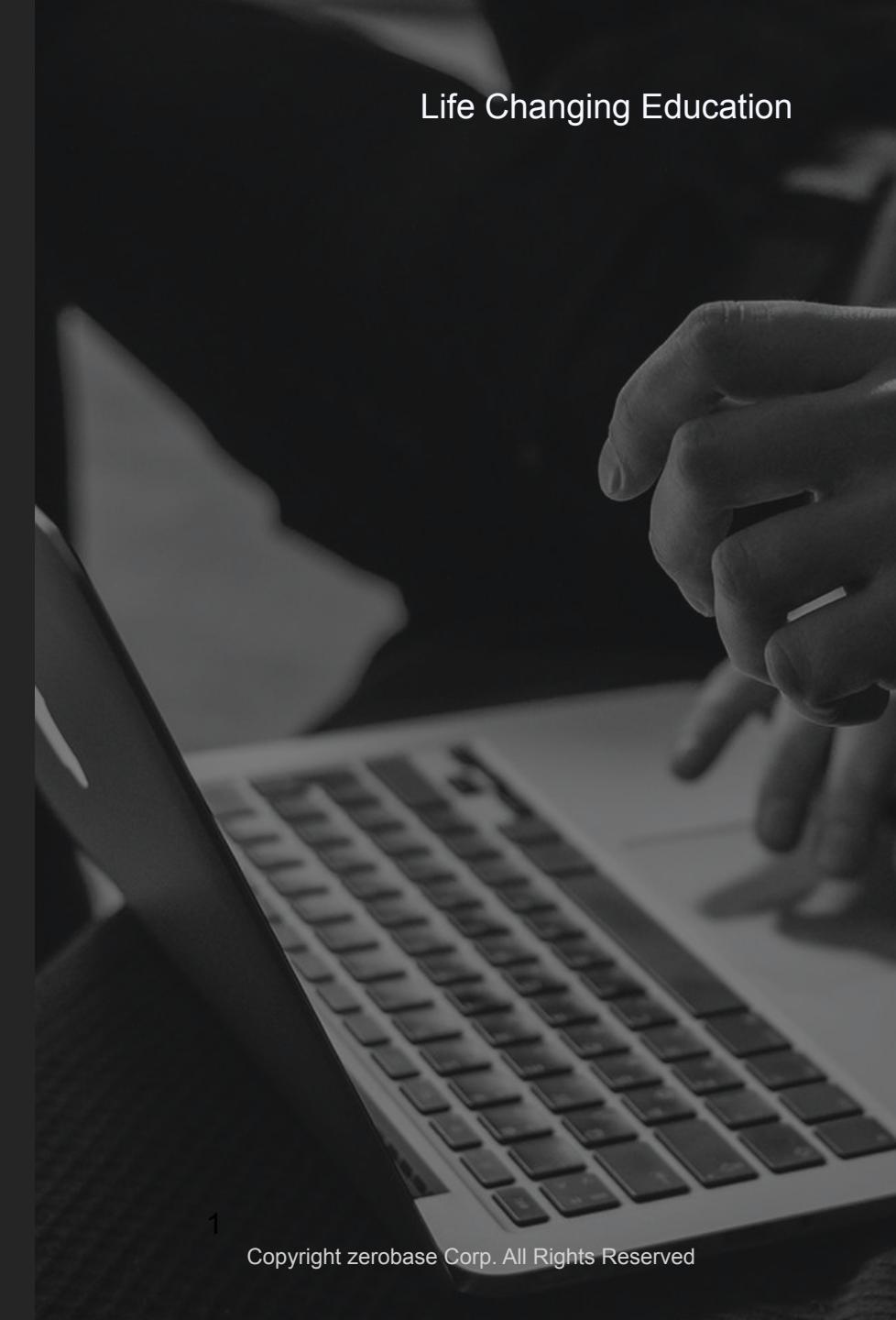
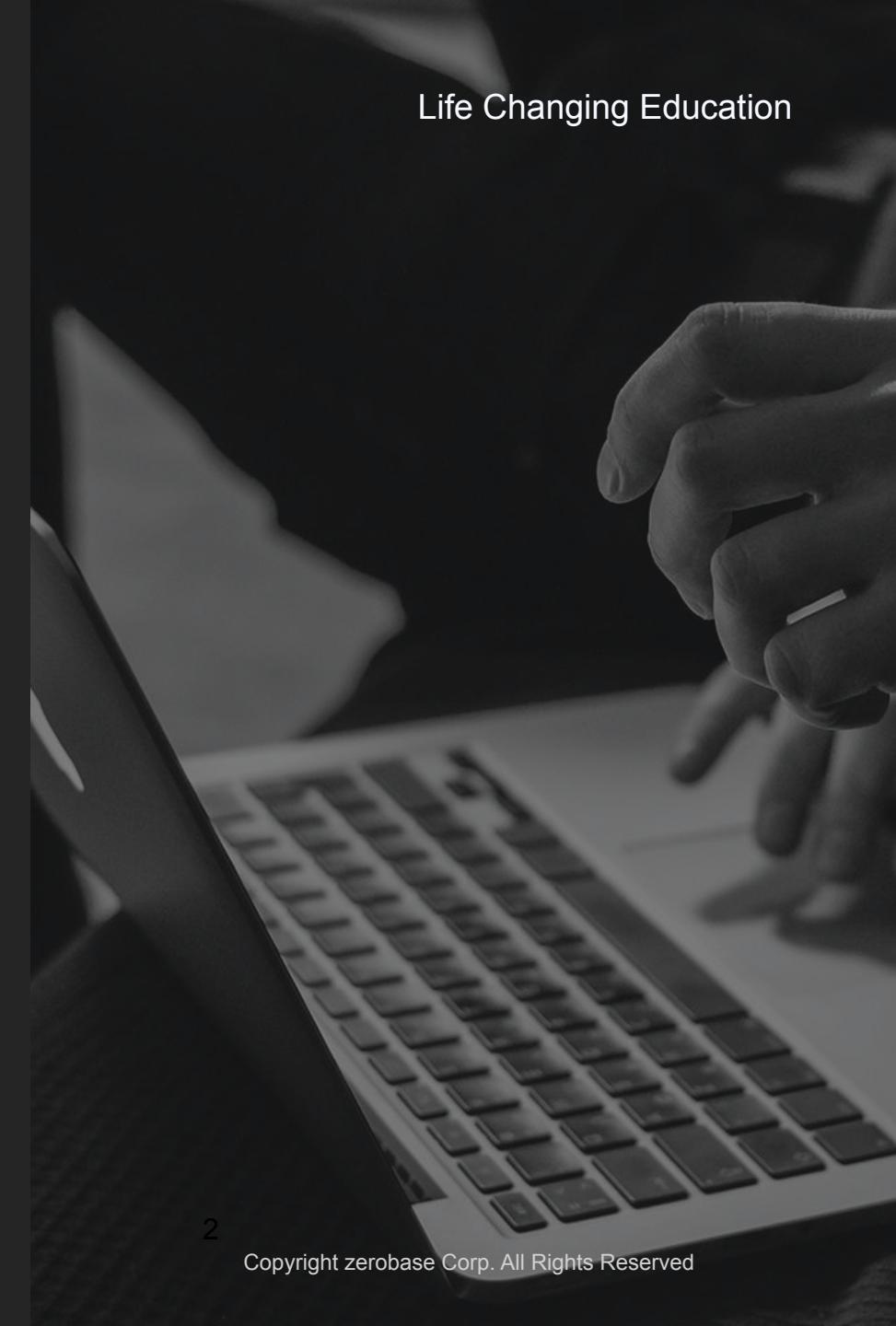


Project 01.

