Python for data analysis

2017년 11월 16일

이병철

테스트 환경

- Anaconda 64 bit windows 용
- Jupyter, numpy 등을 conda 를 이용하여 설치
- Env 를 jupyter 로 이름 지정

> activate jupyter

하에서 실행

3.4.2 %time, %timeit

```python
strings=['foo','foobar','baz','qux','python','Guido Van rossum','scari'] * 100000

def method1(strings):
    [x for x in strings if x.startswith('foo')]
def method2(strings):
    [x for x in strings if x[:3]=='foo']
```

코드를 위와 같이 작성하여 ipython에서 실행 결과

%time: 한번 실행
%timeit: 여러 번 하여 통계치를 줌

책과 달리 CPU time 이 나오지 않음 (아마도 windows 에서 실행한 결과 인 듯)
(Ipython 에서 %timeit? 하면 여러 옵션이 나옴)

```
In [5]: %run 3.4.2.py

In [6]:

In [6]:

In [6]:

In [6]:

In [6]: %time method1(strings)
Wall time: 127 ms

In [7]: %time method1(strings)
Wall time: 107 ms

In [8]: %time method1(strings)
Wall time: 107 ms

In [9]: %time method2(strings)
Wall time: 104 ms

In [10]: %time method2(strings)
Wall time: 97 ms

In [11]: %time method2(strings)
Wall time: 92 ms

In [12]: %timeit method1(strings)
10 loops, best of 3: 92.6 ms per loop

In [13]: %timeit method2(strings)
10 loops, best of 3: 77.2 ms per loop
```

3.4.3 %prun, %run -p

```python
import numpy as np
from numpy.linalg import eigvals

## in python3.6, range is xrange of python 2.x
def run_experiment(niter=100):
    K = 100
    results=[]
    for _ in range(niter):
        mat = np.random.randn(K
        max_eigenvalue = np.abs
        results.append(max_eige
    return results

if __name__=="__main__":
    some_results = run_experime
    print ('Largest one we saw:
```

```
In [26]: %prun -l 7 -s cumulative p3_4_3.py
 *************************************************************************
AttributeError                            Traceback (most recent call last)
<ipython-input-26-fc436eb1c24a> in <module>()
----> 1 get_ipython().magic('prun -l 7 -s cumulative p3_4_3.py')

C:\Anaconda\envs\jupyter\lib\site-packages\IPython\core\interactiveshell.py in
magic(self, arg_s)
In [27]: %prun -l 7 -s cumulative run_experiment()
         3804 function calls in 0.399 seconds

   Ordered by: cumulative time
   List reduced from 31 to 7 due to restriction <7>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.399    0.399 {built-in method builtins.exec}
        1    0.000    0.000    0.399    0.399 <string>:1(<module>)
        1    0.001    0.001    0.399    0.399 3.4.3.py:5(run_experiment)
      100    0.366    0.004    0.370    0.004 linalg.py:819(eigvals)
      100    0.028    0.000    0.028    0.000 {method 'randn' of 'mtrand.RandomS
tate' objects}
      300    0.002    0.000    0.002    0.000 {method 'reduce' of 'numpy.ufunc'
objects}
      100    0.001    0.000    0.001    0.000 linalg.py:214(_assertFinite)
```

%prun 은 함수에 대해서 사용

3.4.3 %prun, %run -p

```python
import numpy as np
from numpy.linalg import eigvals

## in python3.6, range is xrange of python 2.x
def run_experiment(niter=100):
    K = 100
    results=[]
    for _ in range(niter):
        mat = np.random.randn(K,K)
        max_eigenvalue = np.abs(eigvals(mat)).max()
        results.append(max_eigenvalue)
    return results

if __name__=="__main__":
    some_results = run_experimen
    print ('Largest one we saw:
```

```
In [29]: %run -p -s cumulative p3_4_3.py
Largest one we saw: 11.6731217933
         3888 function calls (3887 primitive calls) in 0.406 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      2/1    0.000    0.000    0.406    0.406 {built-in method builtins.exec}
        1    0.000    0.000    0.406    0.406 <string>:1(<module>)
        1    0.000    0.000    0.406    0.406 interactiveshell.py:2417(safe_ex
file)
        1    0.000    0.000    0.406    0.406 py3compat.py:182(execfile)
        1    0.000    0.000    0.405    0.405 p3_4_3.py:1(<module>)
        1    0.001    0.001    0.404    0.404 p3_4_3.py:5(run_experiment)
      100    0.370    0.004    0.374    0.004 linalg.py:819(eigvals)
      100    0.029    0.000    0.029    0.000 {method 'randn' of 'mtrand.Rando
tate' objects}
```

