2018-09-27

데이터 분석을 위한 파이썬
&
데이터 시각화

3기 김현세

Growth Hackers

CONTENTS

1. 기초 파이썬 복습
2. 기초 파이썬 심화
3. Numpy 패키지
4. Pandas 패키지
5. 데이터 시각화

기초 파이썬 복습

Q. 이 코드를 올바르게 고칠 때, 그 실행 결과는?

```
tmp = 0
while ['a']:
tmp += 1
if tmp >= 100:
break
print(tmp)
```

기초 파이썬 복습

A.

```
tmp = 0
while ['a']:
    tmp += 1
    if tmp >= 100:
        break
    print(tmp)
```

빈 리스트가 아닌 이상 True에 대응되므로,
반복문이 무한으로 실행됨

반복되어야 하는
코드들은 들여쓰기

tmp >= 100이 만족될 경우에만
반복문을 벗어나도록 한 번 더 들여씀

결과: 1, 2, 3, ..., 99까지 출력됨

Growth Hackers

기초 파이썬 복습

슬라이싱 연습

```
a = list(range(100))
```

기초 파이썬 복습

슬라이싱 연습

a = [0, 1, 2, ..., 99]

기초 파이썬 복습

슬라이싱 연습

a = [0, 1, 2, ..., 99]

a[1:10] =

a[11:] =

a[-10:] =

a[-10:-1] =

a[-1:-10] =

기초 파이썬 복습

슬라이싱 연습

a = [0, 1, 2, ..., 99]

a[1:10] = [1, 2, 3, ... , 9]

a[11:] = [11, 12, ..., 99]

a[-10:] = [90, 91, ..., 99]

a[-10:-1] = [90, 91, ..., 99]

a[-1:-10] = []

기초 파이썬 복습

재귀 함수: 함수 정의 내에 함수가 다시 사용되는 함수

→ 반복적인 작업을 수행하는 함수를 짤 때 매우 효과적!

Q. 아래 dictionary의 모든 문자열을 원소로 갖는 리스트를 만드시오.

{ 'a' : { 'b' : { 'c' : { 'd' : { 'e' : {}}}}}, 'aa' : { 'bb' : { 'cc' : { 'dd' : { 'ee' : {}}}}}

정답: { 'a' : { 'b' : { 'c' : { 'd' : { 'e' : {}}}}}, 'aa' : { 'bb' : { 'cc' : { 'dd' : { 'ee' : {}}}}}
→ ['a', 'b', 'c', 'd', 'e', 'aa', 'bb', 'cc', 'dd', 'ee']

중첩된 딕셔너리의 key 값을 반복적으로 뽑아내는 것이 풀이의 핵심!

기초 파이썬 복습

A.

```
def dict2list(input):
    ret = []
    for key, val in input.items():
        ret.append(key)
        ret.extend(dict2list(val))
    return ret
```

하위 딕셔너리를 인자로 하는 함수를 실행!
{}가 인자로 보내져서 실행되는 함수가 마지막

기초 파이썬 심화

Iterable: 반복 가능한 객체

e. g.

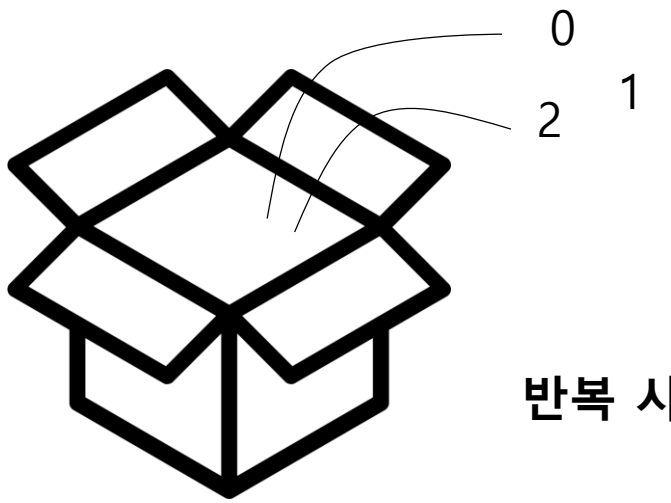
[0, 1, 2, 3, 4, 5]

range(6)

```
range(6)
```

```
range(0, 6)
```

← 0, 1, 2, 3, 4, 5 형태로 반환되지 않음.



