concurrent.futures

Python 2需要手动安装

```
❯ pip install futures
```

模块主要包含下面2个类：

1. ThreadPoolExecutor
2. ProcessPoolExecutor

也就是对threading和multiprocessing的进行了高级别的抽象，暴露出统一的接口，方便开发者使用

```python
import time
from concurrent.futures import ProcessPoolExecutor, as_completed

NUMBERS = range(25, 38)


def fib(n):
    if n<= 2:
        return 1
    return fib(n-1) + fib(n-2)


start = time.time()

with ProcessPoolExecutor(max_workers=3) as executor:
    for num, result in zip(NUMBERS, executor.map(fib, NUMBERS)):
        print(f'fib({num}) = {result}')

print(f'COST: {time.time() - start}')
```

```
❯ python process_pool_executor.py
fib(25) = 75025
fib(26) = 121393
fib(27) = 196418
fib(28) = 317811
fib(29) = 514229
...
```

源码中的「干货」注释


In-process
Out-of-process
Process Pool Executor
Work Ids
6
7
...
Work Items
6: call()
future
...
Local Worker Thread
Call Q
...
5, call()
...
Result Q
...
4, result
3, except
Process #1..n

```python
from concurrent.futures import ThreadPoolExecutor, as_completed

NUMBERS = range(30, 35)


def fib(n):
    if n == 34:
        raise Exception("Don't do this")
    if n<= 2:
        return 1
    return fib(n-1) + fib(n-2)


with ThreadPoolExecutor(max_workers=3) as executor:
    future_to_num = {executor.submit(fib, num): num
                     for num in NUMBERS}
    for future in as_completed(future_to_num):
        num = future_to_num[future]
        try:
            result = future.result()
        except Exception as e:
            print(f'raise an exception: {e}')
        else:
            print(f'fib({num}) = {result}')


with ThreadPoolExecutor(max_workers=3) as executor:
    for num, result in zip(NUMBERS, executor.map(fib, NUMBERS)):
        print(f'fib({num}) = {result}')
```

```
fib(30) = 832040
fib(31) = 1346269
raise an exception: Don't do this
fib(32) = 2178309
fib(33) = 3524578
fib(30) = 832040
fib(31) = 1346269
fib(32) = 2178309
fib(33) = 3524578
Traceback (most recent call last):
  File "thread_pool_executor.py", line 27, in <module>
    for num, result in zip(NUMBERS, executor.map(fib, NUMBERS)):
  File "/usr/local/Cellar/lib/python3.6/concurrent/futures/_base.py", line 586, in result_iterator
    yield fs.pop().result()
  File "/usr/local/Cellar/lib/python3.6/concurrent/futures/_base.py", line 425, in result
    return self.__get_result()
  File "/usr/local/Cellar/lib/python3.6/concurrent/futures/_base.py", line 384, in __get_result
    raise self._exception
  File "/usr/local/Cellar/lib/python3.6/concurrent/futures/thread.py", line 56, in run
    result = self.fn(*self.args, **self.kwargs)
  File "thread_pool_executor.py", line 8, in fib
    raise Exception("Don't do this")
Exception: Don't do this
```

Future

一个Future对象代表了一些尚未就绪（完成）的结果，在「将来」的某个时间就绪了之后就可以获取到这个结果。比如上面的例子，我们期望并发的执行一些参数不同的fib函数，获取全部的结果。传统模式就是在等待queue.get返回结果，这个是同步模式，而在Future模式下，调用方式改为异步

用multiprocessing中的Pool还是concurrent.futures中的PoolExecutor?

```python
import time
from multiprocessing.pool import Pool

NUMBERS = range(25, 38)


def fib(n):
    if n <= 2:
        return 1
    return fib(n-1) + fib(n-2)


start = time.time()

pool = Pool(3)

for num, result in zip(NUMBERS, pool.map(fib, NUMBERS)):
    print(f'fib({num}) = {result}
```

```
❯ python multiprocessing_pool.py
fib(25) = 75025
fib(26) = 121393
fib(27) = 196418
fib(28) = 317811
fib(29) = 514229
fib(30) = 832040
fib(31) = 1346269
fib(32) = 2178309
fib(33) = 3524578
fib(34) = 5702887
fib(35) = 9227465
fib(36) = 14930352
fib(37) = 24157817
COST: 14.73520016670227
```

分析

1. concurrent.futures的架构明显要复杂一些，不过更利于写出高效、异步、非阻塞的并行代码，而ThreadPeool/Pool更像一个黑盒，你用就好了，细节不仅屏蔽定制性也差。

2. concurrent.futures的接口更简单一些。ThreadPool/Pool的API中有processes, initializer，initargs，maxtasksperchild，context等参数，新人看起来容易不解，而concurrent.futures的参数就一个max_workers。

如何选择还是看具体需求和开发习惯了，我比较喜欢用concurrent.futures的

注意 👇

当「版本 < Python 3.5」并且「待处理的任务量比较大时」不应该使用 concurrent.futures

延伸阅读

1. https://www.python.org/dev/peps/pep-3148/
2. https://docs.python.org/3/library/concurrent.futures.html
3. https://pymotw.com/3/concurrent.futures/
4. http://www.dongwm.com/archives/shi-yong-tornadorang-ni-de-qing-qiu-yi-bu-fei-zu-sai/
5. http://www.dongwm.com/archives/%E4%BD%BF%E7%94%A8Python%E8%BF%9B%E8%A1%8C%E5%B9%B6%E5%8F%91%E7%BC%96%E7%A8%8B-PoolExecutor%E7%AF%87/