its ICTAcademy
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Multiprocessing in Python

Intro to Processes

A thread is a separate flow of execution.

This means that your program will have two things happening at once.

But for most Python 3 implementations the different threads do not actually execute at the same time: **they merely appear to.**

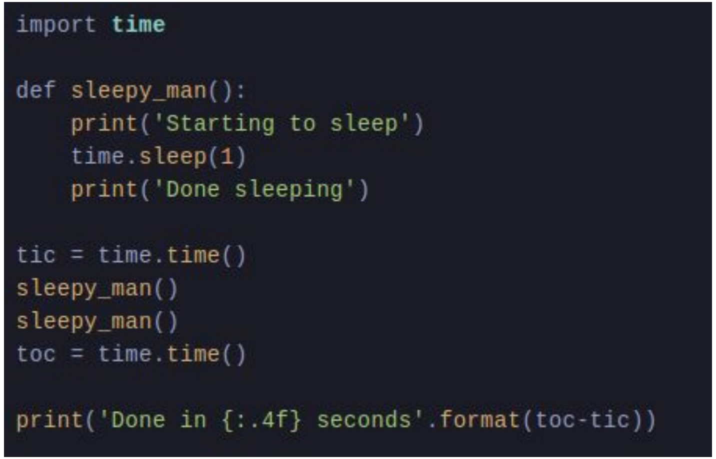
It’s tempting to think of threading as having two (or more) different processors running on your program, each one doing an independent task at the same time.

That’s almost right. The threads may be running on different processors, but they will only be running one at a time. Remember the GIL!

Intro to Multiprocessing in Python

Take a look a this code.

How does it behave?

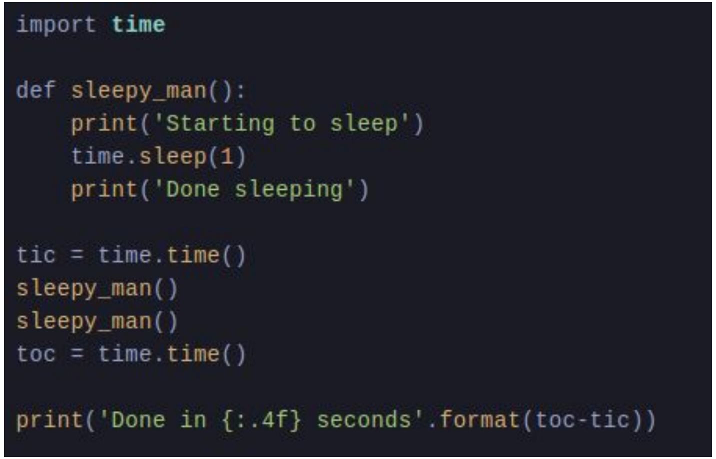


```
import time

def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
sleepy_man()
sleepy_man()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

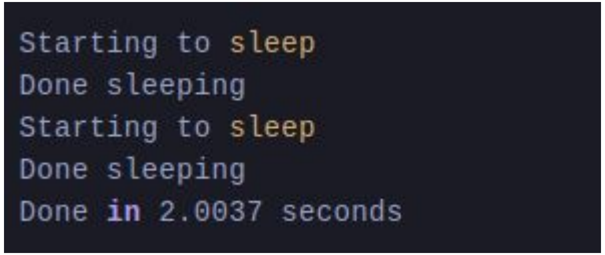
Intro to Multiprocessing in Python

Take a look a this code.

How does it behave?



```
import time

def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
sleepy_man()
sleepy_man()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

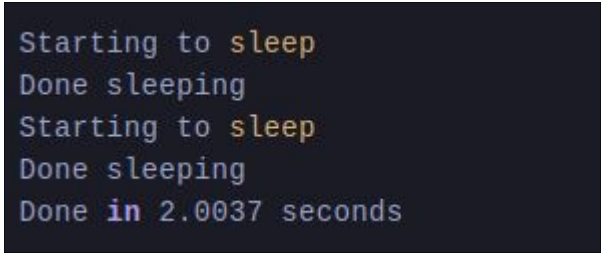


```
Starting to sleep
Done sleeping
Starting to sleep
Done sleeping
Done in 2.0037 seconds
```

