



[스파르타코딩클럽] 파이썬 데이터분석 첫걸음 - 1주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

▼ 단축키 모음

▼ 실행(Run)

- `shift` + `enter`

▼ 자동완성

- `Tab`

[수업 목표]

1. Python 기초 문법을 익힌다.
2. Pandas의 사용법을 익힌다.
3. 실제 데이터를 다룰 수 있게 된다.

[목차]

01. 오늘 배울 것

02. 필수 프로그램 설치

03. 파이썬, 분석환경 구축확인

04. 파이썬 기초 문법 - (1)

05. 파이썬 기초 문법 - (2)

06. Quiz_파이썬 기초 문법!

07. Pandas

08. 기초 데이터 다루기

09. Pandas 연습하기

10. 시각화, Matplotlib

11. Matplotlib 부딪혀보기

12. Matplotlib 그래프 입맛대로 바꿔보기

13. Quiz_다른 컬럼 분석 해보기

14. 끝 & 숙제 설명

15. 1주차 숙제 답안 코드



모든 토글을 열고 닫는 단축키

Windows : `Ctrl` + `alt` + `t`

Mac : `⌘` + `⌥` + `t`

01. 오늘 배울 것

▼ 파이썬과 데이터 분석환경

파이썬이 무엇인지, 파이썬으로 데이터를 분석하기 위해서는 어떤 환경이 필요한지 알아보겠습니다.

▼ Pandas

파이썬으로 데이터 분석을 하면서 가장 많이 쓰는 도구에 대해서 알아보겠습니다.

▼ Matplotlib

데이터를 분석하고 그 결과를 시각화 하는 도구에 대해 알아보겠습니다.

▼ 실제 데이터 분석 방법

실제 데이터를 분석 해보고 그 방법에 대해 알아보겠습니다.

02. 필수 프로그램 설치

▼ Anaconda 설치하기 ([다운로드 링크](#))

▼ ([←눌러보기](#)) Windows

1) [다운로드 링크](#)에 접속하고, [Download] 버튼을 클릭하세요.



Individual Edition




Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

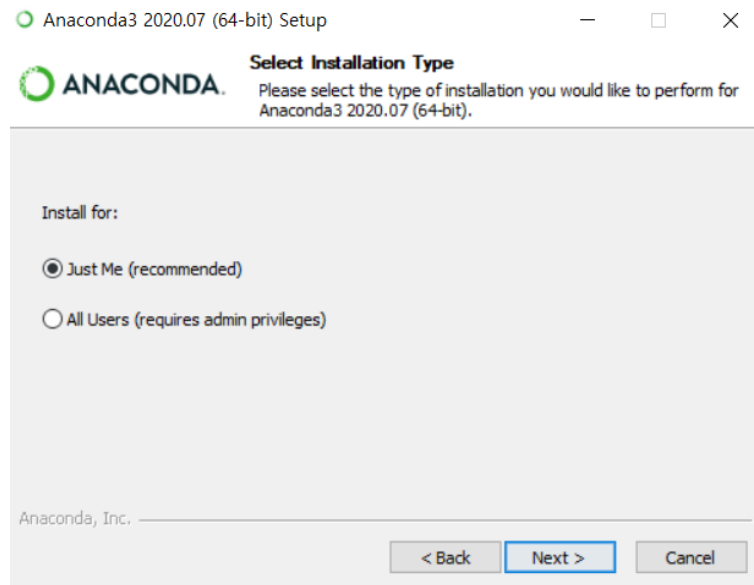
[Download](#)

2) 64-Bit Graphical Installer 를 다운로드 하세요.

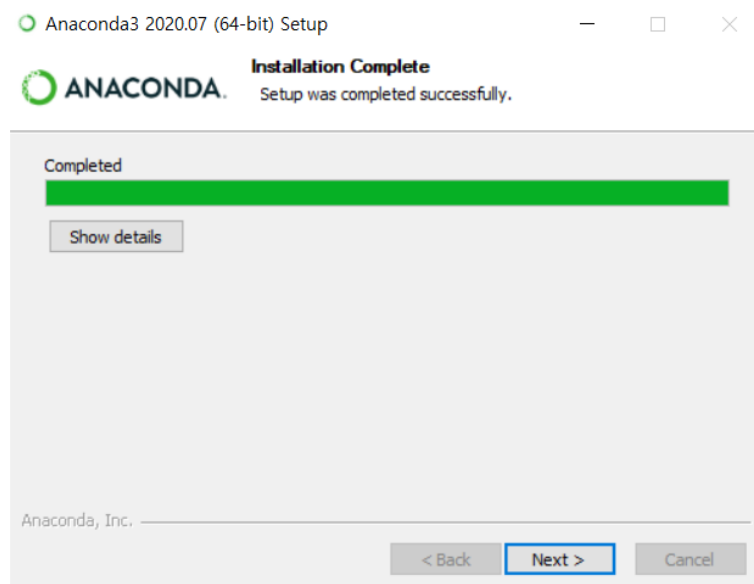
Anaconda Installers

Windows 	MacOS 	Linux 
Python 3.8 64-Bit Graphical Installer (466 MB) 32-Bit Graphical Installer (397 MB)	Python 3.8 64-Bit Graphical Installer (462 MB) 64-Bit Command Line Installer (454 MB)	Python 3.8 64-Bit (x86) Installer (550 MB) 64-Bit (Power8 and Power9) Installer (290 MB)

3) Next → Next → 눌러서 설치해주세요!



4) 설치완료. 끝!



▼ (←눌러보기) Mac

1) 다운로드 링크에 접속하세요.



Individual Edition

Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

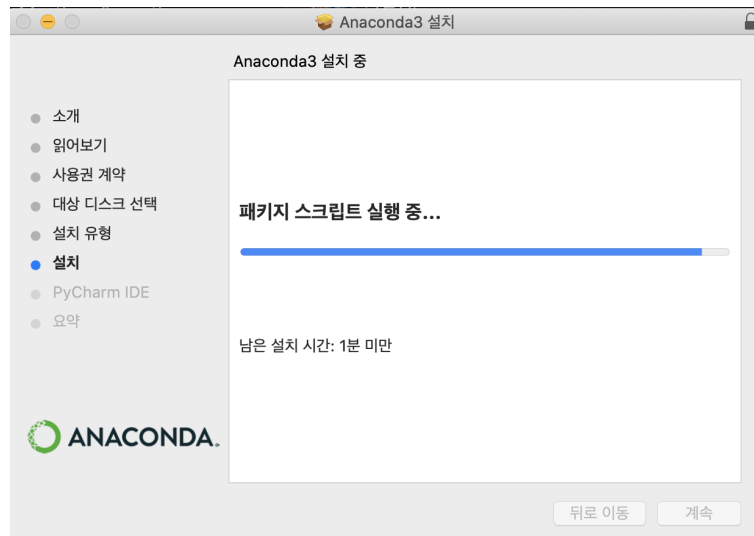
[Download](#)

2) 64-Bit Graphical Installer 다운로드 하세요.

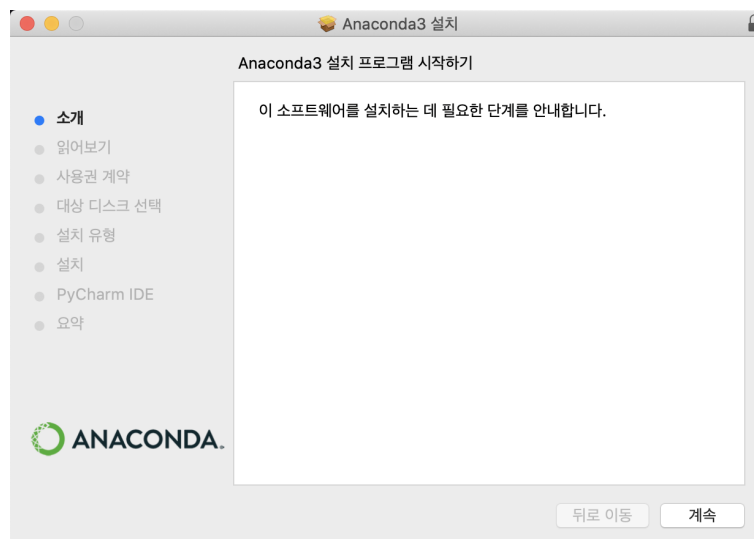
Anaconda Installers

Windows 	MacOS 	Linux 
Python 3.8 64-Bit Graphical Installer (466 MB) 32-Bit Graphical Installer (397 MB)	Python 3.8 64-Bit Graphical Installer (462 MB) 64-Bit Command Line Installer (454 MB)	Python 3.8 64-Bit (x86) Installer (550 MB) 64-Bit (Power8 and Power9) Installer (290 MB)

3) 쪽→쪽→ 설치해주세요!



4) 설치완료. 끝!



03. 파이썬, 분석환경 구축확인

▼ 1) 이 강의에서 우리는 무엇을 하나요?

우리는 파이썬과 아나콘다를 이용해서 데이터를 분석할 거예요.

일단 **파이썬**과 **아나콘다**는 알고 가야겠죠?

그래서 이전에 0주차 사전과제에서 구축하신 분석환경에 대해 간략히 설명드립니다.

파이썬이 무엇인지 또 아나콘다의 사용방법과 사용 꿀팁을 알려드릴게요!

▼ 파이썬을 설치한다?



파이썬을 설치한다는 행위는 일종의 번역팩을 설치한다고 생각하면 됩니다. 컴퓨터는 101010001 과 같은 언어만 알아들을 수 있습니다. 파이썬이라는 언어의 문법으로 된 것을 101010001과 같은 컴퓨터의 언어로 번역해 줄 수 있도록, 번역기를 설치하고 프로그래밍을 쉽게 할 수 있는 기본 코드(예를 들면, 기본 함수)를 설치하는 것입니다.

▼ 아나콘다를 설치한다?