Analysis Seoul CCTV



프로젝트 개요



서울시 CCTV 분석

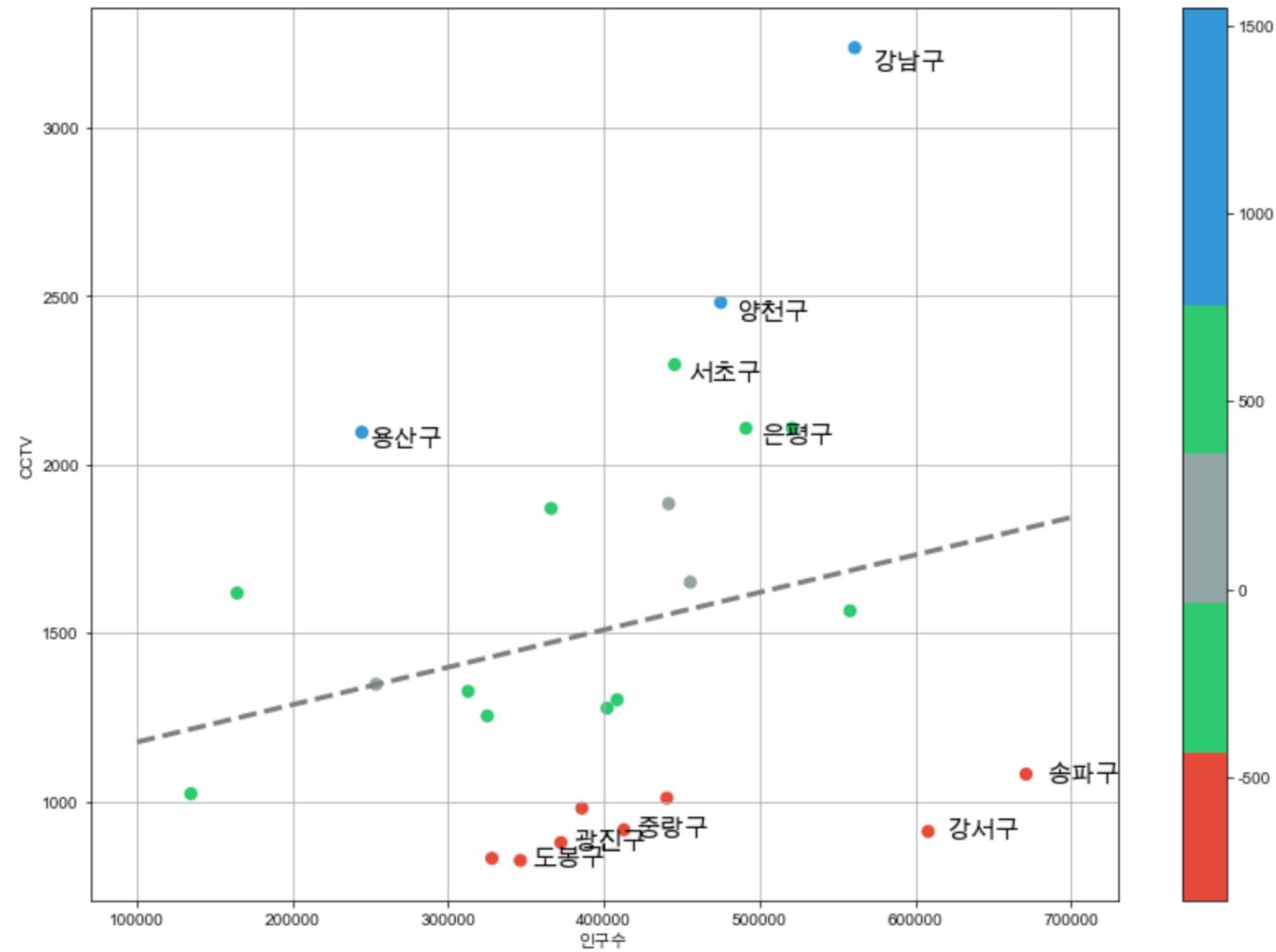
사실은 왕초보용 프로젝트^^

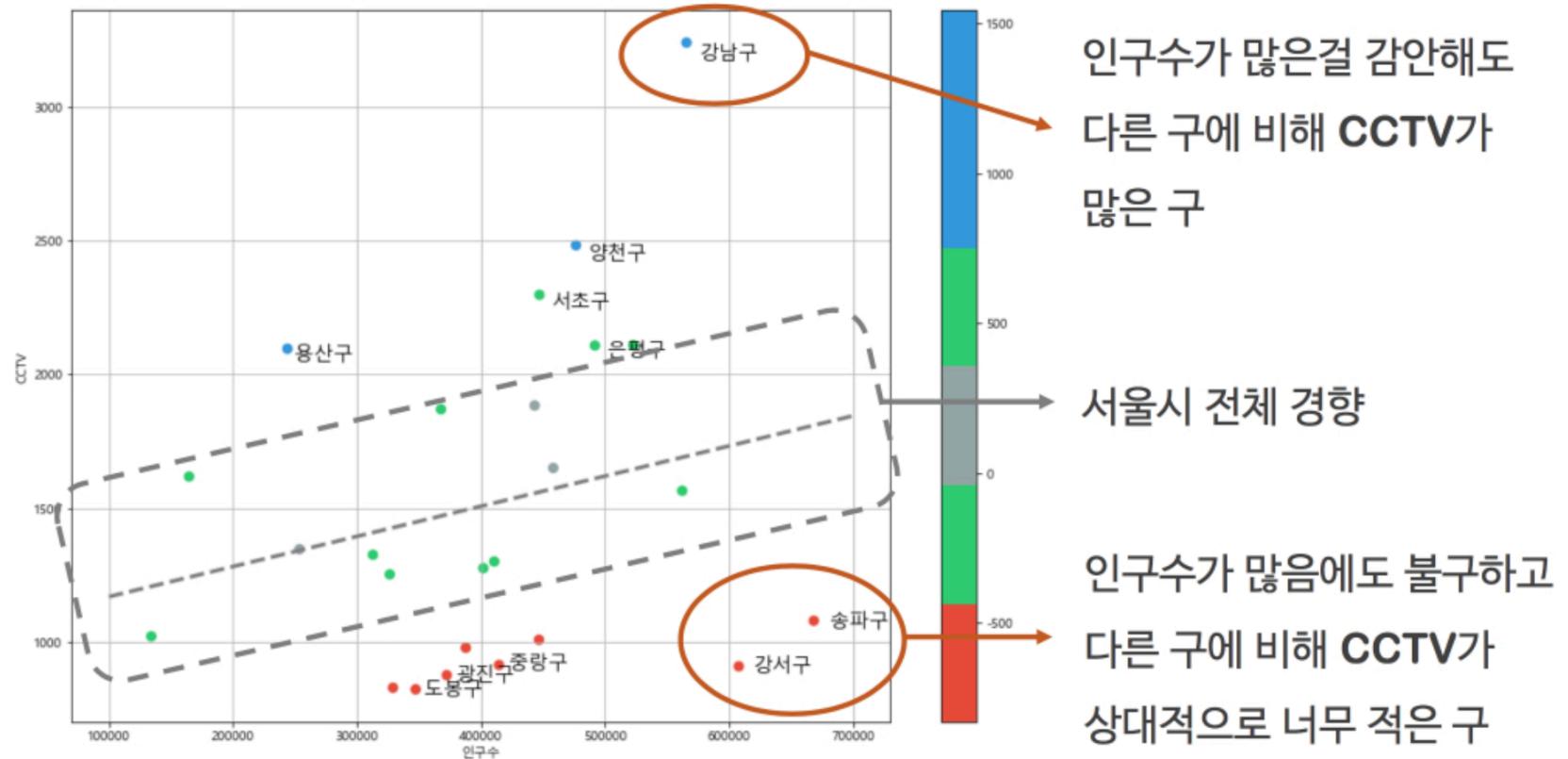
난데없이
왜 서울시 CCTV를 분석하나요??

현실적으로 우리 처지에..

Python / Pandas / Matplotlib 등을
한번에 공부하기 괜찮아서 그렇습니다.

오늘의 목표





1. 당연히 서울시 구별 **CCTV** 현황 데이터 확보
2. 또 당연히 인구 현황 데이터 확보
3. **CCTV** 데이터와 인구 현황 데이터 합치기
4. 데이터를 정리하고 정렬하기
5. 그래프를 그릴 수 있는 능력
6. 전제적인 경향을 파악할 수 있는 능력
7. 그 경향에서 벗어난 데이터를 강조하는 능력

Python, Pandas

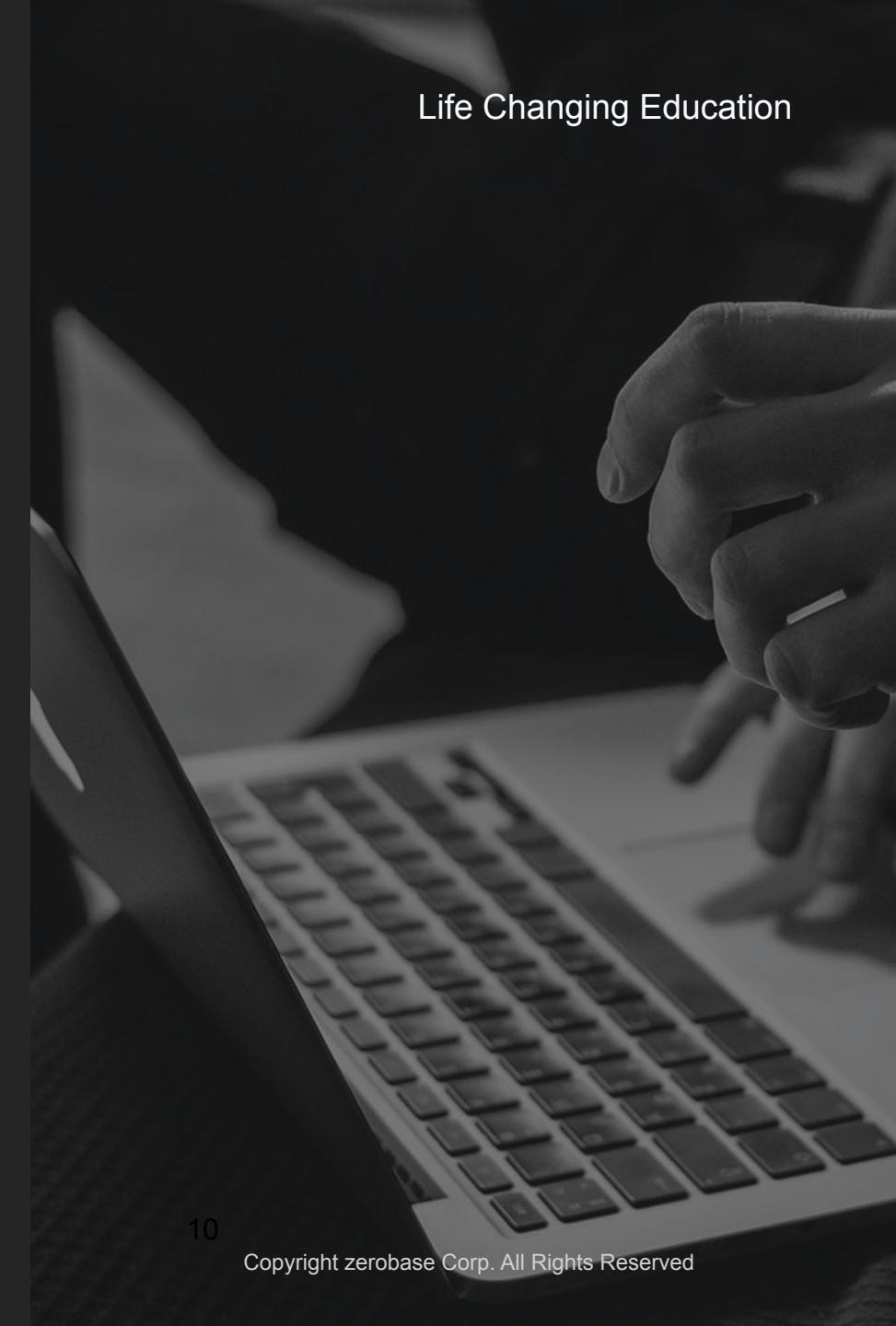
Matplotlib

Regression using Numpy

Insight and Visualization

- 저희 이 후 과정은 전체 흐름을 보는 자료
- 기초 하나하나를 익히는 자료
- 필요한 내용을 필요할 때 적절한 분량을 익히는
기술이 지금 필요할 때입니다.

데이터 얻기



1. 당연히 서울시 구별 CCTV 현황 데이터 확보
2. 또 당연히 인구 현황 데이터 확보

3. CCTV 데이터와 인구 현황 데이터 합치기

4. 데이터를 정리하고 청탁하지
5. 그래프를 그릴 수 있는 능력
6. 전제적인 경향을 파악할 수 있는 능력
7. 그 경향에서 벗어난 데이터를 강조하는 능력
- # 서울시 구별 CCTV 현황과 구별 인구 현황 데이터 얻기



Google

서울시 자치구 연도별 cctv 설치 현황

All News Images 구글신에게 물어보고... Tools

About 167,000 results (0.41 seconds)

<http://data.seoul.go.kr> > dataList > datasetView ▾

서울시 자치구 연도별 CCTV 설치 현황> 데이터셋> 공공데이터 ...

서울특별시 각 자치구의 설치 연도별 CCTV 설치 현황입니다.(2020.12.말 기준)

<https://opengov.seoul.go.kr> > data ▾

서울시 자치구 연도별 CCTV 설치 현황 | 서울시 정보소통광장 ...

서울시 자치구 연도별 CCTV 설치 현황 - 문서정보. 관리번호, D0000028979607, 등록일, 2014-04-02. 분류, 기타. 원본시스템, 공공데이터, 제공부서, 스마트도시정책관 ...

