🚗

Video5

Sharing Data Between Processes Using Queue

if you know the fundamentals of multi processing

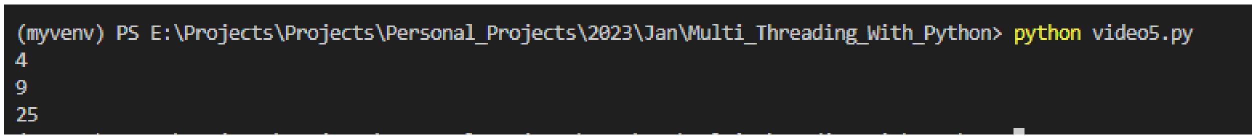
then you are already aware of that multiple process have their own address base, they don’t share the address base, that results in a problem as bellow

```
import time
import multiprocessing
square_results = []
def calc_square(numbers):
    global square_results
    for n in numbers:
        print(f"square {n*n}")
        square_results.append(n*n)
    print(f"within a process: result {square_results}")
if __name__ == "__main__":
    arr = [2,3,8,9]
    p1 = multiprocessing.Process(target=calc_square,args=(arr,))
    t = time.time()
    # start the processes
    p1.start()
    # wait until processes finish successfully!
    p1.join()
    print(f"results {square_results}")
    print(f"Done! at {time.time()-t} secs")
```

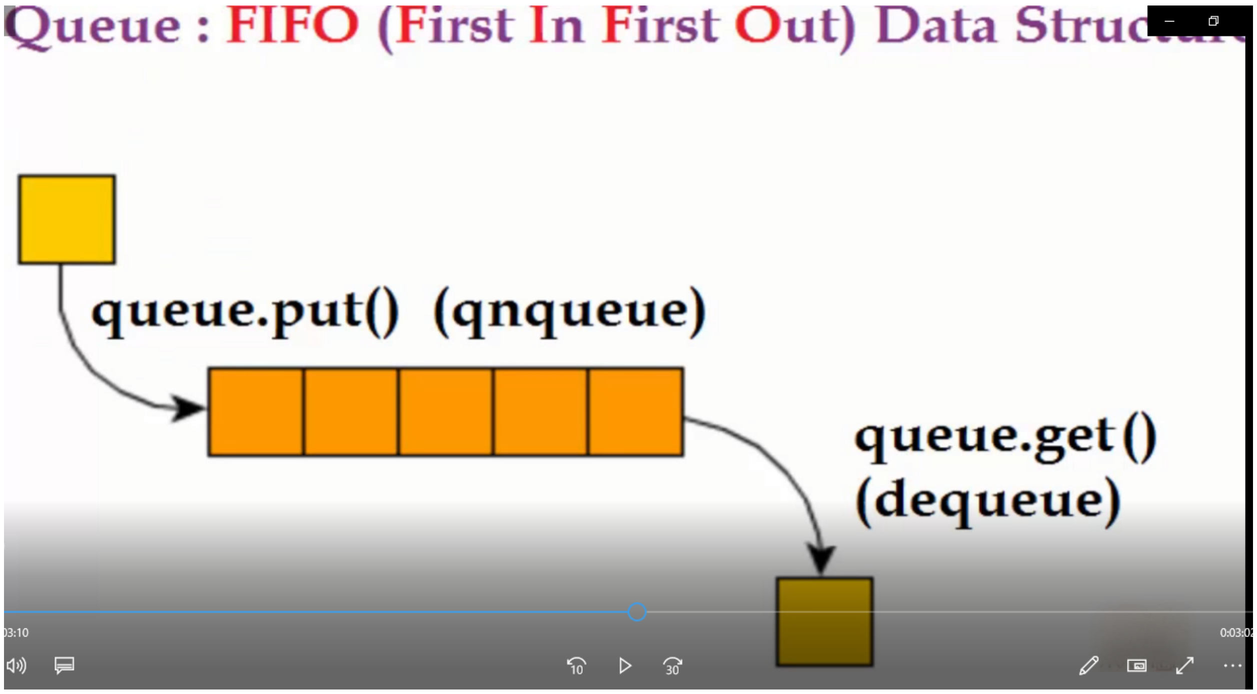
so basically in the above example the program will calculate the square root of a given number and

we will pass in a array of numbers and then it will append the result in a global variable and since we use multiprocessing the new process will create a new address base and if we call that result variable outside of the function we will get a empty array.

