DSAI 3202 – Parallel and distributed computing
Lab – 5: Multiprocessing

1. Objectives:

- Develop Python programs that take advantage of python multiprocessing capabilities.

2. Tools and Concepts:

- Python: Programming language.
- Packages: multiprocessing, concurrent.

3. Exercises in conjunction with the lecture

3.a. Square program

- Create e function `square` that computes the square number of an int.
- Create a list of 10^3 numbers.
- Time the program in these scenarios on the random list.
    - A sequential `for` loop.
    - A `multiprocessing` `for` loop with a process for each number.
    - A `multiprocessing` `pool` with both `map()` and `apply()`.
    - A `concurrent.futures ProcessPoolExecutor.`
- What are your conclusions?
- Redo the test with 10^7 numbers.
- Test both synchronous and asynchronous versions in the pool.
- What are your conclusions?

3.b. Machine Learning

- Go back to Lab 3 - part 2 and redo the program using a parallelization scheme of your choice.
- Quickest program takes 3% in the assignment (On the fastest).

4. Process Synchronization with Semaphores

4.a. Overview

Learn how to use semaphores in Python's multiprocessing module to manage access to a limited pool of resources. Implement a `ConnectionPool` class that simulates a pool of database connections, using a semaphore to control access.

- Create a `ConnectionPool` class with methods to get and release connections, using a semaphore to limit access.
- Write a function that simulates a process performing a database operation by acquiring and releasing a connection from the pool.
- Observe how the semaphore ensures that only a limited number of processes can access the pool at any given time.

4.b. Instructions

i) Create the ConnectionPool Class:

- Define a **`ConnectionPool`** class with an **`__init__`** constructor method that initializes a list of connections and a semaphore.
- Implement **`get_connection`** and **`release_connection`** methods to acquire and release connections using the semaphore.

ii) Implement the Database Operation Function:

- Write a function **`access_database`** that simulates a process performing a database operation. It should:
  - acquire a connection,
  - print a message indicating that it has the connection, sleep for a random duration to simulate work,
  - release the connection and print a message indicating that it has released the connection.

iii) Set Up Multiprocessing:

- In the **`main`** function, create an instance of **`ConnectionPool`** with a limited number of connections.
- Create multiple **`multiprocessing.Process`** instances, each targeting the **`access_database`** function with the connection pool as an argument.
- Start all processes and wait for them to be completed.
- Ensure your program prints messages indicating when a process is waiting for a connection, has acquired a connection, and has released a connection.

iv) Discuss Observations:

- What happens if more processes try to access the pool than there are available connections?
- How does the semaphore prevent race conditions and ensure safe access to the connections?