관리번호: D0000028979607 원본시스템: 공공데이터

서울시 자치구 연도별 CCTV 설치 현황

서울특별시 각 자치구의 설치 연도별 CCTV 설치 현황입니다.(2020.12.말 기준)

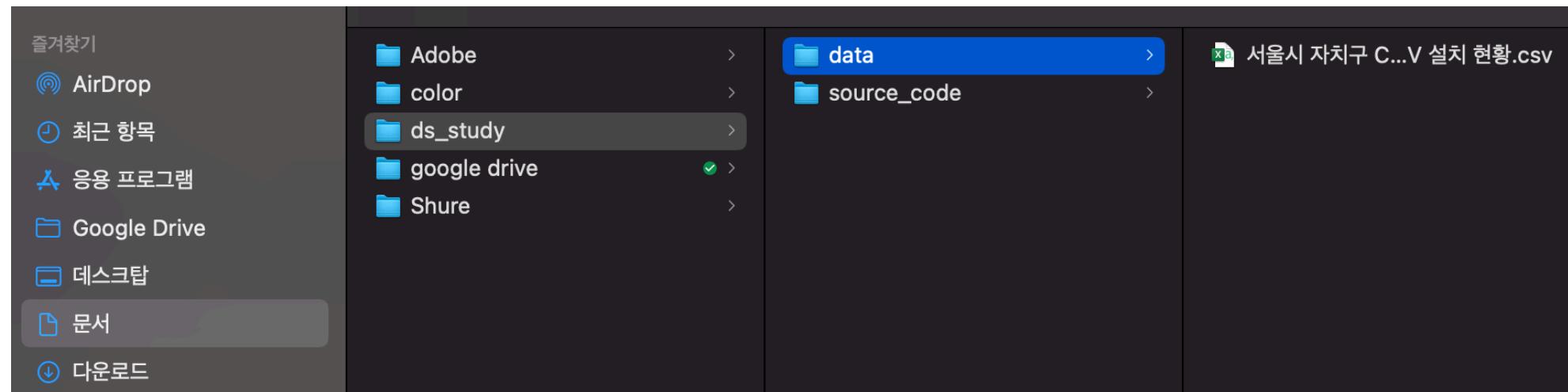
파일내려받기

NO	파일명	내려받기
1	서울시 CCTV 설치운영 현황(자치구)-연도별.csv	

데이터 정보

공개일자	2014.04.02	최신수정일자	2021.04.07
갱신주기	비정기(자료변경시)	분류	안전
원본시스템	서울시 자치구 CCTV	저작권자	자치구
제공기관	서울특별시	제공부서	스마트도시정책관 정보통신보안담당관
담당자	이선혜 (02-2133-2864)		
원본형태	File	제3저작권자	없음
라이센스	저작권자표시(BY): 이용이나 변경 및 2차적 저작물의 작성을 포함한 자유이용을 허락합니다.		
관련 태그	CCTV		

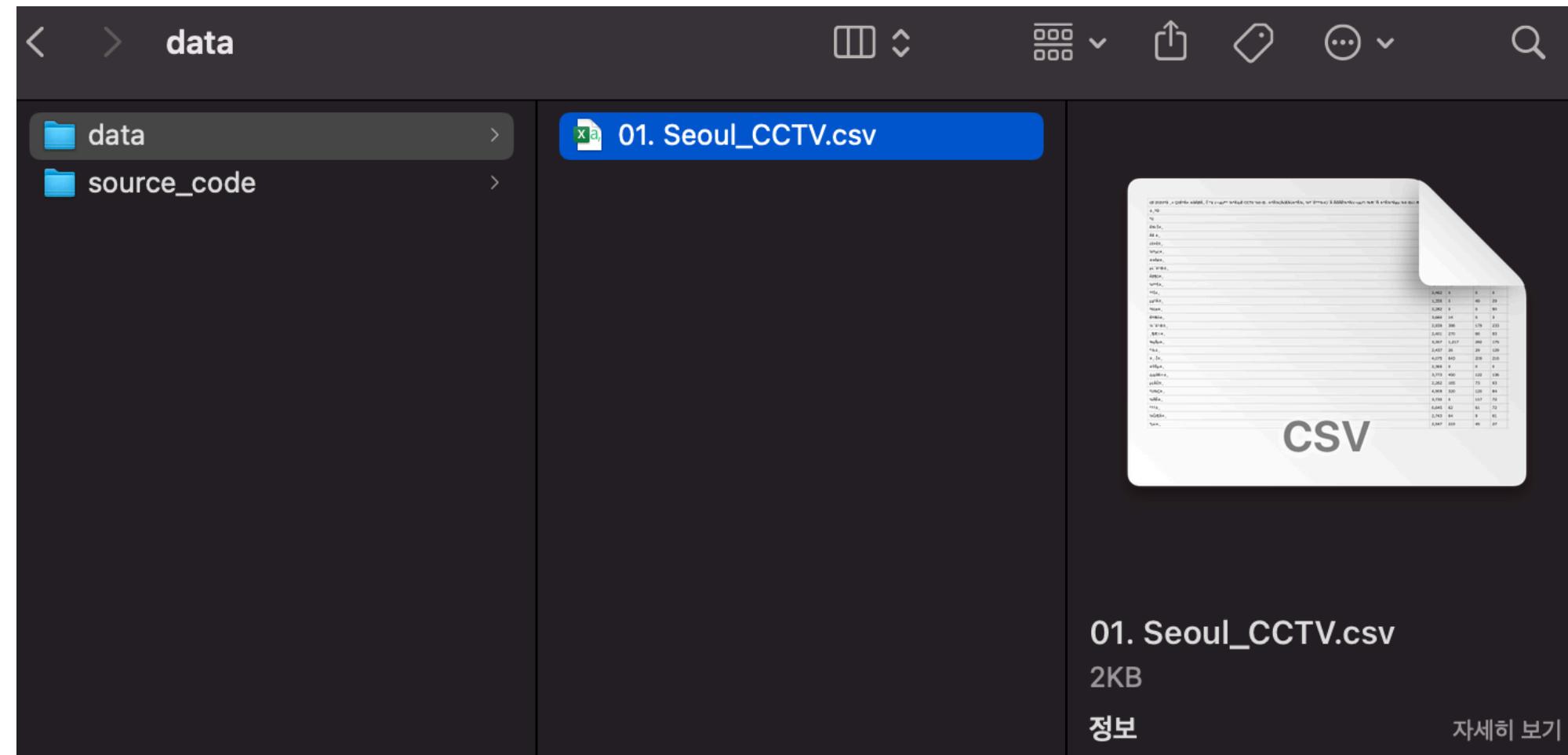
우리가 만든 ds_study => data 폴더에 저장



서울시 CCTV 현황 내용

	서울시 자치구 년도별 CCTV 설치 현황.csv — DeepLearningZeroToAll				
1	["기관명", "소계", "2013년도 이전", "2014년", "2015년", "2016년"]				
2	["강남구", "3238", "1292", "430", "584", "932"]				
3	["강동구", "1010", "379", "99", "155", "377"]				
4	["강북구", "831", "369", "120", "138", "204"]				
5	["강서구", "911", "388", "258", "184", "81"]				
6	["관악구", "2109", "846", "260", "390", "613"]				
7	["광진구", "878", "573", "78", "53", "174"]				
8	["구로구", "1884", "1142", "173", "246", "323"]				
9	["금천구", "1348", "674", "51", "269", "354"]				
10	["노원구", "1566", "542", "57", "451", "516"]				
11	["도봉구", "825", "238", "159", "42", "386"]				
12	["동대문구", "1870", "1070", "23", "198", "579"]				
13	["동작구", "1302", "544", "341", "103", "314"]				
14	["마포구", "980", "314", "118", "169", "379"]				
15	["서대문구", "1254", "844", "50", "68", "292"]				
16	["서초구", "2297", "1406", "157", "336", "398"]				
17	["성동구", "1327", "730", "91", "241", "265"]				
18	["성북구", "1651", "1009", "78", "360", "204"]				
19	["송파구", "1081", "529", "21", "68", "463"]				
20	["양천구", "2482", "1843", "142", "30", "467"]				
21	["영등포구", "1277", "495", "214", "195", "373"]				
22	["용산구", "2096", "1368", "218", "112", "398"]				
23	["은평구", "2108", "1138", "224", "278", "468"]				
24	["종로구", "1619", "464", "314", "211", "630"]				
25	["중구", "1023", "413", "190", "72", "348"]				
26	["중랑구", "916", "509", "121", "177", "109"]				

파일 이름 변경 01. Seoul_CCTV.csv



* 맥 등의 파일명 시스템 호환을 위해 파일 이름을 영문으로 변경

Copyright zero-base Corp. All Rights Reserved

서울 열린 데이터 광장으로 이동

* 데이터 이용하기 --> 오픈 API 서비스 선택

서울 구별 인구 통계 선택

전체	보건	일반행정	문화/관광	산업/경제	복지	환경	전체 데이터를 확인할 수 있습니다.	
	교통	도시관리	교육	안전	인구/가구	주택/건설		
Sheet	Open API	Chart	Map	File	Link	LOD	제공기관을 선택하세요 ▾	<input type="checkbox"/> 결과내 검색
							<input style="width: 100px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 5px; margin-right: 10px; vertical-align: middle;" type="text" value="검색어"/>	

4,383 건이 검색되었습니다.

전체	제목순	최신순	조회순	15개씩
 통계 SHEET OPEN API CHART [인구/가구] 서울시 주민등록인구 (구별) 통계 	 통계 SHEET OPEN API CHART [인구/가구] 서울시 주민등록인구 (동별) 통계 	 통계 SHEET OPEN API CHART [인구/가구] 서울시 고령자현황 (동별) 통계 		
서울시 통계정보 조회수 : 256276 수정일 : 2017-12-26 서울시 통계정보시스템에서 제공하는 주민등록인구(구별)에 대한 통계정보입니다. 주민등록인구수를 자치구	서울시 통계정보 조회수 : 106856 수정일 : 2017-12-26 서울시 통계정보시스템에서 제공하는 주민등록인구(동별)에 대한 통계정보입니다. 주민등록인구수를 동 단위로	서울시 통계정보 조회수 : 65485 수정일 : 2012-02-01 서울시 통계정보시스템에서 제공하는 고령자현황(동별)에 대한 통계정보입니다. 서울시 전체인구 중 고령인구		

엑셀 파일로 저장

Sheet Chart Open Api

자료분석 **XLS** CSV HWP

언어 한국어 소수점 기간 년 2017년 ~ 2017년 자료검색

단위 : 세대, 명

기간	자치구	세대	인구								
			합계			한국인			등록외국인		
			계	남자	여자	계	남자	여자	계	남자	여자
	합계	4,220,082	10,124,579	4,957,857	5,166,722	9,857,426	4,830,206	5,027,220	267,153	127,651	139,1
	종로구	73,594	164,257	80,094	84,163	154,770	75,967	78,803	9,487	4,127	5,30
	중구	60,412	134,593	66,337	68,256	125,709	62,253	63,456	8,884	4,084	4,80
	용산구	107,666	244,444	119,423	125,021	229,161	110,878	118,283	15,283	8,545	6,71
	성동구	132,902	312,711	154,077	158,634	304,808	150,368	154,440	7,903	3,709	4,19
	광진구	160,798	372,298	180,645	191,653	357,703	174,414	183,289	14,595	6,231	8,3
	동대문구	159,938	366,011	181,185	184,826	350,647	175,324	175,323	15,364	5,861	9,50
	중랑구	179,132	412,780	205,125	207,655	408,226	203,325	204,901	4,554	1,800	2,7
	성북구	187,112	455,407	221,103	234,304	444,055	216,556	227,499	11,352	4,547	6,8
	강북구	142,533	328,002	160,252	167,750	324,479	158,918	165,561	3,523	1,334	2,11
	도봉구	137,378	346,234	169,553	176,681	344,166	168,759	175,407	2,068	794	1,2
	노원구	217,619	558,075	271,025	287,050	554,403	269,508	284,895	3,672	1,517	2,1
	은평구	202,839	491,202	238,223	252,979	486,794	236,353	250,441	4,408	1,870	2,5