반복 시마다 1개씩 자료를 반환하는 객체

Iterator: next()를 호출할 때 다음 값을 생성해내는 헬퍼 객체
itearble에 iter() 함수를 적용하여 만들 수 있음

```
In [1]: iter_ex = iter(range(6))

In [2]: next(iter_ex)
Out[2]: 0

In [3]: next(iter_ex)
Out[3]: 1
```

```
In [7]: next(iter_ex)
Out[7]: 5

In [8]: next(iter_ex)
---------------------------------------
StopIteration
<ipython-input-8-4f0e2e2
----> 1 next(iter_ex)
```

통상 한 사이클을 다 돌면 끝!

Growth Hackers

기초 파이썬 심화

자주 쓰이는 함수들

map(function, iterable) – iterable인 자료의 각 원소에 function을 적용한 iterable을 반환.

zip(iterable1, iterable2) – iterable1과 iterable2의 원소 쌍으로 이루어진 새로운 iterable을 반환.

enumerate(iterable) – (인덱스, iterable 원소) 튜플을 원소로 갖는 iterable을 반환.

lambda 변수: 반환 값 – 변수를 원소로 받아 반환 값을 반환하는 일종의 간이 함수를 정의.

기초 파이썬 심화

예시 코드

```
f = lambda x: x**2

a = range(10)

b = map(f, a)

c = zip(range(10), b)

d = enumerate(b)

e = enumerate(d)
```

```
for x in ?:
        print(x)
```


Growth Hackers

기초 파이썬 심화

예시 코드

```
f = lambda x: x**2

a = range(10)

b = map(f, a)

c = zip(range(10), b)

d = enumerate(b)

e = enumerate(d)
```

```
for x in ?:
        print(x)
```

```
for x in a:
    print(x)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for x in b:
    print(x)
```

```
0
1
4
9
16
25
36
49
64
81
```

```
for x in c:
    print(x)
```

```
(0, 0)
(1, 1)
(2, 4)
(3, 9)
(4, 16)
(5, 25)
(6, 36)
(7, 49)
(8, 64)
(9, 81)
```

```
for x in d:
    print(x)
```

```
(0, 0)
(1, 1)
(2, 4)
(3, 9)
(4, 16)
(5, 25)
(6, 36)
(7, 49)
(8, 64)
(9, 81)
```

```
for x in e:
    print(x)
```

```
(0, (0, 0))
(1, (1, 1))
(2, (2, 4))
(3, (3, 9))
(4, (4, 16))
(5, (5, 25))
(6, (6, 36))
(7, (7, 49))
(8, (8, 64))
(9, (9, 81))
```

기초 파이썬 심화

반환: 함수를 호출했을 때 함수의 논리에 따라 특정 값이 도출되는 것

e. g.
```
f = lambda x: x*2
f(2)
```
```
4
```

확장된 개념 - 객체를 호출했을 때 특정 값이 도출되는 것

e. g.
```
a = [0, 1, 2, 3]
a
```
← 리스트를 호출하면, 리스트가 반환됨
```
[0, 1, 2, 3]
```

```
a = range(4)
a
```
```
range(0, 4)
```
```
b = map(f, a)
b
```
```
<map at 0x213e0fae6d8>
```
← 꼭 모든 iterable이 바로 원소들을 반환하는 것은 아님.

기초 파이썬 심화

반환과 출력의 차이

```
a
b
```

```
<map at 0x213e0fae6d8>
```

객체의 이름을 호출하는 방식으로는
마지막 하나의 반환 값만을 출력할 수 있으나,

```
print(a)
print(b)
```

```
range(0, 4)
<map object at 0x00000213E0FAE6D8>
```

print() 등 출력 함수를 사용하면
여러 반환 값을 동시에 출력할 수 있다.

