Celery

Python에서 사용하는 비동기 작업 큐로 message broker가 python process에 연결해서 process를 컨트롤 해줍니다. message broker로는 redis를 사용합니다.

reference - http://www.celeryproject.org/

1. install redis & celery

mac

$ pip3 install celery redis

$ brew install redis

$ redis-server

window

32비트는 메모리를 4G까지만 지원하기 때문에 64비트만 지원합니다. Redis에서는 공식적으로 windows OS를 지원하지 않습니다.

https://github.com/MSOpenTech/redis/releases 에서 msi 파일 다운로드 및 설치

C:\ProgramFiles\Redis> redis-cli.exe

127.0.0.1:6379> shutdown

not connected> exit

C:\ProgramFiles\Redis> redis.server.exe

celery, redis package 설치

celery 4.x 버전은 windows를 지원하지 않습니다.

C:\User\jin> conda install -c conda-forge celery==3.1.25

C:\User\jin> conda install -c conda-forge redis-py

2. test redis

```
import redis

r = redis.StrictRedis(host='localhost', port=6379, db=0)

# set key and value
result = r.set('foo', 'bar')
print("set", result)

# get value decode ascii
result = r.get('foo')
print("get", result)
result = result.decode('ascii')
print("get(after decode)", result)
```

---

3. make task.py

```
%%writefile task.py

from celery import Celery

BROKER_URL = 'redis://localhost:6379/0'
CELERY_RESULT_BACKEND = 'redis://localhost:6379/0'

app = Celery('task', broker=BROKER_URL, backend=CELERY_RESULT_BACKEND)
```

```
@app.task
def add(a, b):
        return a + b
```

---

4. start celery

mac

```
$ celery -A task worker
```

windows

```
$ celery -A task worker -l info
```

5. test celery

```
import task

result = task.add.delay(2,3)
print(result.ready())
result.get()
```

---

6. test celery - prime number

task.py

```
%%writefile task.py

from celery import Celery
```

```
BROKER_URL = 'redis://localhost:6379/0'
CELERY_RESULT_BACKEND = 'redis://localhost:6379/0'

app = Celery('task', broker=BROKER_URL, backend=CELERY_RESULT_BACKEND)

@app.task
def prime_number(n):
    prime_count = 0
    for num1 in range(1, n+1):
        is_prime = True
        for num2 in range(2, num1):
            if num1 % num2 == 0:
                is_prime = False
        if is_prime:
            prime_count += 1
    return prime_count
```

```
import task

result_1 = task.prime_number.delay(20000)
result_2 = task.prime_number.delay(10000)

import time

is_done_1, is_done_2 = False, False
```

```
r_1, r_2 = 0, 0
count = 0


while (not is_done_1) or (not is_done_2):
    is_done_1 = result_1.ready()
    is_done_2 = result_2.ready()
    time.sleep(1)
    count += 1
    print("{} sec : done1-{}, done2-{}, r_1-{}, r_2-{}".format(count, is_done_1,
is_done_2, r_1, r_2))
    if is_done_1:
        r_1 = result_1.get()
    if is_done_2:
        r_2 = result_2.get()


result_1.get(), result_2.get()
```
