# Full Stack Development with AI
## Lab 5.4 – Iterative Control Flow with JavaScript

### Lab Overview

In this lab, you will learn how to work with the iterative control flow statements `while`, `do…while` and `for` in JavaScript through some basic programming exercises.

### Exercise 1 – Times Table

Write a JavaScript that asks user to input a positive integer. Thereafter, the JavaScript should print out the times table of the integer from 1 to 10.

| Sample Input | Sample Output |
|---|---|
| 2 | 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 |
| 13 | 13, 26, 39, 52, 65, 78, 91, 104, 117, 130 |

### Exercise 2 – The Power of Numbers

Write a JavaScript that asks user to input two non-negative integers as the base and exponent. Thereafter, the JavaScript should print out the base raised to the power of exponent, i.e., $\text{base}^{\text{exponent}}$.

Do not use the `Math.pow()` method nor the `**` operator.

| Sample Input | Sample Output |
|---|---|
| 0,1 | 0 |
| 1,0 | 1 |
| 2,2 | 4 |
| 3,3 | 27 |

### Exercise 3 – Sum of Multiples

Write a JavaScript that takes two positive integers `limit` and `x`, and returns the sum of all multiples of `x` that are less than or equal to `limit`.

You are required to use a suitable JavaScript iterative control flow statement instead of a mathematical formula or built-in array methods such as `.filter()` or `.reduce()`.

| Sample Input | Sample Output |
|---|---|
| 10, 3 | 18 |
| 25, 5 | 75 |
| 100, 10 | 550 |

## Exercise 4 – Count the Digits

Write a JavaScript that takes a positive integer `n` as input and returns the number of digits in `n` using a suitable JavaScript iterative control flow statement. Do not convert the number to a string and do not use any built-in methods such as `Math.log10()`.

However, since JavaScript does not have an integer division operator, you may use the `Math.floor()` method to truncate the fractional part of a floating-point division result.

| Sample Input | Sample Output |
|---|---|
| 7 | 1 |
| 42 | 2 |
| 13579 | 5 |

***-- End of Lab --***