아나콘다는 분석을 도와줄 패키지매니징 플랫폼입니다. 물론 아나콘다가 없어도 분석을 할 수는 있어요. 아나콘다를 쓰는 이유는 우리가 개발하는데 필요한 기본적인 도구들을 모아 놓은 공구 상자라고 생각하시면 됩니다. 개발에 필요한 도구들을 미리 설치해 둡시다.

▼ 파이썬 라이브러리?



파이썬에서 관련있는 기능들의 묶음을 모듈, 모듈들의 묶음을 패키지, 패키지들의 묶음을 라이브러리라고 부릅니다. (패키지와 라이브러리는 많이들 혼용해서 쓰기도 합니다 😊)

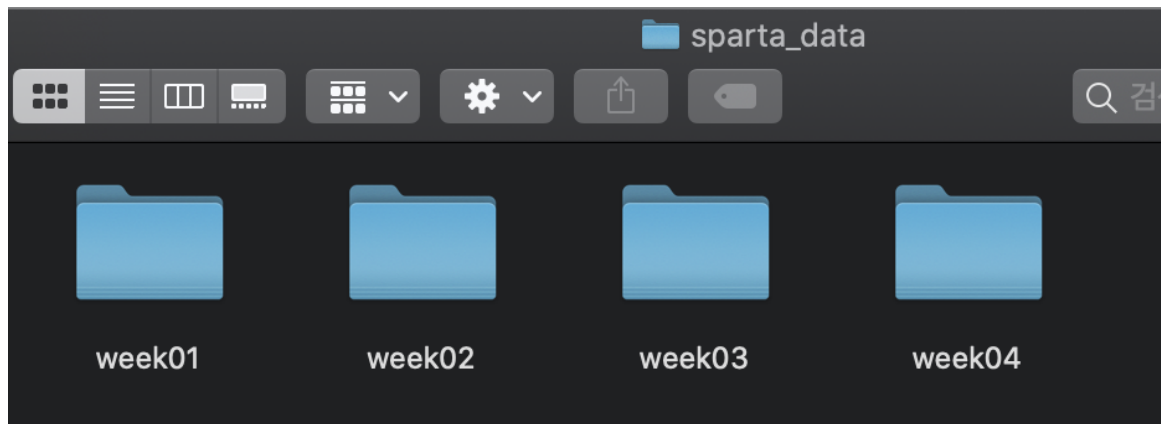
파이썬을 설치할 때 따라오는 내장 라이브러리부터 우리가 오늘 쓸 Pandas, Matplotlib이나 키워드 분석, 딥러닝을 위한 외부 라이브러리들까지 정말 다양한 라이브러리가 있다는 게 파이썬의 수많은 장점 중 하나입니다.

▼ 2) 전체 디렉토리 구조 만들기

▼ [코드스니펫] 다운로드 링크

<https://drive.google.com/drive/folders/1vh20PDVm9XFv2-iLxt-URE4nTFQz5KJH?usp=sharing>

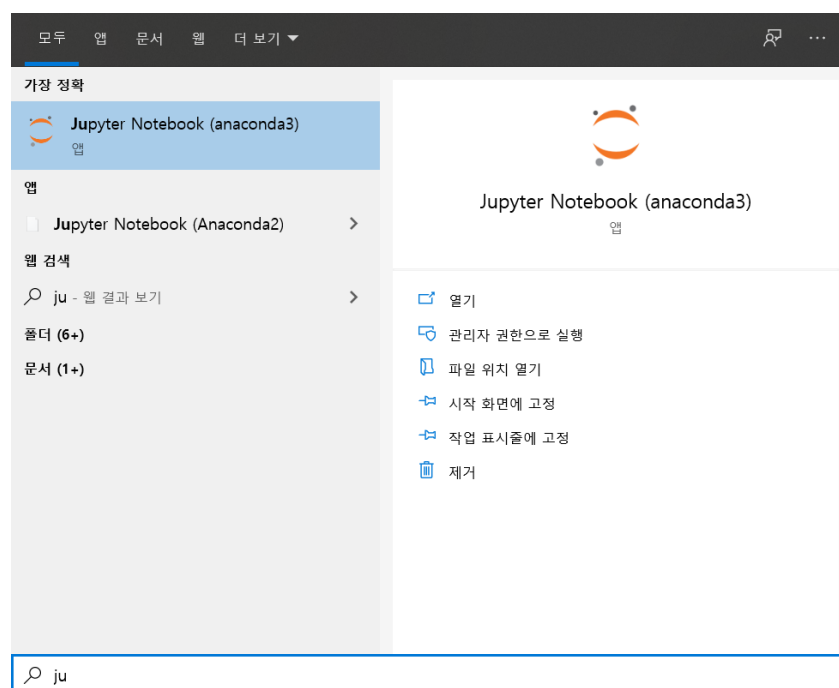
- 바탕화면에 압축 풀기
- 각 주차에 필요한 데이터들은 data 디렉토리 안에 들어있습니다. 여러분의 주피터 노트북은 각 주차 하위에 만들어 주시면 됩니다.
- week01
 - myfile.ipynb
 - data
 - data.csv
- 이렇게 말이죠



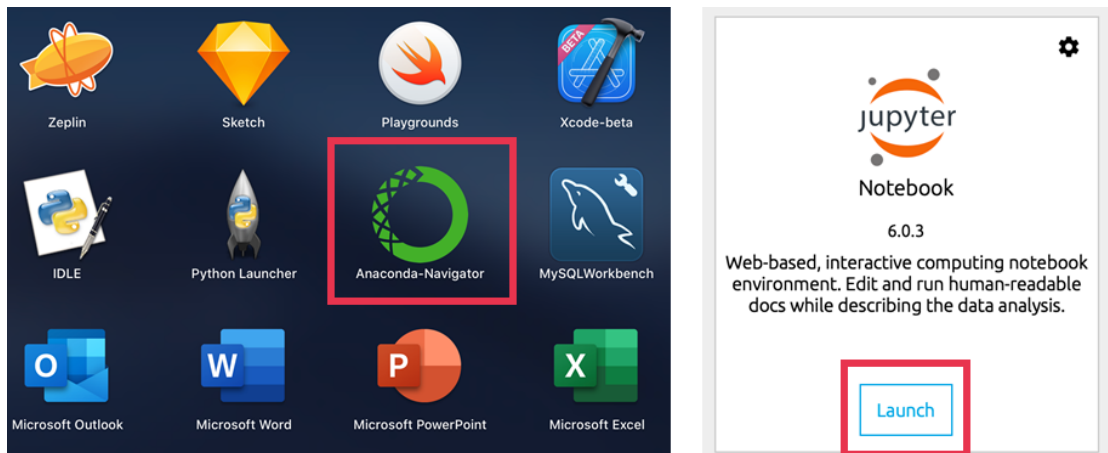
▼ 3) 아나콘다 설치 확인 & 주피터 노트북 실행

1. Jupyter Notebook 을 엽니다

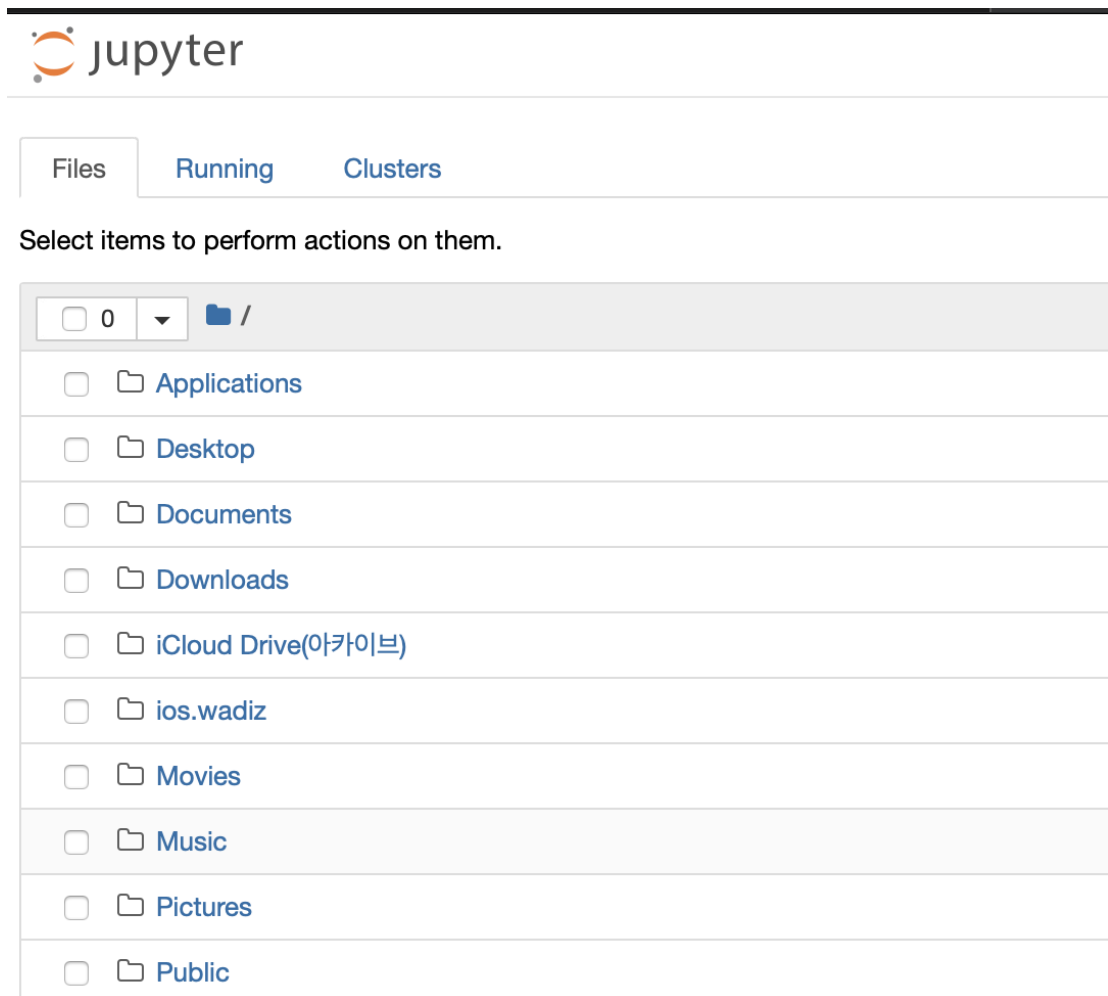
- 윈도우 → 윈도우 키+Q(검색) 를 눌러서 Jupyter Notebook 검색



