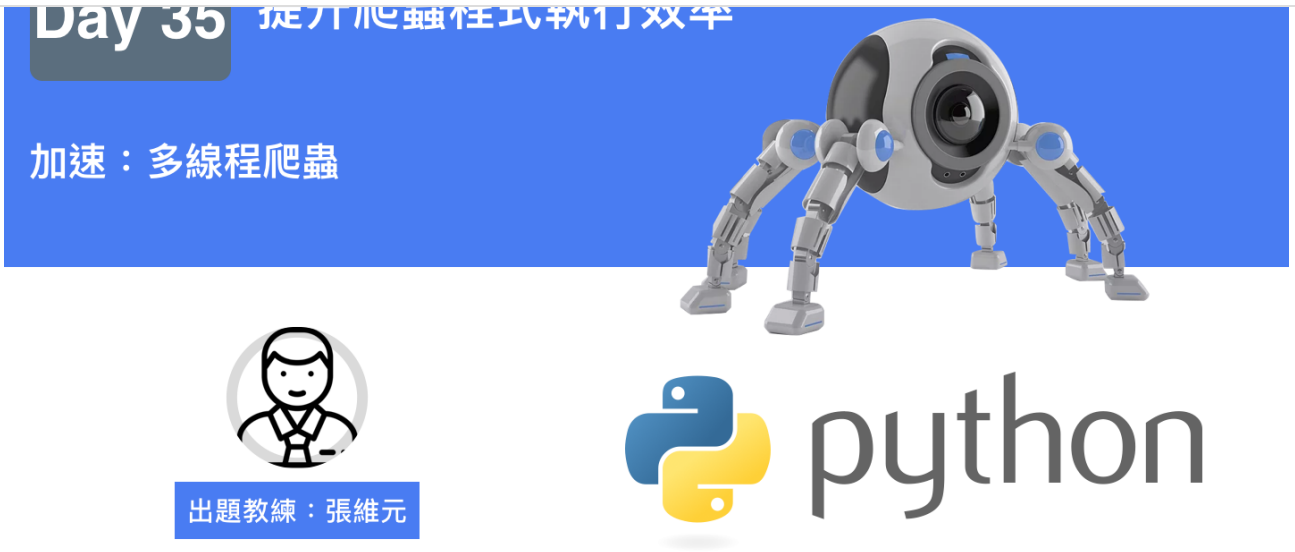
D35：加速：多線程爬蟲

簡報閱讀 範例與作業 問題討論

加速：多線程爬蟲

解題時間



本日知識點目標

- 了解多線程爬蟲加速原理與實作

當資料量龐大或是更新速度較為頻繁的狀況下。依照正常的爬蟲程式，可以會因此受到應用上的限制。所以必須用程式的方法，來思考如何加速爬蟲的處理速度。

多線程爬蟲

第一種加速的方法是「多線程爬蟲」，多線程爬蟲的意思是一次可以多個程式重複執行，因此也可以稱為平行處理。

在 Python 中實作多線程

```
def print_time( threadName, data):
    for d in data:
        time.sleep(2)
        print(threadName, ' => ', d)

_thread.start_new_thread( print_time, ("Thread-1", range(0, 5, 2), ) )
_thread.start_new_thread( print_time, ("Thread-2", range(1, 5, 2), ) )
```

```
Out[7]: 123145395556352

        Thread-1  =>  0
        Thread-1  =>  2
        Thread-2  =>  1
        Thread-1  =>  4
        Thread-2  =>  3
```

執行結果可以看出，

原本的程式被拆成兩個函式平行地交互執行。

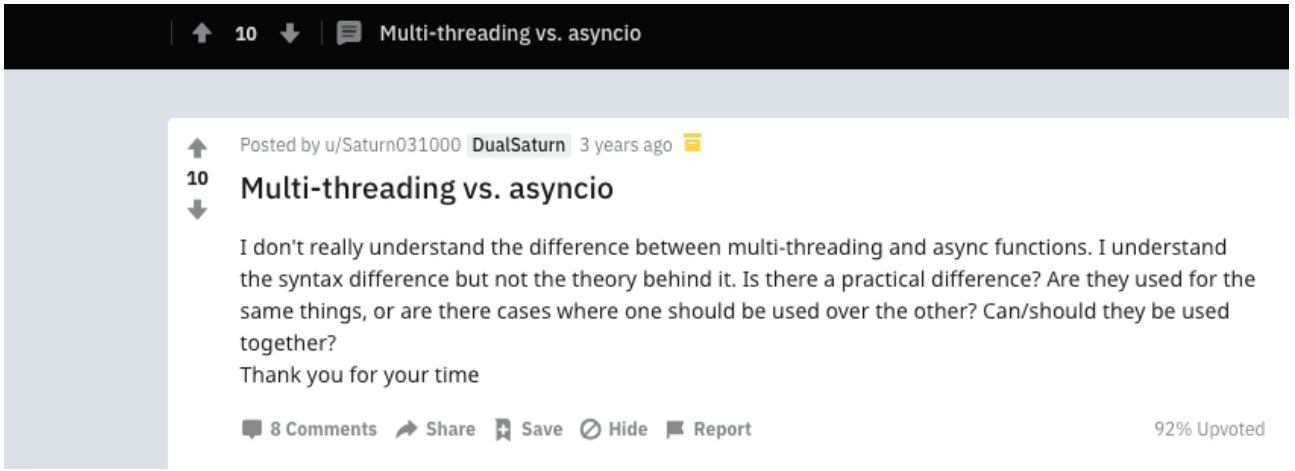
```
1  import _thread
2  import time
3
   def print_time( threadName, data):
       for d in data:
           time.sleep(2)
           print(threadName, ' => ', d)

   _thread.start_new_thread( print_time, ("Thread-1", range(0, 5, 2), ) )
   _thread.start_new_thread( print_time, ("Thread-2", range(1, 5, 2), ) )
```

簡單來說，可以想像成 _thread.start_new_thread 會開一個分支執行，不用等到結束就繼續執行下一行程式。

Multi-threading vs. asyncio Reddit

比較多線程與非同步的差異



解題時間

下一步：閱讀範例與完成作業