

Python網路程式設計

IGS / ACD_RD4
duyhsieh
duyhsieh@igs.com.tw

source file available at:
<https://github.com/mantisa1980/pythontutorial>

Agenda

- Python
 - 基礎觀念
 - Concurrency & Parallelism
 - pip套件管理
- HTTP
 - 基礎觀念
 - JSON
 - WSGI
- Python Web Server library
 - gunicorn
 - gevent coroutine
 - falcon

Part I: Python基礎

Python Basis

- 腳本語言. py -> pyc (bytecode)
- 強制縮排: 4 space (recommended) / tab (等同C大括號)
- Garbage collection
- 跨平台 (但某些三方套件對Windows不太友善: linux is better)
- 官方標準直譯器: CPython
 - 其他: PyPy (JIT), IronPython (for .NET), Stackless Python...
- 整合IDE: PyCharm

Python Basis (Cont.)

- 2.7.x v.s. 3.x
 - 2.x預計只patch到2020，新功能會出在3.x
 - 語法差異上不大: `print 123` v.s. `print(123)`，對於 `unicode/byte` 處理上有些差異
 - 2.x還是三方函式庫主流 (但3.x支援也越來越多了)
 - 初學建議從2.x著手即可，較無第三方函式庫相容問題

Python Hello World

- 練習: `helloworld.py`
 - `python helloworld.py`
 - Try to unmark first line
 - `if __name__ == "__main__":`
...被當輸入腳本時才執行，import不會

Interactive Interpreter

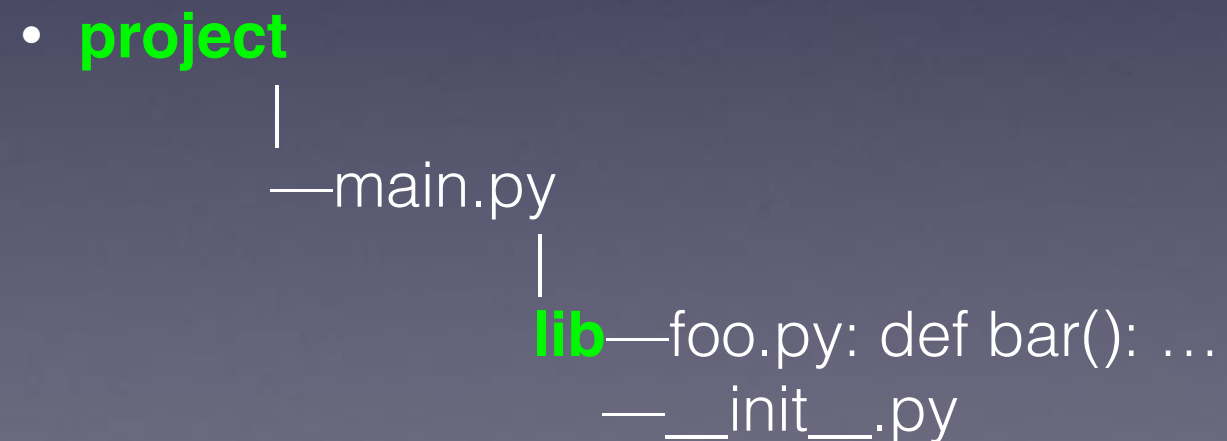
- python
 - 離開: quit()

Package / Module

- 在python的術語中...
- package: 包含多個modules
 - package目錄需要要有__init__.py
- module: single .py file
 - module裡面有function, class...

Import

- 各種可能的import法
 - `import lib.foo`
`lib.foo.bar()`
 - `from lib import foo`
`foo.bar()`
 - **`from lib.foo import bar`**
`bar()` —> 效能最好 (較少 dot reference)



Import Paths

- Ubuntu 14.04
 - 1. Input script (被執行的腳本) 目錄
 - 2. PYTHONPATH環境變數
 - 會被置入sys.path中; 若無設定預設為input script 目錄
 - 3. /usr/local/lib/python2.7/dist-packages/
 - 3rd-party library default installation path
 - 如果自己make install python，則會變成/usr/local/lib/python2.7/site-packages
- 練習: **import_test.py (fix import error)**

Variable

- 變數: 動態型別，指向物件的一個參考(name reference)
- 物件類型
 - Mutable object: list , dictionary...
 - 可透過參考直接修改物件本身
 - Immutable object: string, number, tuple...
 - 無法透過參考修改物件，只能改指到別的地方
- 變數傳入Function只是複製了另一份Variable，指向一樣的地方
- 練習:reference.py

