Python Tutorial 4

socket, Concurrency, HTTP

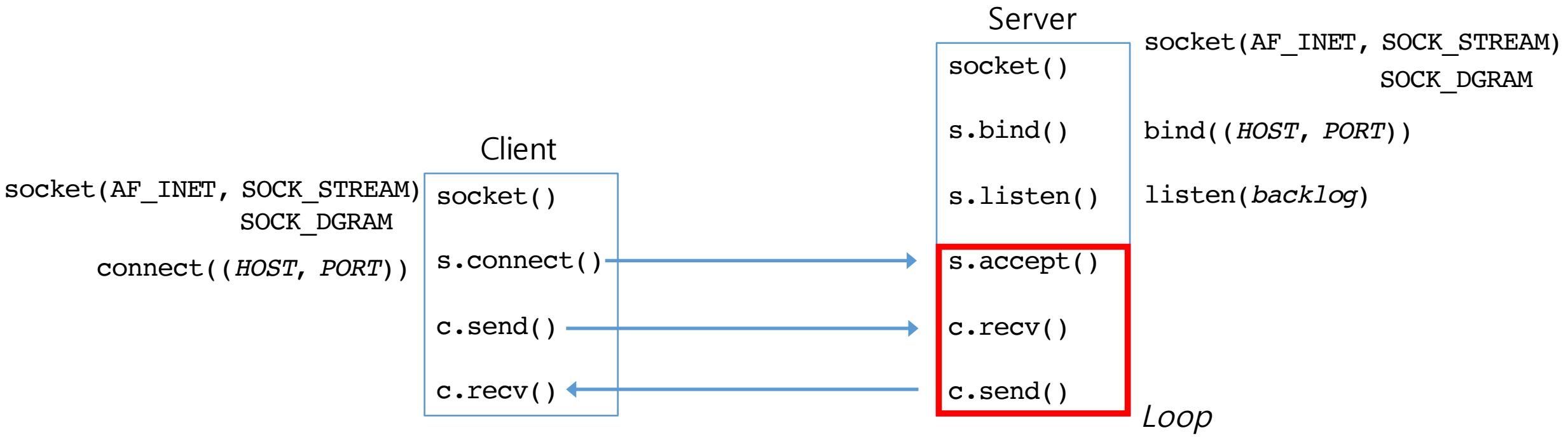
Haesun Park, haesunrpark@gmail.com,

1. socket
2. Concurrency
3. HTTP
4. example

socket

raw socket: server

- UNIX/BSD 계열의 소켓 인터페이스를 그대로 승계하고 있습니다.
- import socket



raw socket example

- 간단한 에코(echo) 프로그램
- 쥬피터 노트북으로 데모

Client

```
import socket

HOST = '127.0.0.1'
PORT = 33300

with socket.socket() as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world')
    data = s.recv(1024)
    print('Received', repr(data))
```

서버 접속

데이터 교환

Server

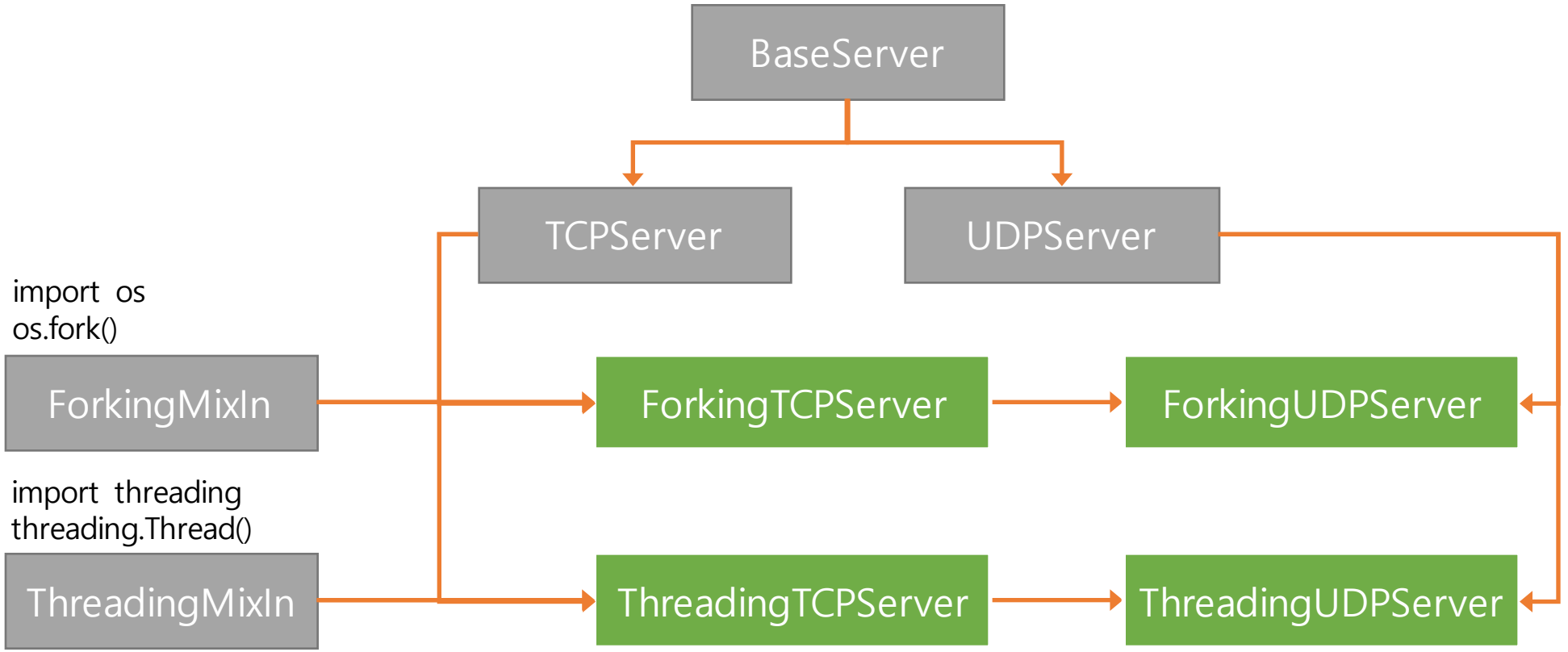
```
import socket

HOST = '127.0.0.1'
PORT = 33300

with socket.socket() as s:
    s.bind((HOST, PORT))
    s.listen(1)

    while True:
        conn, addr = s.accept()
        with conn:
            print('connect :', addr)

            while True:
                data = conn.recv(1024)
                if not data:
                    break
                conn.sendall(data)
```