기초 파이썬 심화

튜플의 각 원소를 각각의 변수에 저장하는 방법

```
def powers(x):
    return x**2, x**3, x**4
```

```
result = powers(2)
```

```
result
```

```
(4, 8, 16)
```

여러 반환 값을 갖는 함수는 반환 값들로 이루어진 튜플을 반환함.

```
result1, result2, result3 = powers(2)
```

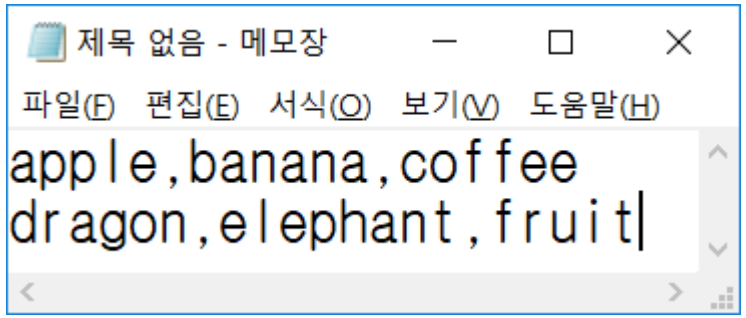
```
result1
```

```
4
```

이렇게 하면, 개별 변수에 각 반환 값을 할당할 수 있음.

기초 파이썬 심화

문자열 다듬기




```
line = open('temp.txt','r').readline()
```

```
line
```

```
'apple,banana,coffee\n'
```

← 위 텍스트 파일의 첫 줄을 읽으면 이와 같다

Q. 이 문자열을 [apple, banana, coffee] 리스트로 바꾸려면?

Growth Hackers

기초 파이썬 심화

문자열 다듬기

A. 맨 끝의 '₩n'을 없애고, ','를 기준으로 문자열을 나누어야!

- strip(): 양 쪽 공백을 지우는 매서드
- split(x): x를 기준으로 문자열을 나누는 매서드

```
line.strip().split(',')
```

```
['apple', 'banana', 'coffee']
```

기타 문자열 매서드

upper(): 대문자로 변환, lower(): 소문자로 변환
count(x): x의 개수를 반환, find(x): x가 나타나는 첫 인덱스 반환
replace(x, y): x를 y로 변환
join(x): x의 문자열 원소들을 해당 문자열을 매개로 결합
ex. ':'.join['Hyunse', 'Handsome'] = 'Hyunse:Handsome'

Growth Hackers

Numpy 패키지

Numpy: 과학 계산을 위한 라이브러리로서 다차원 배열을 처리하는 데 유용

```
import numpy as np
```

Import 시, 통상 'np'로 축약하여 사용함

array: Numpy의 대표적 자료형으로서, '행렬'과 비슷함

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} =$$

```
np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

array와 list는 대응 관계

```
np.array([[1, 2, 3], [4, 5, 6], [7, 8]])
```

```
array([list([1, 2, 3]), list([4, 5, 6]), list([7, 8])], dtype=object)
```

▲ 원소들의 차원이 맞지 않으면, 연산이 어려울 수 있음

```
arrn = np.array([[[8, 6, 7], [9, 8, 7], [5, 1, 6], [2, 9, 7]],
                 [[6, 8, 1], [1, 2, 3], [7, 0, 2], [4, 0, 9]]])
```

```
arrn
```

```
array([[[8, 6, 7],
        [9, 8, 7],
        [5, 1, 6],
        [2, 9, 7]],

       [[6, 8, 1],
        [1, 2, 3],
        [7, 0, 2],
        [4, 0, 9]]])
```

▲ 이런 것은 행렬과 다른 부분

Growth Hackers

Numpy 패키지

내장 함수

np.array(iterable): iterable과 같은 원소를 담은 array를 반환

np.arange(): 기본 내장 함수 range()와 동일하나, array 형태로 반환

np.zeros(shape), np.ones(shape), np.full(shape, value): 각각 0, 1, value로 채워진 array를 반환

np.eye(n): n*n 형태의 Identity Matirx array를 반환

np.random.randint(n, m, size = (x, y,..)): n 이상 m 미만의 임의의 수로 채워진 array를 반환

Array의 매서드

reshape(shape): array의 형태를 변환

ndim, shape, size: array의 차원, 형태, 원소의 개수를 반환

numpy_basic.ipynb 파일 참고!

Numpy 패키지

Array의 인덱싱

[x][y]의 형태가 아니라 [x, y]의 형태로 인덱싱

슬라이싱 또한 [a:b, c:d]의 형태로 하나의 대괄호 안에서 해결

Boolean Indexing: 동일한 형태의 Boolean Array를 인덱스로 넣으면, True에 해당하는 entity만 반환됨

Array의 연산

기본적인 사칙연산은 element-wise로 적용됨

행렬곱은 np.dot() 함수 활용

np.sum(array, axis=n)

n번 째 차원의 축을 기준으로 원소 간 합을 도출해 반환함

Append & delete

append(a, b, axis=n): n번 째 차원의 축을 기준으로 array a에 array b를 덧붙임

delete(arrn, n, m): m번 째 축의 n번 째 array를 삭제함

Pandas 패키지

Pandas: 데이터 처리와 분석을 위한 라이브러리