- 맥 → 런치 패드에서 Anaconda-Naviagator 실행 → Jupyter Notebook Launch

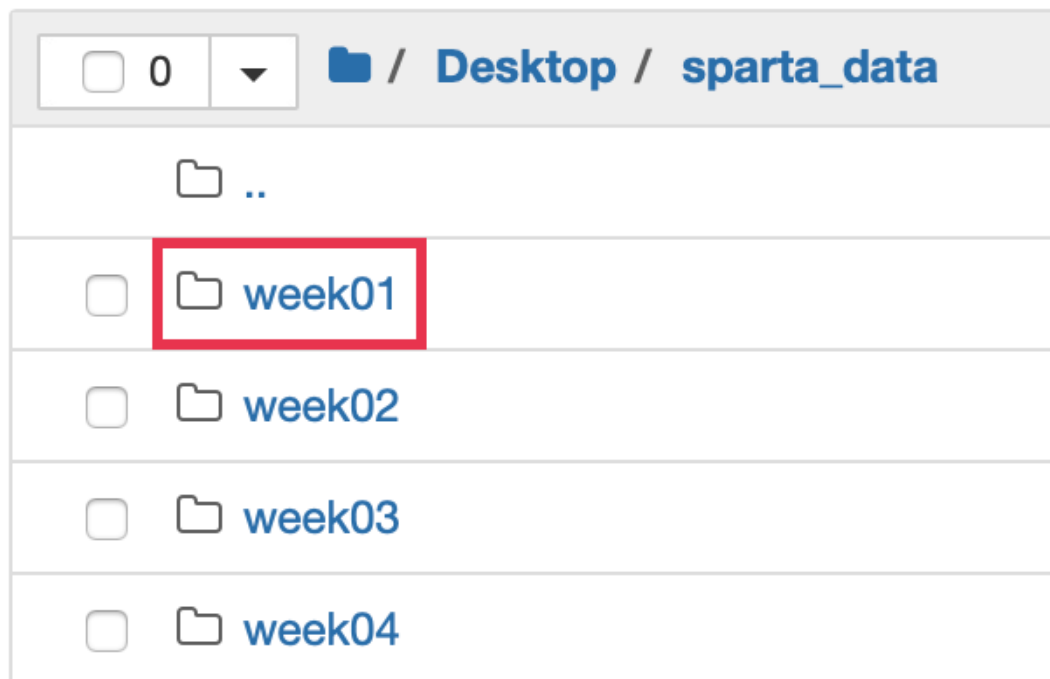


2. 자동으로 브라우저에 주피터 노트북이 뜨지 않는다면, 크롬 브라우저를 띄워
<http://localhost:8888> 로 접속합니다.



3. Desktop > sparta_data > week01 로 이동합니다.

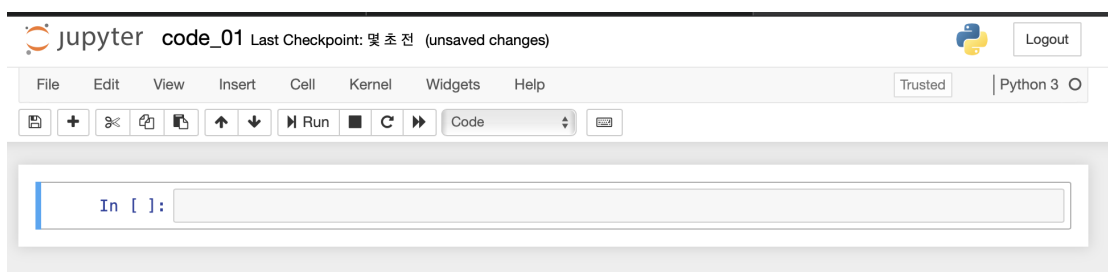
Select items to perform actions on them.



4. 오른쪽 끝에 New를 누르고 Python 3을 누르세요.



5. 자 새로운 노트북 파일을 만들었으면 이제 본격적으로 시작해 봅시다!



▼ 주피터 노트북 사용 팁

▼ +) 주피터 노트북 원하는 폴더에서 여는 법

▼ 윈도우

1. 시작메뉴에서 Anaconda Prompt 실행
2. 아래 코드를 입력하여 configuration 파일 만들기

```
jupyter notebook --generate-config
```

3. 결과로 나오는 경로(`C:\Users\[사용자이름]\.jupyter`)를 파일 탐색기로 찾아 들어가기



.jupyter가 보이지 않는다면?

이 폴더는 숨김 폴더입니다. 탐색기 보기 탭에서 숨김파일 보기 체크!

4. jupyter_notebook_config.py 파일 우클릭 > 다른 프로그램으로 열기 > 메모장
5. Ctrl+f 눌러서 찾기 기능 > 'c.NotebookApp.notebook_dir' 검색
6. 해당 줄을 아래처럼 바꿔주기

```
c.NotebookApp.notebook_dir = r'C:\Users\[사용자이름]\Desktop\sparta'
```

▼ 맥

1. 스포트라이트에서 Terminal 실행
2. 아래 코드를 입력하여 configuration 파일 만들기

```
jupyter notebook --generate-config
```

3. 결과로 나오는 경로(`/Users/[사용자이름]/.jupyter`)를 파인더로 찾아 들어가기



.jupyter가 보이지 않는다면?

이 폴더는 숨김 폴더입니다. Command + Shift + . (마침표) 클릭해서 보이게!

4. jupyter_notebook_config.py 파일 우클릭 > 다른 프로그램으로 열기 > 노트
5. Cmd+f 눌러서 찾기 기능 > 'c.NotebookApp.notebook_dir' 검색
6. 해당 줄을 아래처럼 바꿔주기

```
c.NotebookApp.notebook_dir = r'/Users/[사용자이름]/Desktop/sparta'
```



맥에는 '스마트 인용' 기능이 있어 따옴표(")가 예쁜 따옴표(“)로 자동으로 바뀔 수도 있어요! 바뀌지 않게 편집 > 대체 > 스마트 인용 해제!

04. 파이썬 기초 문법 - (1)

▼ 4) 변수 & 기본연산



변수는 데이터를 저장할 수 있는 공간입니다. 어떤 데이터가 저장되고, 저장된 데이터로 무엇을 할 수 있는지 살펴보겠습니다.

```
a = 3 # 3을 a에 넣는다
print(a)
```

▼ 주피터 노트북 사용 팁!

- 코드 블록을 실행하는 단축키는 - **Shift** + **Enter**
- 노트북에서는 변수 이름만 입력해도 출력

```
b = a # a를 b에 넣는다
print(b)
```

```
a = a + 1 # a+1을 다시 a에 넣는다
print(a)
```

```
num1 = a * b # a*b의 값을 num1이라는 변수에 넣는다
print(num1)
```

```
num2 = 99 # 99의 값을 num2이라는 변수에 넣는다
print(num2)
```

▼ 5) 자료형

1. 숫자, 문자열, 참거짓

```
num = 12 # 숫자가 들어갈 수도 있고,
print(num)
```

```
name = 'Harry' # 변수에는 문자열이 들어갈 수도 있고,  
print(name)
```

```
number_status = True # True 또는 False -> "Boole" 형이 들어갈 수도 있습니다.  
print(number_status)
```

▼ 🍷 깜짝 퀴즈

```
# 출력값을 예측해보세요  
  
num1 = '12'  
  
num2 = '23'  
  
total = num1 + num2  
print(total)
```

2. 리스트 형

```
waiting_list = [] # 비어있는 리스트 만들기  
waiting_list.append('이현호') # 리스트에 문자열 데이터를 넣는다  
print(waiting_list)
```

```
waiting_list.append('이범규') # 리스트에 '이범규'라는 문자열을 하나 더 넣는다  
print(waiting_list)
```

```
waiting_list.append(['고영희', '황철수']) # 리스트에는 또 다른 리스트가 추가될 수 있습니다  
list_food
```

```
# print 로 값 확인해보기  
# waiting_list의 값은? ['이현호', '이범규', ['고영희', '황철수']]  
# waiting_list[0]의 값은? '이현호'  
# waiting_list[2]의 값은? ['고영희', '황철수']  
# waiting_list[2][0]의 값은? '고영희'
```

3. Dictionary 형

```
eng_kor_dict = {} # 비어있는 딕셔너리 만들기  
  
eng_kor_dict = {'apple': '사과', 'pear': '배'}  
eng_kor_dict['apple']
```

```
#eng_kor_dict['사과']
#eng_kor_dict['banana']
```

```
# 딕셔너리에 추가하고 싶을 때
eng_kor_dict['banana'] = '바나나'
eng_kor_dict
```

```
# print 로 값 확인해보기
# eng_kor_dic의 값은? {'apple': '사과', 'pear': '배', 'banana': '바나나'}
# eng_kor_dic['apple']의 값은? '사과'
# eng_kor_dic['pear']의 값은? '배'
# eng_kor_dict['banana']의 값은? '바나나'
```

4. Set 형

```
group1 = set([1, 2, 3, 4, 2, 1])
group2 = set([1, 2, 3, 1, 6])
print(group1)      # {1, 2, 3, 4}
print(group2)      # {1, 2, 3, 6}
```

```
# 교집합
print(group1 & group2) # {1, 2, 3}

# 합집합
print(group1 | group2) # {1, 2, 3, 4, 6}
```



리스트에 있는 데이터에 접근할 때 list_name[0] 와 같은 방법으로 접근합니다.
딕셔너리에 있는 데이터에 접근할 때는 dictionary_name["키값"] 의 방법을 이용합니다.
셋은 리스트를 ()로 감싸주어 사용합니다.

