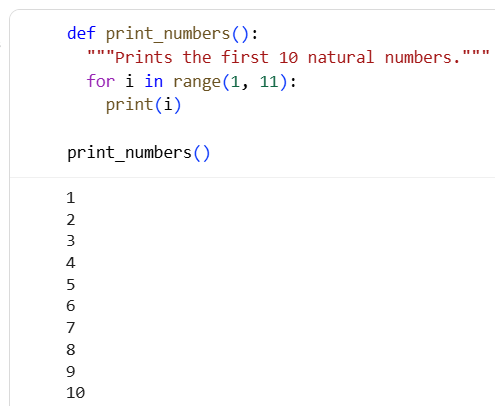
Assingment-19

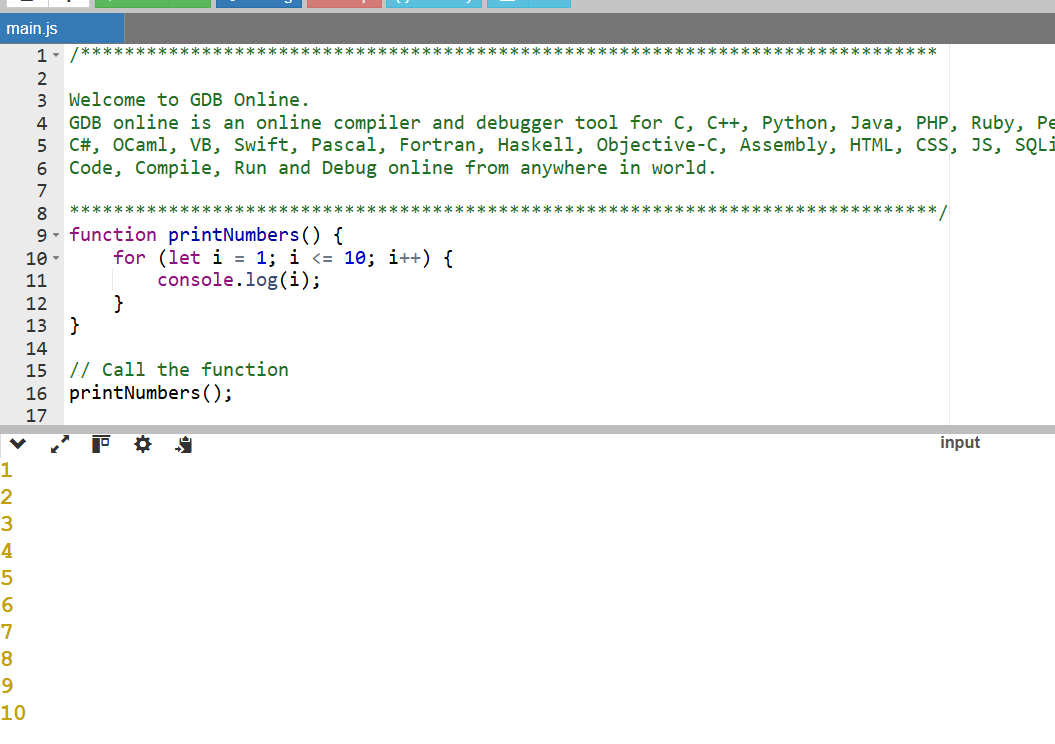
**Task 1: Translate a Simple Program (Python → JavaScript)**