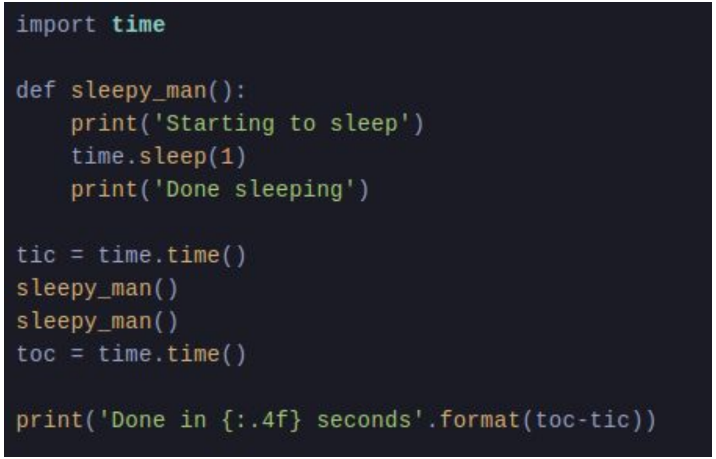
Intro to Multiprocessing in Python

Take a look a this code.

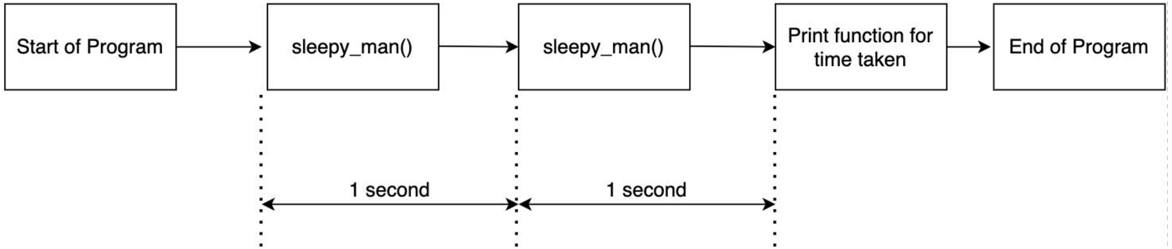
How does it behave?



```
Starting to sleep
Done sleeping
Starting to sleep
Done sleeping
Done in 2.0037 seconds
```



```python
import time

def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
sleepy_man()
sleepy_man()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```



Start of Program

sleepy_man()

sleepy_man()

Print function for time taken

End of Program

1 second

1 second

Intro to Multiprocessing in Python

What if I want to execute the functions **at the same time**?

Intro to Multiprocessing in Python

What if I want to execute the functions **at the same time**?

I could use threads!

Intro to Multiprocessing in Python

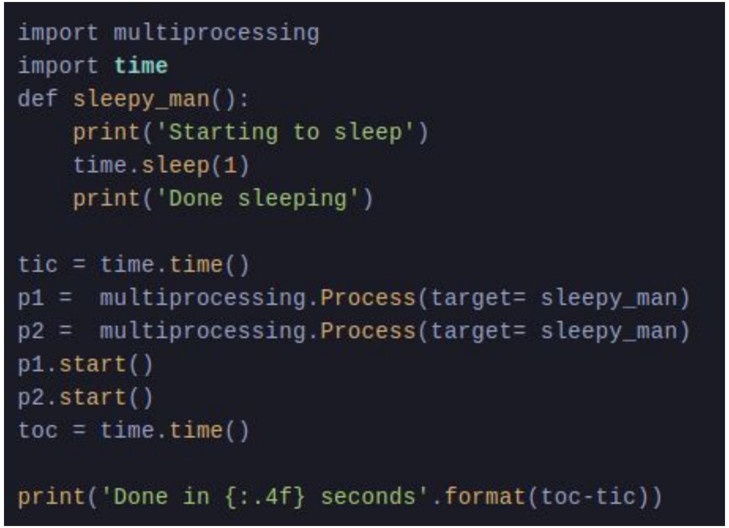
What if I want to execute the functions **at the same time**?

I could use threads!

But there is the GIL so parallelism is effectively just on I/O bounded programs!

Intro to Multiprocessing in Python

I can use the multiprocess library in python!

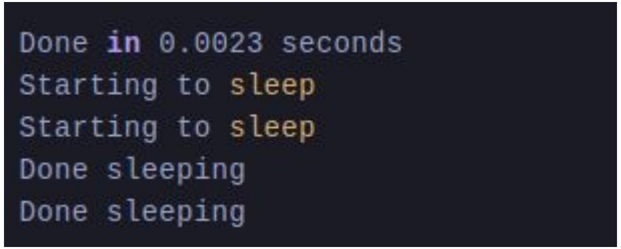