05. 파이썬 기초 문법 - (2)

▼ 6) 조건문

[🖥️ 코드 조건문 01]

```
age = 20

if age >= 20:
    print('성인입니다') # 조건이 참이면 성인입니다를 출력
else:
    print('청소년이에요') # 조건이 거짓이면 청소년이에요를 출력

# age = 17 로 하면 무엇이 출력될까요?
```

[💻 코드 조건문 02] if / elif / else

```
# 조건을 여러 개 사용하고 싶을 때
age = 65

if age > 80:
    print('아직 정정하시군요')
elif age > 60:
    print('인생은 60부터!')
else:
    print('아직어려요!')

# age = 20 하면 무엇이 출력될까요?
```



파이썬은 들여쓰기(indent)로 코드의 블록(시작과 끝) 단위를 나눕니다. 들여쓰기를 잘못 하면 들여쓰기 에러(indentation error) 가 발생하죠! 따라서 들여쓰기가 매우 중요합니다.

▼ 7) 반복문



반복문은, 리스트나 문자열의 요소들을 하나씩 꺼내쓰는 형태입니다. 즉, 임의의 열(sequence, 리스트나 문자열처럼)의 항목들을 그 순서대로 꺼내어 반복합니다.

[💻 코드 반복문 01] 기본

```
fruits = ['사과', '배', '감', '귤']

for fruit in fruits: # fruit 은 우리가 임의로 지어준 이름입니다.
    print(fruit) # 사과, 배, 감, 귤 하나씩 꺼내어 출력합니다.
```

[💻 코드 반복문 02] 살짝 응용해볼까요? - 과일 갯수 세기

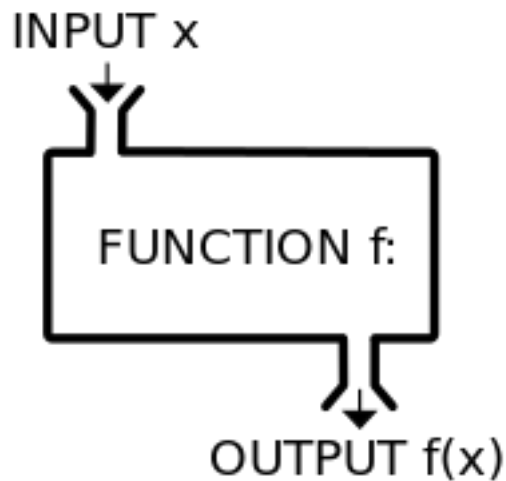
```
fruits = ['사과', '배', '배', '감', '수박', '귤', '딸기', '사과', '배', '수박']

count = 0
for fruit in fruits:
    if fruit == '사과':
        count = count + 1

# 사과의 갯수를 출력합니다.
print(count)
```

▼ 8) 함수

- 처음부터 필요한 내용은 아니에요!
- 컨셉 정도만 기억해주세요.



- 함수는 어떤 값을 넣어주고 값을 반환 해주는 코드 블록
- 같은 일을 줄여줌

[🖥️ 코드 함수 01] 기본

```
def sum(a, b):  
    return a + b  
  
print(sum(3,5))
```

```
def print_name(name):  
    print("반갑습니다 "+name+" 님")  
  
print_name("이현호")
```



리스트와 딕셔너리는 []를 사용합니다.

함수는 ()를 사용하죠! 또 () 나 [] 를 사용하는 것에는 어떤 것이 있는지 떠올려보아요!

06. Quiz_파이썬 기초 문법!

▼ 9) Quiz_01 - 리스트

```
number_list = [0, [1, 2], 3]  
print(number_list[1]) # 예측해 보아요  
  
number_list = [0, [1, 2], 3]  
# 2를 출력하고 싶어요!
```


▼ 정답

```
print(number_list[1]) // [1, 2]

print(number_list[1][1])
first_list = number_list[1] // [1, 2]
print(first_list[1]) // 2
```

▼ 10) Quiz_02 - 딕셔너리

```
students = [{"name": "현호", "age": 23}, {"name": "범규", "age": 24}, {"name": "건희", "age": 25}]
print(students[1]["name"]) # 무엇이 출력될까요?
# 건희를 출력해보세요!
```

▼ [코드스니펫] - 딕셔너리 퀴즈

```
students = [{"name": "현호", "age": 23}, {"name": "범규", "age": 24}, {"name": "건희", "age": 25}]
```

▼ 11) Quiz_03 - Set

```
group1 = set([1, 1, 2, 2, 3, 3])
group2 = set([1, 3, 5, 7, 9])
print(group1) # 무엇이 출력될까요?
print(group2) # 무엇이 출력될까요?
print(group1 & group2) # 교집합
print(group1 | group2) # 합집합
```

▼ [코드스니펫] - set 퀴즈

```
group1 = set([1, 1, 2, 2, 3, 3])
group2 = set([1, 3, 5, 7, 9])
```

▼ 12) Quiz_04 - 조건문

```
answer = (4 + 5) * 2 + 3
x = ? # ?에 알맞은 숫자를 넣어주세요
answer = answer + x

if answer == 30:
    print("정답입니다")
else:
    print("오답입니다")
# 정답입니다를 출력해주세요
```

▼ 정답

```

answer = (4 + 5) * 2 + 3
x = 9
answer = answer + x

if answer == 30:
    print("정답입니다")
else:
    print("오답입니다")
// 정답입니다

```

▼ 13) Quiz_05 - 반복문

```

data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
total = 0

for number in data:
    total = total + number

print(total) # 무엇이 출력될까요?

```

▼ 14) Quiz_06 - 함수

```

# 데이터 부분
data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
data2 = {"첫번째값" : 4, "두번째값" : 10}
# 함수 정의
def sum(a, b):
    return a + b

print(sum(1, 4)) # 무엇이 출력될까요?
print(sum[0, 2])) # 무엇이 출력될까요?
print(sum(data[0], data[3])) # 무엇이 출력될까요?
print(sum(data(3), data(5))) # 무엇이 출력될까요?
print(sum(data2["첫번째값"], data2["두번째값"])) # 무엇이 출력될까요?

```

▼ [코드스니펫] - 함수 퀴즈

```

data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
data2 = {"첫번째값" : 4, "두번째값" : 10}

```

▼ 정답

```

# 데이터 부분
data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
data2 = {"첫번째값" : 4, "두번째값" : 10}
# 함수 정의
def sum(a, b):
    return a + b

print(sum(1, 4)) # 함수 예제와 같습니다
print(sum[0, 2])) # error! 함수는 ( ) 기호를 사용합니다.
print(sum(data[0], data[3])) # 리스트에 있는 데이터에 접근합니다
print(sum(data(3), data(5))) # error! 리스트에는 []를 써야합니다
print(sum(data2["첫번째값"], data2["두번째값"])) # dictionary에는 []를 써야합니다.

```

07. Pandas

▼ 15) 엑셀로 분석해보기

우리가 쉽게 접할 수 있는 엑셀로 데이터 분석을 한번 해 볼까요?

엑셀로 분석을 해보면서 어떤 점이 불편한지, 이런 것들이 파이썬으로는 가능한지 한번 이야기 해 봅시다.

기준일	요일	성별	연령대	시도	시군구	읍면동	업종	통화건수
20190701	월	남	30대	서울특별시	강남구	논현동	치킨	8
20190701	월	여	30대	서울특별시	강남구	개포동	치킨	10
20190701	월	여	30대	서울특별시	강남구	논현동	치킨	16
20190701	월	여	30대	서울특별시	강남구	수서동	치킨	5
20190701	월	여	30대	서울특별시	강남구	대치동	치킨	5
20190701	월	남	40대	서울특별시	강남구	수서동	치킨	5
20190701	월	남	40대	서울특별시	강남구	세곡동	치킨	5
20190701	월	남	50대	서울특별시	강남구	역삼동	치킨	5
20190701	월	남	50대	서울특별시	강남구	도곡동	치킨	5
20190701	월	남	10대	서울특별시	강남구	개포동	치킨	5

- 기본적인 데이터 분석은 엑셀로도 가능!
- 월별 데이터를 하나의 파일로 만들어서 계산하기 → 연간 데이터 만들기
- 비어있는 데이터 채우기
- 엑셀을 배우는 것과 파이썬을 배우는 것 중 어떤 문법이 쉬울까요
- 250메가 파일 다루는데 걸리는 시간



그럼 이제 파이썬을 가지고 데이터 분석을 해 봅시다.
먼저 가장 기초적인 도구인 **Pandas**부터 다루어 보겠습니다!

▼ 16) Pandas란?



파이썬에서 사용되는 데이터 분석 라이브러리입니다. 관계형 데이터를 행과 열로 구성된 객체로 만들어 줍니다. 우리가 불러온 데이터를 다루기 쉽게 도와주는 도구입니다. 글로 읽어서는 잘 이해가 되지 않으니 실제로 써보면서 익혀봅시다.

08. 기초 데이터 다루기

▼ 17) 판다스 불러오기

- 파이썬에는 다양한 라이브러리들이 있습니다. 내가 원하는 기능을 구현해서 쓰기보다는, 이미 구현된 기능이 있는지 찾아보고 어떻게 쓰는 건지 공부해서 활용하는 경우가 훨씬 많을거예요.
- 판다스를 사용하려면 가장 먼저 해야할 일은 '판다스를 불러오는 것' 입니다.
- 우리는 pandas를 불러와 보겠습니다. 그리고 pandas라고 부르면 너무 기니까 pd라고 부르도록 할게요.

