Jupyter Notebook 사용방법

주피터 노트북은 오픈소스의 웹 어플리케이션으로 브라우져상에서 라이브 코딩이 가능합니다. 코딩과 동시에 바로바로 결과를 확인할수 있고 그래프나 이미지등도 브라우져 내에서 바로 확인이 가능하기 때문에 주로 데이터 사이언스 분야에서 많이 사용합니다. 또한 파이썬 이외의 다른 언어도 지원하며 내부에 설명을 위한 Markdown도 지원 합니다. 여기에서는 Mode, Markdown, Magic Command, Notebook Expression를 자주 사용하는 기능들을 위주로 설명합니다.

1. Mode

주피터 노트북에는 명령모드와 편집모드 2가지 모드가 있습니다. 명령모드(command mode)에서는 코드가 실행되는 셀을 편집해주고 편집모드에서는 셀안의 코드와 같은 내용을 편집합니다. 명령모드 상태에서는 셀의 좌측 선이 파란색으로 표시되며 편집모드에서는 초록색으로 표시됩니다.

1.1 명령모드 (command mode)

셀을 선택한 상태에서 ESC키를 누르면 명령모드로 모드가 전환됩니다. 명령모드에서는 셀에 대한 편집을 할수 있으면 자주 사용하는 단축키는 아래와 같습니다.

- dd : 셀삭제
- a : 현재 선택된 셀 위로 새로운 셀 생성
- b : 현재 선택된 셀 아래로 새로운 셀 생성
- m : 현재 선택된 셀을 마크다운 스타일로 변환
- y : 현재 선택된 셀을 코드 스타일로 변환
- l : 셀의 라인을 표시

1.2 편집모드 (edit)

명령모드 상태에서 Enter키를 입력하면 선택된 셀이 편집모드로 전환됩니다. 편집모드에서는 셀 내부의 내용을 편집하며 코딩시에 항상 편집모드 상태에서 셀 내부에 코딩을 하게 됩니다.

- cmd(ctrl) + Enter : 현재 셀의 코드가 실행되고 다음 셀로 이동 하지 않음
- option(alt) + Enter : 현재 셀의 코드가 실행되고 아래 셀 생성
- shift + Enter : 현재 셀의 코드가 실행되고 다음 셀로 이동
- tab : 몇글자를 입력한 상태에서 tab키를 누르면 해당 글자가 포함된 명령어리스트가 표시되어 명령어를 자동완성이 가능
- shift + tab : 함수나 클래스 명 뒤에서 입력하면 내부 정보를 보여줌
- (function or class name)? : 내부정보를 보여줌
- (function or class name)?? : 코드를 보여줌

2. Markdown

2.1 머리말 (Heading)

머릿말 앞에 '#' 를 넣어주며 1개에서 5개 까지 가능하며 '#'이 많아질수록 글자 크기가 작아집니다.

```
# Head
## Head
### Head
#### Head
##### Head
```

2.2 인용 (Blockquotes)

앞에 '>' 기호를 넣어주며 갯수에 따라서 레벨을 나눠줄수 있습니다.

```
> dss
>> datascience
```

2.3 리스트 (List)

-, * 의 기호를 넣어 리스트 스타일을 나타낼수 있으며 띄어쓰기나 들여쓰리고 레벨을 나눠줄수 있습니다.

```
 - data1
  - data2
* data3
 * data4
    * data5
```

2.4 코드 (Code Block)

` `, ``` ```로 코드 블럭을 지정할수 있습니다. 한단어나 한줄을 나타낼때는 한개의 기호를 사용하며 여러줄을 나타낼때는 세개의 기호를 사용합니다. 그리고 코드 블럭 안에서는 -, *, > 기호등과 같이 markdown에 예약 되어 있는 기호도 그대로 나타납니다.

```
`> code`

```
- code
* list
```
```

2.5 링크 (Link)

[text](link)의 형태로 URL이 숨겨진 링크를 나타낼수 있습니다.

```
[Go Fastcampus!](http://www.fastcampus.co.kr/)
```

2.6 가로선 긋기

--- 로 가로선을 그을수 있습니다.

```
---
```

3. Magic Command

셀 내부에서 특별한 동작을 할수 있는 커멘드를 의미하며 커멘드에 대한 내용은 ipython 매직커멘드에 대한 웹 페이지 (http://ipython.readthedocs.io/en/stable/interactive/magics.html) 를 참고하시면 됩니다. 여기에서는 주로 사용되는 커멘드를 정리하였습니다.