```
import multiprocessing
import time
def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
p1 =  multiprocessing.Process(target= sleepy_man)
p2 =  multiprocessing.Process(target= sleepy_man)
p1.start()
p2.start()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

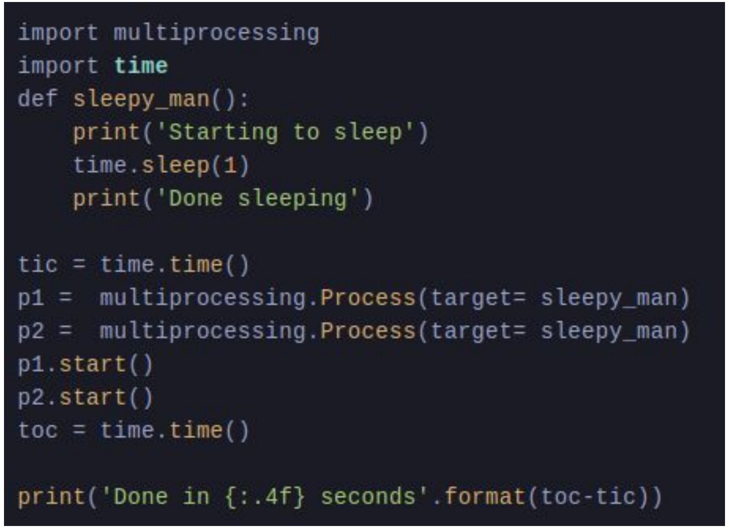
Intro to Multiprocessing in Python

I can use the multiprocess library in python!

The execution is the following



```
Done in 0.0023 seconds
Starting to sleep
Starting to sleep
Done sleeping
Done sleeping
```

