

Forum:[Hack Forums](#)

Thread: [Multiprocessing in "for loop"](#)

The Pulp Fiction - May 7, 2022, 8:55 am

I'm handling a big chunk of data and executing the "for loop" one after the other seems inefficient.

Example:

Code for i in range (0,5000):

```
print(i)
```

So, basically, what I'm looking for is, splitting the task as 10 at a time parallelly(print 0 to 9 at the same time).

Can someone explain which package is best for this and the code for doing it?

Thanks ?

3phase - May 9, 2022, 4:17 am

You might use "asyncio" for asynchronous programming

you can read about that here

<https://docs.python.org/3/library/asyncio.html>

regarding your specific example, with asyncio it would look like that

Code import asyncio

```
def chunks(lst, n):
```

```
    for i in range(0, len(lst), n):
```

```
        yield lst[i:i + n]
```

```
async def something(num):
```

```
    print(num)
```

```
async def run_script():
```

```
    for chunk in chunks(range(5000), 10):
```

```
        answers = await asyncio.gather(*[something(num) for num in chunk])
```

```
        await asyncio.sleep(5)
```

```
if __name__ == '__main__':
```

```
    ans = asyncio.run(run_script())
```

ive made a generator to better handle the chunks,

and inserted each chunk into asyncio.gather.

note that declaring "something(num)" will not execute the function, you gotta use

"await" before the async function to actually execute it (so the syntax does make sense)

"answers" param will contain a list of the return values (if exists) from the executed function in
asyncio.gather

ive added the correct way to use "sleep" with asyncio programming, just so you can actually check that its working

Hope that's help you