SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s) Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week10 - Thursday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	
AssignmentNumber:19.4(Present assignment number)/**24**(Total number of assignments)			

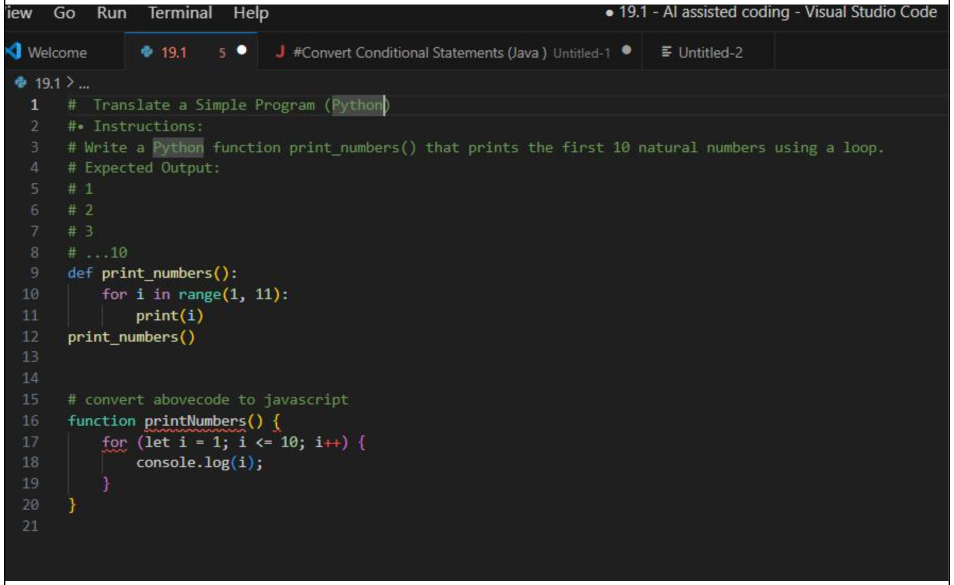
Q.No.	**Question**	***Expected Time to complete***
1	**Lab 19 – Code Translation: Converting Between Programming Languages** **Lab Objectives:** • Understand how AI tools can assist in translating code between different programming languages.	Week10 - Thursday

- Learn to verify correctness and functionality after translation.
- Explore syntactic and semantic differences between languages (e.g., Python, Java, C++).
- Practice debugging and optimizing AI-translated code.

Task 1: Translate a Simple Program (Python → JavaScript)

- **Instructions:**
    - Write a Python function print_numbers() that prints the first 10 natural numbers using a loop.
    - Translate the function into JavaScript as a reusable function printNumbers().
    - Call the function in both languages to display results.
- **Expected Output:**
- **1**
- **2**
- **3**
- **...10**

Code:


iew Go Run Terminal Help
● 19.1 - AI assisted coding - Visual Studio Code
Welcome
19.1 5
#Convert Conditional Statements (Java) Untitled-1
Untitled-2
19.1 > ...

```
#  Translate a Simple Program (Python)
#• Instructions:
# Write a Python function print_numbers() that prints the first 10 natural numbers using a loop.
# Expected Output:
# 1
# 2
# 3
# ...10
def print_numbers():
    for i in range(1, 11):
        print(i)
print_numbers()


# convert abovecode to javascript
function printNumbers() {
    for (let i = 1; i <= 10; i++) {
        console.log(i);
    }
}

