Part 9: Pooling

Bookmark this page

Pooling

A processing pool contains worker processes with only a configured number running at one time.

```
from multiprocessing import Pool
pool = Pool(processes=4)
```

The Pool module has several methods for adding jobs to the pool.

`apply_async(func[, args[, kwargs[, callback]]])`

`map_async(func, iterable[, chunksize[, callback]])`

Pooling example

```
from multiprocessing import Pool
def f(x):
    return x*x
if __name__ == '__main__':
    pool = Pool(processes=4)

    result = pool.apply_async(f, (10,))
    print(result.get(timeout=1))
    print(pool.map(f, range(10)))

    it = pool.imap(f, range(10))
    print(it.next())
    print(it.next())
    print(it.next(timeout=1))

    import time
    result = pool.apply_async(time.sleep, (10,))
    print(result.get(timeout=1))
```

ThreadPool

Threading also has a pool.

Confusingly, it lives in the multiprocessing module.

```
from multiprocessing.pool import ThreadPool
pool = ThreadPool(processes=4)
```

Previous Next