

# Video 2

## Multiprocessing with Python

with are gonna create two processes,

- First will calculate square of all numbers
- Second one is to calculate cube of numbers






```
import time
import multiprocessing
def calc_square(numbers):
    for n in numbers:
        time.sleep(5) # we add this delay for program to work slower in order to check
        them in task mananger
        print('square',n*n)
def calc_cube(numbers):
    for n in numbers:
        time.sleep(5) # we add this delay for program to work slower in order to check
        them in task mananger
        print('cube',n*n*n)
if __name__ == "__main__":
    arr = [2,3,8,9]
    p1 = multiprocessing.Process(target=calc_square,args=(arr,))
    p2 = multiprocessing.Process(target=calc_cube,args=(arr,))
    t = time.time()
    # start the processes
    p1.start()
    p2.start()
```

```

# wait until processes finish successfully!
p1.join()
p2.join()

print(f"Done! at {time.time()-t} secs")

```

ne	Status	CPU	Memory	DISK	NETWORK
 wsappx		0%	1.1 MB	0 MB/s	0 Mbps
 Python		0%	7.2 MB	0 MB/s	0 Mbps
 Python		0%	7.0 MB	0 MB/s	0 Mbps
 Python		0%	6.4 MB	0 MB/s	0 Mbps
 Python		0%	0.4 MB	0 MB/s	0 Mbps

```

(myvenv) PS E:\Projects\Projects\Personal_Projects\2023\Jan\Multi_Threading_With_Python> python video2.py
square 4
cube 8
square 9
cube 27
square 64
cube 512
square 81
cube 729
Done! at 20.160309314727783 secs

```

```

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)
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")

```

```
(myvenv) PS E:\Projects\Projects\Personal_Projects\2023\Jan\Multi_Threading_With_Python> python video2.py
square 4
square 9
square 64
square 81
results []
Done! at 0.13168859481811523 secs
```

as you can see still with creating a global variable named square results, the result is a empty set.

**So Why is that?**

Every process has its own address space (virtual memory). Thus program variables are not shared between two processes. You need to use interprocess communication (IPC) techniques if you want to share data between two processes

2

```
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")
```

```
(myvenv) PS E:\Projects\Projects\Personal_Projects\2023\Jan\Multi_Threading_With_Python> python video2.py
square 4
square 9
square 64
square 81
within a process: result [4, 9, 64, 81]
results []
Done! at 0.1255502700805664 secs
```

as you can see above when we call the function inside of a process we can print the results but when we call it outside of that process even with a global variable we can't access the results, that is because every process will create a new copy of that global variable and this is the main difference between multi threading and multi processing so for instance if we create a thread instead of process and then call the print method over the square-results out side of the thread we can get results!