아이피와 포트 점유

해당 포트의 접속을 대기

socketserver

- 네트워크 서버를 만들기 쉽도록 미리 제공되는 모듈입니다.
- TCP와 UDP 서비스를 위한 클래스가 있습니다.
- ForkingMixIn, ThreadingMixIn과 함께 사용하여 TCP와 UDP에 대해 각각 프로세스방식과 스레드 방식의 멀티플렉싱을 지원합니다.



ThreadingTCPServer

- ThreadingTCPServer는 TCPServer와 ThreadingMixIn을 상속받은 클래스입니다.
- 호스트와 포트를 지정하고 데이터 처리를 담당할 핸들러를 정의하여 서버를 만듭니다.
- 서버의 실행을 스레드 클래스에 위임하고 스레드를 시작합니다.
- 쥬피터 노트북으로 데모

Client

```
import socket

HOST = '127.0.0.1'
PORT = 33300

with socket.socket() as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world')
    data = s.recv(1024)
    print('Received', repr(data))
```

처리 위임

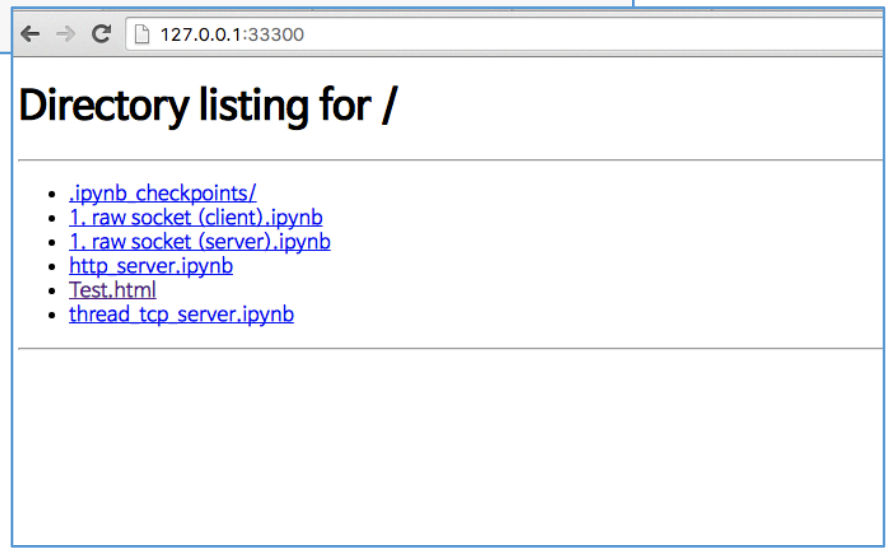
Server

```
import socket
import socketserver
import threading

HOST = '127.0.0.1'
PORT = 33300

class MyRequestHandler(socketserver.BaseRequestHandler):

    def handle(self):
        data = self.request.recv(1024)
        print(data)
        self.request.sendall(data)

server = socketserver.ThreadingTCPServer((HOST, PORT), MyRequestHandler)
server_thread = threading.Thread(target=server.serve_forever)
server_thread.start()
```