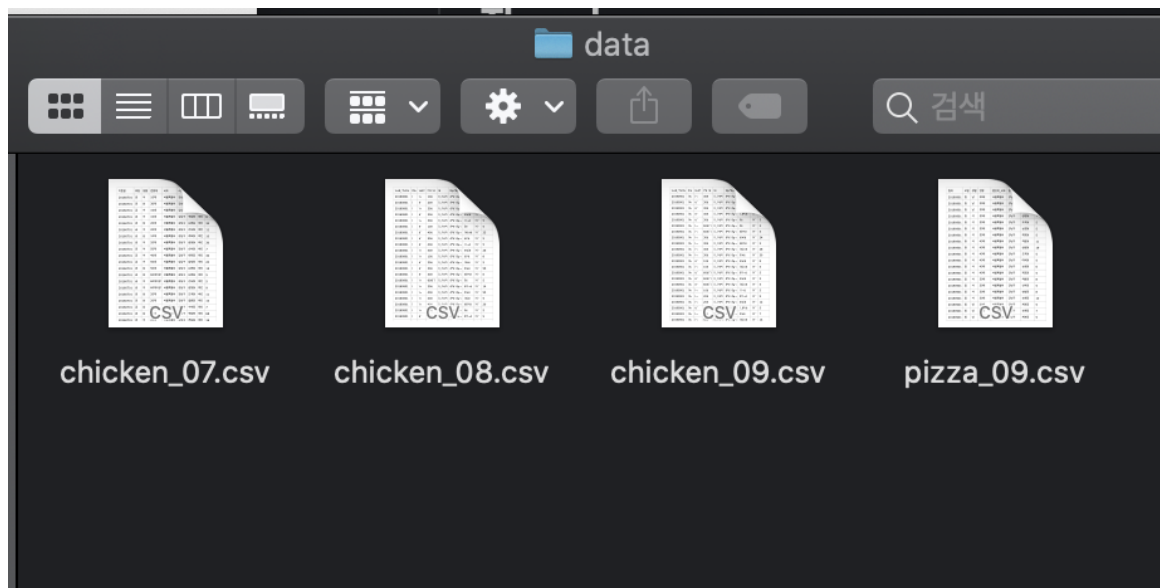
```
import pandas as pd
```

네 아무일도 일어나지 않아요. 그럼 잘 불러왔는지 어떻게 아나요?

pandas의 기능을 한번 써볼게요!

▼ 18) 데이터 불러오기

데이터를 다루기 위해서는 데이터를 불러와야 합니다.



이미 다 준비 해 놓은 데이터를 어떻게 불러올 수 있을까요?

아래와 같이 입력해서 불러 와 보겠습니다.

```
chicken07 = pd.read_csv('./data/chicken_07.csv')
```

네 눈치 채셨겠지만 `pd.read_csv('파일의경로+파일의이름')` 이렇게 파일을 불러옵니다.

./는 현재 디렉토리를 의미해요. 그러니까 ./data/chicken_07.csv 는 현재 디렉토리의 하위에 data 라는 디렉토리에 있는 chicken_07.csv 파일을 가리킵니다.

▼ 19) 데이터 살펴보기

데이터를 불러왔으면 한번 살펴봐야겠죠?

데이터를 살펴봐야 한다면 무엇부터 봐야할까요?

- 몇개의 열들로 이루어져 있나요?
- 각각의 열들은 어떤 데이터들이 들어있나요?
- 성별, 종목, 성공여부와 같은 데이터 인가요?
- 횟수, 갯수와 같은 데이터 인가요?
- 시간과 같이 연속된 데이터 인가요?
- 몇개의 행으로 이루어져 있나요?

다음과 같은 코드를 입력해서 데이터를 살펴봅시다.

이렇게 보입니다.

	기준일	요일	성별	연령대	시도	시군구	읍면동	업종	통화건수
26374	20190731	수	여	60대이상	서울특별시	중랑구	면목동	치킨	20
26375	20190731	수	여	30대	서울특별시	중랑구	중화동	치킨	5
26376	20190731	수	여	10대	서울특별시	중랑구	면목동	치킨	5
26377	20190731	수	여	60대이상	서울특별시	중랑구	망우동	치킨	5
26378	20190731	수	남	40대	서울특별시	중랑구	면목동	치킨	22

요일, 성별, 연령 등의 컬럼은 데이터를 분류할 수 있는 성질을 가지고 있죠. 반면에 통화 건수나 날짜 등의 데이터는 범위가 존재합니다. 총합 혹은 범위를 지정해서 데이터를 자르거나 합할 수 있습니다.

```
## 데이터의 기본 통계치
chicken07.describe()
```

	기준일	통화건수
count	2.637900e+04	26379.000000
mean	2.019072e+07	12.346109
std	8.869258e+00	14.961707
min	2.019070e+07	5.000000
25%	2.019071e+07	5.000000
50%	2.019072e+07	5.000000
75%	2.019072e+07	14.000000
max	2.019073e+07	279.000000

- `count` : 갯수
- `mean` : 평균
- `std` : 표준편차
- `min` : 최솟값
- `max` : 최댓값

범위 데이터가 기준일과 통화건수 두 개 뿐이라, 이 두 개에 대해서만 나왔습니다.

09. Pandas 연습하기

▼ 20) 성별 데이터 살펴보기

예측이 가능한 쉬운 데이터들 부터 살펴보겠습니다.

우리의 데이터 chicken07 중 성별 컬럼을 선택해서, 집합으로 만들어 보겠습니다. 데이터를 집합으로 만들면 중복값이 제거되어 구성 요소들을 알 수 있거든요!

그리고 구성 요소들의 갯수도 세어줄게요. `len()` 은 리스트의 길이(length)를 측정해주는 함수입니다.

```
gender_range = set(chicken07['성별'])
print(gender_range, len(gender_range))
# {'남', '여'} 2
```

▼ 21) 연령대 데이터 살펴보기

연령대는 어떻게 구성 되어있나 볼까요?

총 6개의 종류로 이루어진 데이터군요, 각 연령 별로 분포를 본다면 의미있는 결과를 도출할 수 도 있어 보이네요. 이어서 다른 컬럼들도 살펴보겠습니다.

```
age_range = set(chicken07['연령대'])
print(age_range, len(age_range))
# {'50대', '60대이상', '40대', '30대', '10대', '20대'} 6
```

▼ [👉Quiz] 직접 해 볼까요?



앞으로 실습이 필요한 부분이 나오면 "👉"모양으로 알려줄게요!

퀴즈설명 영상을 먼저 보고 → 정해진 시간동안 혼자 한다음 → 함께하기 영상을 보세요!

▼ Q. 퀴즈 설명

- 요일 데이터 살펴보기
- 시군구 데이터 살펴보기
- 읍면동 데이터 살펴보기

▼ A. 함께 하기

```
day_range = set(chicken07['요일'])
print(day_range, len(day_range))
# {'일', '수', '금', '화', '목', '월', '토'} 7

city_range = set(chicken07['시군구'])
print(city_range, len(city_range))
# {'관악구', '광진구', '도봉구', ...} 25

town_range = set(chicken07['읍면동'])
print(town_range, len(town_range))
# {'시흥동', '창성동', '상계동', ...} 425
```

▼ 22) 데이터 합치기



데이터는 여러 방향으로 합칠 수 있어요. 가로, 세로 무슨말인지 말 모르시겠다구요?

한번에 다 설명드리는게 아니라 하나씩 해보면서 이해시켜 드릴게요.

이번 주에 합쳐 볼 데이터는 같은 형태의 데이터를 아래로 합쳐 보겠습니다. 지금 우리가 불러온 데이터는 7월 한달치입니다. 그렇다면 8월, 9월 더 나아가서 1년치의 데이터를 분석 해야한다면 어떻게 해야할까요?

```
chicken07 = pd.read_csv('./data/chicken_07.csv')
chicken08 = pd.read_csv('./data/chicken_08.csv')
chicken09 = pd.read_csv('./data/chicken_09.csv')

# 3분기 데이터
chicken_data = pd.concat([chicken07, chicken08, chicken09])
chicken_data
```

데이터는 잘 합쳐진 것 같은데 인덱스가 이상합니다!

76495 rows × 9 columns - 9개의 열에 76495개의 행이 있는데 우리의 인덱스는 24116로 끝나네요. 기존의 9월 데이터가 갖고 있던 인덱스를 그대로 유지해서 그렇습니다. 인덱스를 다시 넣어줍니다.

```
chicken_data = chicken_data.reset_index(drop=True)
chicken_data
```

성공!

10. 시각화, Matplotlib

▼ 23) Matplotlib 이란?



파이썬에서 사용되는 시각화 라이브러리입니다. 판다스가 관계형 데이터를 다루는데 사용된다면, Matplotlib은 그 데이터들을 시각화 하는데 사용합니다. 가장 기초가 되는 Matplotlib을 이용해서 바 차트들을 그려봅시다.

▼ 24) Matplotlib 불러오기

- Matplotlib도 사용하려면 불러와야 합니다.
- 판다스를 불러왔던 것 처럼 Matplotlib도 불러오겠습니다. 마찬가지로 너무 길기 때문에 plt라고 부르도록 하겠습니다.