Reference

Variable

Object

A = 3

A

3

B = 3

B

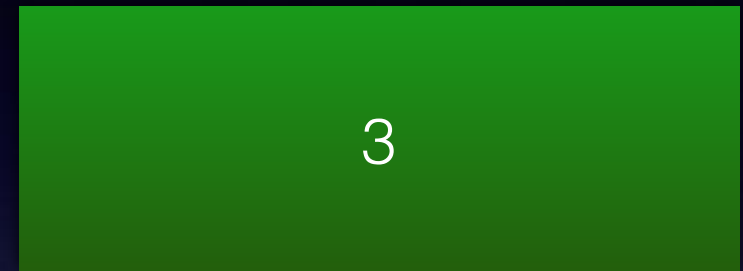
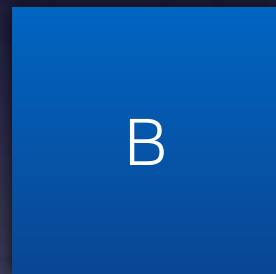
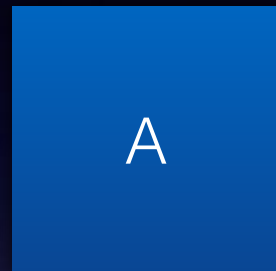
C = [1,2,3]

C

[1,2,3]

C = D

D



Reference

Variable

Object

A = 3

A

3

B = 4

B

4

C

[1,2,3,4]

D

D.append(4)
連帶影響C



常用資料結構: Dictionary (Hash table)

- `dic = {'name': 'John', 'coin': 100 } #或用dic = dict()`
`dic['age'] = 30`
`dic.pop('name')`
- 用來快速查找key / value
- 無法保證key的順序性
- `dic.keys(), dic.values():` 回傳keys / values
- 尋訪元素
- `for k in dic.keys():`
`dic[k] = ...`
- `for k,v in dic.items():`
`....`

常用資料結構 (Cont.)

- list: dynamic array，類似C++ vector的東西
 - Direct indexing很快，但搜尋複雜度= $O(n)$
 - `A = [1, 2, 'xyz']`
- tuple # immutable
 - `a = (2,)` # 注意`a=(2)`會解讀成2 (整數)
 - `a = (2,3)` # ok
`a = (2,3,)` # ok
`a[0]` # 2
- 練習:`data_structure.py`

Exception Handling

- ```
import traceback
try:
 raise Exception("Error message")
except:
 print traceback.format_exc() # print call stack
```



# Class

- ```
class Foo(object): # new style class in python
    def __init__(self, name):
        self.name = name
        self.__pdata = 0 # private

    def hello(self):
        print self.name

    @staticmethod
    def bar(param1):
        print param1
```
- ```
x = Foo('Alice')
print x.name, x.hello()
Foo.bar(1234)
```

# Parallelism & Concurrency

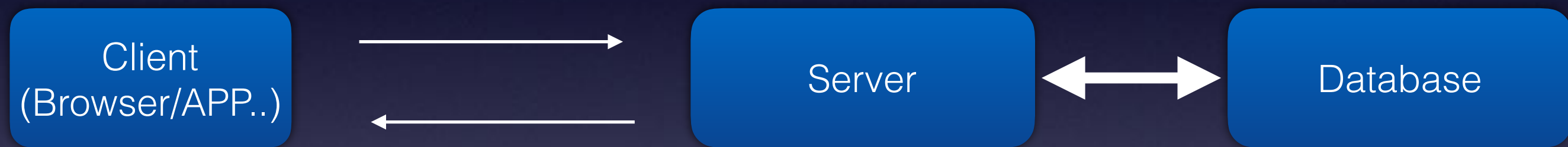
- Parallelism: 硬體在“單一時間點”多核同時執行
- Concurrency: “一段時間內”同時執行多項任務
  - multi-thread
  - single thread + asynchronous library
- Python 全域鎖(GIL: Global Interpreter Lock)
  - one active thread per python process
    - No parallelism (even multi-threaded)

# pip套件管理

- PyPI (Python Package Index)
  - Python的公開第三方套件庫
  - <https://pypi.python.org/pypi>
- 安裝第三方套件:
  - `pip install package_name`
- 反安裝:
  - `pip uninstall package_name`
- 查看安裝套件:
  - `pip list`

# Part II: 基礎HTTP觀念

# Common HTTP Architecture



Over TCP/IP

Connection closed after request

Use Keep-alive header to keep connections open

# HTTP Request Format

|                                                                                                                                                                                  |                 |                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------|
| GET /index.html HTTP/1.1                                                                                                                                                         | Request Line    | HTTP<br>Request |
| Date: Thu, 20 May 2004 21:12:55 GMT<br>Connection: close                                                                                                                         | General Headers |                 |
| Host: www.myfavoriteamazingsite.com<br>From: joeblow@somewebsitesomewhere.com<br>Accept: text/html, text/plain<br>User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) | Request Headers |                 |
|                                                                                                                                                                                  | Entity Headers  |                 |
|                                                                                                                                                                                  | Message Body    |                 |