```
import pandas as pd
```

Import 시, 통상 'pd'로 축약하여 사용함

Pandas의 기본 자료형
- Series: 1차원 자료 구조
- DataFrame: 2차원 자료 구조
- Panel: 3차원 자료 구조

위 자료형들의 특징은 0 이상 정수가 아닌, 별도의 인덱스를 지정할 수 있다는 점!

```
a = pd.Series(['a','b','c'], index=['no1','no2','no3'])
```

```
a
```

```
no1    a
no2    b
no3    c
dtype: object
```

Pandas 패키지

DataFrame

2차원 자료형으로, 기반이 되는 자료형은 딕셔너리

딕셔너리의 key-value 쌍이 DataFrame에 하나의 column으로 들어감

DataFrame의 인덱싱

Column을 인덱싱/슬라이싱 할 때에는 [col_name] or [[col_name1, col_name2,...]] 형태로

Boolean 인덱싱 동일하게 사용 가능함.

iloc 매서드 -> [row, column] / loc 매서드 -> [index, col_name]

DataFrame 조작

사칙연산은 numpy.array와 마찬가지로 element-wise로 이루어짐

새로운 column 추가는 dictionary에서와 동일

새로운 row의 추가는 append 매서드 이용

loc 매서드 사용 시 개별 원소 추가 가능

Pandas 패키지

csv 파일 입출력

입력: pd.read_csv('파일 경로', engine = x, index_col = y)

출력: DataFrame.to_csv('파일 경로', encoding = x)

DataFrame의 주요 매서드

평균 – mean(), 분산 – var(), 표준편차 – std(), 최소값 – min(), 최대값 – max()

apply(function): DataFrame의 특정 부분에 function을 적용한 것을 반환

head(): DataFrame의 앞 5줄만을 보여줌

groupby([col_name]): 특정 column의 값을 기준으로 데이터를 묶어줌

pandas_basic.ipynb 파일 참고!

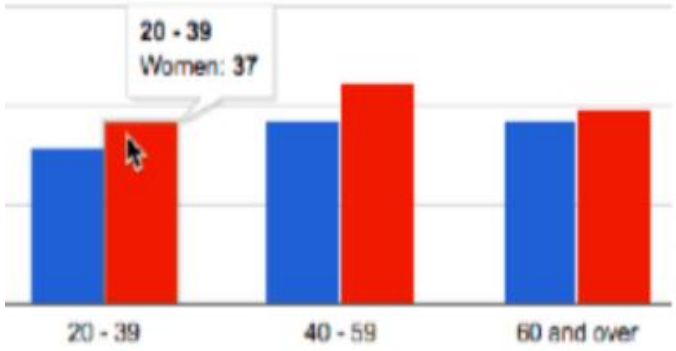
데이터 시각화

시각화의 종류

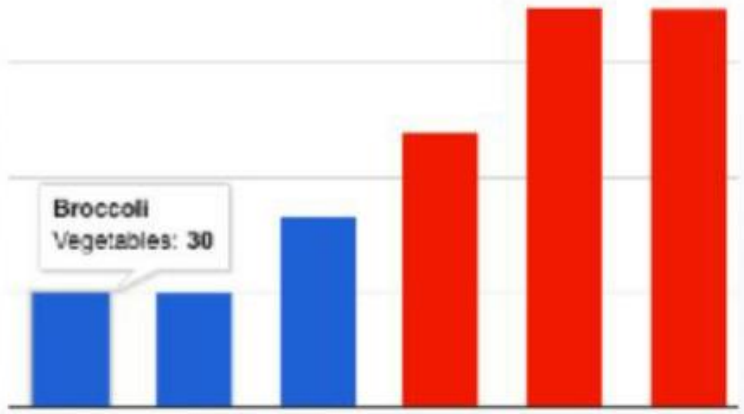
정보 시각화 방법				
시간 시각화	분포 시각화	관계 시각화	비교 시각화	공간 시각화
