SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
Course Coordinator Name	Dr. Rishabh Mittal		
Instructor(s)Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpulasumalatha Mr. Bingi Raju		
Course Code	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	Week3 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	23CSBTB01 To 23CSBTB52
AssignmentNumber:6.2(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals **Lab Objectives:** • To explore AI-powered auto-completion features for core Python	Week3 - Tuesday

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 ≥ 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 Python class `Student` with attributes name, roll_no, and marks. Add a method `is_pass()` that returns whether the student has passed (marks ≥ 40).**
- **CODE**

```
class Student:
    def __init__(self, name, roll_no, marks):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks

    def is_pass(self):
        return self.marks >= 40
s1 = Student("Mukesh", 101, 75)
s2 = Student("Ravi", 102, 35)

print(s1.name, "Pass Status:", s1.is_pass())
print(s2.name, "Pass Status:", s2.is_pass())

```

CODE O/P
Mukesh Pass Status: True
Ravi Pass Status: False

EXPLANATION

- The constructor initializes student details.
- The `is_pass()` method checks if marks are ≥ 40.
- Returns `True` if passed, otherwise `False`.
- The logic is clear and correctly implemented.

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 to print a right-angled triangle star pattern using a for loop and then using a while loop.**
- **CODE**

```
def triangle_for(n):
    for i in range(1, n+1):
        print("*" * i)

n=5
print(triangle_for(n))
```

- **CODE O/P**

```
*
* *
* * *
* * * *
* * * * *
```

- **EXPLANATION**

- The function `triangle_for(n)` uses a **for loop** to iterate from 1 to `n` and prints stars in increasing order.
- In each iteration, `"*"` is multiplied by `i`, which creates a right-angled triangle pattern.
- Since the function does not return any value, `print(triangle_for(n))` prints the pattern first and then prints `None`.

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**
- Write a function that checks whether a number is positive, negative, or zero using if-elif-else.
- **CODE**

```
def check_number(num):
    if num > 0:
        return "Positive"
    elif num < 0:
        return "Negative"
    else:
        return "Zero"


# Testing
print(check_number(10))
print(check_number(-5))
print(check_number(0))
```

- **CODE O/P**
- Positive
- Negative
- Zero

EXPLANATION

- Uses `if-elif-else` decision structure.
- Covers all possible conditions.
- Correct logical classification.

Task Description-4 (Nested Conditionals)

- Generate a function check_discount(age, is_member) that determines discount eligibility:
- Age ≥ 60 → 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) using nested if statements.
- **CODE**

```
def check_discount(age, is_member):
    if age >= 60:
        print("Eligible for Senior Discount")
        if is_member:
            print("Eligible for Additional Member Discount")
    else:
        if is_member:
            print("Eligible for Member Discount Only")
        else:
            print("No Discount")


# Testing
check_discount(65, True)
check_discount(30, True)
check_discount(25, False)
```

- **CODE O/P**
- Eligible for Senior Discount
- Eligible for Additional Member Discount
- Eligible for Member Discount Only
- No Discount
- **EXPLANATION**
- Outer `if` checks age.
- Inner `if` checks membership.
- Proper nested decision flow implemented.

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.
- **CODE**

```
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

    def circumference(self):
        return 2 * math.pi * self.radius


# Testing
c1 = Circle(7)

print("Area:", c1.area())
print("Circumference:", c1.circumference())
```

- **CODE O/P**
- **Area: 153.93804002589985**
- **Circumference: 43.982297150257104**

EXPLANATION

- Uses mathematical formulas:
- Area = πr^2
- Circumference = $2\pi r$
- Proper class structure with methods.
- Uses `math.pi` for accurate calculation.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.