서울 구별 인구 현황 내용

A1 ▾ ✎ ✓ fx | 기간

기간	자치구	세대	인구												세대당인구	65세이상고령자		
			합계			한국인			등록외국인									
			계	남자	여자	계	남자	여자	계	남자	여자							
2017	합계	4,220,082	10,124,579	4,957,857	5,166,722	9,857,426	4,830,206	5,027,220	267,153	127,651	139,502	2,34	1,365,126					
	종로구	73,594	164,257	80,094	84,163	154,770	75,967	78,803	9,487	4,127	5,360	2.1	26,182					
	중구	60,412	134,593	66,337	68,256	125,709	62,253	63,456	8,884	4,084	4,800	2.08	21,384					
	용산구	107,666	244,444	119,423	125,021	229,161	110,878	118,283	15,283	8,545	6,738	2.13	36,882					
	성동구	132,902	312,711	154,077	158,634	304,808	150,368	154,440	7,903	3,709	4,194	2.29	41,273					
	광진구	160,798	372,298	180,645	191,653	357,703	174,414	183,289	14,595	6,231	8,364	2.22	43,953					
	동대문구	159,938	366,011	181,185	184,826	350,647	175,324	175,323	15,364	5,861	9,503	2.19	55,718					
	중랑구	179,132	412,780	205,125	207,655	408,226	203,325	204,901	4,554	1,800	2,754	2.28	59,262					
	성북구	187,112	455,407	221,103	234,304	444,055	216,556	227,499	11,352	4,547	6,805	2.37	66,251					
	강북구	142,533	328,002	160,252	167,750	324,479	158,918	165,561	3,523	1,334	2,189	2.28	56,530					
	도봉구	137,378	346,234	169,553	176,681	344,166	168,759	175,407	2,068	794	1,274	2.51	53,488					
	노원구	217,619	558,075	271,025	287,050	554,403	269,508	284,895	3,672	1,517	2,155	2.55	74,243					
	은평구	202,839	491,202	238,223	252,979	486,794	236,353	250,441	4,408	1,870	2,538	2.4	74,559					
	서대문구	137,266	325,028	155,250	169,778	312,800	151,144	161,656	12,228	4,106	8,122	2.28	49,266					
	마포구	169,408	385,783	183,248	202,535	374,915	178,939	195,976	10,868	4,309	6,559	2.21	49,615					
	양천구	176,649	475,018	234,279	240,739	471,154	232,518	238,636	3,864	1,761	2,103	2.67	55,234					
	강서구	254,257	608,255	296,175	312,080	601,691	292,989	308,702	6,564	3,186	3,378	2.37	76,032					
	구로구	171,570	441,559	221,386	220,173	410,742	204,003	206,739	30,817	17,383	13,434	2.39	58,794					
	금천구	106,066	253,491	129,775	123,716	235,154	119,497	115,657	18,337	10,278	8,059	2.22	34,170					
	영등포구	167,355	402,024	202,400	199,624	368,550	183,975	184,575	33,474	18,425	15,049	2.2	53,981					
	동작구	172,995	408,493	198,827	209,666	396,217	193,255	202,962	12,276	5,572	6,704	2.29	57,255					
	관악구	255,352	520,929	262,006	258,923	503,297	253,661	249,636	17,632	8,345	9,287	1.97	70,046					
	서초구	173,594	445,401	213,422	231,979	441,102	211,236	229,866	4,299	2,186	2,113	2.54	53,205					
	강남구	231,612	561,052	268,941	292,111	556,164	266,442	289,722	4,888	2,499	2,389	2.4	65,060					
	송파구	264,628	671,173	326,407	344,766	664,496	323,147	341,349	6,677	3,260	3,417	2.51	76,582					
	강동구	177,407	440,359	218,699	221,660	436,223	216,777	219,446	4,136	1,922	2,214	2.46	56,161					

파일 이름 변경 및 복사

The screenshot shows a file manager interface. At the top, there's a toolbar with icons for back, forward, search, and others. Below the toolbar, there are two file entries:

- 01_Seoul_CCTV.csv
- 01.Seoul_Population.xls

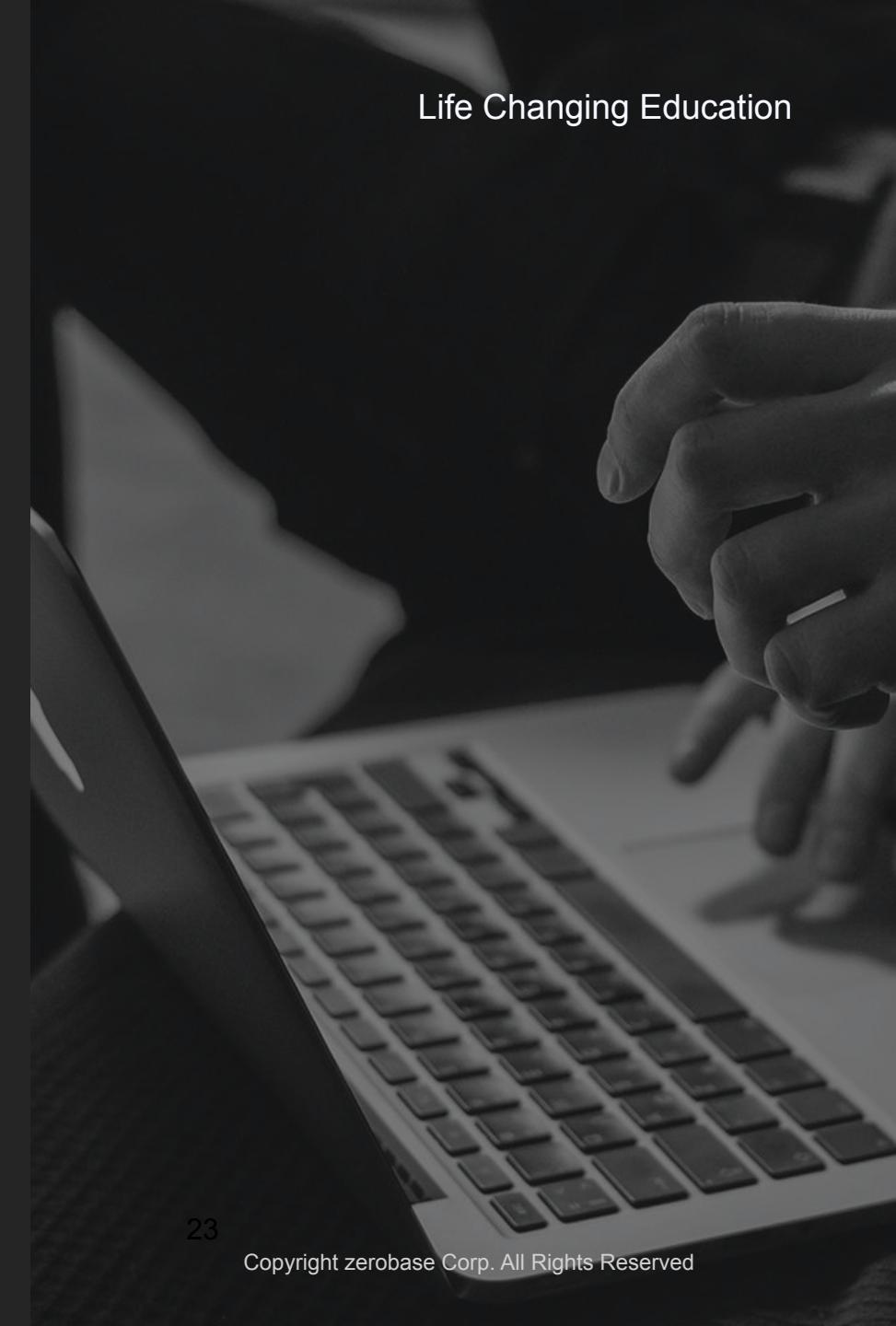
The second item, "01.Seoul_Population.xls", is highlighted with a red box. Above the file list, there's a folder icon labeled "data" which is also highlighted with a red box. To the right of the file list, there's a preview window showing a table of data with columns labeled "기준", "개체구", and "자녀". The preview window has a red border around its content area. Below the preview window, the file name "01.Seoul_Population.xls" is displayed again.

기준	개체구	자녀
2017	강남구	4,139,063
	중랑구	73,994
	성동구	68,412
	성북구	107,000
	마포구	132,962
	용산구	168,798
	광진구	199,638
	성동구	179,152
	성북구	187,112
	마포구	143,513
	용산구	197,318
	성동구	207,849
	성북구	267,679
	마포구	174,726
	용산구	176,761
	성동구	176,761
	성북구	294,397
	마포구	267,0
	용산구	258,276
성동구	296,179	

01.Seoul_Population.xls

이 데이터들은 시간이 지나면 바뀔 수도 있습니다. 그래서 수업때는 데이터를 얻었던 경로만 소개하고 실제로는 데이터를 배포합니다.
배포한 데이터 파일을 확인해 주세요

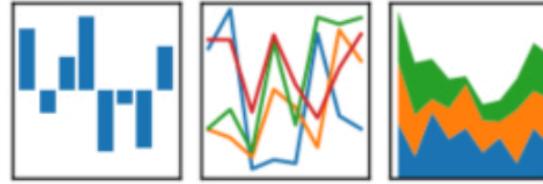
데이터 읽기



Pandas로 CSV, 엑셀 파일 읽기

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



[overview](#) // [get pandas](#) // [documentation](#) // [community](#) // [talks](#) // [donate](#)

pandas: powerful Python data analysis toolkit

- Python에서 R 만큼의 강력한 데이터 핸들링 성능을 제공하는 모듈
- 단일 프로세스에서는 최대 효율
- 코딩 가능하고 응용 가능한 엑셀로 받아들여도 됨
- 누군가는 **스테로이드를 맞은 엑셀**로 표현함

```
In [1]: import pandas as pd
```

- 원하는 모듈이 설치되어 있다면
- import 명령을 통해 사용하겠다고 선언한다

- **import MODULE : MODULE을 사용하겠다**
 - 사용 : MODULE.function
- **import MODULE as md**
 - MODULE을 사용할 건데, 앞으로는 md라는 이름으로 부르겠다
 - 사용 : md.function
- **from MODULE import function**
 - MODULE에 포함된 function이라는 함수만 사용하겠다
 - 사용 : function

```
In [2]: CCTV_Seoul = pd.read_csv("../data/01. Seoul_CCTV.csv", encoding="utf-8")
CCTV_Seoul.head()
```