막대그래프 (Bar graph) 누적 막대그래프 (Stacked Bar graph) 점그래프 (Point graph)	파이차트 (Pie chart) 도넛 차트 (Donut chart) 트리맵 (Tree map) 누적 연속 그래프 (Cumulative continuous graph)	스캐터 플롯 (Scatter plot) 버블 차트 (Bubble chart) 히스토그램 (Histogram)	히트맵 (heat map) 체르노프 페이스 (Chernoff face) 별그래프 (Star graph) 평행 좌표계 (Parallel coordinate system) 다차원 척도법 (Multi-dimensional scaling)	지도 매핑 (Dataviz on map)

Growth Hackers

데이터 시각화



- Grouped column or bar
  - Best to compare categories side-by-side. Vertical columns, or horizontal bars for long labels.



- Separated column or bar
  - Best to compare categories in separate clusters.

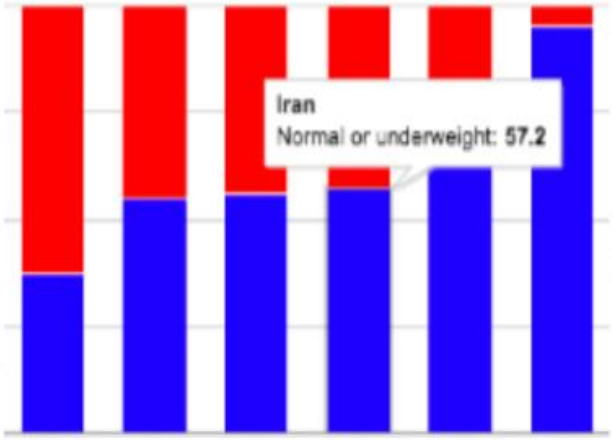
2차원

OR

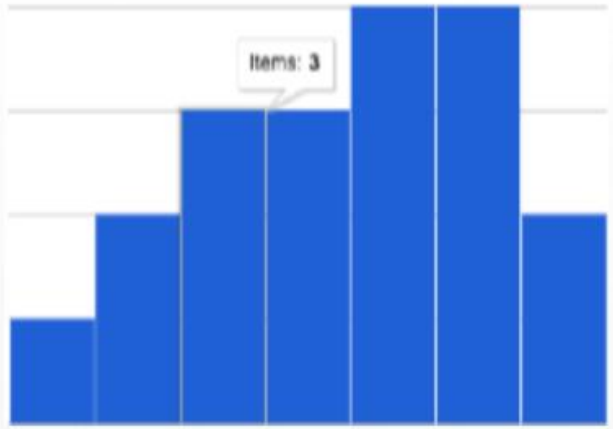
Index + 1차원

Growth Hackers

데이터 시각화



- Stacked column or bar
  - Best to compare sub-categories, or parts of a whole. Vertical columns, or horizontal bars for long labels.



- Histogram
  - Best to show distribution of raw data, with number of values in each bucket.

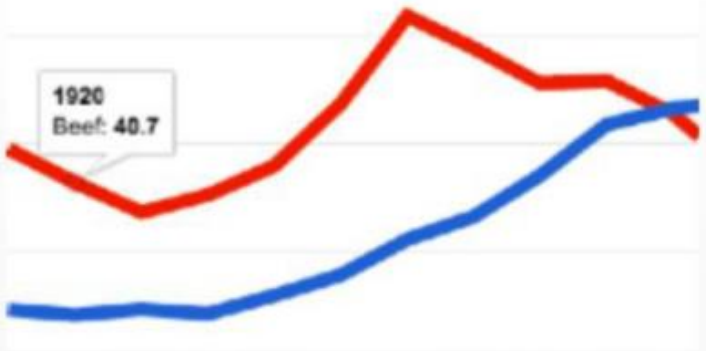
2차원

OR

Index + 1차원

Growth Hackers

데이터 시각화



- Pie chart
  - Best to show parts of a whole, but hard to estimate size of slices.
- Line chart
  - Best to show continuous data, such as change over time.

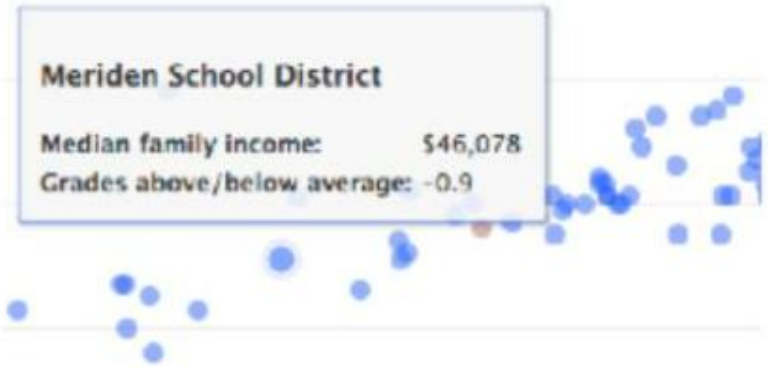
2차원

OR

Index + 1차원

Growth Hackers

데이터 시각화



- Scatter chart **2차원**