http.server

- socketserver.TCPServer의 서브클래스 입니다.
- 간단한 HTTP 서비스를 제공합니다.
- 서버를 시작하고 브라우저로 노트북이 위치한 디렉토리 목록과 파일을 열 수 있습니다.
- 쥬피터 노트북으로 데모

```
import http.server
import socketserver

HOST = '127.0.0.1'
PORT = 33300

httpd = socketserver.TCPServer((HOST, PORT), http.server.SimpleHTTPRequestHandler)
httpd.serve_forever()
```



http.client

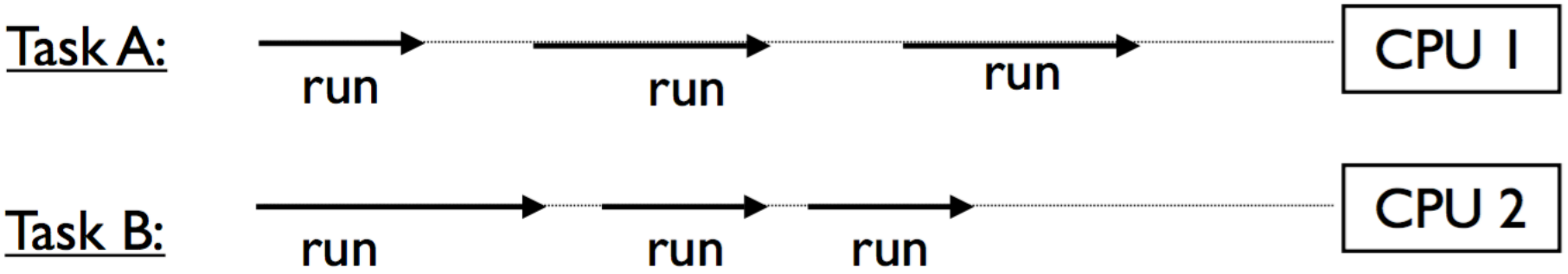
- 이 모듈을 직접 사용하지 않고 urllib.request 를 사용하는 것이 보통입니다.
- HTTP 프로토콜을 이용하여 서버와 데이터를 주고 받을 수 있는 클라이언트 프로그램을 만들 때 사용합니다.
- 쥬피터 노트북으로 데모

```
import http.client

HOST = '127.0.0.1'
PORT = 33300

conn = http.client.HTTPConnection('127.0.0.1', 33300)
conn.request("GET","/Test.html")

r = conn.getresponse()
print(r.status, r.reason)

r.read(200)
```

Concurrency

Parallel Processing

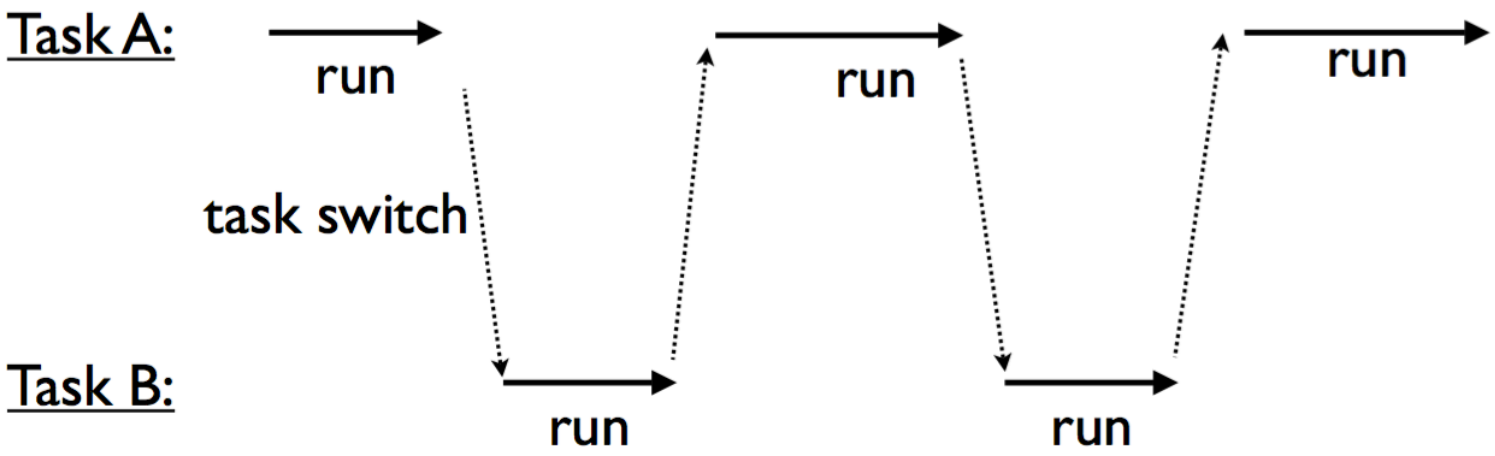
- You may have parallelism (many CPUs)
- Here, you often get simultaneous task execution