```

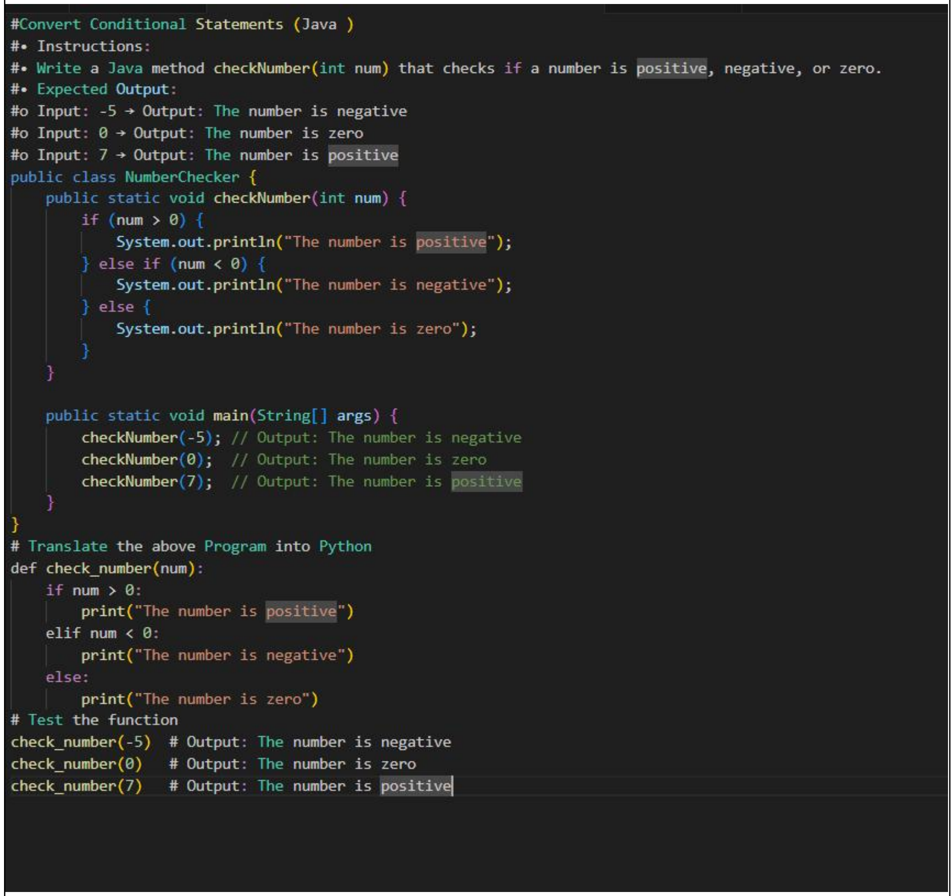
Task 2: Convert Conditional Statements (Java → Python)

- **Instructions:**
    - Write a Java method checkNumber(int num) that checks if a number is positive, negative, or zero.

- Translate the method into a Python function check_number(num).
- Call the function/method with different inputs and compare outputs.

- **Expected Output:**
    - Input: -5 → Output: The number is negative
    - Input: 0 → Output: The number is zero
    - Input: 7 → Output: The number is positive

Code:



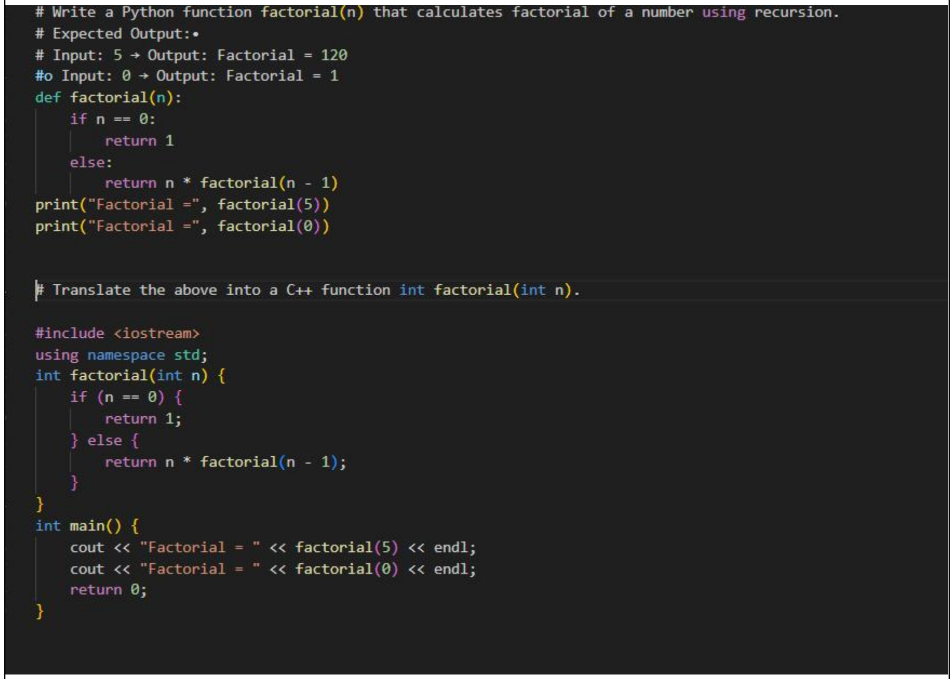
```
#Convert Conditional Statements (Java )
#• Instructions:
#• Write a Java method checkNumber(int num) that checks if a number is positive, negative, or zero.
#• Expected Output:
#o Input: -5 → Output: The number is negative
#o Input: 0 → Output: The number is zero
#o Input: 7 → Output: The number is positive
public class NumberChecker {
    public static void checkNumber(int num) {
        if (num > 0) {
            System.out.println("The number is positive");
        } else if (num < 0) {
            System.out.println("The number is negative");
        } else {
            System.out.println("The number is zero");
        }
    }

    public static void main(String[] args) {
        checkNumber(-5); // Output: The number is negative
        checkNumber(0);  // Output: The number is zero
        checkNumber(7);  // Output: The number is positive
    }
}
# Translate the above Program into Python
def check_number(num):
    if num > 0:
        print("The number is positive")
    elif num < 0:
        print("The number is negative")
    else:
        print("The number is zero")
# Test the function
check_number(-5)  # Output: The number is negative
check_number(0)   # Output: The number is zero
check_number(7)   # Output: The number is positive
```