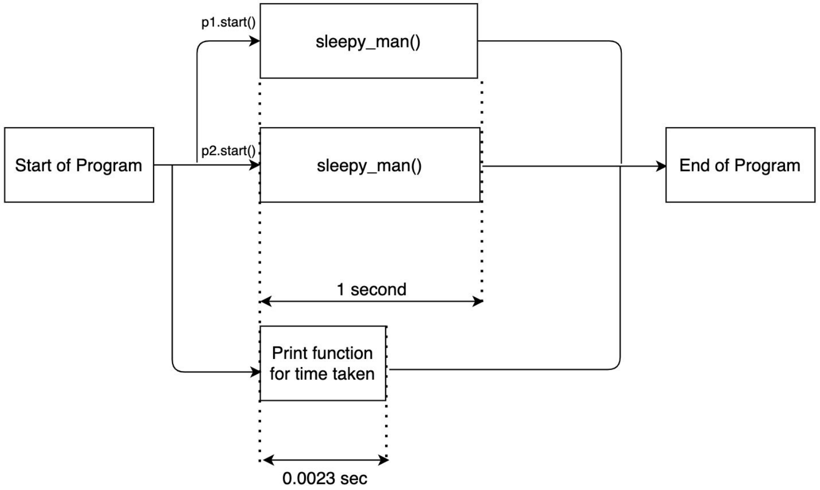


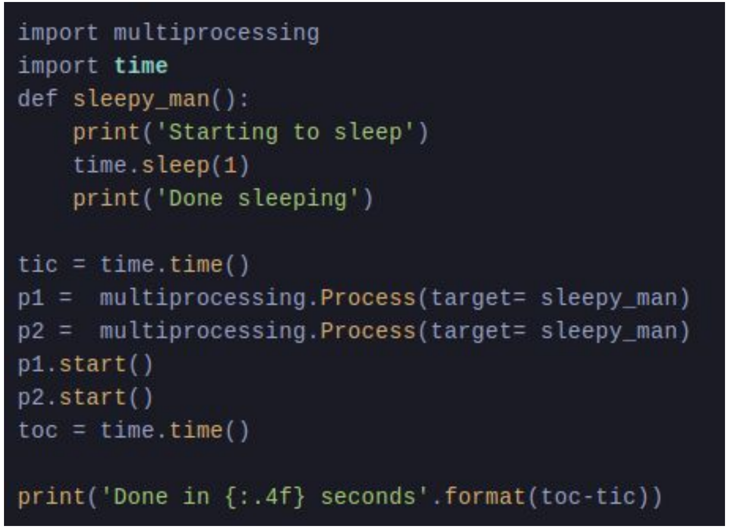
```
import multiprocessing
import time
def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
p1 =  multiprocessing.Process(target= sleepy_man)
p2 =  multiprocessing.Process(target= sleepy_man)
p1.start()
p2.start()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

Intro to Multiprocessing in Python

I can use the multiprocess library in python!

The execution is the following

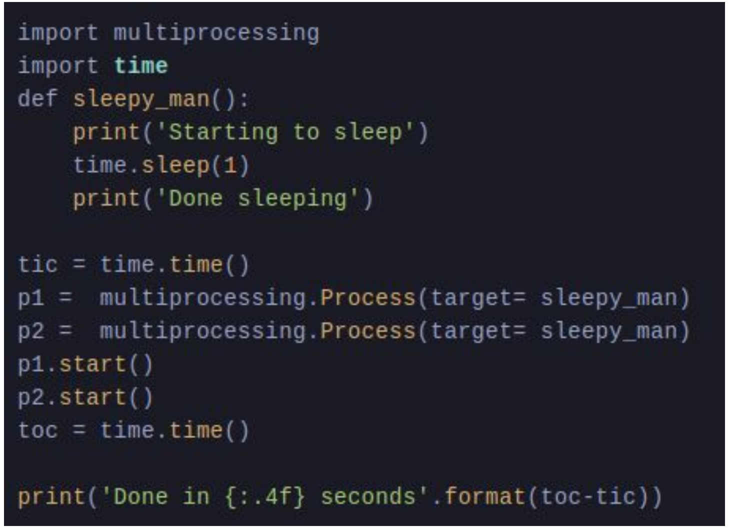




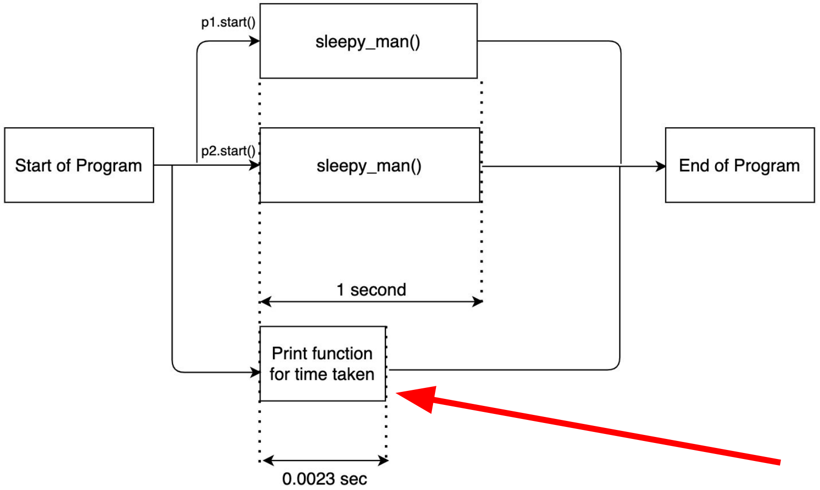
```
import multiprocessing
import time
def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
p1 =  multiprocessing.Process(target= sleepy_man)
p2 =  multiprocessing.Process(target= sleepy_man)
p1.start()
p2.start()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