- Note: If the total number of tasks exceeds the number of CPUs, then each CPU also multitasks

Multitasking

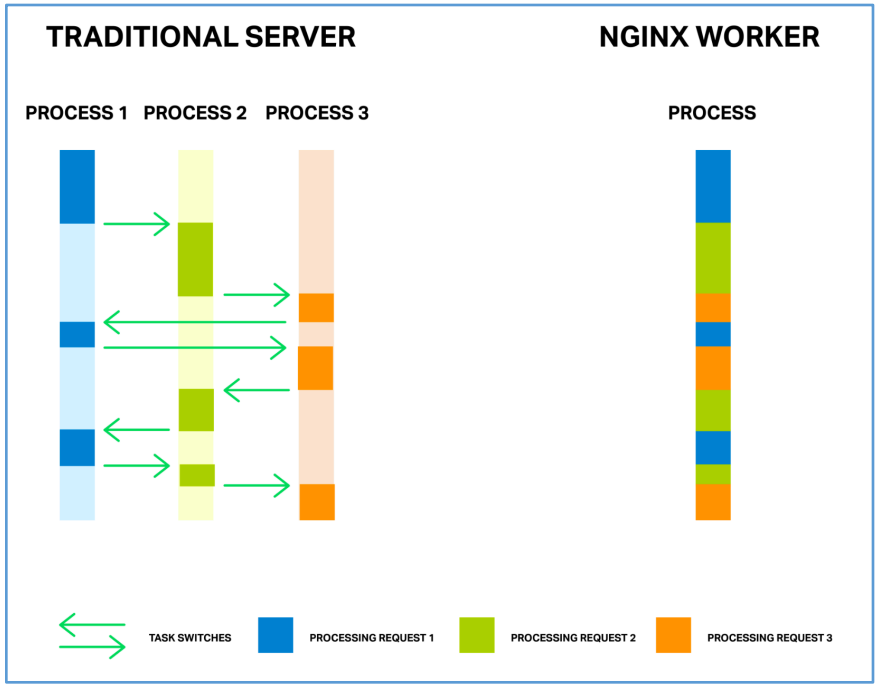
- Concurrency typically implies "multitasking"



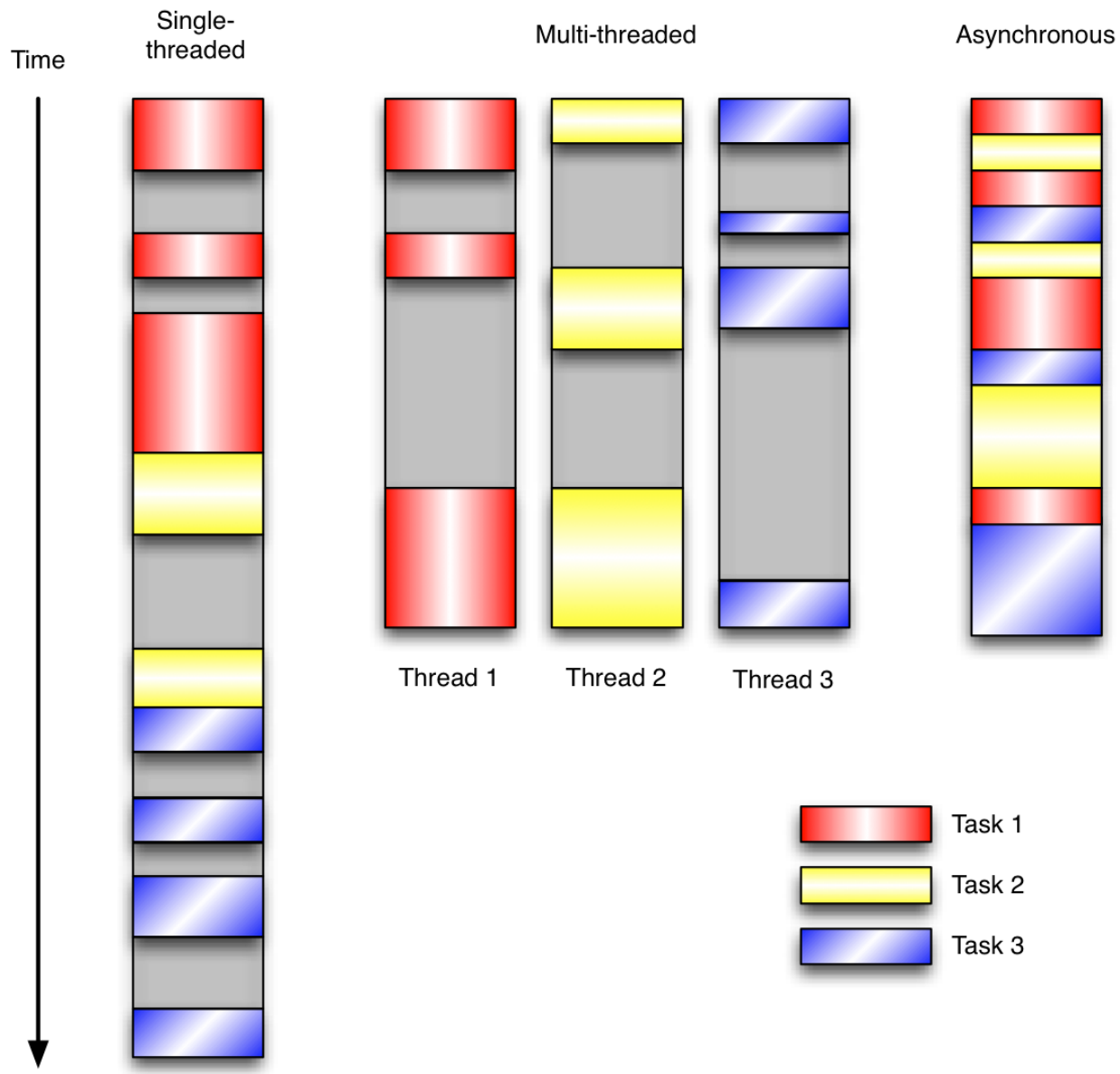
- If only one CPU is available, the only way it can run multiple tasks is by rapidly switching between them