Out[2]:

	기관명	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	3238	1292	430	584	932
1	강동구	1010	379	99	155	377
2	강북구	831	369	120	138	204
3	강서구	911	388	258	184	81
4	관악구	2109	846	260	390	613

- 통상 csv는 띄어쓰기로 구분되니 그냥 read_csv 명령으로 읽기만 해도 된다
- 긴 파일명을 끝까지 입력하지 말고 적당한 곳에서 TAB 키를 눌러보자
- 한글은 encoding 설정이 필수

Column Name

	기관명	소계	2013년도 이전	2014년	2015년	2016년
Index	강남구	3238	1292	430	584	932
0	강동구	1010	379	99	155	377
1	강북구	831	369	120	138	204
2	강서구	911	388	258	184	81
3	관악구	2109	846	260	390	613
4						

Values

Column

```
In [3]: CCTV_Seoul.columns
```

```
Out[3]: Index(['기관명', '소계', '2013년도 이전', '2014년', '2015년', '2016년'], dtype='object')
```

```
In [4]: CCTV_Seoul.columns[0]
```

```
Out[4]: '기관명'
```

- column의 이름을 조회할 수 있다

```
In [5]: CCTV_Seoul.rename(columns={CCTV_Seoul.columns[0]: "구별"}, inplace=True)  
CCTV_Seoul.head()
```

Out[5]:

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	3238	1292	430	584	932
1	강동구	1010	379	99	155	377
2	강북구	831	369	120	138	204
3	강서구	911	388	258	184	81
4	관악구	2109	846	260	390	613

- 컬럼 이름을 바꾸고 싶다면 ~

```
In [7]: pop_Seoul = pd.read_excel("../data/01. Seoul_Population.xls")
pop_Seoul.head()
```

Out[7]:

	기간	자치구	세대	인구	인구.1	인구.2	인구.3	인구.4	인구.5	인구.6	인구.7	인구.8	세대당인구	65세이상고령자
0	기간	자치구	세대	합계	합계	합계	한국인	한국인	한국인	등록외국인	등록외국인	등록외국인	세대당인구	65세이상고령자
1	기간	자치구	세대	계	남자	여자	계	남자	여자	계	남자	여자	세대당인구	65세이상고령자
2	2017	합계	4220082	10124579	4957857	5166722	9857426	4830206	5027220	267153	127651	139502	2.34	1365126
3	2017	종로구	73594	164257	80094	84163	154770	75967	78803	9487	4127	5360	2.1	26182
4	2017	중구	60412	134593	66337	68256	125709	62253	63456	8884	4084	4800	2.08	21384

- 이제 아까 받은 엑셀 파일도 열어보자
- 그런데…?

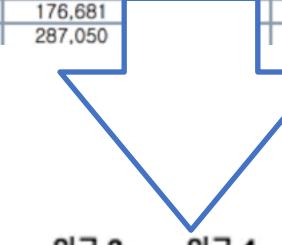
	기간	자치구	세대	인구	인구.1	인구.2	인구.3	인구.4	인구.5	인구.6	인구.7	인구.8	세대당인구	65세이상고령자
0	기간	자치구	세대	합계	합계	합계	한국인	한국인	한국인	등록외국인	등록외국인	등록외국인	세대당인구	65세이상고령자
1	기간	자치구	세대	계	남자	여자	계	남자	여자	계	남자	여자	세대당인구	65세이상고령자
2	2017	합계	4220082	10124579	4957857	5166722	9857426	4830206	5027220	267153	127651	139502	2.34	1365126
3	2017	종로구	73594	164257	80094	84163	154770	75967	78803	9487	4127	5360	2.1	26182
4	2017	중구	60412	134593	66337	68256	125709	62253	63456	8884	4084	4800	2.08	21384

- 앤 또 왜 이모양이지?

A1																
1	기간	자치구	세대	인구									세대당인구	65세이상고령자		
				합계			한국인			등록외국인						
				계	남자	여자	계	남자	여자	계	남자	여자				
2	2017	합계	4,220,082	10,124,579	4,957,857	5,166,722	9,857,426	4,830,206	5,027,220	267,153	127,651	139,502	2.34	1,365,126		
3	2017	종로구	73,594	164,257	80,094	84,163	154,770	75,967	78,803	9,487	4,127	5,360	2.1	26,182		
4	2017	중구	60,412	134,593	66,337	68,256	125,709	62,253	63,456	8,884	4,084	4,800	2.08	21,384		
5	2017	용산구	107,666	244,444	119,423	125,021	229,161	110,878	118,283	15,283	8,545	6,738	2.13	36,882		
6	2017	성동구	132,902	312,711	154,077	158,634	304,808	150,368	154,440	7,903	3,709	4,194	2.29	41,273		
7	2017	광진구	160,798	372,298	180,645	191,653	357,703	174,414	183,289	14,595	6,231	8,364	2.22	43,953		
8	2017	동대문구	159,938	366,011	181,185	184,826	350,647	175,324	175,323	15,364	5,861	9,503	2.19	55,718		
9	2017	종로구	179,132	412,780	205,125	207,655	408,226	203,325	204,901	4,554	1,800	2,754	2.28	59,262		
10	2017	성북구	187,112	455,407	221,103	234,304	444,055	216,556	227,499	11,352	4,547	6,805	2.37	66,251		
11	2017	강북구	142,533	328,002	160,252	167,750	324,479	158,918	165,561	3,523	1,334	2,189	2.28	56,530		
12	2017	도봉구	137,378	346,234	169,553	176,681	344,166	168,759	175,407	2,068	794	1,274	2.51	53,488		
13	2017	노원구	217,619	558,075	271,025	287,050	554,403	269,508	284,895	3,672	1,517	2,155	2.55	74,243		
14	2017	은평구	202,839	491,202	238,223	252,979	486,794	236,353	250,441	4,408	1,870	2,538	2.4	74,559		
15	2017	서대문구	137,266	325,028	155,250	169,778	312,800	151,144	161,656	12,228	4,106	8,122	2.28	49,266		
16	2017	마포구	169,408	385,783	183,248	202,535	374,915	178,939	195,976	10,868	4,309	6,559	2.21	49,615		
17	2017	양천구	176,649	475,018	234,279	240,739	471,154	232,518	238,636	3,864	1,761	2,103	2.67	55,234		
18	2017	강서구	254,257	608,255	296,175	312,080	601,691	292,989	308,702	6,564	3,186	3,378	2.37	76,032		
19	2017	구로구	171,570	441,559	221,386	220,173	410,742	204,003	206,739	30,817	17,383	13,434	2.39	58,794		
20	2017	금천구	106,066	253,491	129,775	123,716	235,154	119,497	115,657	18,337	10,278	8,059	2.22	34,170		
21	2017	영등포구	167,355	402,024	202,400	199,624	368,550	183,975	184,575	33,474	18,425	15,049	2.2	53,981		
22	2017	동작구	172,995	408,493	198,827	209,666	396,217	193,255	202,962	12,276	5,572	6,704	2.29	57,255		
23	2017	관악구	255,352	520,929	262,006	258,923	503,297	253,661	249,636	17,632	8,345	9,287	1.97	70,046		
24	2017	서초구	173,594	445,401	213,422	231,979	441,102	211,236	229,866	4,299	2,186	2,113	2.54	53,205		
25	2017	강남구	231,612	561,052	268,941	292,111	556,164	266,442	289,722	4,888	2,499	2,389	2.4	65,060		
26	2017	송파구	264,628	671,173	326,407	344,766	664,496	323,147	341,349	6,677	3,260	3,417	2.51	76,582		
27	2017	강동구	177,407	440,359	218,699	221,660	436,223	216,777	219,446	4,136	1,922	2,214	2.46	56,161		
28	2017															
29	2017															

- 엑셀 원본을 확인하니 애초 그렇게 저장된 아이였다

기간	자치구	세대	인구												세대당인구	
			합계			한국인			등록외국인							
			계	남자	여자	계	남자	여자	계	남자	여자					
4	합계	4,220,082	10,124,579	4,957,857	5,166,722	9,857,426	4,830,206	5,027,220	267,153	127,651	139,502	2.34	1	65A		
5	종로구	73,594	164,257	80,094	84,163	154,770	75,967	78,803	9,487	4,127	5,360	2.1				
6	중구	60,412	134,593	66,337	68,256	125,709	62,253	63,456	8,884	4,084	4,800	2.08				
7	용산구	107,666	244,444	119,423	125,021	229,161	110,878	118,283	15,283	8,545	6,738	2.13				
8	성동구	132,902	312,711	154,077	158,634	304,808	150,368	154,440	7,903	3,709	4,194	2.29				
9	광진구	160,798	372,298	180,645	191,653	357,703	174,414	183,289	14,595	6,231	8,364	2.22				
10	동대문구	159,938	366,011	181,185	184,826	350,647	175,324	175,323	15,364	5,861	9,503	2.19				
11	중랑구	179,132	412,780	205,125	207,655	408,226	203,325	204,901	4,554	1,800	2,754	2.28				
12	성북구	187,112	455,407	221,103	234,304	444,055	216,556	227,499	11,352	4,547	6,805	2.37				
13	강북구	142,533	328,002	160,252	167,750	324,479	158,918	165,561	3,523	1,334	2,189	2.28				
14	도봉구	137,378	346,234	169,553	176,681	316,759	168,759	175,407	2,068	794	1,274	2.51				
15	노원구	217,619	558,075	271,025	287,050	269,508	284,895	3,672	1,517	2,155	2.55					



기간	자치구	세대	인구	인구.1	인구.2	인구.3	인구.4	인구.5	인구.6	인구.7	인구.8	세대당인구	65세이상고령자	
0	기간	자치구	세대	합계	합계	한국인	한국인	한국인	등록외국인	등록외국인	등록외국인	세대당인구	65세이상고령자	
1	기간	자치구	세대	계	남자	여자	계	남자	여자	계	남자	여자	세대당인구	65세이상고령자
2	2017	합계	4220082	10124579	4957857	5166722	9857426	4830206	5027220	267153	127651	139502	2.34	1365126
3	2017	종로구	73594	164257	80094	84163	154770	75967	78803	9487	4127	5360	2.1	26182
4	2017	중구	60412	134593	66337	68256	125709	62253	63456	8884	4084	4800	2.08	21384

```
In [8]: pop_Seoul = pd.read_excel(
    "../data/01. Seoul_Population.xls", header=2, usecols="B, D, G, J, N"
)
pop_Seoul.head()
```

Out[8] :

	자치구	계	계.1	계.2	65세이상고령자
0	합계	10124579	9857426	267153	1365126
1	종로구	164257	154770	9487	26182
2	중구	134593	125709	8884	21384
3	용산구	244444	229161	15283	36882
4	성동구	312711	304808	7903	41273