  - Best to show relationship between two sets of data. Also called an XY chart.

- Bubble chart **3+차원**
  - Best to show relationship between three or four sets of data, using bubble size and color.

Growth Hackers

데이터 시각화

matplotlib: 자료를 시각화하는 데 유용한 패키지

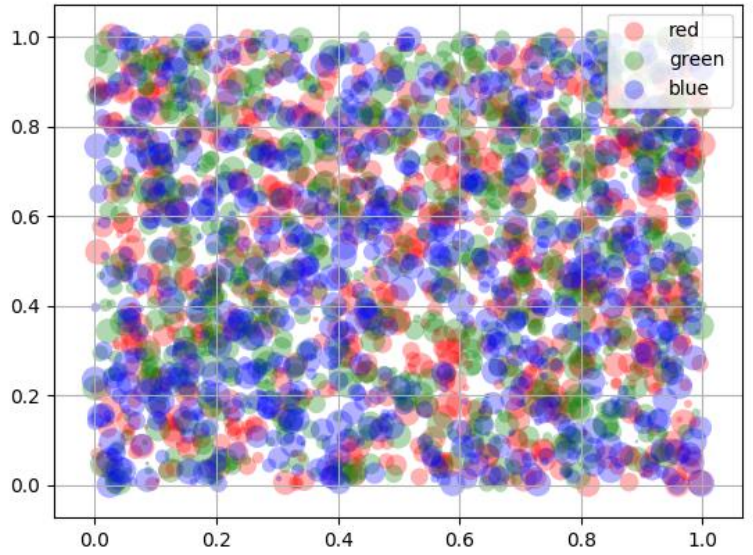
```
from matplotlib import pyplot as plt
```

pyplot만 import하며 통상 plt로 축약함

선 그래프
막대 그래프
히스토그램
산포도
…

다양한 종류의 시각화를 할 수 있음!

e. g.



matplotlib_basic.ipynb 파일 참고!

데이터 시각화

차트

Line Chart: plt.plot(x, y, ...) 색: color, 점 모양: marker, 선 모양: linestyle, 라벨: label ...

Bar Chart: plt.bar(x, y, ...) 색: color, 너비: width, 라벨: label ...

Histogram: plt.hist(x, ...) 구간: range, 구간 개수: bins, 스타일: histtype, 색: color ...

Scatter Plot: plt.scatter(x, y, ...) 색: color, 점 크기: size, 점 모양: marker, alpha: 투명도 ...

주요 함수

plt.show(): 설정된 차트를 display하는 함수
plt.figure(): 차트에 대한 각종 설정을 넣는 함수 차트 크기: figsize
plt.subplot(n, m, x): n*m등분한 칸 중 x번 째 칸에 차트를 그리겠다는 선언
plt.title(x): x를 타이틀로 넣음 / plt.xlabel(x): x를 횡축 라벨로 넣음 / plt.ylabel(x): x를 종축 라벨로 넣음
plt.grid(True): 그리드를 넣음 / plt.legend(): 범례를 넣음
plt.xtics(xs, labels): xs가 나타내는 x축 위치들에 labels의 원소들을 라벨로 넣음
plt.annotate(label, (x, y)): (x, y)가 나타내는 위치에 label을 라벨로 넣음

QUEST

1. numpy, pandas를 활용하여, 'sex ratio.csv'로부터 아래와 같은 형태를 가진 'new sex ratio.csv'를 반환하는 코드를 구현해주세요.

