**Task L1: (GIL)**

Write a Python script that uses both threads and processes to perform a CPU-bound task. The task should calculate the factorial of 100 multiple times using GIL

1. Define a function cpu\_intensive\_task(n) that performs the heavy calculations.
2. Implement a function run\_with\_threads(num\_threads, iterations) that creates and starts multiple threads to perform the task.
3. Implement a function run\_with\_processes(num\_processes, iterations) that creates and starts multiple processes to perform the task.
4. Create a main() function to execute these tasks with 4 threads and 4 processes, and measure the execution time for both.

Your script should print the time taken using threads and processes, as well as calculate and display the speedup between threads and processes.

**L1 Task 2 (Memory Management)**

Write a Python program that demonstrates how memory is used for different object types and how Python manages memory for small integers and larger objects. Begin by importing the sys module to access memory-related functions. Define a function named memory\_usage that takes one parameter, obj, and uses the sys.getsizeof() function to calculate the memory size of the object. The function should print the object's representation (e.g., 42 for an integer or "Python" for a string), its data type, and its memory size in bytes.

1. create objects of various types, including an integer of your choice, a string, a list with at least five integers, and a dictionary with at least three key-value pairs. Use the memory\_usage function to display memory usage for each object. Then, demonstrate memory reuse by creating two integer variables with the same small integer value (e.g., 10) and showing their memory usage. Use the is operator to check if the variables refer to the same memory location and print the result.
2. create a large list containing integers from 0 to 999 using a list comprehension and use the memory\_usage function to display its memory usage.

**L1 task 3 (Profiling)**

You have been tasked with optimizing a small Python program that processes a list of numbers. The program takes a list of integers and calculates the sum of all the numbers. However, the current implementation is taking longer than expected, and you suspect there might be some inefficiency in the way it processes the data. Your goal is to profile this program to identify potential performance bottlenecks.