%run -p 은 script 에 대해서 사용

%lprun

- Ipython profile 에서 line_profiler 를 먼저 넣어 주어야 한다. 다음과 같은 명령어로 기본 configure 파일 위치를 알 수 있다. (참고: https://ipython.org/ipython-doc/3/config/intro.html)

```
(C:\Anaconda\envs\jupyter) D:\git_clone\selfstudy\20171115>ipython locate profil
e
C:\Users\이병철.MACROGEN\.ipython\profile_default
```

- 그리고 현재의 환경에서 line_profiler 를 설치해 준다
  - Anaconda navigator 에서 line_profiler 검색 또는
  - > conda install line_profiler
- 그 후 ipython_config.py 를 열고 아래 행을 추가

```
c.TerminalIPythonApp.extensions = ['line_profiler']
```

%lprun

%lprun

```
In [5]: %lprun -f add_and_sum -f call_function call_function()
Timer unit: 3.00467e-07 s

Total time: 0.00410918 s
File: D:¥git_clone¥selfstudy¥20171115¥p3_4_4.py
Function: add_and_sum at line 3

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     3                                           def add_and_sum(x,y):
     4         1        10452  10452.0     76.4      added = x+y
     5         1         3222   3222.0     23.6      summed = added.sum(axis=1)
     6         1            2      2.0      0.0      return summed

Total time: 0.071418 s
File: D:¥git_clone¥selfstudy¥20171115¥p3_4_4.py
Function: call_function at line 8

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     8                                           def call_function():
     9         1       135387 135387.0     57.0      x=randn(1000,1000)
    10         1        87206  87206.0     36.7      y=randn(1000,1000)
    11         1        15097  15097.0      6.4      return add_and_sum(x,y)
```

%prun

```
In [7]:  %prun -l 7 -s cumulative call_function()
         10 function calls in 0.062 seconds

   Ordered by: cumulative time
   List reduced from 9 to 7 due to restriction <7>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.062    0.062 {built-in method builtins.exec}
        1    0.001    0.001    0.062    0.062 <string>:1(<module>)
        1    0.001    0.001    0.061    0.061 p3_4_4.py:8(call_function)
        2    0.057    0.029    0.057    0.029 {method 'randn' of 'mtrand.RandomS
tate' objects}
        1    0.002    0.002    0.003    0.003 p3_4_4.py:3(add_and_sum)
        1    0.000    0.000    0.001    0.001 {method 'sum' of 'numpy.ndarray' o
bjects}
        1    0.000    0.000    0.001    0.001 _methods.py:31(_sum)
```