	A	B	C	D	E	F
1		2006	2007	2008	2009	2010
2	11	102	102	102	102	101
3	21	104	104	105	105	106
4	22	102	103	103	104	105
5	23	103	104	105	105	105
6	24	101	101	101	101	101
7	25	102	102	102	103	103
8	26	103	104	106	108	108
9	31	102	102	102	103	103
10	32	112	112	112	113	113
11	33	109	109	109	110	110
12	34	113	113	114	114	114

sex ratio

	A	B	C	D	E	F
1		지역	연도	성비		
2	0	11	2006	102		
3	1	11	2007	102		
4	2	11	2008	102		
5	3	11	2009	102		
6	4	11	2010	101		
7	5	11	2011	101		
8	6	11	2012	101		
9	7	11	2013	101		
10	8	11	2014	101		
11	9	11	2015	100		
12	10	11	2016	100		

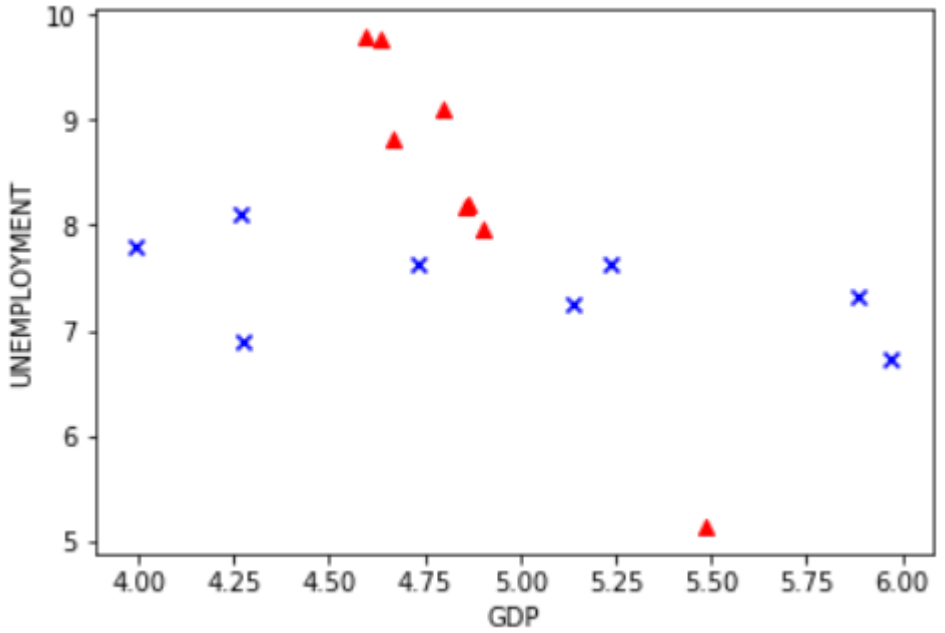
new sex ratio

지역별 성비 row 데이터를 column 형태로 바꾼 뒤 이를 일렬로 이어 붙인 거대한 column으로 만듦.
그리고 그 왼쪽에는 해당 데이터의 지역 코드와 연도가 쓰여있는 2개의 colum이 있도록 해야 함.
※ 파일 참고

Growth Hackers

QUEST

2. ‘province data.csv’로부터 지역별로 전 기간에 걸친 평균 gdp와 unemploymen를 구한 뒤, 이를 산포도로 표현하세요. 단, 평균 성비가 110 이상인 지역의 산포도는 파란색 X 모양의 점, 110 미만인 지역의 산포도는 빨간색 세모 모양의 점으로 표현하세요.



Growth Hackers

참고 자료

- 이영준, 고병욱(GH 2기) – 파이썬 세션 자료
- 정재근, 빈다은, 고병욱(GH 2기) – 데이터 시각화 & 다루기 세션 자료

- 예제로 배우는 Python 프로그래밍 – http://pythonstudy.xyz/python/article
- 이터레이터와 제너레이터 - https://mingrammer.com/translation-iterators-vs-generators/
- https://pandas.pydata.org/pandas-docs/
- https://docs.scipy.org/doc/numpy/reference/
- https://matplotlib.org/api/_as_gen/