- 엑셀 설정
- 자료를 읽기 시작할 행(header)를 지정
- 읽어올 엑셀의 컬럼을 지정(usecols)

```
In [9]: pop_Seoul.rename(
    columns={
        pop_Seoul.columns[0]: "구별",
        pop_Seoul.columns[1]: "인구수",
        pop_Seoul.columns[2]: "한국인",
        pop_Seoul.columns[3]: "외국인",
        pop_Seoul.columns[4]: "고령자",
    },
    inplace=True,
)
pop_Seoul.head()
```

Out [9] :

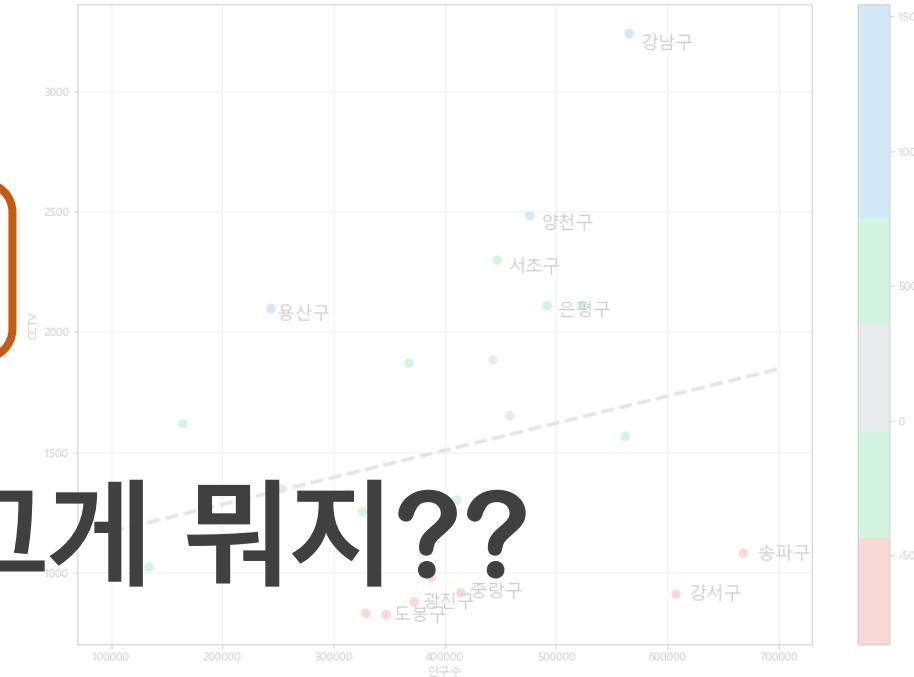
	구별	인구수	한국인	외국인	고령자
0	합계	10124579	9857426	267153	1365126
1	종로구	164257	154770	9487	26182
2	중구	134593	125709	8884	21384
3	용산구	244444	229161	15283	36882
4	성동구	312711	304808	7903	41273

- 컬럼 이름을 바꾸자

Pandas 기초

1. 당연히 서울시 구별 CCTV 현황 데이터 확보
2. 또 당연히 인구 현황 데이터 확보
3. **CCTV 데이터와 인구 현황 데이터 합치기**
4. 데이터를 정리하고 정렬하기
5. 그래프를 그릴 수 있는 능력
6. 전제적인 경험을 파악할 수 있는 능력
7. 그 경향에서 벗어난 데이터를 강조하는 능력

Pandas?? 그게 뭐지??



```
In [10]: import pandas as pd  
         import numpy as np
```

- pandas는 통상 pd로 import 하고
- 수치해석적 함수가 많은 numpy는 통상 np로 import 한다

```
In [11]: s = pd.Series([1, 3, 5, np.nan, 6, 8])
```

```
s
```

```
Out[11]: 0    1.0
          1    3.0
          2    5.0
          3    NaN
          4    6.0
          5    8.0
          dtype: float64
```

- Pandas의 데이터형을 구성하는 기본은 **Series**이다

```
In [12]: dates = pd.date_range("20130101", periods=6)  
dates
```

```
Out[12]: DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',  
                       '2013-01-05', '2013-01-06'],  
                      dtype='datetime64[ns]', freq='D')
```

- 날짜(시간)를 이용할 수 있다

```
In [13]: df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=["A", "B", "C", "D"])
df
```

Out[13]:

	A	B	C	D	Column Name
Index	2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626	
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355	
2013-01-04	0.180506	-1.581461	0.334148	0.287644	
2013-01-05	0.271466	-0.837816	1.279099	1.088684	
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077	

Values

- Pandas에서 가장 많이 사용되는 데이터형은 DataFrame이다
- index와 columns를 지정하면 된다

```
In [14]: df.head()
```

Out[14]:

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684

- 앞 부분 5개의 데이터 확인

```
In [15]: df.index
```

```
Out[15]: DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
                           '2013-01-05', '2013-01-06'],
                          dtype='datetime64[ns]', freq='D')
```

- DataFrame의 index 확인

```
In [15]: df.columns
```

```
Out[15]: Index(['A', 'B', 'C', 'D'], dtype='object')
```

- DataFrame의 컬럼 확인

```
In [17]: df.values
```

```
Out[17]: array([[ 0.58792475,  0.71820561, -1.92847049,  0.77137575],
   [ 0.60413004,  0.07743885,  0.81404559, -0.57662612],
   [-1.3203935 , -1.46053789,  0.18238411, -1.52535479],
   [ 0.18050633, -1.5814612 ,  0.33414779,  0.28764361],
   [ 0.27146551, -0.83781619,  1.27909936,  1.08868438],
   [-0.16754939, -2.23231641,  0.21965778, -0.24207741]])
```

- DataFrame의 value 확인

In [18]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 6 entries, 2013-01-01 to 2013-01-06
Freq: D
Data columns (total 4 columns):
 #   Column   Non-Null Count   Dtype  
--- 
 0   A         6 non-null      float64
 1   B         6 non-null      float64
 2   C         6 non-null      float64
 3   D         6 non-null      float64
dtypes: float64(4)
memory usage: 240.0 bytes
```

- DataFrame의 기본 정보 확인
- 여기서는 각 컬럼의 크기와 데이터형태를 확인하는 경우가 많다

```
In [19]: df.describe()
```

```
Out[19]:
```

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000
mean	0.026014	-0.886081	0.150144	-0.032726
std	0.718938	1.107280	1.102331	0.956491
min	-1.320393	-2.232316	-1.928470	-1.525355
25%	-0.080535	-1.551230	0.191703	-0.492989
50%	0.225986	-1.149177	0.276903	0.022783
75%	0.508810	-0.151375	0.694071	0.650443
max	0.604130	0.718206	1.279099	1.088684

- DataFrame의 통계적 기본 정보를 확인

In [20]: # 데이터 정렬

```
df.sort_values(by="B", ascending=False)
```

Out[20]:

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

- sort_values : 데이터를 정렬

```
In [21]: df["A"]
```

```
Out[21]: 2013-01-01    0.587925
          2013-01-02    0.604130
          2013-01-03   -1.320393
          2013-01-04    0.180506
          2013-01-05    0.271466
          2013-01-06   -0.167549
Freq: D, Name: A, dtype: float64
```

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

- 특정 컬럼만 읽기

```
In [22]: df[0:3]
```

```
Out[22]:
```

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355

- [n:m] : n 부터 m-1 까지
- 그러나 인덱스나 컬럼의 이름으로 slice하는 경우는 끝을 포함함

Pandas Slice - option LOC

```
In [23]: df["20130102":"20130104"]
```

Out[23]:

	A	B	C	D
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

Pandas Slice - option LOC

```
In [24]: df.loc[:, ["A", "B"]]
```

Out[24]:

	A	B
2013-01-01	0.587925	0.718206
2013-01-02	0.604130	0.077439
2013-01-03	-1.320393	-1.460538
2013-01-04	0.180506	-1.581461
2013-01-05	0.271466	-0.837816
2013-01-06	-0.167549	-2.232316

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

- 이름으로도 사용 가능
- Pandas의 보편적인 slice 옵션

Pandas Slice - option LOC

```
In [25]: df.loc["20130102":"20130104", ["A", "B"]]
```

```
Out[25]:
```

	A	B
2013-01-02	0.604130	0.077439
2013-01-03	-1.320393	-1.460538
2013-01-04	0.180506	-1.581461

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

Pandas Slice - option LOC

```
In [26]: df.loc["20130102", ["A", "B"]]
```

```
Out[26]: A    0.604130
          B    0.077439
Name: 2013-01-02 00:00:00, dtype: float64
```

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

Pandas Slice - option iLOC

```
In [27]: df.iloc[3]
```

```
Out[27]: A    0.180506
          B   -1.581461
          C    0.334148
          D    0.287644
Name: 2013-01-04 00:00:00, dtype: float64
```

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

- iloc 옵션을 이용해서 번호로만 접근

Pandas Slice - option iLOC

```
In [28]: df.iloc[3:5, 0:2]
```

Out[28]:

	A	B
2013-01-04	0.180506	-1.581461
2013-01-05	0.271466	-0.837816

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

Pandas Slice - option iLOC

```
In [29]: df.iloc[[1, 2, 4], [0, 2]]
```

Out[29]:

	A	C
2013-01-02	0.604130	0.814046
2013-01-03	-1.320393	0.182384
2013-01-05	0.271466	1.279099

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

Pandas Slice - option iLOC

```
In [30]: df.iloc[:, 1:3]
```

```
Out[30]:
```

	B	C
2013-01-01	0.718206	-1.928470
2013-01-02	0.077439	0.814046
2013-01-03	-1.460538	0.182384
2013-01-04	-1.581461	0.334148
2013-01-05	-0.837816	1.279099
2013-01-06	-2.232316	0.219658

A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470
2013-01-02	0.604130	0.077439	0.814046
2013-01-03	-1.320393	-1.460538	0.182384
2013-01-04	0.180506	-1.581461	0.334148
2013-01-05	0.271466	-0.837816	1.279099
2013-01-06	-0.167549	-2.232316	0.219658

Pandas Slice under condition

In [31]: df

Out[31]:

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

- df[condition]과 같이 사용하는 것이 일반적
- Pandas의 버전에 따라 조금씩 허용되는 문법이 다르다
- 인터넷에서 확보한 소스코드를 돌릴 때는 Pandas의 버전을 확인하는 것이 필요

Pandas Slice under condition

zero-base /

```
In [32]: df[df[ "A" ] > 0]
```

```
Out[32]:
```

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.107549	2.232316	0.219658	-0.242077

Pandas Slice under condition

zero-base /

In [33]: df[df > 0]

Out[33]:

	A	B	C	D
2013-01-01	0.587925	0.718206	NaN	0.771376
2013-01-02	0.604130	0.077439	0.814046	NaN
2013-01-03	NaN	NaN	0.182384	NaN
2013-01-04	0.180506	NaN	0.334148	0.287644
2013-01-05	0.271466	NaN	1.279099	1.088684
2013-01-06	NaN	NaN	0.219658	NaN

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.561461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

Pandas Slice under condition