Cost

- 새로운 클라이언트가 서버에 접속할 때 마다 스레드를 만들거나 프로세스를 포킹 해야 합니다.
- 만약 수천 수만개의 클라이언트가 접속을 한다면.. (C10K Problem)
- 스레드와 프로세스 안에서 여러개의 블럭킹 IO를 효율적으로 대기할 수 있어야 합니다.
- 예) 비동기 처리를 하는 Nginx가 Apache보다 성능이 뛰어납니다.



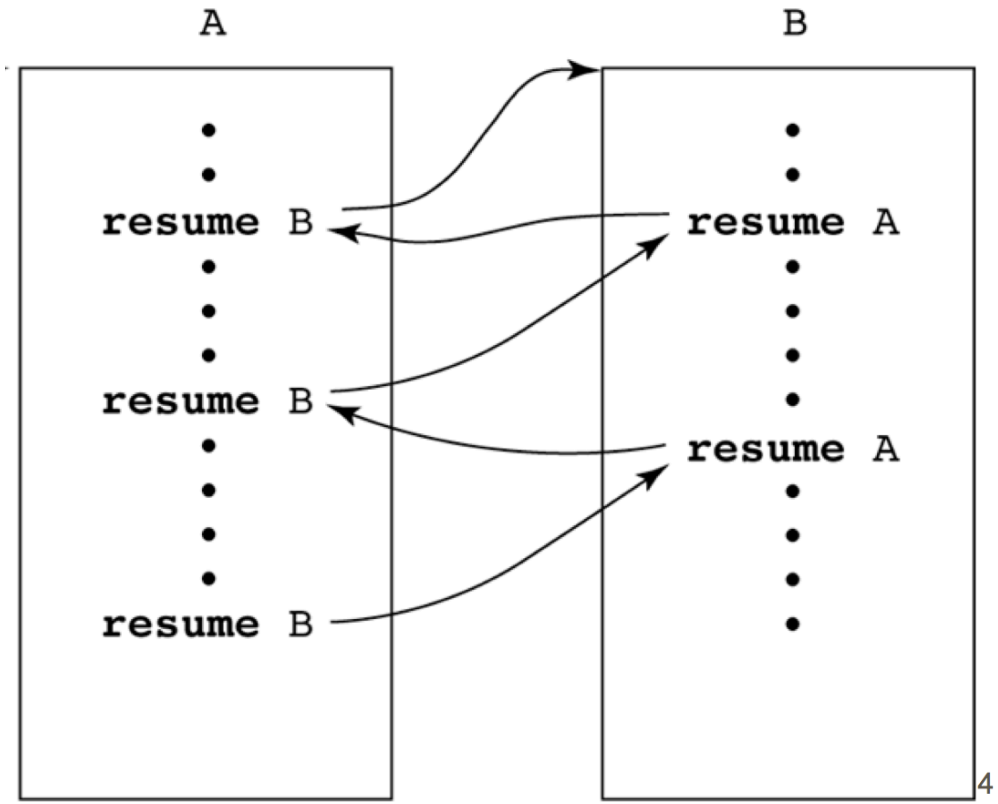
Thread vs Event driven



Coroutine

- yield 기능으로 함수의 중간에서 멈추고 다른 함수를 실행할 수 있습니다.
- 즉 함수의 진입점을 여러개로 가질 수 있습니다.
- 스레드 스위칭 보다 빠르게 여러개의 함수를 동시에 실행할 수 있습니다.
- 스레드나 프로세스 간의 공유 자원에 대한 경쟁/동기화가 없습니다.

