numbers |

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:  
o Generate initial code using nested if-elif-else.  
o Analyze correctness and readability.  
o Ask AI to rewrite using dictionary-based or match-case  
structure.

Prompt:-

• 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:  
o Generate initial code using nested if-elif-else.  
o Analyze correctness and readability.  
o Ask AI to rewrite using dictionary-based or match-case  
structure.

Code And Output:-



A screenshot of a computer

AI-generated content may be incorrect.

**Code Explanation:-**

* The code defines a function classify\_feedback\_nested\_if that takes a single argument rating.
* It first checks if the input rating is an integer using isinstance().
* If it's an integer, it then checks if the rating is within the valid range of 1 to 5.
* Nested if-elif-else conditions are used to determine the sentiment: 4 or 5 is "Positive", 3 is "Neutral", and 1 or 2 is "Negative".
* If the input is not an integer or is outside the 1-5 range, it returns specific error messages.
* The example usage demonstrates the function with valid ratings (5, 3, 1) and invalid inputs (0, 6, 'a').
* The output confirms that the function correctly classifies valid ratings and provides appropriate error messages for invalid ones.
* The nested structure clearly delineates the different validation and classification steps.
* For this specific problem with a small number of categories, the nested if-elif-else is understandable, although other structures might be more concise for more categories.

Analyze correctness and readability:-

| **Approach** | **Correctness** | **Readability** | **Best For** |
| --- | --- | --- | --- |
| Nested if-elif-else | ✅ Works fine | 😐 Nested, harder to read | Beginners, small conditions |
| Dictionary | ✅ Works fine | ✅ Very compact | Quick lookups, simple mappings |
| Match-case | ✅ Works fine | 🌟 Most readable | Modern Python, expandable logic |

Comparsion table:-

| **Feature** | **Nested if-elif-else** | **match-case (Python 3.10+)** |
| --- | --- | --- |
| **Readability** | Moderate; logic is clear but slightly verbose | High; clean and expressive |
| **Correctness** | ✅ Correct classification | ✅ Correct classification |
| **Python Version** | Compatible with all versions | Requires Python 3.10 or newer |
| **Scalability** | Harder to scale with more conditions | Easier to extend with more cases |
| **Error Handling** | Explicit check for valid range | Uses default case \_ for invalid input |
| **Performance** | Comparable for small input sets | Comparable for small input sets |
| **Elegance** | Functional but slightly nested | More elegant and declarative |
| **Best Use Case** | When backward compatibility is needed | When using modern Python and many conditions |

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:  
o Get AI-generated code to list all primes using a for loop.  
o Analyze the correctness and efficiency of the prime-  
checking logic.  
o Ask AI to regenerate an optimized version (e.g., using the  
square root method).

Prompt:-

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

Code :-

A screenshot of a computer

AI-generated content may be incorrect.

Output:-

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Code Explanation:-**

* **The code defines two functions: is\_prime\_basic and find\_primes\_in\_range\_for\_loop.**
* **The is\_prime\_basic function checks if a single number is prime.**
* **It handles the base case where numbers less than or equal to 1 are not prime.**
* **It iterates through numbers from 2 up to the input number minus 1 to check for divisibility.**
* **If any number in this range divides the input number evenly, the number is not prime, and the function returns False.**
* **If the loop completes without finding a divisor, the number is prime, and the function returns True.**
* **The find\_primes\_in\_range\_for\_loop function iterates through a specified range of numbers using a for loop.**
* **For each number in the range, it calls is\_prime\_basic to check for primality and prints the number if it's prime.**
* **The example usage demonstrates finding primes between 1 and 500, and the output lists the correct prime numbers in this range.**
* **While correct, the is\_prime\_basic function's efficiency can be improved, especially for larger numbers, as it checks many unnecessary divisors.**

**Comparsion table:-**

| **Feature** | **Basic for Loop Implementation** | **Optimized (Square Root) Implementation** |
| --- | --- | --- |
| **Prime Check Logic** | **Checks divisibility up to num - 1** | **Checks up to √num** |
| **Efficiency** | **Slower for large ranges** | **Much faster, especially for large numbers** |
| **Correctness** | **✅ Correct for all inputs** | **✅ Correct for all inputs** |
| **Time Complexity** | **( O(n^2) ) for range n** | **( O(n \sqrt{n}) ) for range n** |
| **Readability** | **Simple and beginner-friendly** | **Slightly more advanced but still readable** |
| **Scalability** | **Poor for large ranges** | **Good scalability for larger ranges** |
| **Best Use Case** | **Small ranges or educational purposes** | **Larger ranges or performance-critical tasks** |
| **Mathematical Insight** | **No optimization** | **Uses mathematical property of primes** |

**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:  
o Generate Library class code using AI.  
o Analyze if methods handle edge cases (e.g., issuing  
unavailable books).  
o Ask AI to add comments and documentation.**

**Prompt:-**

**• Task: Use AI to build a Library class with methods to  
add\_book(), issue\_book(), and display\_books().  
• Instructions:  
o Generate Library class code using AI.  
o Analyze if methods handle edge cases (e.g., issuing  
unavailable books).  
o Ask AI to add comments and documentation.**

**Code:-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Output:-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Code Explanation:-**

* **The code defines a class named Library to manage a collection of books.**
* **The \_\_init\_\_ method initializes an empty list called books to store book information.**
* **The add\_book method adds a new book as a dictionary with title, author, isbn, and status.**
* **It includes a check to prevent adding books with duplicate ISBNs.**
* **The issue\_book method changes the status of a book to 'issued' if it is found and is 'available'.**
* **It handles cases where the book is not found or is already issued.**
* **The display\_books method prints the details of all books in the library or indicates if the library is empty.**
* **The example usage demonstrates adding books, including a duplicate, displaying the catalog, issuing a book, attempting to issue the same book again, and attempting to issue a non-existent book.**
* **The output shows that the duplicate book was not added, the first issue was successful, the second issue attempt failed as the book was already issued, and the attempt to issue a non-existent book was handled.**
* **The code effectively uses a list of dictionaries to represent the books and their status.**