Python網路程式設計

商用研四
謝明泰

Agenda

- Python基礎
  - 基礎語法
  - Concurrency & Parallelism
  - pip套件管理
- 基礎HTTP觀念
  - HTTP request/response format
  - HTTP METHODS
  - JSON
  - WSGI
- Python Web Server library
  - Gunicorn (Synchronous / Asynchronous server)
- Python Web Framework library
  - Coroutine & Gevent
  - Falcon

Part I: Python基礎

Python Basis

- 腳本語言
  - .py -> pyc (bytecode)
- 強制縮排: 不可混用
  - 4 space (recommended)
  - tab
- 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)
  - 2.x 社群太強大: 2.x還是三方函式庫支援主流 (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
- 整合IDE: PyCharm

Python Basis (Cont.)

- Garbage collection
- 跨平台
  - 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…

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

- 概念上
  - package: 包含多個modules
    - package目錄需要有__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)

- **project**

```
        |
        —main.py
                |
                lib—foo.py: def bar(): …
                   —__init__.py
```

Import Paths

- Ubuntu 14.04
  - 1. Input script 目錄 (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**

Variable

- 變數: 動態型別，指向記憶體的一個參考(name reference):
  - Mutable object: list , dictionary…
    - 修改參考將**直接影響**被指向的物件
      - 參考到的位址不變
  - Immutable object: string, number, tuple…
    - 修改參考只是把參考指到另一塊記憶體
      - 參考到的位址改變
- **練習:reference.py**

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.iteritems():
      ….

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2 (整數)
  - a = (2,2) # 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

    @staticmethod
    def bar(param1):
        ...

- x = Foo()
  print x.name
  Foo.bar()

Concurrency & Parallelism

- 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
    - 系統層面: 不會有blocking等待
- Python 全域鎖(GIL: Global Interpreter Lock)
    - one active thread per python process
        - No parallelism for single process (even multi-threaded)
        - Concurrency via async library (gevent)

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

HTTP Request Format

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

HTTP Request

HTTP Response Format

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

HTTP Response

HTTP Methods

- HEAD
  - 只取得資源的metadata, 不取得資源本文
- GET
  - 讀取資源
    - 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
    - 用multi process解決python GIL的限制