* **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**

**Python programme**

****

**JAVASCRIPT code**

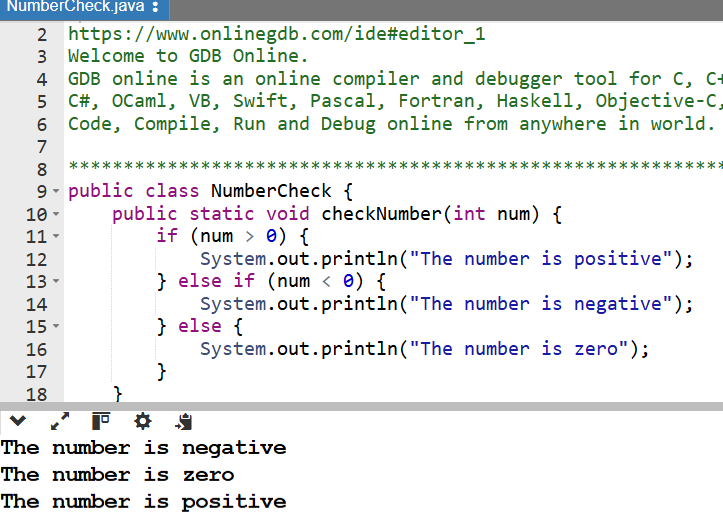


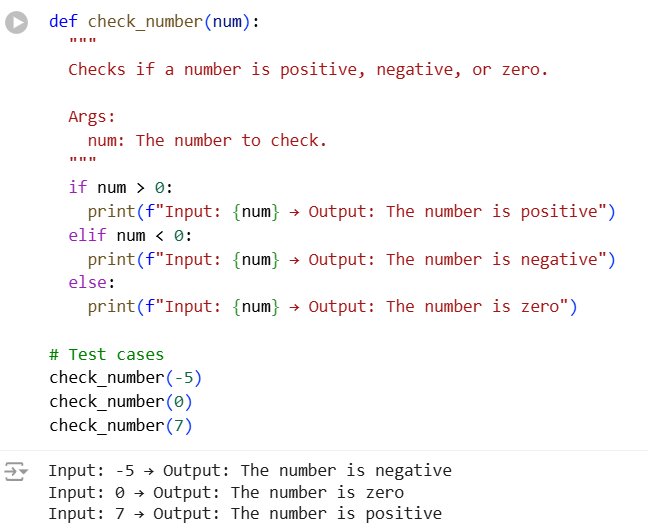
Explaination:

In Python, the print\_numbers() function uses a for loop with range(1, 11) to print the first 10 natural numbers. In JavaScript, the equivalent printNumbers() function uses a for loop starting from 1 to 10 with console.log(i) to display each number. Both functions demonstrate basic loop control and output, making them reusable for printing sequences in their respective languages.

**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:
  1. Input: -5 → Output: The number is negative
  2. Input: 0 → Output: The number is zero
  3. Input: 7 → Output: The number is positive