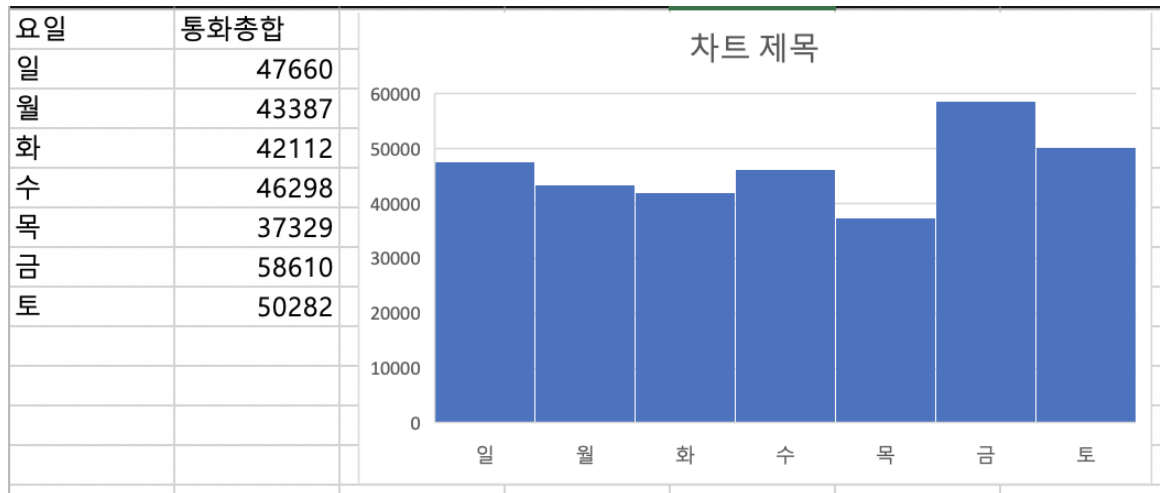
```
import pandas as pd
import matplotlib.pyplot as plt
```

마찬가지로 아무 일도 일어나지 않습니다. Matplotlib을 사용하러 가 봅시다.

11. Matplotlib 부딪혀보기

▼ 25) 요일에 따른 총 통화건수

- 첫 번째로 요일별로 총 통화 건수 살펴보겠습니다.
- 머릿속에서는 이미 완성된 그래프가 그려지는군요



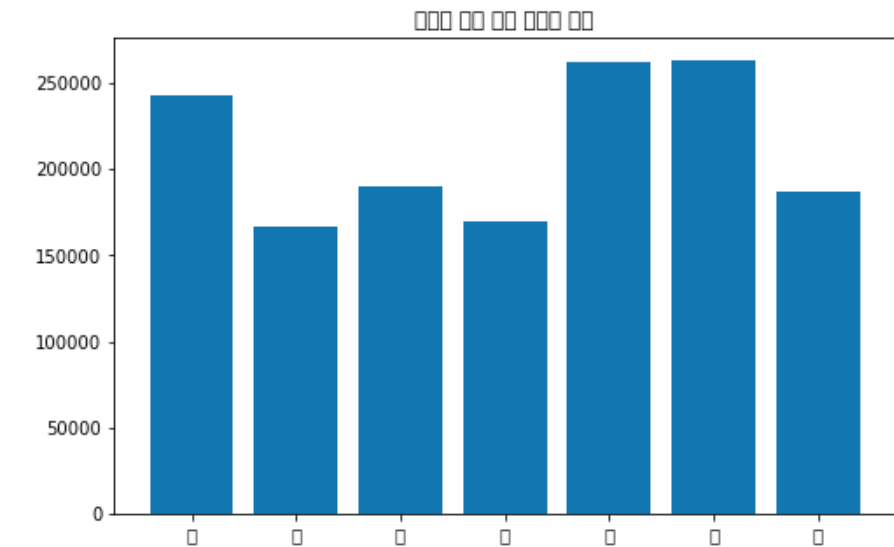
- 위 그래프를 그리는데 적어도 x축, y축, 그래프 제목이 필요해 보이네요
- 아래 코드를 통해 어떻게 그릴 수 있는지 살펴봅시다.
- 요일별 치킨 데이터는 굉장히 많겠죠? 월 - 15, 화 - 5.... 그 다음주 월 - 9, 화 - 19... 이런식입니다.
- 우리가 원하는 데이터는 요일을 기준으로 통화 건수를 모두 더한 데이터입니다.

```
sum_of_calls_by_week = chicken_data.groupby('요일')['통화건수'].sum()
sum_of_calls_by_week
```

```
요일
금    243013
목    166913
수    189884
월    169859
일    262655
토    263438
화    186617
Name: 통화건수, dtype: int64
```

네 이렇게 보니 요일별 통화건수의 총합의 데이터가 보이네요. 이제 이 데이터를 가지고 그래프를 그려봅시다.

```
plt.figure(figsize=(8,5)) # 그래프의 사이즈
plt.bar(sum_of_calls_by_week.index, sum_of_calls_by_week) # bar 그래프에 x축, y축 값을 넣어줍니다.
plt.title('요일에 따른 치킨 주문량 합계') # 그래프의 제목
plt.show() # 그래프 그리기
```



으악! 🤖

▼ 26) 한글 설정하기



폰트를 설정해주지 않아 한글이 깨지는 현상을 만난 것 입니다. 한글을 지원하는 폰트를 설정해서 그래프를 그려봅시다.

- 현재 설정되어 있는 폰트를 살펴봅시다. 환경 마다 다를 수 있겠죠?

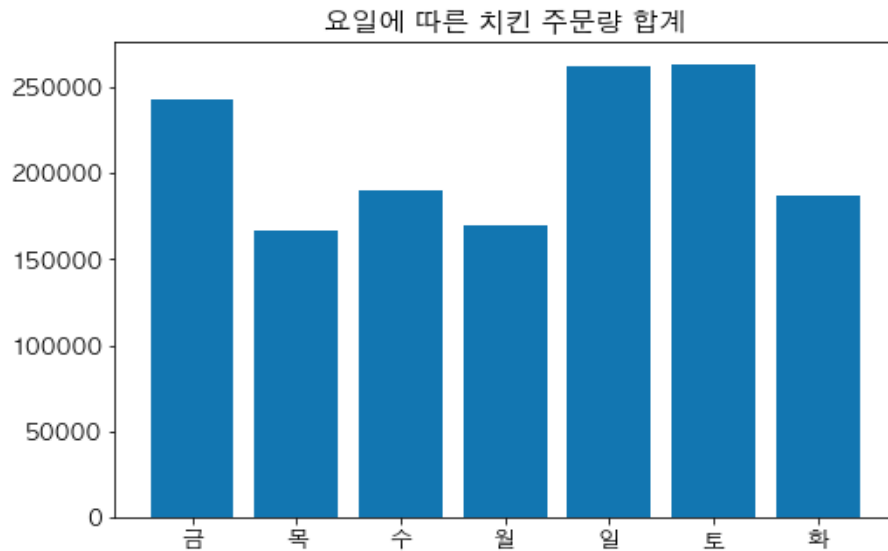
```
print('설정 되어 있는 폰트 사이즈 :', plt.rcParams['font.size'])
print('설정 되어 있는 폰트 글꼴 :', plt.rcParams['font.family'])
```

- 한글을 지원하는 폰트로 바꾸어줍니다.

```
# Apple은 'AppleGothic', Windows는 'Malgun Gothic'을 추천
plt.rcParams['font.family'] = "Malgun Gothic"
```

- 다시 그래프를 그려봅시다.

```
plt.figure(figsize=(8,5)) # 그래프의 사이즈
plt.bar(sum_of_calls_by_week.index, sum_of_calls_by_week) # bar 그래프에 x축, y축 값을 넣어줍니다.
plt.title('요일에 따른 치킨 주문량 합계') # 그래프의 제목
plt.show() # 그래프 그리기
```



▼ 여전히 한글이 뜨지 않는다면?

선택한 글꼴이 한글을 지원하지 않거나 matplotlib에서 이용할 수 없는 경우, 설치가 안되어있는 경우 등등 다양한 이유가 있을 수 있습니다.

▼ 나눔글꼴 설치하기 ([다운로드 링크](#))

나눔글꼴을 설치하고 다시 폰트를 설정해줍니다. (주피터 노트북 재실행이 필요할 수 있어요!)

```
plt.rcParams['font.family'] = "NanumGothic"
```

폰트를 설치했는데도, 주피터 노트북을 재실행해도 여전히 인식을 못한다면 아래처럼 matplotlib이 참조하는 폰트 목록을 새로고침해주세요.

```
import matplotlib.font_manager as fm
fm._rebuild()
```

또는 이용 가능한 폰트를 찾아줄 수도 있습니다.

- font_manager 를 불러와 줍니다.

```
# fontmanager 임포트하기
import matplotlib.font_manager as fm

# 이용 가능한 폰트 중 '고딕'만 선별
fonts_list = [f.name for f in fm.fontManager.ttflist if 'Gothic' in f.name]

fonts_list
# ['Franklin Gothic Book', 'Kozuka Gothic Pro', 'Showcard Gothic', ...]
```

- 위에서 나온 폰트 중 하나를 골라 설정해줍니다.

```
plt.rcParams['font.family']="Malgun Gothic"
```

12. Matplotlib 그래프 입맛대로 바꿔보기

▼ 27) 정렬을 해봅시다



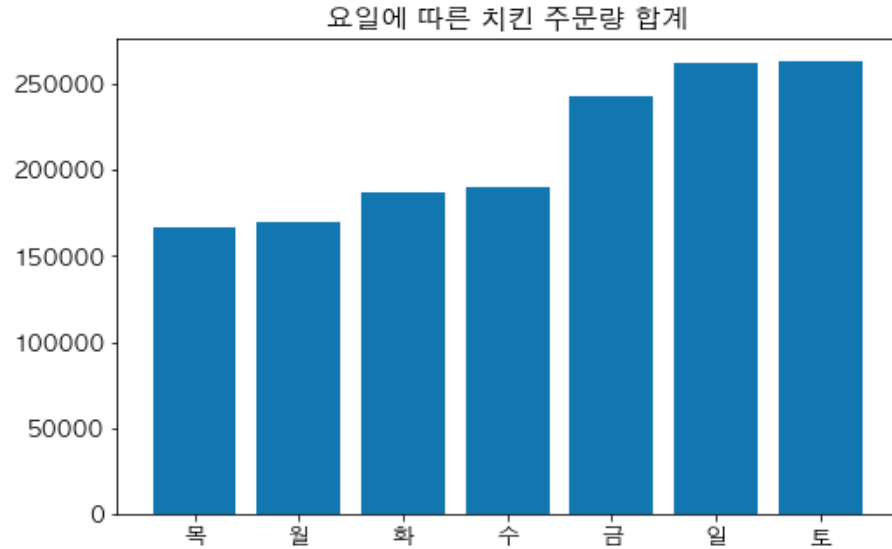
우리가 그린 그래프도 의미가 있으나 보통의 바 그래프는 우 상향 그래프나, 좌 상향 그래프로 정렬해서 어떤 항목이 가장 높은지 보면서 인사이트를 추가적으로 얻을 수 있습니다. 가장 적은 요일 순으로 정렬 해봅시다.

```
sum_of_calls_by_week = chicken_data.groupby('요일')['통화건수'].sum()
sum_of_calls_by_week
```