# HTTP Response Format

|                                                                        |                  |               |
|------------------------------------------------------------------------|------------------|---------------|
| HTTP/1.1 200 OK                                                        | Status Line      | HTTP Response |
| Date: Thu, 20 May 2004 21:12:58 GMT                                    | General Headers  |               |
| Connection: close                                                      |                  |               |
| Server: Apache/1.3.27                                                  | Response Headers |               |
| Accept-Ranges: bytes                                                   |                  |               |
| Content-Type: text/html                                                | Entity Headers   |               |
| Content-Length: 170                                                    |                  |               |
| Last-Modified: Tue, 18 May 2004 10:14:49 GMT                           |                  |               |
| <html>                                                                 | Message Body     |               |
| <head>                                                                 |                  |               |
| <title>Welcome to the Amazing Site!</title>                            |                  |               |
| </head>                                                                |                  |               |
| <body>                                                                 |                  |               |
| <p>This site is under construction. Please come back later. Sorry!</p> |                  |               |
| </body>                                                                |                  |               |
| </html>                                                                |                  |               |

# HTTP Methods

- HEAD
  - 只取得資源的metadata, 不取得資源本文
- GET
  - 讀取資源.
  - 參數夾帶在url之後. ex. `http://serverurl/api?param1=abc&param2=efg`
    - 不要用來修改資料; 可能會有爬蟲程式來呼叫
- POST
  - 修改資源
- Others
  - PUT, DELETE, TRACE, CONNECT, PATCH



# JSON (JavaScript Object Notation)

- Represented by python “dict” type
  - `json.loads(json_string)` to dict\_object, `json.dumps(dict_object)` to json\_string
  - {
    - `"str_key":"bbb",`
    - `"int_key":1,`
    - `"array_key":[`
      - `{"some_key":123 },`
      - `{"some_key":456 }`
    - `],`
    - `"sub_doc_key":{`
      - `"mmm":"nnn",`
      - `"xxx":"yyy"`
    - `}`
  - `}`

# WSGI (Web Server Gateway Interface)

- 規範Python web server的request handler格式
- 執行環境繼承了傳統CGI變數，以及新增自定義的變數
  - 傳統CGI環境變數: REQUEST\_METHOD, QUERY\_STRING
  - 自定義變數: wsgi.version, wsgi.url\_scheme...

```
def simple_app(environ, start_response):
 status = '200 OK'
 response_headers = [('Content-type', 'text/plain')]
 start_response(status, response_headers)
 return ['Hello world!n']
```

- 練習: **wsgi.py**

# Part III: Python Web Server 函式庫

# Web development stack

- gunicorn + gevent + falcon
  - falcon: web API development framework
  - gevent: asynchronous coroutine library
  - gunicorn: web server binary

# gunicorn

- Python本身或部分python web framework 內建的web server不適合用來正式環境 (註解有寫)
  - 安全性不佳
  - 不能處理GIL對於多核心執行的限制
- gunicorn:
  - ported from Ruby's Unicorn project
  - 1 master process + N worker processes (pre-fork model)
    - 用multi process解決python GIL的限制

# run gunicorn server

- 練習測試
  - gunicorn -w 5 gunicorn:app
    - app: a WSGI compatible handler
- test workers
  - curl localhost:8000 some times

# Synchronous server

- 一個request完全處理完才處理下一個. 例如: 剪頭髮
  - gunicorn default mode
    - Good: CPU-bound applications (scientific computation)
    - Bad: High I/O bound applications (ex. DB read/write)
- 練習測試
  - `gunicorn -w 2 guni_sleep:app`
  - `gunicorn -w 1 guni_sleep:app`
    - 可測試同時開兩terminal request: `curl localhost:8000`