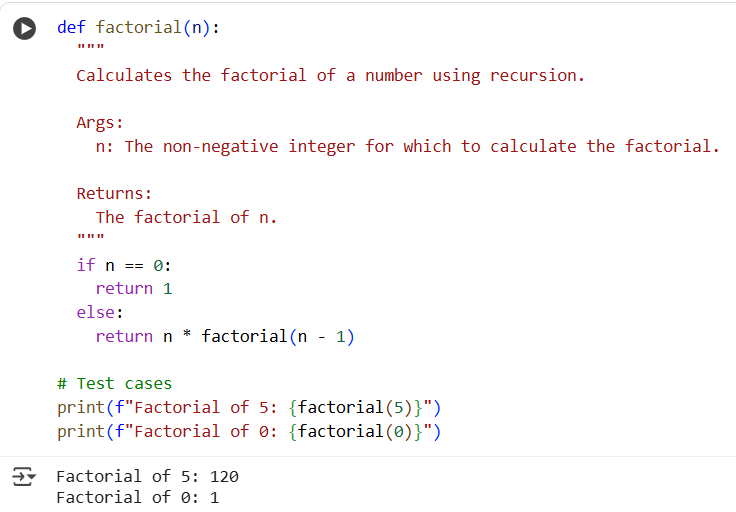


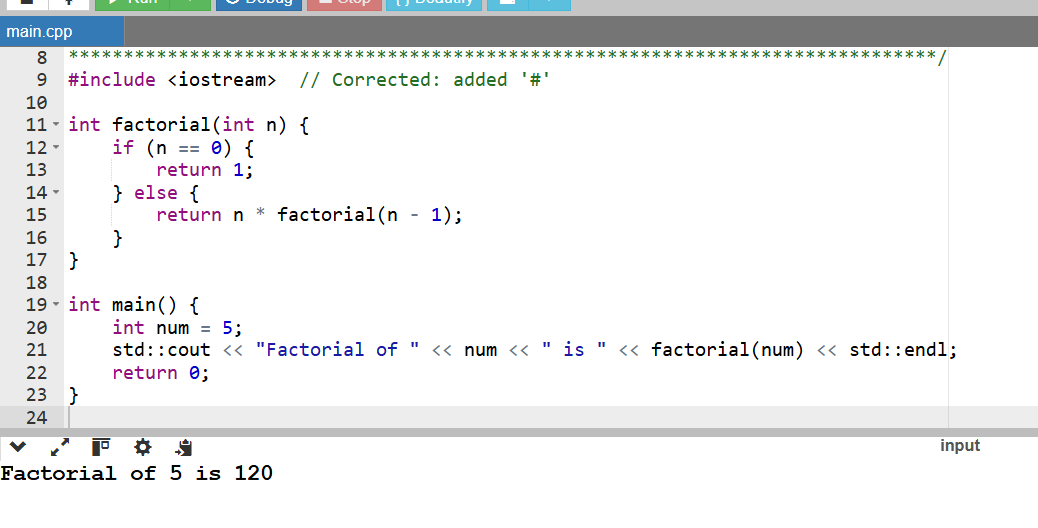
Explaination:

The Java method checkNumber(int num) and its Python counterpart check\_number(num) both serve the same purpose: to determine whether a given number is positive, negative, or zero. In Java, the method is defined with explicit data types and access modifiers—public static void—which indicate that it belongs to the class and doesn't return a value. It uses curly braces {} to define code blocks and System.out.println() to print output. The conditional logic is structured with if, else if, and else statements to evaluate the number. In Python, the function is defined using def without specifying data types, as Python is dynamically typed. Instead of braces, Python relies on indentation to define code blocks, and it uses print() for output. The else if in Java becomes elif in Python, which is more concise. Both versions are tested with sample inputs like -5, 0, and 7, and they produce identical outputs, confirming that the logic has been accurately translated across languages. This exercise highlights how Python simplifies syntax while preserving the core logic of Java, making it easier to write and read conditional statements.

**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:**
  1. **Input: 5 → Output: Factorial = 120**
  2. **Input: 0 → Output: Factorial = 1**

****

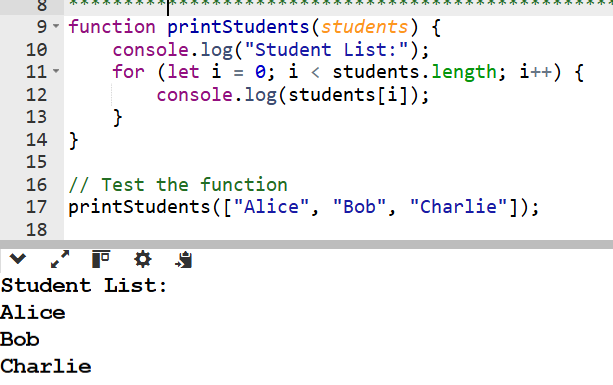


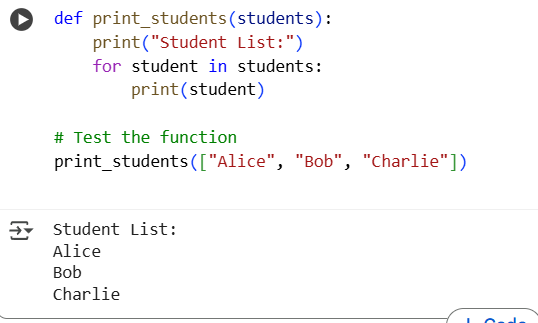
Explaination:

* Create python function: Write a Python function factorial(n) to calculate the factorial of a number using recursion and test it with inputs 5 and 0.
* Create c++ function: Write the equivalent C++ function int factorial(int n).
* Execute c++ code: Execute the C++ code in the Colab environment, which may require using a magic command or setting up a C++ environment. Test the C++ function with inputs 5 and 0.
* Compare outputs: Compare the outputs of the Python and C++ functions to ensure they are the same.
* Finish task: Summarize the process and present the final Python and C++ code**.**