- 코드가 너무 길죠?
- 체이닝(Chaining)이라고 하는 나열법인데, 익숙하면 편해요. 하지만 지금은 잘라서 살펴보겠습니다.
- `chicken_data.groupby('요일')` : 일단 데이터를 요일 별로 그룹을 지어줍니다.
- 모든 데이터는 월~일요일을 기준으로 7 덩어리가 되었겠죠?
- `chicken_data.groupby('요일')['통화건수']` : 그 중에서 통화 건수를 선택합니다
- `chicken_data.groupby('요일')['통화건수'].sum()` : 요일 별 통화 건수의 총합을 `sum_of_calls_by_week` 변수에 저장합니다.

```
# 요일 별로 모아주기
groupdata = chicken_data.groupby('요일')
# '통화건수' 열만 떼어보기
call_data = groupdata['통화건수']
# 요일 별로 더해주기
sum_of_calls_by_week = call_data.sum()
sorted_sum_of_calls_by_week = sum_of_calls_by_week.sort_values(ascending=True)

plt.figure(figsize=(8,5)) # 그림의 사이즈
plt.bar(sorted_sum_of_calls_by_week.index, sorted_sum_of_calls_by_week) # 바 그래프
plt.title('요일에 따른 치킨 주문량 합계') # 그래프의 제목
plt.show() # 그래프 그리기
```



이렇게 살펴보니 목요일이 가장 적고, 토요일이 전화 주문량이 가장 많음을 알 수 있습니다.

▼ 28) 요일 순서대로 그려봅시다



그릴 그래프의 종류는 같습니다. 데이터도 그대로입니다. 단지 우리는 요일별로 정렬을 다시 할 뿐입니다. 앞에서 정렬되지 않은 index를 사용했다면, 이번에는 index를 우리가 정한 순서대로 정렬해서 사용해 봅시다.

```
weeks = ['월', '화', '수', '목', '금', '토', '일'] # 우리가 정한 순서
sum_of_calls_by_weeks = chicken_data.groupby('요일')['통화건수'].sum().reindex(weeks) # 인덱스 다시 정렬

plt.figure(figsize=(8,5)) # 그림의 사이즈
plt.bar(sum_of_calls_by_weeks.index, sum_of_calls_by_weeks) # 바 그래프
plt.title('요일에 따른 치킨 주문량 합계') # 그래프의 제목
plt.show() # 그래프 그리기
```

13. Quiz_다른 컬럼 분석 해보기

▼ 29) 🍴 연령대 별로 치킨 주문량은 다를까?

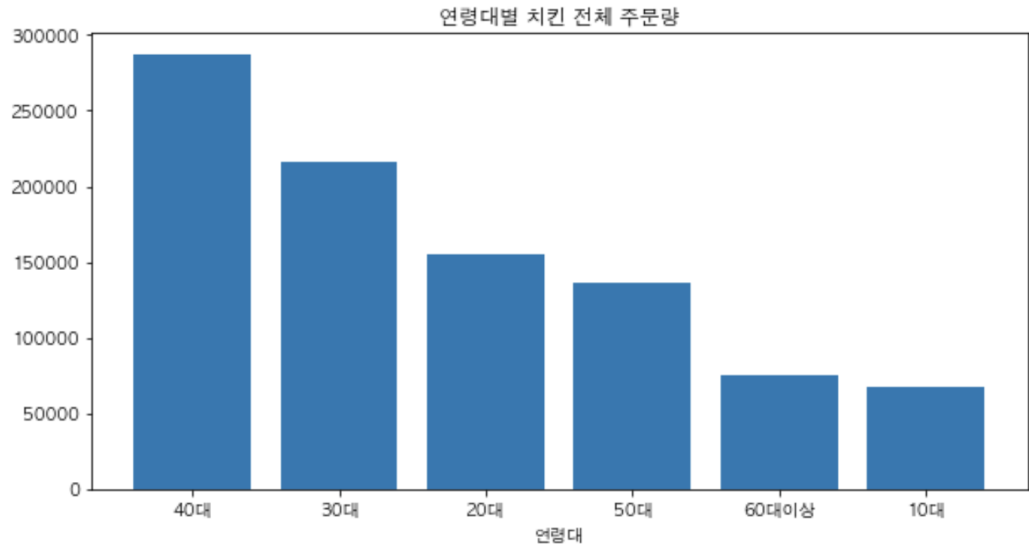
▼ Q. 퀴즈 설명



연령대 별로 치킨의 주문량이 어떻게 다른지 분석해 봅시다. 위에서 요일 데이터를 가지고 총 합을 구했다면, 이번에는 '연령대' 데이터를 분석 해봅시다.

▼ 모습보기

- x축에는 연령대가 y축에는 전체 주문량이 나올 수 있도록 해줍니다.



▼ A. 정답 보기

```
# 연령대 별로 모아주기
groupdata = chicken_data.groupby('연령대')
# '통화건수' 열만 떼어보기
call_data = groupdata['통화건수']
# 요일 별로 더해주기
sum_of_calls_by_age = call_data.sum()
sorted_sum_of_calls_by_age = sum_of_calls_by_age.sort_values(ascending=False)
sorted_sum_of_calls_by_age
```

```
plt.figure(figsize=(10,5))
plt.bar(sorted_sum_of_calls_by_age.index, sorted_sum_of_calls_by_age)
plt.xlabel('연령대') # x축에 이름을 붙여줍니다
plt.title('연령대별 치킨 전체 주문량')
plt.show()
```

▼ 30) 📍 서울에서 치킨은 어디에서 가장 많이 먹을까?

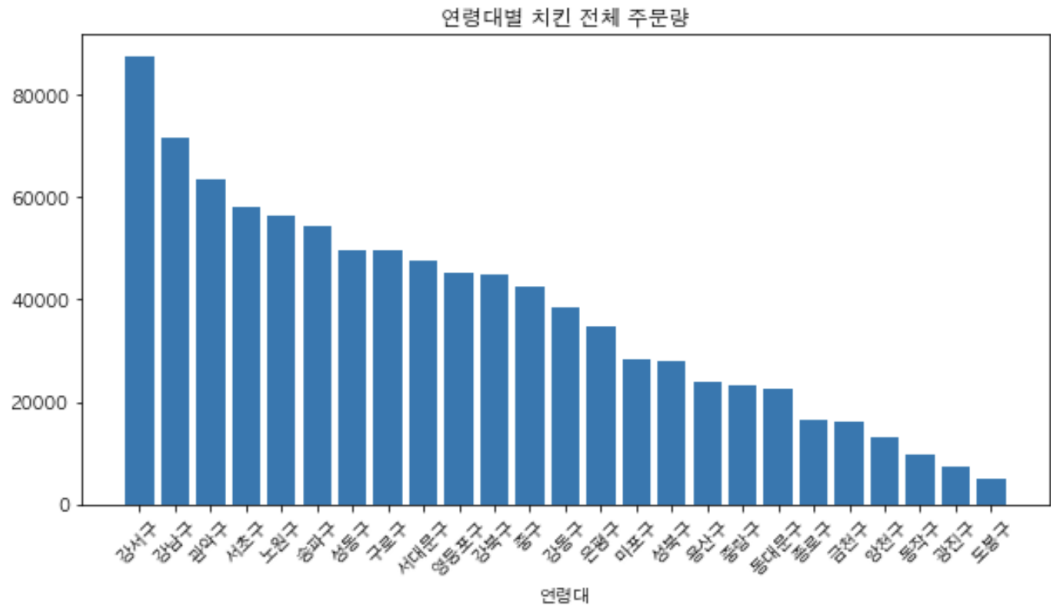
▼ Q. 퀴즈 설명



서울시에서 가장 많은 치킨을 시켜 먹는 '시군구'를 알아봅시다. 이전의 데이터들과 마찬가지로 우리가 가지고 있는 데이터를 사용해 분석할 수 있겠죠? 자 바 그래프를 정복해봅시다.

▼ 모습보기

- x축에는 시군구가 y축에는 전체 주문량이 나올 수 있도록 해줍니다.