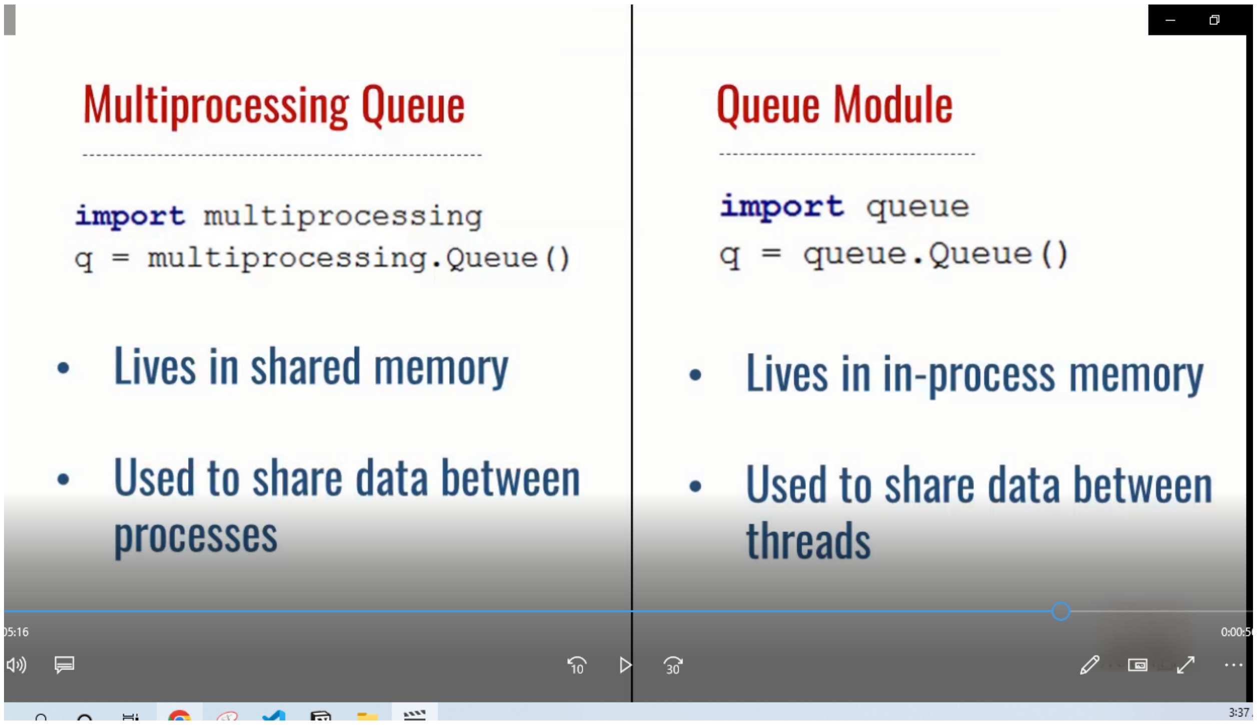
```
import multiprocessing
def calc_square(numbers,q):
    for n in numbers:
        q.put(n*n)

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())
```



```
(myvenv) PS E:\Projects\Projects\Personal_Projects\2023\Jan\Multi_Threading_With_Python> python video5.py
4
9
25
```



Queue : FIFO (First In First Out) Data Struc



Multiprocessing Queue

```
import multiprocessing
q = multiprocessing.Queue()
```

- Lives in shared memory
- Used to share data between processes

Queue Module

```
import queue
q = queue.Queue()
```

- Lives in in-process memory
- Used to share data between threads