Task 3: Translate Recursive Function (Python → C++)

- **Instructions:**
    - Write a Python function factorial(n) that calculates factorial of a number using recursion.
    - Translate the same into a C++ function int factorial(int n).
    - Call the function in both languages with inputs 5 and 0.
- **Expected Output:**
    - **Input: 5 → Output: Factorial = 120**

- ○ **Input: 0 → Output: Factorial = 1**

Code:

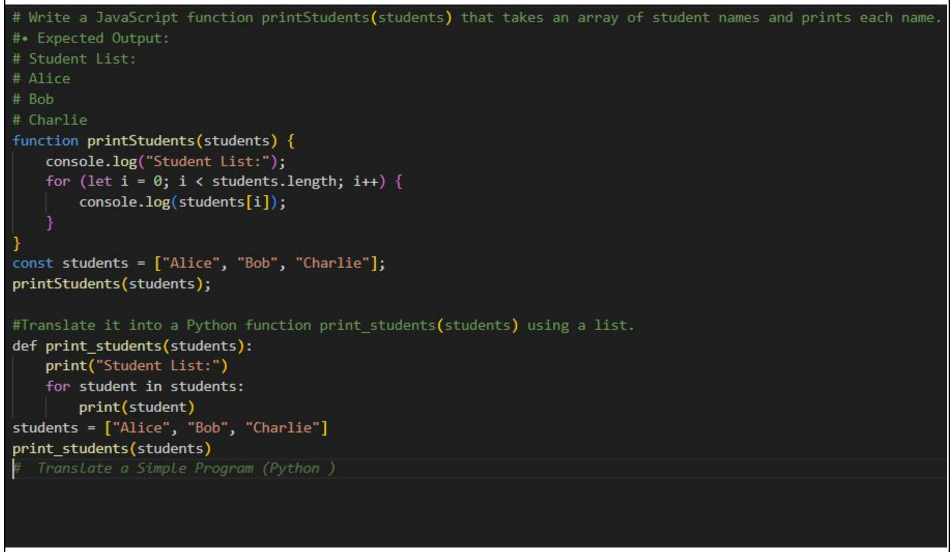


```
# Write a Python function factorial(n) that calculates factorial of a number using recursion.
# Expected Output:•
# Input: 5 → Output: Factorial = 120
#o Input: 0 → Output: Factorial = 1
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print("Factorial =", factorial(5))
print("Factorial =", factorial(0))


# Translate the above into a C++ function int factorial(int n).

#include <iostream>
using namespace std;
int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
int main() {
    cout << "Factorial = " << factorial(5) << endl;
    cout << "Factorial = " << factorial(0) << endl;
    return 0;
}
```

Task 4: Data Structures with Functions (JavaScript → Python)

- **Instructions:**
    - Write a JavaScript function printStudents(students) that takes an array of student names and prints each name.
    - Translate it into a Python function print_students(students) using a list.
    - Test both functions with sample student names.
- **Expected Output:**
- Student List:
- Alice
- Bob
- Charlie

Code:



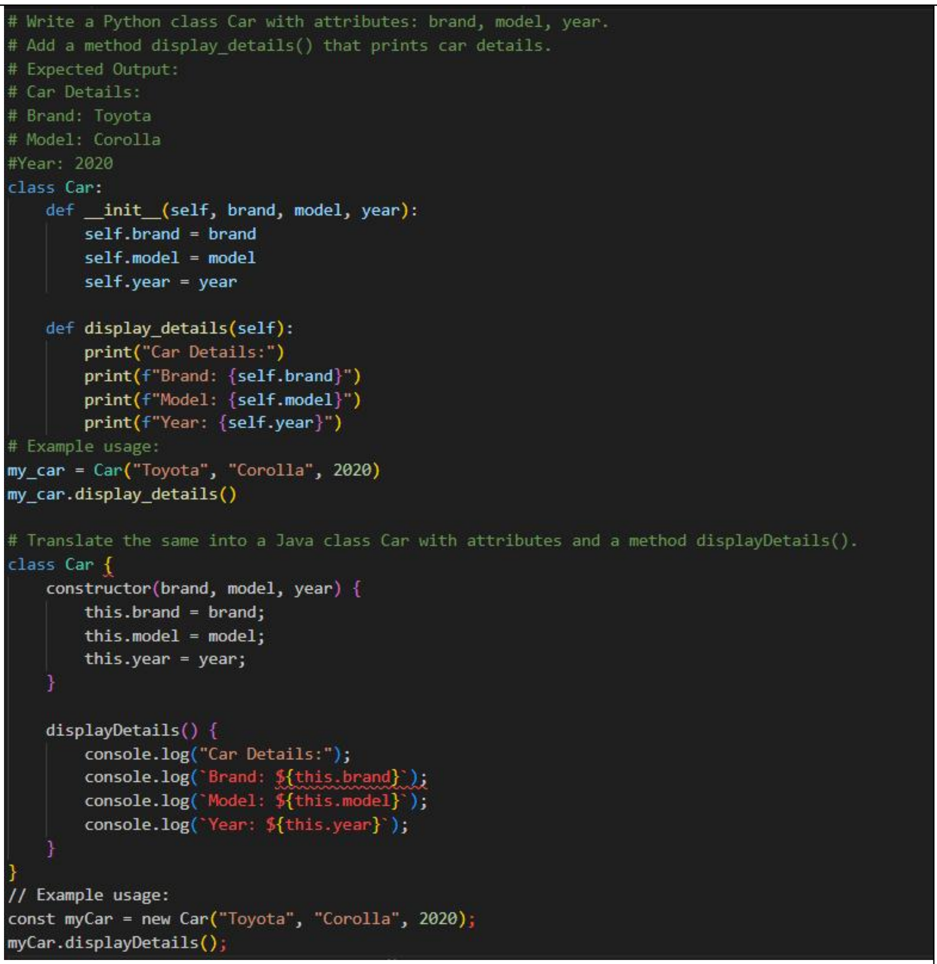
```
# Write a JavaScript function printStudents(students) that takes an array of student names and prints each name.
#• Expected Output:
# Student List:
# Alice
# Bob
# Charlie
function printStudents(students) {
    console.log("Student List:");
    for (let i = 0; i < students.length; i++) {
        console.log(students[i]);
    }
}
const students = ["Alice", "Bob", "Charlie"];
printStudents(students);

#Translate it into a Python function print_students(students) using a list.
def print_students(students):
    print("Student List:")
    for student in students:
        print(student)
students = ["Alice", "Bob", "Charlie"]
print_students(students)
#  Translate a Simple Program (Python )
```

Task 5: Class & Object Translation (Python → Java)

- **Instructions:**
  1. Write a **Python class** Car with attributes: brand, model, year.
  2. Add a **method** display_details() that prints car details.
  3. Translate the same into a **Java class** Car with attributes and a **method** displayDetails().
  4. Create an object in both languages and call the method.
- **Expected Output:**
- Car Details:
- Brand: Toyota
- Model: Corolla

  Year: 2020

Code:

```
# Write a Python class Car with attributes: brand, model, year.
# Add a method display_details() that prints car details.
# Expected Output:
# Car Details:
# Brand: Toyota
# Model: Corolla
#Year: 2020
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def display_details(self):
        print("Car Details:")
        print(f"Brand: {self.brand}")
        print(f"Model: {self.model}")
        print(f"Year: {self.year}")
# Example usage:
my_car = Car("Toyota", "Corolla", 2020)
my_car.display_details()

# Translate the same into a Java class Car with attributes and a method displayDetails().
class Car {
    constructor(brand, model, year) {
        this.brand = brand;
        this.model = model;
        this.year = year;
    }

    displayDetails() {
        console.log("Car Details:");
        console.log(`Brand: ${this.brand}`);
        console.log(`Model: ${this.model}`);
        console.log(`Year: ${this.year}`);
    }
}
// Example usage:
const myCar = new Car("Toyota", "Corolla", 2020);
myCar.displayDetails();
```

☑ Deliverables (For All Tasks)

1. AI-generated prompts for code and test case generation.
2. At least 3 assert test cases for each task.
3. AI-generated initial code and execution screenshots.
4. Analysis of whether code passes all tests.
5. Improved final version with inline comments and explanation.
6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.