**Lab Exercise 2 – Python Decorators**

**Objective:** In this lab exercise, you will learn about Python decorators, how to use built-in decorators, and how to create and apply custom decorators.

**Instructions:**

**Part 1: Understanding Built-in Decorators**

* Open a Python IDE or text editor.
* Create a function named say\_hello that prints "Hello, World!"
* Run the say\_hello function and observe the output.
* Create a decorator function named uppercase\_decorator that converts the text generated by any function to uppercase.
* Apply the uppercase\_decorator to the say\_hello function and create a decorated version of it.
* Run the decorated say\_hello function and observe the output.

**Part 2: Creating Custom Decorators**

* Create a custom decorator named timer\_decorator that measures and prints the time taken by any function to execute.
* Create a function named slow\_function that simulates a time-consuming operation by sleeping for a few seconds.
* Apply the timer\_decorator to the slow\_function and create a decorated version of it.
* Run the decorated slow\_function and observe the time taken for execution.

Conclusion: In this lab exercise, you learned about Python decorators, how to use built-in decorators, and how to create and apply custom decorators. Decorators are a powerful tool for modifying and extending the behavior of functions without modifying their source code.