3.1 사용 방법

- %(command) : 한줄을 매직커멘드로 사용할때 사용
- %%(command) : 여러줄을 매직커멘드로 사용할때 사용
- %(command)? : 매직키워드 뒤에 ? 를 붙여서 실행하면 키워드에 대한 상세 설명이 나옴

3. 2 주요 Magic Command

- pwd : 현재 위치
- ls : 현재 디렉토리 파일 조회
- whos : 현재 정의된 변수 확인
- reset : 현재 정의된 변수 삭제
- time : 실행시간을 알려줌
- timeit : 평균실행 시간을 알려줌
- writefile : 파일작성
- magic : 각 키워드 상세 도움말 출력
- lsmagic : magic 키워드 출력
- run : python file code 실행
- history : 사용한 명령을 출력
- matplotlib inline : matplotlib을 내부에서 실행할수 있도록 함

3. 3 사용 예시

3. 3. 1 함수 내용 설명

```
def test():
    """
    docstring test
    """
    print("test")
```

```
test?
```

```
test??
```

3. 3. 2 파이썬 파일 작성하여 파일로 저장

Magic Command 아래의 코드를 test.py라는 파일로 저장합니다.

```
%%writefile test.py
print("test python")
print(2 + 3)
```

작성한 코드가 test.py로 저장되었는지 파일 리스트 명령으로 확인 합니다.

```
%ls
```

작성한 test.py 코드를 실행합니다.

```
%run test.py
```

3. 3. 3 저장된 파일 호출

방금 작성한 test.py 코드를 노트북 셀로 호출합니다.

```
%load test.py
```

3. 3. 4 실행시간 확인

time 명령어를 사용하여 코드의 실행 시간을 확인할수 있습니다.

```
%%time
2 ** 1000
```

timeit 명령어를 사용하면 코드를 여러번 실행하여 평균 실행 시간과 오차를 확인할수 있습니다.

```
%%timeit
2 ** 1000
```

4. Shell Command

쉘 명령어는 셀 가장 앞에 ! 기호를 붙여서 노트북 내에서 쉘 명령어를 사용할수 있습니다.

4. 1 주요 Shell Command

- ls : 파일 리스트 출력
- echo : 문자열 출력
- touch : 파일 생성
- cat : 파일 출력
- mkdir : 디렉토리 생성
- rmdir : 디렉토리 삭제
- mv : 파일 이동
- cp : 파일 복사
- rm : 파일 삭제
- df : (disk free) 사용량 확인

4. 2 Shell Command 결과를 변수로 할당 및 출력

쉘 명령어의 결과를 노트북 내에서 변수로 받아서 사용할수 있으며, 노트북 내의 변수를 쉘 명령어에도 사용 할수 있습니다.

```
file_list = ! ls
file  list
```

```
! echo {file_list}
```

4. 3 Shell Command 사용 연습

(1) 아래 코드가 작성된 test.py 파일 생성

```
def sum(a, b):
    return a + b
```

(2) test.py가 생성되었는지 확인

(3) test.py를 dss.py로 변경

(4) dss.py를 test.py로 복사

(5) test.py 파일 로드

(6) test.py 실행 (run 명령어 사용 또는 호출된 코드 실행)

(7) dss.py 파일 삭제
(8) sum(1, 2) 실행
(9) sum(1, 2) 코드 실행시간 확인
(10) 현재 선언된 변수 확인
(11) a = 1 변수 선언
(12) 선언된 변수 삭제
(13) 선언된 변수가 삭제되었는지 확인

```
# 1
%%writefile test.py
def sum(a, b):
    return a + b

# 2
! ls

% ls

# 3
! mv test.py dss.py

! ls

# 4
! cp dss.py test.py

! ls

# 5
%load test.py

# 6
% run test.py

! ls

# 7
! rm dss.py

! ls

# 8
sum(1,2)

# 9
%%time
sum(1,2)

%%timeit
sum(1,2)

# 10
%whos

# 11
a = 1

%whos

# 12
%reset

# 13
%whos
```

5. Jupyter Notebook Expression

바로 앞의 셀에서 실행한 결과를 다음 셀에서 사용하고자 할때 사용합니다. _(언더바)를 사용하면 이전 셀의 결과를 가져와서 사용할수 있습니다.

```
1 + 3
```

4

```
_ * 2
```

8