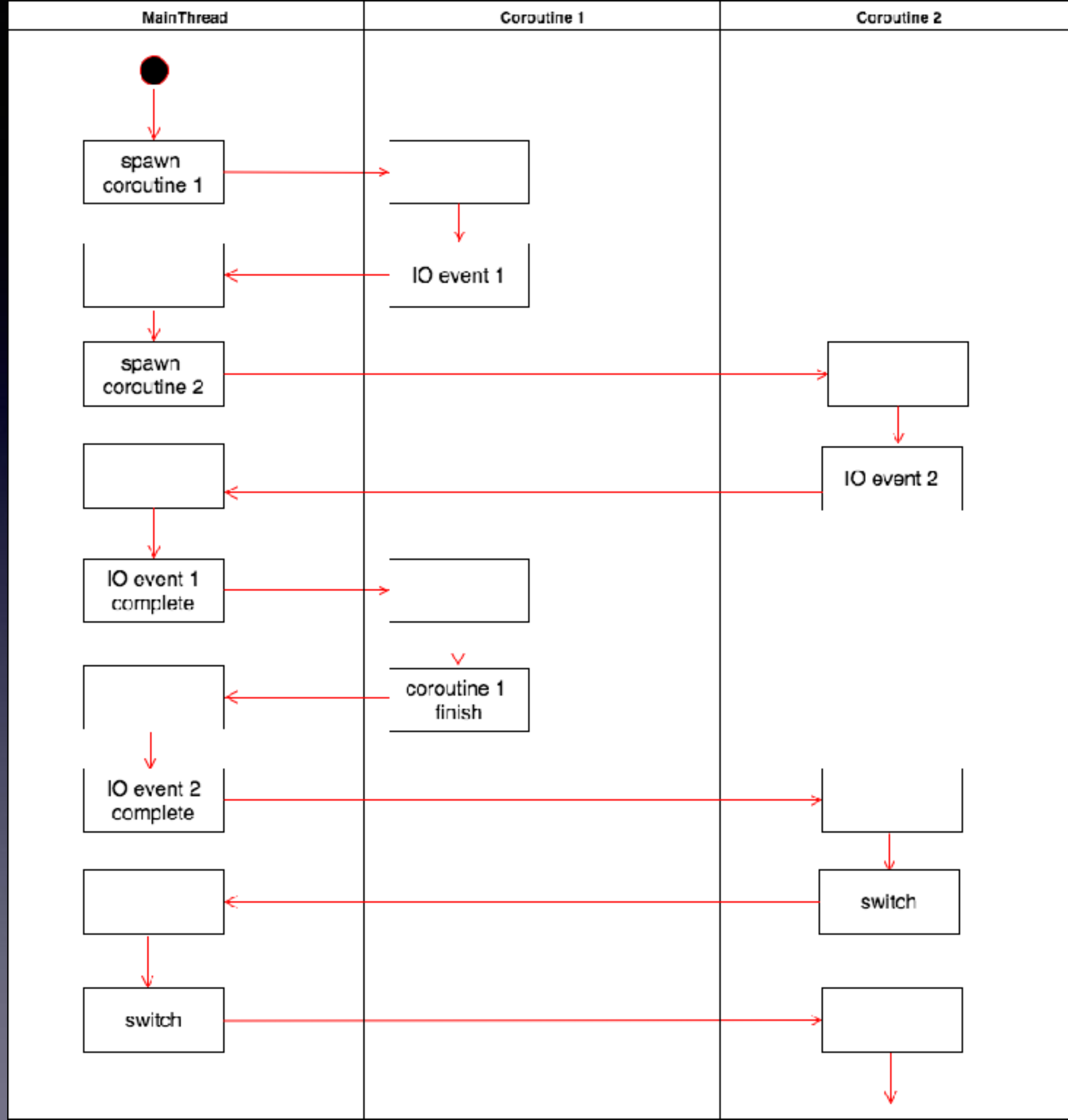
# Asynchronous Server

- 一個request處理過程中，還可以處理其他request
  - 例如: 老闆幫顧客A炸雞排，同時幫顧客B裝飲料
- gunicorn async mode:
  - **gevent: coroutine**
  - Others: tornado(event-loop) ...



# Coroutine

- Light-weight thread
- single thread
- no OS context switch overhead
  - 可以不透過OS自行切換控制權



# gevent

- Coroutine network library
  - based on greenlet library
  - *synchronous-style code that runs asynchronously*
- IO events trigger coroutine switch
  - Socket / file / database operations

# Gevent & coroutine

- **測試練習**
  - gr\_test.py: greenlet coroutine library
  - gevent\_test.py: gevent based on greenlet

# Asynchronous server with Gevent

- 每個request都是一個greenlet coroutine
- 練習測試
  - `gunicorn -w 1 asyn:app --worker-class gevent`

# Falcon

- a fast / minimal python web framework to build backend applications
- building web API easier
  - `on_get` / `on_post` / `on_delete` / `on_patch` ...

# Falcon API

- Falcon API initialize:  
`api_router = falcon.API()`  
`api_router.add_route('/', RootHandler())`
- `add_route` parameters:
  - 1. API路徑
  - 2. 實作GET/POST...的類別實體
- API implementation format: (post / patch...亦同)
  - `on_get(self, req, resp):` # req: dict型態，包含WSGI環境變數  
`resp.body = "Hello World!"`

# 練習

- 修改homework/server.py
- 使用falcon + gevent + gunicorn架一個後端伺服器，並新增下列web API
  - /account (GET , POST)
    - get自帶兩個參數account, nickname
    - post自帶兩個參數account, nickname並轉成json