```
In [34]: df["E"] = ["one", "one", "two", "three", "four", "three"]
df
```

Out[34]:

	A	B	C	D	E
2013-01-01	0.587925	0.718206	-1.928470	0.771376	one
2013-01-02	0.604130	0.077439	0.814046	-0.576626	one
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355	two
2013-01-04	0.180506	-1.581461	0.334148	0.287644	three
2013-01-05	0.271466	-0.837816	1.279099	1.088684	four
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077	three

Pandas Slice under condition

```
In [35]: # 특정 요소가 있는지 확인  
df[ "E" ].isin([ "two" , "four" ])
```

```
Out[35]: 2013-01-01    False  
2013-01-02    False  
2013-01-03     True  
2013-01-04    False  
2013-01-05     True  
2013-01-06    False  
Freq: D, Name: E, dtype: bool
```

	A	B	C	D	E
2013-01-01	0.587925	0.718206	-1.928470	0.771376	one
2013-01-02	0.604130	0.077439	0.814046	-0.576626	one
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355	two
2013-01-04	0.180506	-1.581461	0.334148	0.287644	three
2013-01-05	0.271466	-0.837816	1.279099	1.088684	four
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077	three

Pandas Slice under condition

In [36]: # 특정 요소가 있는 행만 선택
df[df["E"].isin(["two", "four"])]

Out[36]:

	A	B	C	D	E
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355	two
2013-01-05	0.271466	-0.837816	1.279099	1.088684	four

	A	B	C	D	E
2013-01-01	0.587925	0.718206	-1.928470	0.771376	one
2013-01-02	0.604130	0.077439	0.814046	-0.576626	one
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355	two
2013-01-04	0.180506	-1.581461	0.334148	0.287644	three
2013-01-05	0.271466	-0.837816	1.279099	1.088684	four
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077	three

Pandas Column remove

zero-base /

In [37]: df

Out[37]:

	A	B	C	D	E
2013-01-01	0.587925	0.718206	-1.928470	0.771376	one
2013-01-02	0.604130	0.077439	0.814046	-0.576626	one
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355	two
2013-01-04	0.180506	-1.581461	0.334148	0.287644	three
2013-01-05	0.271466	-0.837816	1.279099	1.088684	four
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077	three

특정 컬럼 제거

```
In [38]: del df["E"]  
df
```

Out[38]:

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	0.604130	0.077439	0.814046	-0.576626
2013-01-03	-1.320393	-1.460538	0.182384	-1.525355
2013-01-04	0.180506	-1.581461	0.334148	0.287644
2013-01-05	0.271466	-0.837816	1.279099	1.088684
2013-01-06	-0.167549	-2.232316	0.219658	-0.242077

Pandas apply function

In [39]:

각 컬럼 누적합

df.apply(np.cumsum)

Out[39]:

	A	B	C	D
2013-01-01	0.587925	0.718206	-1.928470	0.771376
2013-01-02	1.192055	0.795644	-1.114425	0.194750
2013-01-03	-0.128339	-0.664893	-0.932041	-1.330605
2013-01-04	0.052168	-2.246355	-0.597893	-1.042962
2013-01-05	0.323633	-3.084171	0.681206	0.045723
2013-01-06	0.156084	-5.316487	0.900864	-0.196355

- 함수를 적용

CCTV 데이터 훑어보기



```
In [40]: CCTV_Seoul.head()
```

```
Out[40]:
```

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	3238	1292	430	584	932
1	강동구	1010	379	99	155	377
2	강북구	831	369	120	138	204
3	강서구	911	388	258	184	81
4	관악구	2109	846	260	390	613

- CCTV 앞 부분 데이터 확인

```
In [40]: CCTV_Seoul.sort_values(by="소계", ascending=True).head(5)
```

Out[40]:

	구별	소계	2013년도 이전	2014년	2015년	2016년
9	도봉구	825	238	159	42	386
2	강북구	831	369	120	138	204
5	광진구	878	573	78	53	174
3	강서구	911	388	258	184	81
24	중랑구	916	509	121	177	109

- 가장 CCTV를 적게 보유한 구

```
In [42]: CCTV_Seoul.sort_values(by="소계", ascending=False).head(5)
```

Out[42]:

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	3238	1292	430	584	932
18	양천구	2482	1843	142	30	467
14	서초구	2297	1406	157	336	398
4	관악구	2109	846	260	390	613
21	은평구	2108	1138	224	278	468

- 가장 CCTV를 많이 보유한 구

```
In [42]: CCTV_Seoul[ "최근증가율" ] = (
    (CCTV_Seoul[ "2016년" ] + CCTV_Seoul[ "2015년" ] + CCTV_Seoul[ "2014년" ])
    / CCTV_Seoul[ "2013년도 이전" ]
    * 100
)

CCTV_Seoul.sort_values(by="최근증가율", ascending=False).head(5)
```

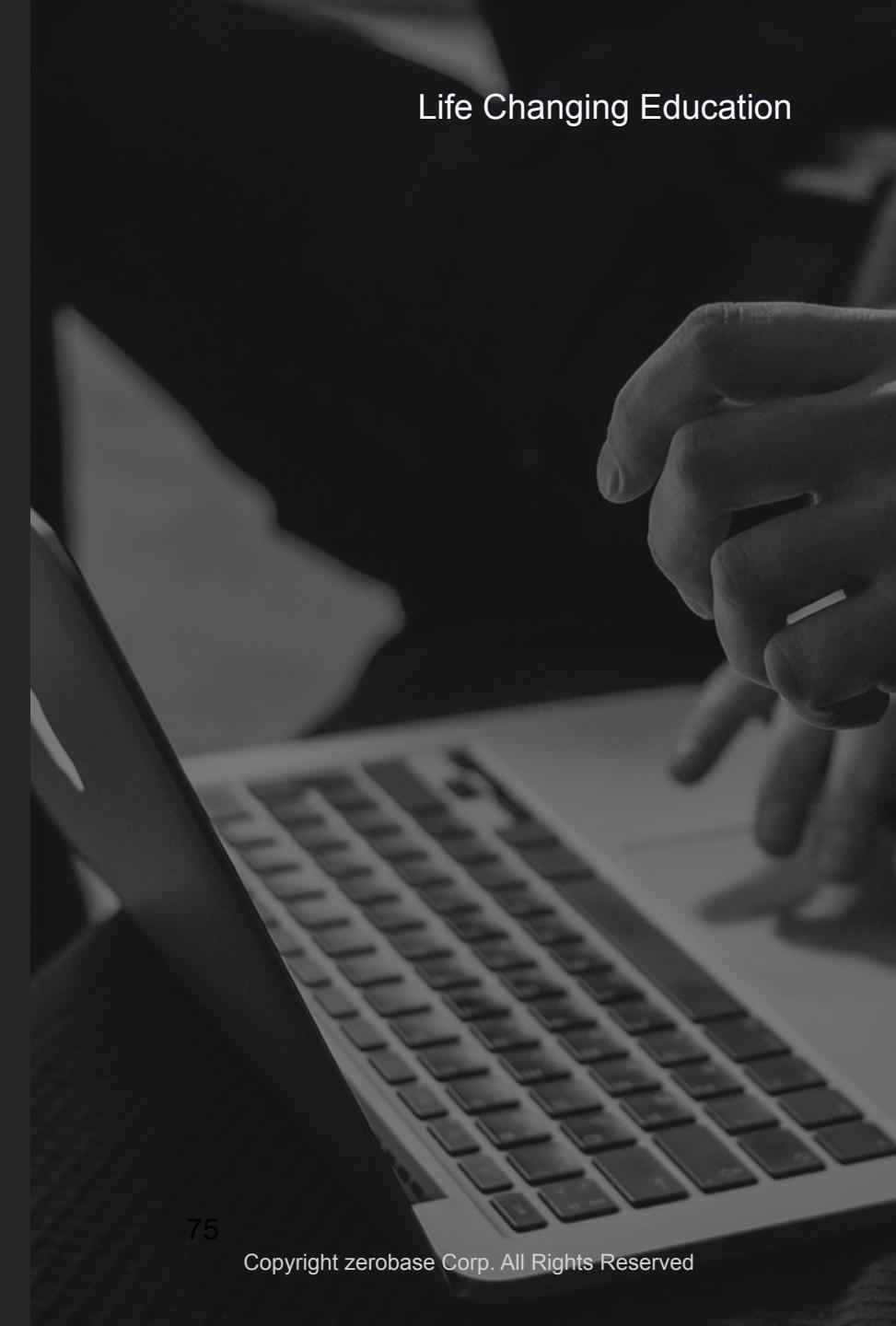
Out[42]:

	구별	소계	2013년도 이전	2014년	2015년	2016년	최근증가율
22	종로구	1619	464	314	211	630	248.922414
9	도봉구	825	238	159	42	386	246.638655
12	마포구	980	314	118	169	379	212.101911
8	노원구	1566	542	57	451	516	188.929889
1	강동구	1010	379	99	155	377	166.490765

- 최근 3년간 그 전 보유한 갯수 대비 CCTV를 많이 설치한 구는

단순히 구별 CCTV 개수로는 한계가 있습니다.
인구 현황과 함께 비교를 해야할 것 같습니다.

인구현황 데이터 훑어보기



In [43]: `pop_Seoul.head()`

Out[43]:

	구별	인구수	한국인	외국인	고령자
0	합계	10124579	9857426	267153	1365126
1	종로구	164257	154770	9487	26182
2	중구	134593	125709	8884	21384
3	용산구	244444	229161	15283	36882
4	성동구	312711	304808	7903	41273

- 서울시 인구 데이터 확인

```
In [44]: pop_Seoul.drop([0], inplace=True)  
pop_Seoul.head()
```

Out[44]:

	구별	인구수	한국인	외국인	고령자
1	종로구	164257	154770	9487	26182
2	중구	134593	125709	8884	21384
3	용산구	244444	229161	15283	36882
4	성동구	312711	304808	7903	41273
5	광진구	372298	357703	14595	43953

- 첫 행(0번)의 합계 데이터는 필요 없다
- 행을 지우는 명령 → drop

```
In [46]: pop_Seoul["구별"].unique()
```

```
Out[46]: array(['종로구', '중구', '용산구', '성동구', '광진구', '동대문구', '중랑구', '성북구', '강북구',
   '도봉구', '노원구', '은평구', '서대문구', '마포구', '양천구', '강서구', '구로구', '금천구',
   '영등포구', '동작구', '관악구', '서초구', '강남구', '송파구', '강동구'], dtype=object)
```

```
In [47]: len(pop_Seoul["구별"].unique())
```

```
Out[47]: 25
```

- unique 조사
- 지금은 데이터양이 작아서 괜찮지만…
- 데이터가 많아지면 unique 조사를 통해 데이터를 초반 검증(눈으로^^) 한다

```
In [47]: pop_Seoul[ "외국인비율" ] = pop_Seoul[ "외국인" ] / pop_Seoul[ "인구수" ] * 100
pop_Seoul[ "고령자비율" ] = pop_Seoul[ "고령자" ] / pop_Seoul[ "인구수" ] * 100
pop_Seoul.head()
```

Out[47]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
1	종로구	164257	154770	9487	26182	5.775705	15.939656
2	중구	134593	125709	8884	21384	6.600640	15.887899
3	용산구	244444	229161	15283	36882	6.252148	15.088118
4	성동구	312711	304808	7903	41273	2.527254	13.198448
5	광진구	372298	357703	14595	43953	3.920247	11.805865

- 외국과 고령자 비율을 만들어 둔다
- 데이터가 행이 25개인데, 딱 한줄로 의도하는 바를 이룬다
- 컬럼 연산이 편하다는 것이 Python의 장점

```
In [48]: pop_Seoul.sort_values(by="인구수", ascending=False).head(5)
```

Out[48]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
24	송파구	671173	664496	6677	76582	0.994825	11.410173
16	강서구	608255	601691	6564	76032	1.079153	12.500021
23	강남구	561052	556164	4888	65060	0.871220	11.596073
11	노원구	558075	554403	3672	74243	0.657976	13.303409
21	관악구	520929	503297	17632	70046	3.384722	13.446362

- 인구수가 많은 구는?