- C# : await
- Javascript : yield
- Lua



asyncio

- python3 에 추가된 기능입니다.
- 코루틴(coroutine)을 사용한 비동기 처리를 지원하고 이벤트 스케줄링을 제공합니다.

- aiohttp
- aioredis
- aiohttp
- aiozmq

listed at http://asyncio.org/

```
import asyncio
import datetime

@asyncio.coroutine
def display_date():
    dt = datetime.datetime
    end_time = dt.now().timestamp() + 5.0
    while True:
        dt = datetime.datetime
        print(dt.now())
        if (dt.now().timestamp() + 1.0) >= end_time:
            break
        yield from asyncio.sleep(1)

loop = asyncio.get_event_loop()
loop.run_until_complete(display_date())
loop.close()
```

aiohttp 설치

- Anaconda Prompt 를 실행하여 *pip install aiohttp* 명령 입력

```
(C:₩Users₩Ricky win7 32bit₩Anaconda3) C:₩Users₩Ricky win7 32bit>pip install aioh
ttp
Collecting aiohttp
  Downloading aiohttp-0.22.1-cp35-cp35m-win32.whl (129kB)
    100% |################################| 133kB 767kB/s
Collecting multidict>=1.1.0 (from aiohttp)
  Downloading multidict-1.2.0-cp35-cp35m-win32.whl (123kB)
    100% |################################| 133kB 3.1MB/s
Collecting chardet (from aiohttp)
  Downloading chardet-2.3.0.tar.gz (164kB)
    100% |################################| 174kB 2.0MB/s
Building wheels for collected packages: chardet
  Running setup.py bdist_wheel for chardet ... done
  Stored in directory: C:₩Users₩Ricky win7 32bit₩AppData₩Local₩pip₩Cache₩wheels₩
28₩8c₩bf₩a69199bd4901d84e13362f95a9ea7bc9a691fed2d655a90bc4
Successfully built chardet
Installing collected packages: multidict, chardet, aiohttp
Successfully installed aiohttp-0.22.1 chardet-2.3.0 multidict-1.2.0

(C:₩Users₩Ricky win7 32bit₩Anaconda3) C:₩Users₩Ricky win7 32bit>ipython
Python 3.5.1 |Anaconda 4.1.0 (32-bit)| (default, Jun 15 2016, 15:33:59) [MSC v.1
900 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import aiohttp

In [2]:
```

