Sharing Data Between Processes Using Value and Array

Using Array :

import multiprocessing

def calc\_square(numbers,result):

for idx, n in enumerate(numbers):

result[idx] = n\*n

if \_\_name\_\_ == "\_\_main\_\_":

numbers = [2,3,5]

result = multiprocessing.Array('i',3)

p= multiprocessing.Process(target=calc\_square, args=(numbers, result ))

p.start()

p.join()

print(result[:])

Using Value:

import multiprocessing

def calc\_square(numbers,result,v):

v.value = 5.67

for idx, n in enumerate(numbers):

result[idx] = n\*n

if \_\_name\_\_ == "\_\_main\_\_":

numbers = [2,3,5]

result = multiprocessing.Array('i',3)

v = multiprocessing.Value('d', 0.0)

p= multiprocessing.Process(target=calc\_square, args=(numbers, result, v ))

p.start()

p.join()

print(v.value)

Sharing data between Processes Using Queue:

import multiprocessing

def calc\_square(numbers, q):

for n in numbers:

q.put(n\*n)

if \_\_name\_\_ == "\_\_main\_\_":

numbers = [2,3,5]

q = multiprocessing.Queue()

p = multiprocessing.Process(target=calc\_square, args=(numbers, q))

p.start()

p.join()

while q.empty() is False:

print(q.get())

Difference between Multiprocessing Queue and Queue Module:

Multiprocessing : Syntax – import multiprocessing

Q= multiprocessing.Queue()

Lives in shared memory

Used to share data between processes

Queue Module: Syntax- import queue

Q = queue.Queue()

Lives in in-process memory

Used to share data between threads

Multiprocessing Lock:

import time

import multiprocessing

def deposit(balance, lock):

for i in range(100):

time.sleep(0.01)

lock.acquire()

balance.value = balance.value + 1

lock.release()

def withdraw(balance, lock):

for i in range(100):

time.sleep(0.01)

lock.acquire()

balance.value = balance.value - 1

lock.release()

if \_\_name\_\_ == '\_\_main\_\_':

balance = multiprocessing.Value('i', 100)

lock = multiprocessing.Lock()

d = multiprocessing.Process(target=deposit, args=(balance, lock))

w = multiprocessing.Process(target=withdraw, args=(balance, lock))

d.start()

w.start()

d.join()

w.join()

print(balance.value)