Exercises

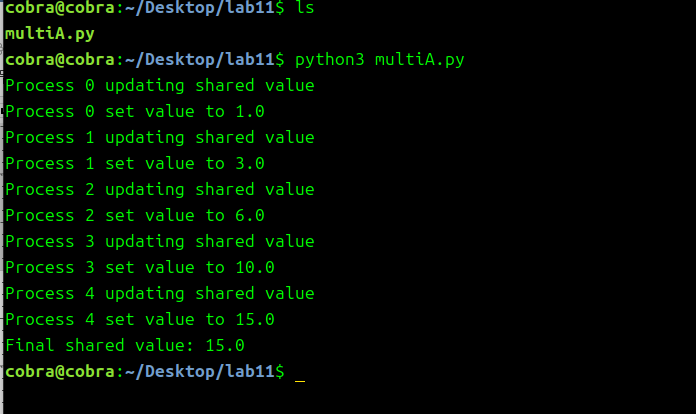
1. Start five processes using multiprocessing. Process objects, each process will update shared memory Value object using their own target function (callable object to be invoked by the run() method). After execution of all child processes, parent process should display the value of the object.

Ans:

multiA.py:

|  |
| --- |
| import multiprocessing  def update\_shared\_value(process\_id, shared\_value):  print(f"Process {process\_id} updating shared value")  with shared\_value.get\_lock(): # Synchronize access  shared\_value.value += process\_id + 1 # Add process\_id + 1  print(f"Process {process\_id} set value to {shared\_value.value}")  def main():  # Initialize shared Value (double-precision float)  shared\_value = multiprocessing.Value('d', 0.0)    # Create 5 processes  processes = [  multiprocessing.Process(target=update\_shared\_value, args=(i, shared\_value))  for i in range(5)  ]    # Start all processes  for p in processes:  p.start()    # Wait for all processes to complete  for p in processes:  p.join()    # Print final value  print(f"Final shared value: {shared\_value.value}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

OUTPUT:



2. Generate 10 random numbers between 0 and 10, and calculate square of each number such that process#1 calculates square of first five numbers and process#2 calculates square of remaining five numbers, Store the square results in an array (shared memory region) using multiprocessing module

MultiB.py

|  |
| --- |
| import multiprocessing  import random  def compute\_squares(process\_id, numbers, shared\_array):  start\_idx = 0 if process\_id == 0 else 5 # First 5 for Process 0, last 5 for Process 1  end\_idx = 5 if process\_id == 0 else 10  print(f"Process {process\_id} computing squares for {numbers[start\_idx:end\_idx]}")  with shared\_array.get\_lock(): # Synchronize access  for i in range(start\_idx, end\_idx):  shared\_array[i] = numbers[i] \* numbers[i]  print(f"Process {process\_id} updated array: {list(shared\_array)[start\_idx:end\_idx]}")  def main():  # Generate 10 random numbers  random\_numbers = [random.randint(0, 10) for \_ in range(10)]  print(f"Generated random numbers: {random\_numbers}")    # Initialize shared Array (signed integers)  shared\_array = multiprocessing.Array('i', [0] \* 10)    # Create 2 processes  processes = [  multiprocessing.Process(target=compute\_squares, args=(i, random\_numbers, shared\_array))  for i in range(2)  ]    # Start all processes  for p in processes:  p.start()    # Wait for all processes to complete  for p in processes:  p.join()    # Print final array  print(f"Final squared array: {list(shared\_array)}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

OUTPUT:

