Магические команды Jupiter Notebook (отладка, профилирование)

Практикум на ЭВМ 2017/2018

Филимонов Владислав Аскольдович, студент 317 группы

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

14 ноября 2017 г.

Введение

Отладка кода вдвое сложнее, чем его написание.
Так что если вы пишете код настолько умно, насколько можете, то вы по определению недостаточно сообразительны, чтобы его отлаживать.

Брайан Керниган.

Отладчик

Jupiter Notebook поддерживает аппарат магических команд (команды, начинающиеся с %), реализованный в IPython. С помощью магических команд имеется возможность использовать отладчик pdb. Этот отладчик поддерживает два режима работы:

- режим «посмертной» отладки;
- режим с предустановленными точками останова (работа в этом режиме весьма неудобна в JN, значительно лучше в IPython).

Отладчик: режим «посмертной» отладки

Для работы в этом режиме достаточно выполнить команду **%debug**. Если работа одной из ячеек Jupiter Notebook'а была прервана исключением, то выполнение этой команды приведет к вызову отладчика для данной ячейки.
Замечание: Следует учитывать, что при таком режиме имеется возможность анализировать только последнее исключение.
С помощью **%pdb** имеется возможность автоматически вызывать отладчик в случае прерывания работы ячейки исключением.

Некоторые команды отладчика

Некоторые команды отладчика (жирным выделены команды для режима с предустановленными точками останова):

- where (w) — печатает stack frame;
- up (u) — перейти на уровень выше в stack frame;
- down(d) — перейти на уровень ниже в stack frame;
- **step(s)** — выполнить одну команду с возможным заходом в вызов функции;
- **next(n)** — выполнить одну команду, не заходя в вызов функции;
- **cont(c)** — продолжить выполенение до следующей точки останова.

Пример

```
>>> def f(x,y):
...         return x + y

>>> f(np.zeros((2, 3)), np.ones((2, 2)))
>>> %debug
      2 def f(x,y):
----> 3     return x + y

>ipdb> x.shape
(2,3)
>ipdb> y * 4
array([[ 4.,  4.],
       [ 4.,  4.]])
```

Пример

```
>>> def f(x, y):
...         return x + y
... def g(x, y, z):
...         return z + f(x, y)
>>> g(np.ones((2, 2)), np.array([5,6,7,8]), np.zeros((2, 2)))
# Тут исключение
>>>%debug
      1 def f(x, y):
----> 2     return x + y
      3 def g(x, y, z):
      4     y.reshape(x.shape)
      5     return z + f(x, y)
>ipdb> x + y.reshape(x.shape)
array([[ 6.,  7.],
       [ 8.,  9.]])
```

Пример

```
>ipdb> up
      3 def g(x, y, z):
      4     y.reshape(x.shape)
----> 5     return z + f(x, y)
      6 def h(x, y, z, t=0):
      7     if(t <= 0):
>ipdb> z + f(x, y.reshape(x.shape))
array([[ 6.,  7.],
       [ 8.,  9.]])
 >ipdb> y = y.reshape(x.shape)
 >ipdb> z + f(x, y)
array([[ 6.,  7.],
       [ 8.,  9.]])
```

Отладчик: режим с предустановленными точками останова

Для работы в этом режиме нужно определить точки останова до запуска кода. Для этого необходимо использовать команду **%debug [–breakpoint FILE:LINE]**.
Замечание: при работе в JN могут возникать сложности, так как файл *.ipynb имеет другую структуру по сравнению с файлом *.py.
Для работы в этом режиме лучше использовать IPython (подробнее об этом можно почитать тут https://habrahabr.ru/post/104086/)

Профилирование

Для профилирования в JN имеется несколько магических команд:

- %prun — для анализа сколько времени занимает каждая функция;
- %lprun — для анализа сколько времени занимает каждая строчка;

Замечание: команды, начинающиеся с % являются строковыми (т.е. применяются к одной строке написанной после них). Если необходимо анализировать все ячейки нужно использовать %%.

Профилирование: примеры

```
>>> def useless_sum(x, y):
...     tmp = np.array(x.shape)
...     for i in range(x.shape[0]):
...         for j in range (x.shape[1]):
...             tmp = x[i][j] + y[i][j]
...     return tmp
>>> def useless_function(x, y):
...         return(useless_sum(x,y), x.dot(y))
>>> x = np.random.rand(2000, 2000)
>>> y = np.random.rand(2000, 2000)
```

Профилирование: примеры

```
>>> %prun useless_function(x, y)
7 function calls in 3.212 seconds
 Ordered by: internal time
 ncalls  tottime  percall     filename:lineno(function)
 1          2.922    2.922          useless_sum
 1          0.285    0.285        method 'dot'
 ...
 1          0.000    3.207       useless_function
 ...
```

Профилирование: примеры

Для работы с %lprun в python 3 необходимо загрузить дополнительное расширение. При этом вызов имеет другую структуру.

```
%load_ext line_profiler
%lprun -f useless_sum useless_sum(x, y)
Total time: 7.91255 s
File: <ipython-input-22-360c7ab4bb4c>
Function: useless_sum at line 2
Line   Hits         Time      Line Contents
==============================================================
2                                def useless_sum(x, y):
3         1           52          tmp = np.array(x.shape)
4      2001         3818          for i in range(x.shape[0]):
5   4002000      8222494             for j in range (x.shape[1]):
6   4000000     22682784                 tmp = x[i][j] + y[i][j]
7         1            2           return tmp
```

Список литературы

http://ipython.readthedocs.io/en/stable/interactive/magics.html

https://habrahabr.ru/post/104086/

https://docs.python.org/3.5/library/pdb.html