▼ A. 정답 보기

```
# 시군구 별로 모아주기
groupdata = chicken_data.groupby('시군구')
# '통화건수' 열만 떼어보기
call_data = groupdata['통화건수']
# 요일 별로 더해주기
sum_of_calls_by_city = call_data.sum()
sorted_sum_of_calls_by_city = sum_of_calls_by_city.sort_values(ascending=False)
sorted_sum_of_calls_by_city
```

```
plt.figure(figsize=(10,5))
plt.bar(sorted_sum_of_calls_by_city.index, sorted_sum_of_calls_by_city)
plt.xlabel('지역별') # x축에 이름을 붙여줍니다
plt.xticks(rotation=45)
plt.title('지역별 치킨 전체 주문량')
plt.show()
```

14. 끝 & 숙제 설명



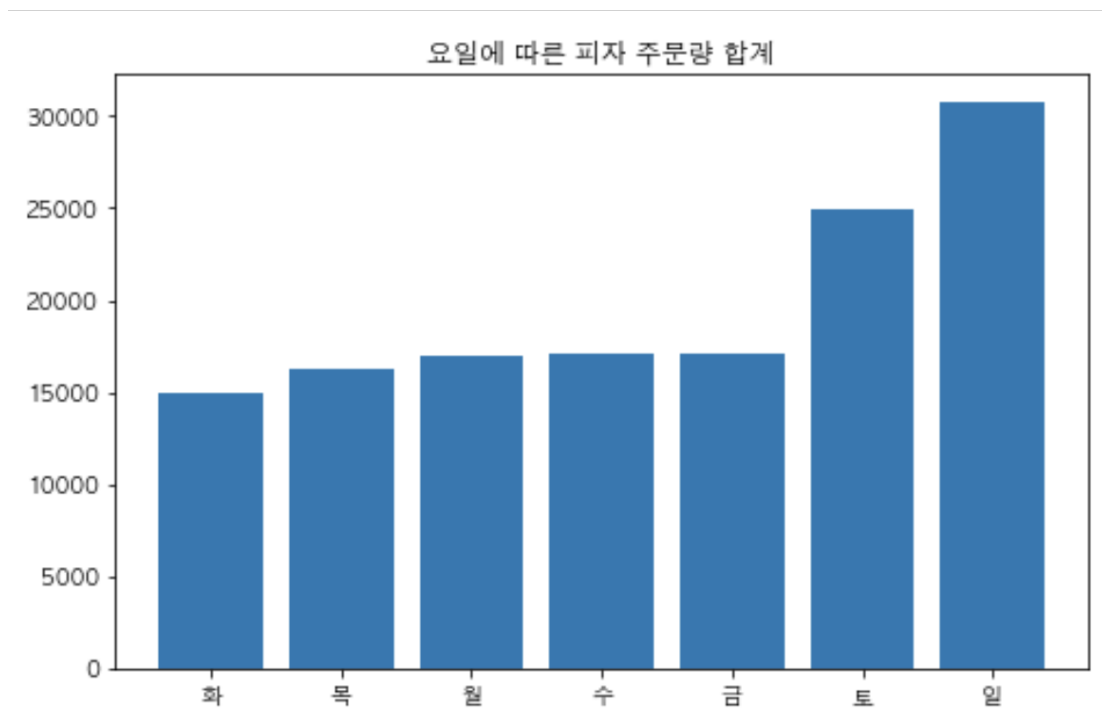
치킨에는 피자! 이번에는 피자 데이터를 분석 해봅시다.

▼ 설명 보기

일자	요일	성별	연령	발신지_시도	발신지_구	발신지_동	통화건수
20190901	일	남	30대	서울특별시	강남구	삼성동	9
20190901	일	남	30대	서울특별시	강남구	논현동	10
20190901	일	남	30대	서울특별시	강남구	개포동	5
20190901	일	여	50대	서울특별시	강남구	삼성동	7
20190901	일	여	50대	서울특별시	강남구	도곡동	5
20190901	일	여	50대	서울특별시	강남구	논현동	5
20190901	일	여	50대	서울특별시	강남구	개포동	5
20190901	일	여	40대	서울특별시	강남구	역삼동	11
20190901	일	여	40대	서울특별시	강남구	삼성동	19
20190901	일	여	40대	서울특별시	강남구	도곡동	5
20190901	일	여	40대	서울특별시	강남구	대치동	5
20190901	일	여	40대	서울특별시	강남구	논현동	6

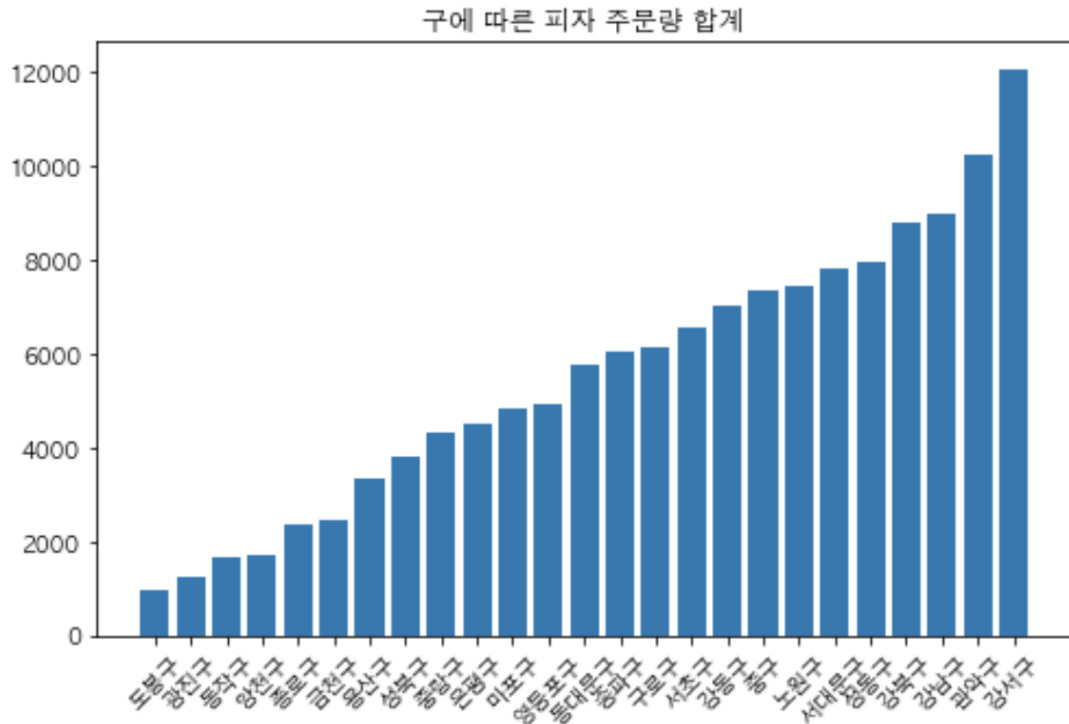
▼ Q1. 피자를 가장 많이 시켜먹는 요일은 언제인가요?

결과 화면 예시)



▼ Q2. 피자를 가장 많이 시켜먹는 구는 어디인가요?

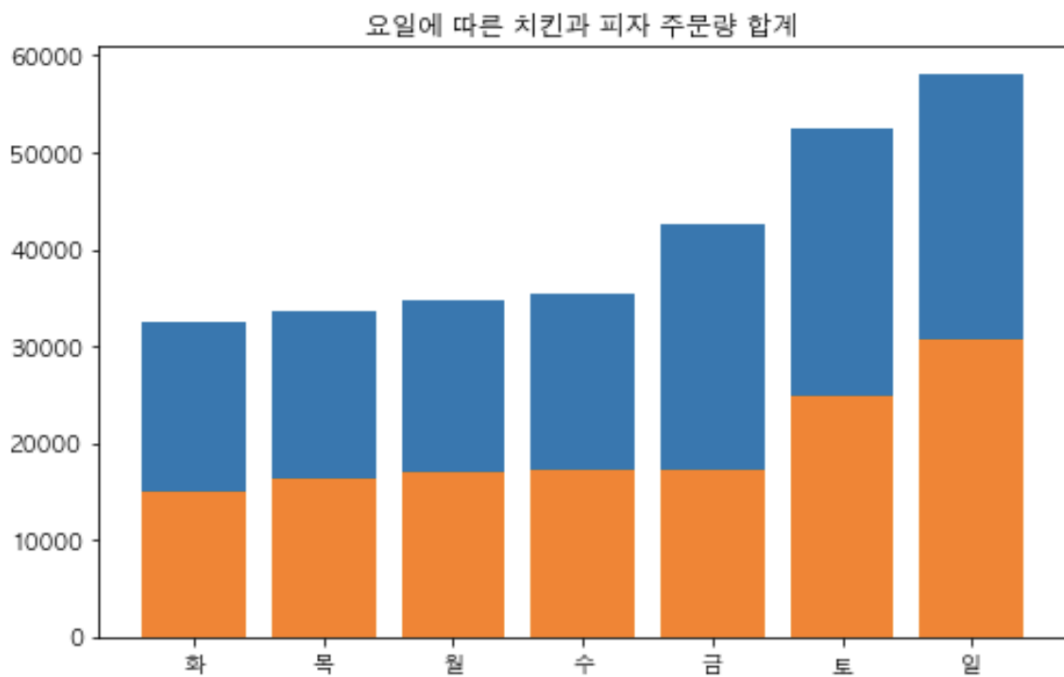
결과 화면 예시)



▼ 바 그래프 2개 그리기

- 그래프를 그릴 때 다음과 같이 2번 그리면 그래프 위에 그래프를 그릴 수 있습니다

```
plt.bar(chicken_data.index, chicken_data) # 바 그래프
plt.bar(pizza_data.index, pizza_data)
```





plt.plot()을 사용하면 line 차트를 그릴 수 있습니다.

15. 1주차 숙제 답안 코드

▼ [코드스니펫] - 1주차 숙제 답안 코드

전체 코드

```
import pandas as pd
pizza = pd.read_csv('./data/pizza_09.csv')
```

```
pizza_sum = pizza.groupby('요일').sum()['통화건수']
sorted_pizza_data = pizza_sum.sort_values(ascending=True)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
plt.bar(sorted_pizza_data.index, sorted_pizza_data)
plt.title('요일별 피자 주문 총량')
```

```
pizza_gu_data = pizza.groupby('발신지_구').sum()['통화건수'].sort_values(ascending=True)

plt.figure(figsize=(20,5))
plt.bar(pizza_gu_data.index, pizza_gu_data)
plt.title('구별 피자 주문 총량')
plt.xticks(rotation=90)
```

Copyright © TeamSparta All rights reserved.