

AI ASSISTANT CODING

ASSIGNMENT-6

B.Sri Laxmi Gayathri

2303A52033

Batch-38

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, 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 – Data Validation)

- Prompt AI to generate a Student class with attributes: name, roll_no, and marks. Add a method is_pass() that returns whether the student has passed (marks \geq 40).

Expected Output-1

- Python class with constructor and validation logic

- Clear method implementation returning pass/fail status
- Students analyze correctness and clarity of class design

Prompt:

generate a Student class with attributes: name, roll_no, and marks. Add a method is_pass() that returns whether the student has passed (marks \geq 40). the expected output Clear method implementation returning pass/fail status , Students analyze correctness and clarity of class design the input should be from user side

```
class Student:
    def __init__(self, name, roll_no, marks):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks
    def is_pass(self):
        """
        Returns True if the student's marks are 40 or greater, indicating a pass
        Otherwise, returns False.
        """
        return self.marks >= 40
    def display_student_info(self):
        print(f"\nStudent Name: {self.name}")
        print(f"Roll Number: {self.roll_no}")
        print(f"Marks: {self.marks}")
# Get student details from the user
student_name = input("Enter student's name: ")
student_roll_no = input("Enter student's roll number: ")
student_marks = int(input("Enter student's marks: "))
# Create a Student object
student1 = Student(student_name, student_roll_no, student_marks)
# Display student information
student1.display_student_info()
# Check if the student passed
if student1.is_pass():
    print("Result: Pass")
else:
    print("Result: Fail")

Enter student's name: gayathri
Enter student's roll number: 2303a52033
Enter student's marks: 90

Student Name: gayathri
Roll Number: 2303a52033
Marks: 90
Result: Pass
```

Explanation:

The program creates a Student class to store a student's name, roll number, and marks. It takes input from the user, checks if the marks are 40 or above using the is_pass() method, and then displays the student details along with the pass or fail result.

Task Description-2 (Loops – Pattern Generation)

- Ask AI to generate a function that prints a right-angled triangle

star pattern using a for loop. Then regenerate the same pattern using a while loop.

Expected Output-2

- Correct pattern output using both loop types
- Logical loop structure with proper conditions

Prompt:

generate a function that prints a right-angled triangle star pattern using a for loop. Then regenerate the same pattern using a while loop. the input should be user based

```
def print_right_triangle_for(n):  
    """  
    Prints a right-angled triangle star pattern using a for loop.  
  
    Args:  
        n (int): The number of rows for the triangle.  
    """  
    print(f"\nRight-angled triangle pattern (for loop) for n={n}:")  
    for i in range(1, n + 1):  
        print(" " * (n - i) + "*" * i)  
  
    # Get user input for n  
    n_for_loop = int(input("Enter the number of rows for the for loop triangle: "))  
  
    # Example usage for the for loop function  
    print_right_triangle_for(n_for_loop)  
  
... Enter the number of rows for the for loop triangle: 5  
  
Right-angled triangle pattern (for loop) for n=5:  
*  
**  
***  
****  
*****
```

```

def print_right_triangle_while(n):
    """
    Prints a right-angled triangle star pattern using a while loop.

    Args:
        n (int): The number of rows for the triangle.
    """
    print(f"\nRight-angled triangle pattern (while loop) for n={n}:")
    i = 1
    while i <= n:
        print(" " * (n - i) + "*" * i)
        i += 1

# Get user input for n
n_while_loop = int(input("Enter the number of rows for the while loop triangle: "))

# Example usage for the while loop function
print_right_triangle_while(n_while_loop)

```

... Enter the number of rows for the while loop triangle: 5

```

Right-angled triangle pattern (while loop) for n=5:
*
**
***
****
*****

```

Explanation:

This program defines two separate functions to print a right-angled triangle star pattern: one using a for loop and the other using a while loop. In both cases, the number of rows is taken as input from the user, ensuring user-based input handling. Each loop increases the number of stars per row logically to form the triangle shape. The output confirms that the same correct pattern is generated using both loop structures.

Task Description-3 (Conditional Statements – Number Analysis)

- Ask AI to write a function that checks whether a given number is positive, negative, or zero using if-elif-else. Test the function with multiple inputs.