**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





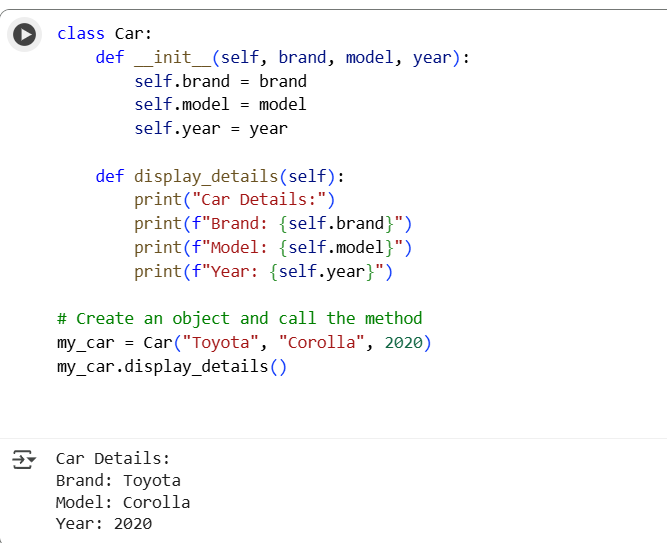
Explaination:

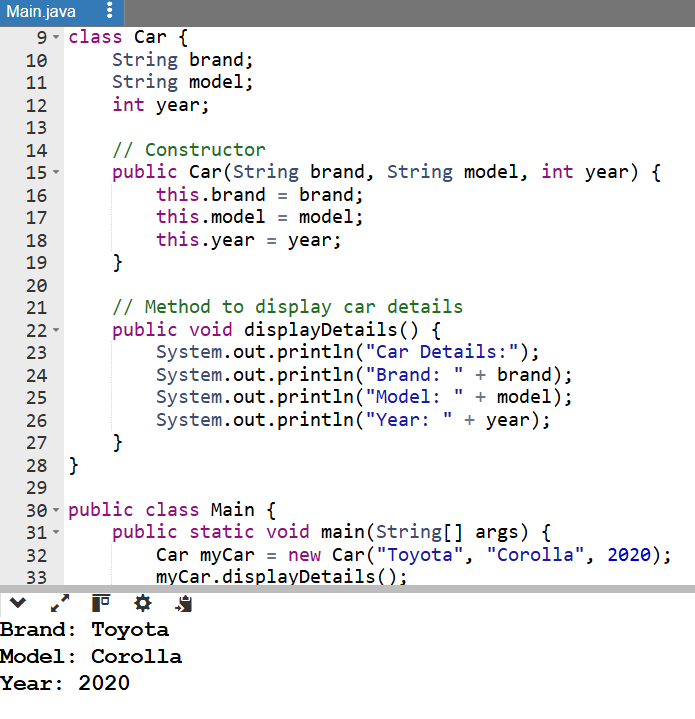
In JavaScript, the function printStudents(students) takes an array of student names and uses a for loop to print each name using console.log(). In Python, the equivalent function is print\_students(students), which takes a list of names and uses a for loop with print() to display each one. Both functions produce the same output: they print a header "Student List:" followed by each name—Alice, Bob, and Charlie. The main difference is in

**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





Explaination:

The Python and Java versions of the Car class share the same core structure but differ in syntax and execution style. In Python, the class is defined using the class keyword, and attributes are initialized inside the \_\_init\_\_ constructor method. Python uses dynamic typing, so there's no need to declare variable types explicitly. The method display\_details() prints the car's brand, model, and year using formatted strings, and the object is created directly without a main function.

In contrast, Java requires explicit data types and uses curly braces to define blocks. The Car class includes a constructor with parameters for brand, model, and year, and the method displayDetails() uses System.out.println() for output. Java also mandates a main() method inside a class (typically named Main in environments like OnlineGDB) to execute the program. Object creation uses the new keyword, and the method is called on the instantiated object. While both languages achieve the same output—displaying car details—their structural and syntactic differences reflect Python’s simplicity and Java’s formality.