Intro to Multiprocessing in Python

Problem! Here the program ends **BEFORE** the processes end!



```
import multiprocessing
import time
def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
p1 =  multiprocessing.Process(target= sleepy_man)
p2 =  multiprocessing.Process(target= sleepy_man)
p1.start()
p2.start()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

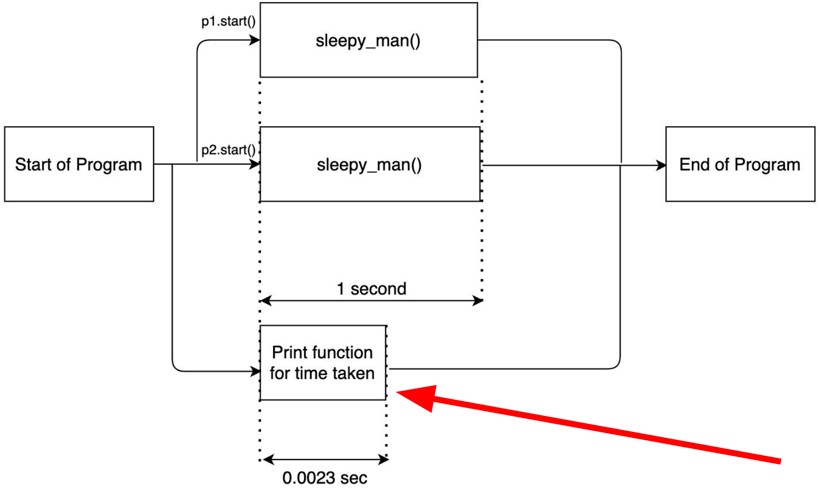


Main Process

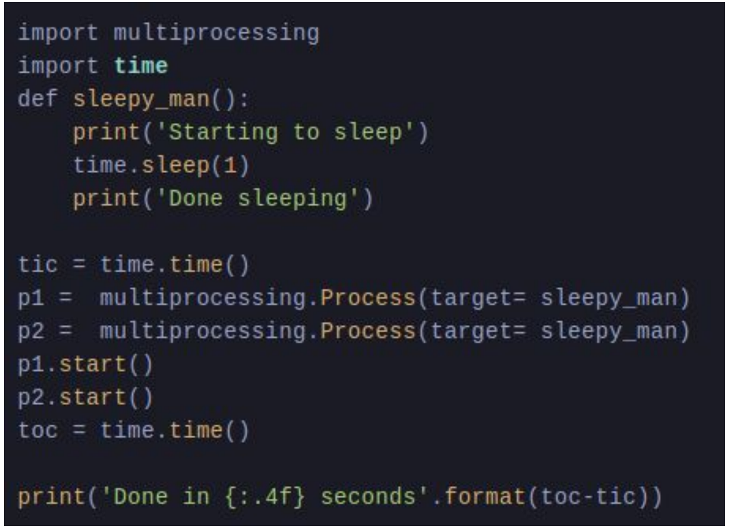
Intro to Multiprocessing in Python

How can I wait until the children process end?

Use the join!



Main Process

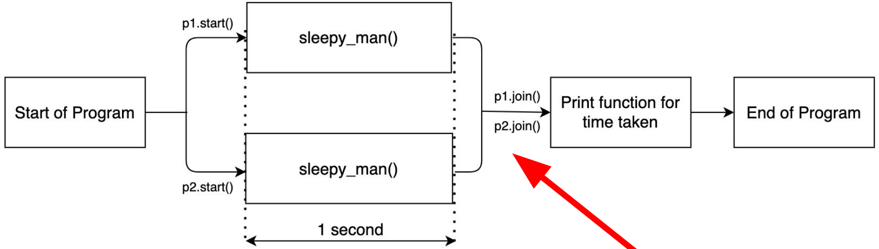


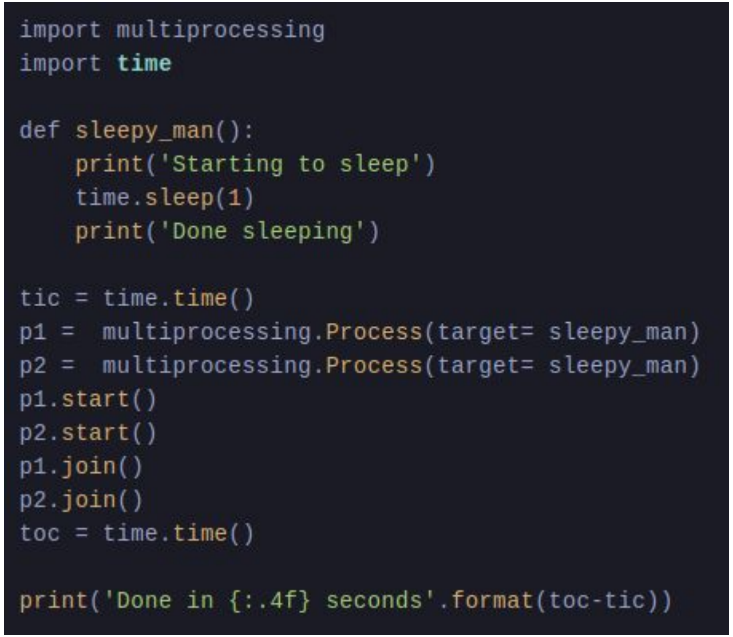
```
import multiprocessing
import time
def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
p1 =  multiprocessing.Process(target= sleepy_man)
p2 =  multiprocessing.Process(target= sleepy_man)
p1.start()
p2.start()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

Intro to Multiprocessing in Python

How can I wait until the children process end?

