Jupyter Notebook 사용법

Jupyter Notebook의 생산성을 높이는 주요 단축키와 편의 기능, 그리고 일반적인 문제 해결 방법을 정리합니다.

0. 경고 메세지 삭제

- 평소에는 코드 효율이나 통일 등을 위해 경고를 켜두는게 좋을 수 있으나, 시험 제출을 위해서 쓸데없는 정보가 추가로 뜨는 것은 삭제가 좋음.

```
import warnings
warnings.filterwarnings('ignore')
```

1. 단축키

Jupyter Notebook은 **명령 모드(Command Mode)**와 **편집 모드(Edit Mode)**를 가집니다.

- **편집 모드 (초록색 테두리)**: 셀에 코드를 입력할 수 있는 상태. `Enter` 키로 진입합니다.
- **명령 모드 (파란색 테두리)**: 노트북의 셀을 관리하는 상태. `Esc` 키로 진입합니다.

기능	단축키 (편집 모드)	단축키 (명령 모드)	설명
셀 실행	`Ctrl + Enter`	`Ctrl + Enter`	현재 셀을 실행합니다.
실행 후 다음 셀 이동	`Shift + Enter`	`Shift + Enter`	현재 셀을 실행하고 다음 셀로 이동합니다. (없으면 새로 생성)
실행 후 아래 셀 추가	`Alt + Enter`	`Alt + Enter`	현재 셀을 실행하고 바로 아래에 새 코드 셀을 추가합니다.
주석 처리/해제	`Ctrl + /`	-	선택한 라인 또는 현재 라인을 주석 처리하거나 해제합니다.
노트북 저장	`Ctrl + S`	`Ctrl + S`	현재 노트북을 저장합니다.
명령 팔레트 열기	`Ctrl + Shift + P`	`Ctrl + Shift + P`	모든 명령어를 검색하고 실행할 수 있는 창을 엽니다.
문서(docstring) 보기	`Shift + Tab`	-	함수/객체에 대한 설명을 표시합니다. (여러 번 누르면 상세 정보)
셀 분할	`Ctrl + Shift + -`	-	커서 위치에서 셀을 분할합니다.
선택 셀 병합	-	`Shift + M`	선택된 셀(들)을 병합합니다. 아래 셀과 병합 시에는 하나만 선택.
라인 넘버 토글	-	`Shift + L`	전체 셀의 라인 넘버 표시를 켜고 끕니다.

기능	단축키 (편집 모드)	단축키 (명령 모드)	설명
셀 삭제	-	`D, D`	현재 셀을 삭제합니다. (D를 두 번 누름)
위에 셀 추가	-	`A`	현재 셀 위에 새로운 코드 셀을 추가합니다.
아래에 셀 추가	-	`B`	현재 셀 아래에 새로운 코드 셀을 추가합니다.
셀 잘라내기	-	`X`	현재 셀을 잘라내기 합니다.
셀 복사	-	`C`	현재 셀을 복사합니다.
셀 붙여넣기	-	`V`	현재 셀을 붙여넣습니다.
마크다운 셀로 변경	-	`M`	현재 셀을 마크다운 셀로 변경합니다.
코드 셀로 변경	-	`Y`	현재 셀을 코드 셀로 변경합니다.
실행 취소	-	`Z`	셀 삭제 등 마지막 작업을 취소합니다.

2. 자동완성 기능 제어

Jupyter의 기본 자동완성(Jedi)이 너무 느리거나 불편할 경우, 아래 매직 명령어로 제어할 수 있습니다.

```
# 자동완성 기능 끄기
%config Completer.use_jedi = False

# 자동완성 기능 다시 켜기
%config Completer.use_jedi = True
```

만약 단어(문장) 드래그 후 좌우 자동으로 괄호나 따옴표 닫는게 안된다면,
`Settings > Auto Close Brackets`를 설정하면 된다.