Expected Output-3

- Function correctly classifies numbers
- Proper handling of all conditions
- Students analyze decision logic

Prompt:

generate a code using a function checks whether a given number is positive, negative, or zero using if-elif-else. Test the function with multiple inputs, the inputs should be from user side

```
def check_number_type(number):  
    """  
    Checks if a given number is positive, negative, or zero.  
  
    Args:  
        number (int or float): The number to be checked.  
    """  
    if number > 0:  
        print(f"The number {number} is positive.")  
    elif number < 0:  
        print(f"The number {number} is negative.")  
    else:  
        print(f"The number {number} is zero.")  
  
    # Get input from the user  
    user_input = float(input("Enter a number: "))  
  
    # Test the function with the user's input  
    check_number_type(user_input)  
  
    # You can test with other inputs if you run this cell again  
    # Example with zero:  
    # check_number_type(0)  
    # Example with a negative number:  
    # check_number_type(-5)  
    # Example with a positive number:  
    # check_number_type(10.5)  
  
... Enter a number: -5  
The number -5.0 is negative.
```

Explanation:

This program defines a function that uses if-elif-else conditions to check whether a given number is positive, negative, or zero. The number is taken as input from the user, making the program interactive. Based on the value entered, the function prints the appropriate result.

Task Description-4 (Nested Conditionals)

- Generate a function `check_discount(age, is_member)` that determines discount eligibility:
 - Age $\geq 60 \rightarrow$ Senior discount

- Member → Additional discount

Use nested if statements.

Expected Output-4

- Python code using nested conditionals
- Clear explanation of decision flow

Prompt:

Generate a function `check_discount(age, is_member)` that determines discount eligibility: Age ≥ 60 : Senior discount , Member : Additional discount, Use nested if statements.the input should be from user sidw

```
def check_discount(age, is_member):
    """
    Determines discount eligibility based on age and membership status using nested if statements.

    Args:
        age (int): The customer's age.
        is_member (bool): True if the customer is a member, False otherwise.
    """
    print(f"\nChecking discount for Age: {age}, Member: {is_member}")
    if age >= 60:
        print("Eligible for Senior discount.")
        if is_member:
            print("Eligible for Additional member discount (Senior).")
        else:
            if is_member:
                print("Eligible for Member discount.")
            else:
                print("No discount applicable.")

    # Get input from the user
    user_age = int(input("Enter customer's age: "))
    user_is_member_str = input("Is the customer a member? (yes/no): ").lower()

    # Convert string input to boolean
    user_is_member = True if user_is_member_str == 'yes' else False

    # Test the function with user's input
    check_discount(user_age, user_is_member)
```

```
... Enter customer's age: 56
Is the customer a member? (yes/no): yes

Checking discount for Age: 56, Member: True
Eligible for Member discount.
```

Explanation:

This program defines a `check_discount` function that uses nested if statements to determine discount eligibility based on the customer's age and membership status. The user provides their age and whether they are a member as input. If the age is 60 or above, the customer receives a senior discount, and members may get an additional discount. The function clearly displays the applicable discount based on these conditions.

Task Description-5 (Class – Mathematical Opera)

- Ask AI to create a Circle class with methods to calculate area () and circumference () given the radius.

Expected Output-5

- Correct mathematical computation
- Well-structured class with methods
- Code explanation provided

Prompt:

create a Circle class with methods to calculate area () and circumference () given the radius. include the code explanation, the input should be user based

```
import math
class Circle:
    def __init__(self, radius):
        if radius <= 0:
            raise ValueError("Radius must be a positive number.")
        self.radius = radius
    def area(self):
        """
        Calculates and returns the area of the circle.
        Area = pi * r^2
        """
        return math.pi * (self.radius ** 2)
    def circumference(self):
        """
        Calculates and returns the circumference of the circle.
        Circumference = 2 * pi * r
        """
        return 2 * math.pi * self.radius
# Get radius from the user
try:
    user_radius = float(input("Enter the radius of the circle: "))

    # Create a Circle object
    my_circle = Circle(user_radius)

    # Calculate and display area and circumference
    print(f"\nCircle with radius: {my_circle.radius}")
    print(f"Area: {my_circle.area():.2f}")
    print(f"Circumference: {my_circle.circumference():.2f}")
except ValueError as e:
    print(f"Error: {e}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")

... Enter the radius of the circle: 2

Circle with radius: 2.0
Area: 12.57
Circumference: 12.57
```

Explanation:

This program defines a Circle class that takes the radius as input from the user and validates that it is a positive value. It includes methods to calculate the area and circumference using standard mathematical formulas. A Circle object is created with the user-entered radius, and the results are displayed. Error handling is used to manage invalid inputs gracefully.