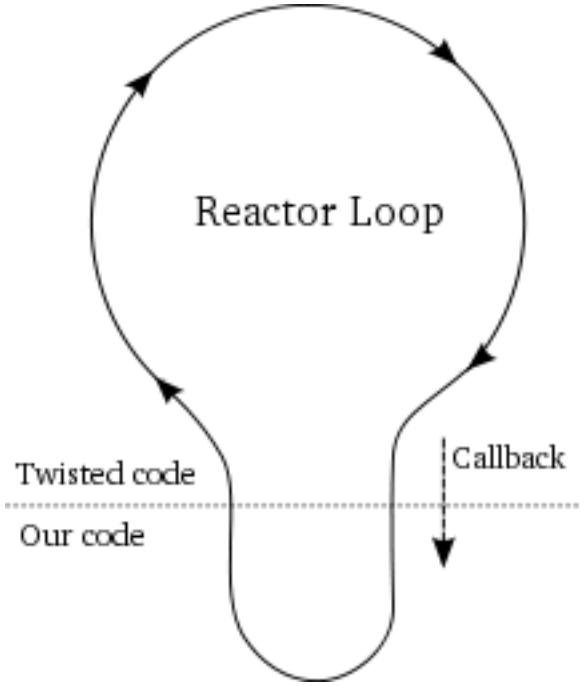
aiohttp: asyncio client & server

- http://aiohttp.readthedocs.io/

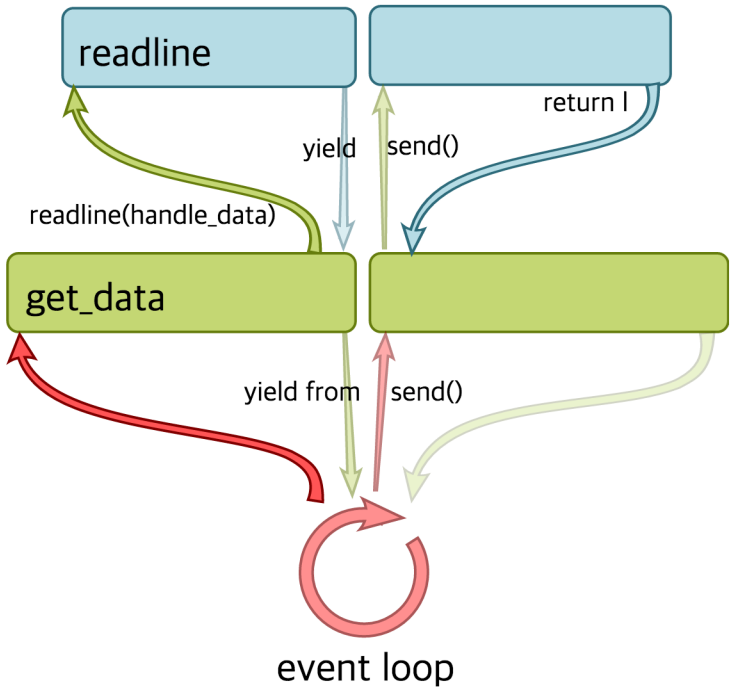
```
import aiohttp
import asyncio

@asyncio.coroutine
def fetch(url):
    print('Start', url)
    res = yield from aiohttp.request('GET', url)
    data = yield from res.read()
    print(len(data))
    print('Done', url)

@asyncio.coroutine
def fetch_all(urls):
    fetches = [asyncio.Task(fetch(url)) for url in urls]
    yield from asyncio.gather(*fetches)

urls = ['http://naver.com', 'https://google.com', 'https://apple.com']

asyncio.get_event_loop().run_until_complete(fetch_all(urls))
```

그밖에

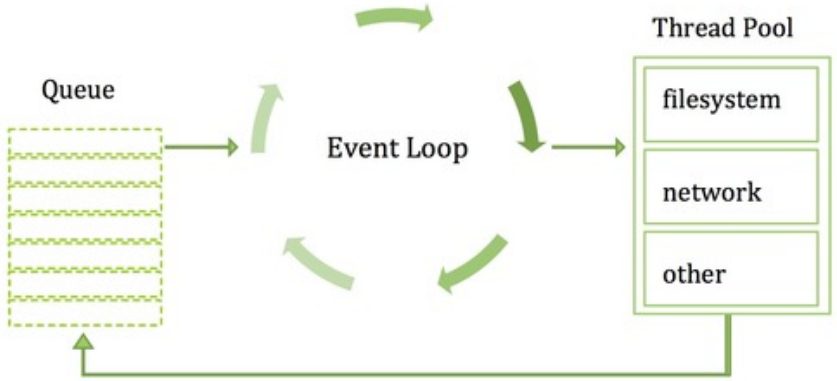
- Twisted: Reactor



- Gevent: libevent



- Tornado: Event Driven



Gunicorn, uWSGI, ...

HTTP

urllib.request

- HTTP 통신과 관련된 여러 복잡한 문제(인증, 쿠키, 청크, 리디렉션 등)을 처리하기 위한 고수준 클래스 입니다.
- urlopen 메소드로 인터넷 리소스를 오픈합니다.
- urlopen의 리턴 값은 http.client.HTTPResponse 클래스의 객체입니다.
- HTTPResponse 의 주요 메소드와 속성
  - read(*n*) : 바이트(n)가 지정되지 않으면 전체 내용을 읽습니다
  - readline() : 한줄씩 읽어서 리턴합니다
  - getheader(*name*) : name에 지정된 헤더를 리턴합니다.
  - getheaders() : 전체 HTTP 헤더를 리스트 만들어 리턴합니다.
  - status : HTTP 상태 코드 (200, 404 등)
  - reason : HTTP 코드 정보 (OK, NOT FOUND 등)
- 쥬피터 노트북으로 데모

```
import urllib.request
with urllib.request.urlopen('http://www.python.org/') as r:
    print(r.read())
```

urllib.parse

- urlparse 메소드는 URL 의 여러 요소를 파싱하는데 사용합니다.
- scheme: http, https, ftp 등
- netloc: 도메인과 포트
- path: URL의 경로 부분
- query: ? 이후의 쿼리스트링 파라메타
- fragment: # 이후의 키워드
- urlparse의 반대 역할을 하는 urlunparse와 urljoin 메소드도 있습니다.
- parse_qs 는 쿼리스트링을 딕셔너리 형태로 parse_qsl 은 리스트 형태로 변경시켜 줍니다.
- 쥬피터 노트북으로 데모

```
from urllib.parse import urlparse, urlunparse
o = urlparse('http://sports.news.naver.com:80/wfootball/news/read.nhn?oid=413&aid=0000036958#news')
print(type(o))
print(o.scheme)
print(o.netloc)
print(o.path)
print(o.query)
print(o.fragment)
print(urlunparse(o))
```