3. 한글 폰트 깨짐 해결 (Matplotlib)

Matplotlib 시각화 시 한글이 깨져 보일 경우, 아래 코드를 실행하여 시스템에 맞는 한글 폰트를 설정합니다.

```
import matplotlib.pyplot as plt
import platform

# 운영체제에 맞는 한글 폰트 설정
if platform.system() == 'Windows':
    plt.rc('font', family='Malgun Gothic')
elif platform.system() == 'Darwin':  # Mac OS
    plt.rc('font', family='AppleGothic')
else:  # Linux
    # 사전에 나눔고딕 설치 필요
    # sudo apt-get install -y fonts-nanum*
    # fc-cache -fv
```

```
    plt.rc('font', family='NanumGothic')

# 마이너스 부호 깨짐 방지
plt.rcParams['axes.unicode_minus'] = False

# --- 예시 ---
plt.figure(figsize=(10, 5))
plt.plot([0, 1, 2, 3], [0, 1, 4, 9])
plt.title('한글 제목 테스트')
plt.xlabel('X축 라벨')
plt.ylabel('Y축 라벨')
plt.show()
```

아니면 아래 코드를 통해 설치된 폰트 리스트를 출력 후 선택해도 된다.

```
import matplotlib.font_manager as fm

font_list = fm.findSystemFonts(fontpaths = None, fontext = 'ttf')
font_list[:]

# 아래 방식으로 하나씩 불러올 수도 있다.
#font_list = [font.name for font in fm.fontManager.ttflist]
```

제35회 ADP 실기 시험 기준, 폰트 파일 위치를 직접 지정해서 설정 시 아래와 같이 진행하면 된다.

```
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
import matplotlib as mpl

# 폰트 경로 지정
font_path = '/usr/share/fonts/nanum/NanumGothic.ttf'

# 폰트 이름 가져오기
font_name = font_manager.FontProperties(fname=font_path).get_name()

# 전역 설정
mpl.rc('font', family=font_name)

# 음수 부호 깨짐 방지
mpl.rcParams['axes.unicode_minus'] = False

# 테스트
plt.title('한글 폰트 테스트')
plt.plot([1, 2, 3], [1, 4, 9])
plt.xlabel('가로축')
plt.ylabel('세로축')
plt.show()
```

이외에 `matplot` 전역 설정값은 아래와 같이 작성 가능하다.

```
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (20,10) # 그래프 기본 크기 설정

plt.rcParams['font.size'] = 12 # 기본 폰트 크기 설정
plt.rcParams['axes.labelsize'] = 10 # x,y 레이블 폰트 크기
plt.rcParams['axes.titlesize'] = 10 # 축 제목 글꼴 크기
plt.rcParams['xtick.labelsize'] = 6 # x축 tick 레이블의 글꼴 크기
plt.rcParams['ytick.labelsize'] = 6 # y축 tick 레이블의 글꼴 크기
plt.rcParams['legend.fontsize'] = 10 # 범례 글꼴 크기
```

4. PDF로 내보내기 (Export to PDF)

HTML로 변환 후 브라우저의 인쇄 기능을 사용하는 것이 간편합니다.

1. **HTML로 다운로드**: `File > Download as > HTML (.html)`
2. **브라우저에서 열기**: 다운로드된 HTML 파일을 Chrome, Edge 등 웹 브라우저에서 엽니다.
3. **PDF로 인쇄**: 브라우저의 인쇄 기능(`Ctrl + P`)을 열고, 대상을 **'PDF로 저장'**으로 선택하여 저장합니다.

- 코드가 너무 길 경우, 백슬래시(`\`)를 이용해서 줄을 나눌 수 있습니다.
- 전체 코드를 괄호를 활용해 감쌀 수도 있습니다.

참고용 함수

현재 사용하는 함수에 대해 어떤 메소드를 지원하고 하이퍼파라미터가 있는지 알 수 없다면, `help()`로 감싸서 출력하면 됩니다.
오픈 라이브러리 사용이 불가한 시험 환경 특성상 유용하게 사용 가능합니다.