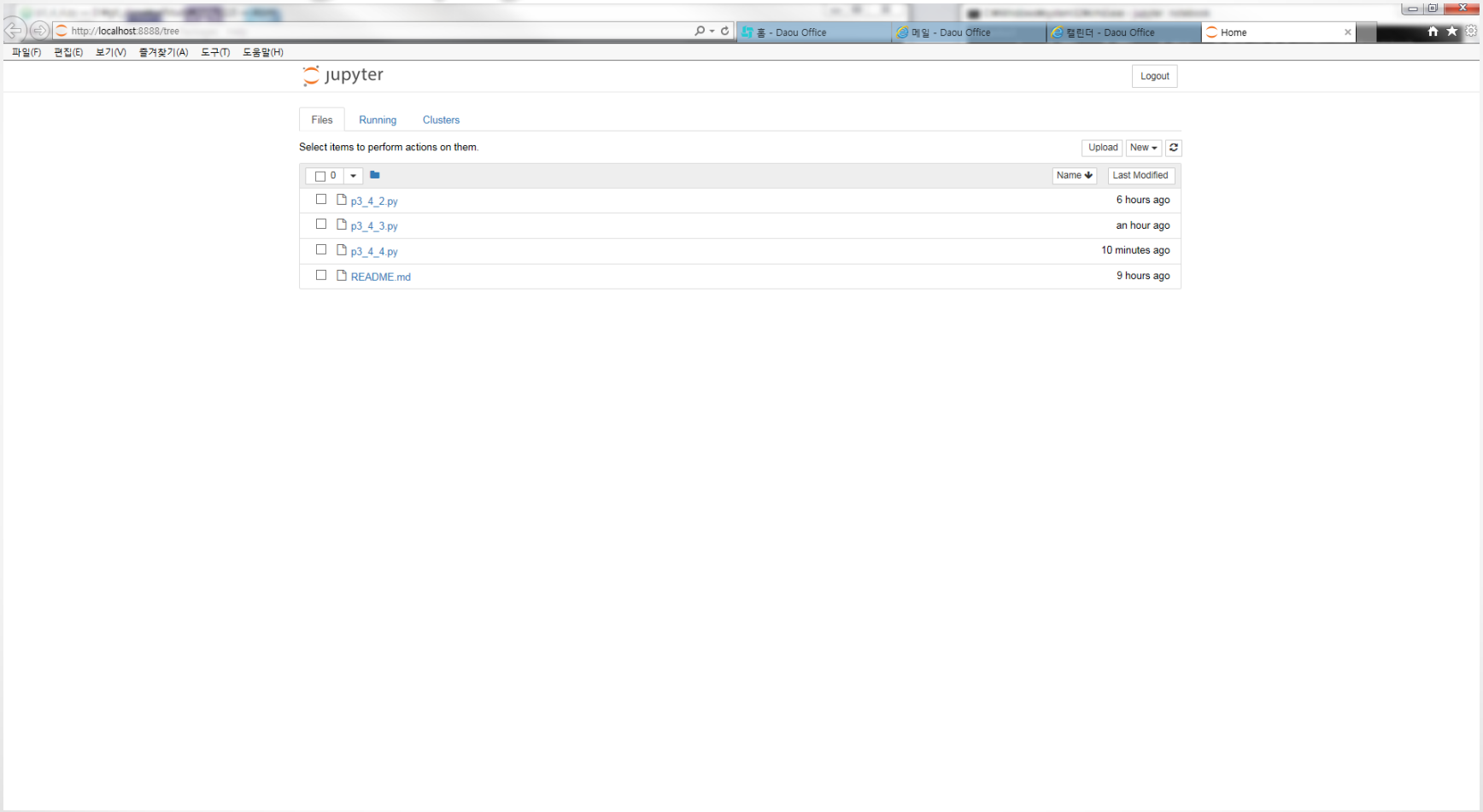
Jupyter nobebook

- 책에 있는 ipython notebook 은 jupyter 라는 이름으로 대체되었음

```
(C:\Anaconda\envs\jupyter) D:\git_clone\selfstudy\20171115>ipython notebook --py
lab=inline
[TerminalIPythonApp] WARNING | Subcommand `ipython notebook` is deprecated and w
ill be removed in future versions.
[TerminalIPythonApp] WARNING | You likely want to use `jupyter notebook` in the
future
[IE 21:35:24.730 NotebookApp] Support for specifying --pylab on the command line
has been removed.
[IE 21:35:24.731 NotebookApp] Please use `%pylab inline` or `%matplotlib inline`
in the notebook itself.
```

- --pylab=inline 이라는 옵션도 jupyter 안에서 직접 사용하라고 되어있음

> jupyter notebook



고맙습니다