Use the join!





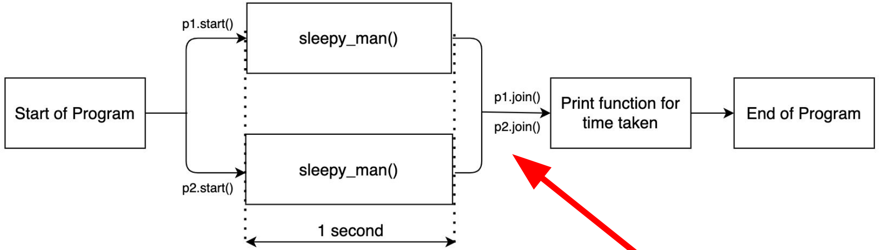
```python
import multiprocessing
import time

def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
p1 =  multiprocessing.Process(target= sleepy_man)
p2 =  multiprocessing.Process(target= sleepy_man)
p1.start()
p2.start()
p1.join()
p2.join()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

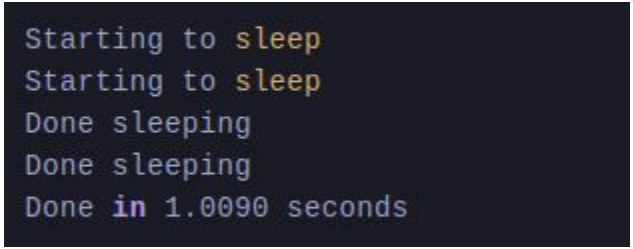
Main Process waits until children end!

Intro to Multiprocessing in Python

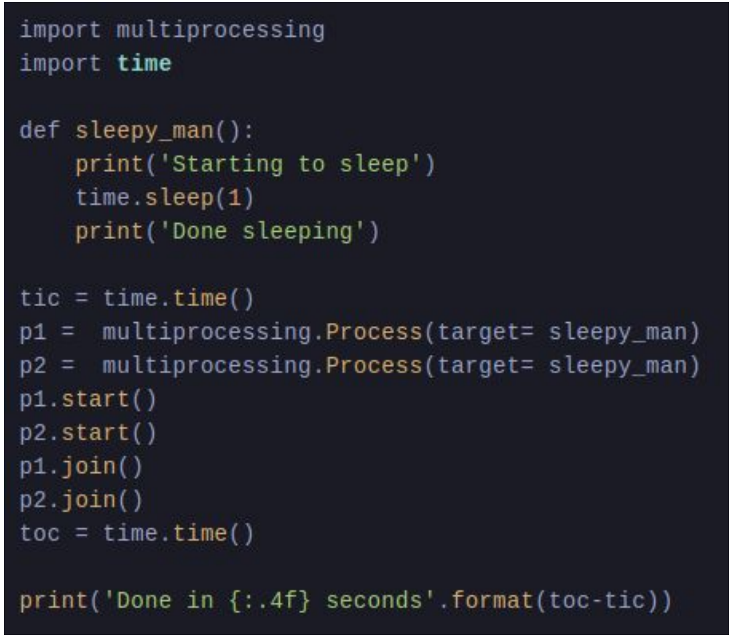
How can I wait until the children process end?

Use the join!





```
Starting to sleep
Starting to sleep
Done sleeping
Done sleeping
Done in 1.0090 seconds
```



```
import multiprocessing
import time

def sleepy_man():
    print('Starting to sleep')
    time.sleep(1)
    print('Done sleeping')

tic = time.time()
p1 =  multiprocessing.Process(target= sleepy_man)
p2 =  multiprocessing.Process(target= sleepy_man)
p1.start()
p2.start()
p1.join()
p2.join()
toc = time.time()

print('Done in {:.4f} seconds'.format(toc-tic))
```

Main Process waits until children end!