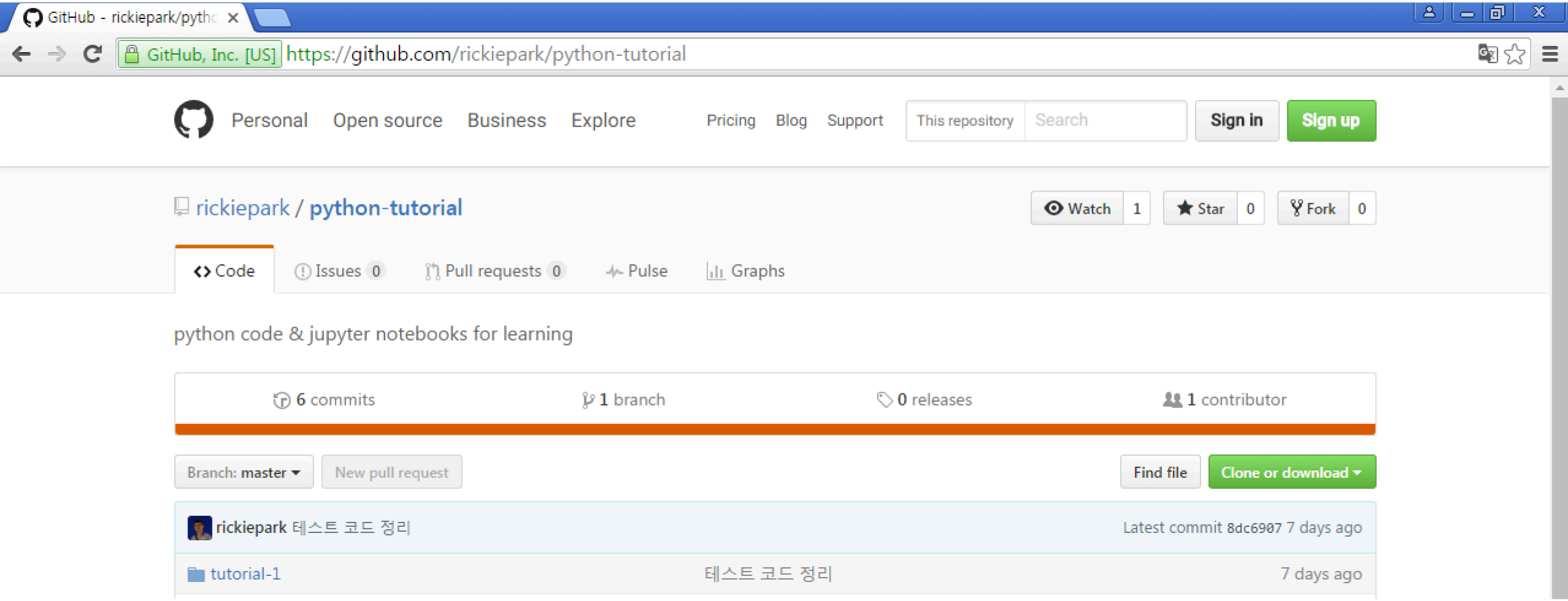
requests

- python-requests.org
- 아나콘다 배포판에 기본적으로 설치되어 있습니다.
- 쉽고 직관적인 인터페이스, 다양한 기능을 제공합니다.
- RESTful API를 사용하기 편리하도록 get, post, del, put 메소드를 제공합니다.
- 쿼리스트링, 헤더, 쿠키, 타임아웃 등의 기본 옵션을 제공합니다.
- 쥬피터 노트북으로 데모

```
import requests
endpoint = 'https://raw.githubusercontent.com/rickiepark/python-tutorial/master/tutorial-4/sample.json'
r = requests.get(endpoint)
print(r.status_code)
print(r.headers)
print(r.text)
print(r.json())
```

Setup & Example

Github Download



내 문서 밑으로 압축 해제

웹 크롤링 클래스

- 웹 페이지를 크롤링하는 클래스를 작성합니다.
  예) class WebFetch(object): …
- 클래스는 사용자에게 제공하는 Public API 를 제공합니다.
  예) fetch = WebFetch()
  html = fetch.get_page('http://www.naver.com/')
- 여러 페이지를 입력받아 차례로 혹은 비동기적으로 웹 페이지를 읽어옵니다.
- 가져온 웹 페이지를 URL의 마지작 이름으로 파일로 저장합니다.
  예) http://news.naver.com/main/main.nhn → main.nhn.html

Q&A

Any Question: haesunrpark@gmail.com

github URL: https://github.com/rickiepark/python-tutorial

This slide available at
http://tensorflowkorea.wordpress.com