run gunicorn server

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

Synchronous server

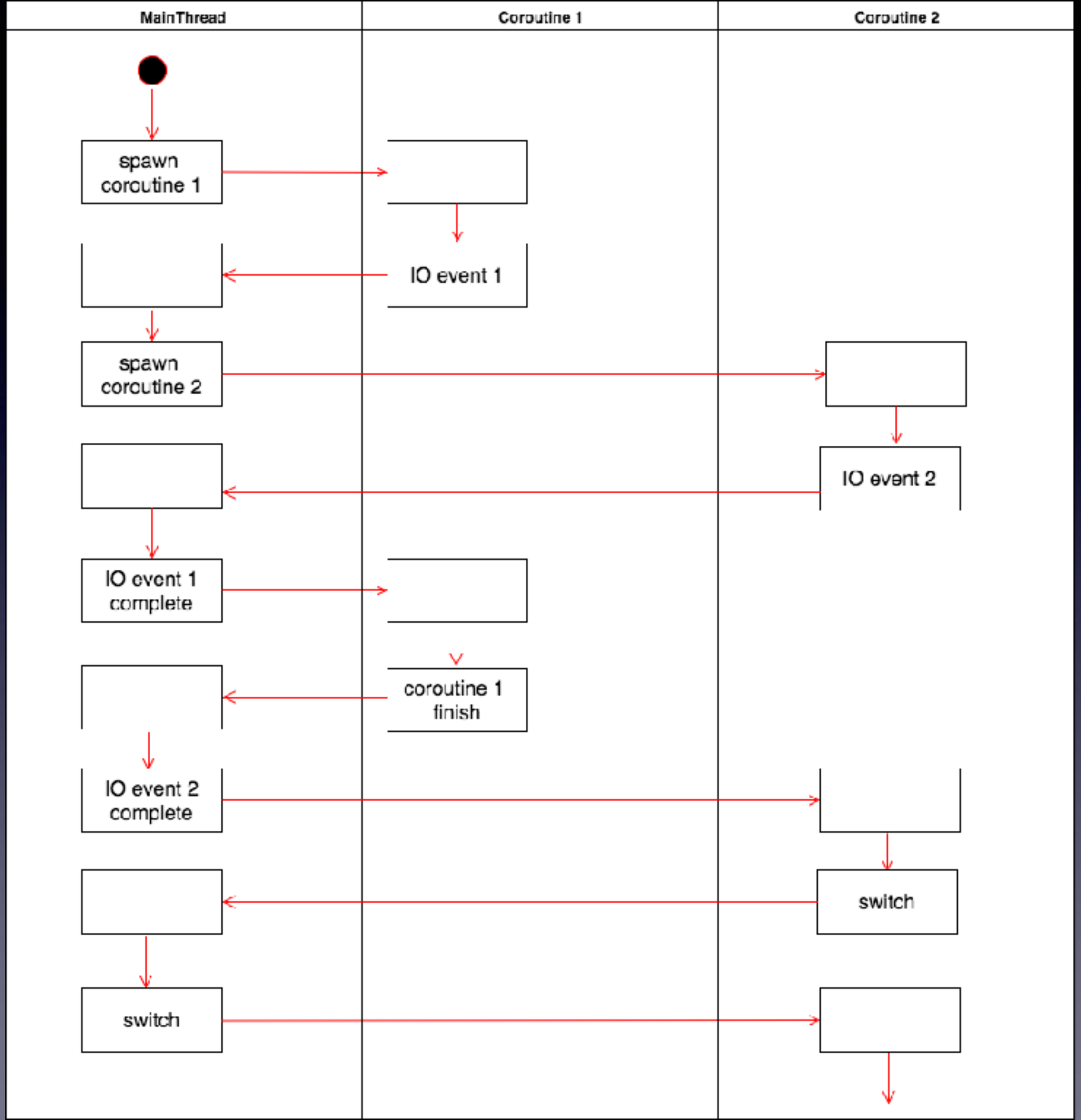
- 一個request完全處理完才處理下一個. 例如: 剪頭髮
  - gunicorn預設模式
  - 也是某些web framework預設內建server的模式
    - 對於需要跟DB溝通的遊戲伺服器來說太慢
- 練習測試
  - gunicorn -w 2 guni_sleep:app
  - gunicorn -w 1 guni_sleep:app
    - 同時開兩個terminal request: curl localhost:8000

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可以搭配非同步函式庫運作
  - gevent, tornado …
- **練習測試**
  - gunicorn -w 1 asyn:app --worker-class gevent

Coroutine

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


MainThread
Coroutine 1
Coroutine 2
spawn coroutine 1
IO event 1
spawn coroutine 2
IO event 2
IO event 1 complete
coroutine 1 finish
IO event 2 complete
switch
switch

gevent

- based on greenlet
  - synchronous code that runs asynchronously
- gevent http server
  - one coroutine / per http request
  - socket operation (DB access) = IO event

Gevent & coroutine

- **測試練習**

- gr_test.py: greenlet coroutine library
- gevent_test.py: gevent based on greenlet

Falcon

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

Falcon API

- Falcon API初始化:
  api_router = falcon.API()
  api_router.add_route('/', RootHandler())
    - add_route參數: API路徑，實作GET/POST...的類別實體
- API implementation format:
    - on_get(self, req, resp):
      resp.body ="Hello World!”
      # req: dictionary，帶上WSGI環境變數
      # resp: 要回傳的資料容器類別
    - on_post / on_patch /on_delete亦同
- See example code in server.py