```
In [49]: pop_Seoul.sort_values(by="외국인", ascending=False).head(5)
```

Out[49]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
19	영등포구	402024	368550	33474	53981	8.326369	13.427308
17	구로구	441559	410742	30817	58794	6.979135	13.315095
18	금천구	253491	235154	18337	34170	7.233787	13.479769
21	관악구	520929	503297	17632	70046	3.384722	13.446362
6	동대문구	366011	350647	15364	55718	4.197688	15.223040

- 외국인이 많은 구는?

```
In [50]: pop_Seoul.sort_values(by="외국인비율", ascending=False).head(5)
```

Out[50]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
19	영등포구	402024	368550	33474	53981	8.326369	13.427308
18	금천구	253491	235154	18337	34170	7.233787	13.479769
17	구로구	441559	410742	30817	58794	6.979135	13.315095
2	중구	134593	125709	8884	21384	6.600640	15.887899
3	용산구	244444	229161	15283	36882	6.252148	15.088118

- 외국인 비율이 높은 구는?

```
In [51]: pop_Seoul.sort_values(by="고령자", ascending=False).head(5)
```

Out[51]:

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
24	송파구	671173	664496	6677	76582	0.994825	11.410173
16	강서구	608255	601691	6564	76032	1.079153	12.500021
12	은평구	491202	486794	4408	74559	0.897390	15.178888
11	노원구	558075	554403	3672	74243	0.657976	13.303409
21	관악구	520929	503297	17632	70046	3.384722	13.446362

- 고령자가 많은 구는?

```
In [52]: pop_Seoul.sort_values(by="고령자비율", ascending=False).head(5)
```

```
Out[52]:
```

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
9	강북구	328002	324479	3523	56530	1.074079	17.234651
1	종로구	164257	154770	9487	26182	5.775705	15.939656
2	중구	134593	125709	8884	21384	6.600640	15.887899
10	도봉구	346234	344166	2068	53488	0.597284	15.448512
6	동대문구	366011	350647	15364	55718	4.197688	15.223040

- 고령자 비율이 높은 구는?

CCTV 데이터로 어느 구가 CCTV가 많은지...
인구현황 데이터로 각 구별 순위 확인...
이런걸로는 뭐 얻을게 없다...
두 데이터를 같이 비교하고 관계도 조사해보자...

두 데이터 합치기



```
In [53]: left = pd.DataFrame(  
    {  
        "key": ["K0", "K4", "K2", "K3"],  
        "A": ["A0", "A1", "A2", "A3"],  
        "B": ["B0", "B1", "B2", "B3"],  
    }  
)  
  
right = pd.DataFrame(  
    {  
        "key": ["K0", "K1", "K2", "K3"],  
        "C": ["C0", "C1", "C2", "C3"],  
        "D": ["D0", "D1", "D2", "D3"],  
    }  
)
```

- merge를 이용한 데이터 병합
- Pandas DataFrame 데이터끼리의 병합은 빈번히 발생한다
- 병합 후 데이터가 엉망이 되지 않도록 익히자 ~~

Pandas 데이터 merge를 이용해서 병합하기

zero-base /

In [55]: left

Out[55]:

	key	A	B
0	K0	A0	B0
1	K4	A1	B1
2	K2	A2	B2
3	K3	A3	B3

In [56]: right

Out[56]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

Pandas 데이터 merge를 이용해서 병합하기

```
In [57]: pd.merge(left, right, on="key")
```

Out[57]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
2	K3	A3	B3	C3	D3

```
In [55]: left
```

Out[55]:

key	A	B	
0	K0	A0	B0
1	K4	A1	B1
2	K2	A2	B2
3	K3	A3	B3

```
In [56]: right
```

Out[56]:

key	C	D	
0	K0	D0	
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

- Key 컬럼을 기준으로 병합

Pandas 데이터 merge를 이용해서 병합하기

```
In [58]: pd.merge(left, right, how="left", on="key")
```

Out[58]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K4	A1	B1	NaN	NaN
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

```
In [55]: left
```

Out[55]:

	key	A	B
0	K0	A0	B0
1	K4	A1	B1
2	K2	A2	B2
3	K3	A3	B3

```
In [56]: right
```

Out[56]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

- left에 key를 기준으로 right 병합

Pandas 데이터 merge를 이용해서 병합하기

In [59]: `pd.merge(left, right, how="right", on="key")`

Out[59]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	NaN	NaN	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

In [55]: `left`

Out[55]:

	key	A	B
0	K0	A0	B0
1	K4	A1	B1
2	K2	A2	B2
3	K3	A3	B3

In [56]: `right`

Out[56]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

- right에 key를 기준으로 left 병합

In [60]: `pd.merge(left, right, how="outer", on="key")`

Out[60]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K4	A1	B1	NaN	NaN
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3
4	K1	NaN	NaN	C1	D1

In [55]: left

Out[55]:

	key	A	B
0	K0	A0	B0
1	K4	A1	B1
2	K2	A2	B2
3	K3	A3	B3

In [56]: right

Out[56]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

- 둘 다 손상되지 않도록 key를 기준으로 병합

```
In [61]: pd.merge(left, right, how="inner", on="key")
```

Out[61]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
2	K3	A3	B3	C3	D3

```
In [55]: left
```

Out[55]:

	key	A	B
0	K0	A0	B0
1	K4	A1	B1
2	K2	A2	B2
3	K3	A3	B3

```
In [56]: right
```

Out[56]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

- key 컬럼에서 두 데이터에 공통분모만 병합

```
In [62]: data_result = pd.merge(CCTV_Seoul, pop_Seoul, on="구별")
data_result.head()
```

Out[62]:

	구별	소계	2013년도 이전	2014년	2015년	2016년	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
0	강남구	3238	1292	430	584	932	150.619195	561052	556164	4888	65060	0.871220	11.596073
1	강동구	1010	379	99	155	377	166.490765	440359	436223	4136	56161	0.939234	12.753458
2	강북구	831	369	120	138	204	125.203252	328002	324479	3523	56530	1.074079	17.234651
3	강서구	911	388	258	184	81	134.793814	608255	601691	6564	76032	1.079153	12.500021
4	관악구	2109	846	260	390	613	149.290780	520929	503297	17632	70046	3.384722	13.446362

- 우리 데이터도 합치자

```
In [63]: del data_result["2013년도 이전"]
del data_result["2014년"]
del data_result["2015년"]
del data_result["2016년"]
data_result.head()
```

Out[63]:

	구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
0	강남구	3238	150.619195	561052	556164	4888	65060	0.871220	11.596073
1	강동구	1010	166.490765	440359	436223	4136	56161	0.939234	12.753458
2	강북구	831	125.203252	328002	324479	3523	56530	1.074079	17.234651
3	강서구	911	134.793814	608255	601691	6564	76032	1.079153	12.500021
4	관악구	2109	149.290780	520929	503297	17632	70046	3.384722	13.446362

- 필요 없는 컬럼 제거

```
In [63]: data_result.set_index("구별", inplace=True)
data_result.head()
```

Out[63]:

	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
구별								
강남구	3238	150.619195	561052	556164	4888	65060	0.871220	11.596073
강동구	1010	166.490765	440359	436223	4136	56161	0.939234	12.753458
강북구	831	125.203252	328002	324479	3523	56530	1.074079	17.234651
강서구	911	134.793814	608255	601691	6564	76032	1.079153	12.500021
관악구	2109	149.290780	520929	503297	17632	70046	3.384722	13.446362

- Pandas Index 지정
- 데이터를 정리하는 과정에서 index를 재지정할 때가 있다
- 여기서는 unique한 데이터인 구별로 index를 잡자
- index를 재지정하는 명령은 set_index 이다

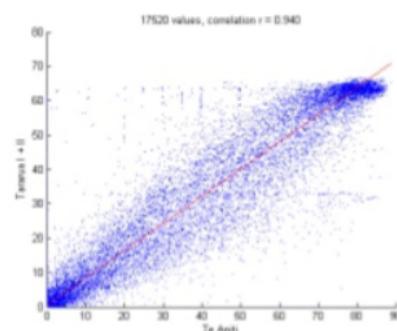
여기서 잠깐...

우리 주제는

인구대비 상대적으로 CCTV가 적은 구를 찾는 것

그런데 인구데이터와 CCTV는 상관관계가 있을까??

상관관계(Correlation)란?

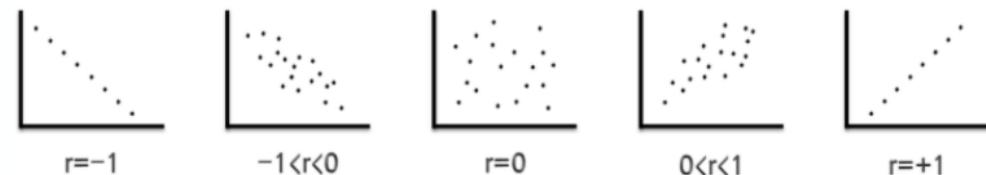


상관관계

두 변량 사이에 한쪽이 증가하면 다른 쪽도 증가(또는 감소)하는 경향이 있을 때, 이 두 변량 사이에는 상관관계가 있다고 함.

단, 상관관계가 있다하여 두 변량이 인과관계인 것은 아님.

0.2 이하	상관관계가 없거나 무시해도 좋은 수준
0.4 이하	약한 상관관계
0.6 이상	강한 상관관계



음의 상관관계가 강하다.

음의 상관관계가 있는 하다.

상관관계가 없다.

양의 상관관계가 있는 하다.

양의 상관관계가 강하다.

In [64]: `data_result.corr()`

Out[64]:

	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
소계	1.000000	-0.264378	0.232555	0.227852	0.030421	0.163905	-0.045956	-0.267841
최근증가율	-0.264378	1.000000	-0.097165	-0.086341	-0.156421	-0.072251	-0.047102	0.190396
인구수	0.232555	-0.097165	1.000000	0.998151	-0.167243	0.936737	-0.601076	-0.637414
한국인	0.227852	-0.086341	0.998151	1.000000	-0.226853	0.936155	-0.645463	-0.628360
외국인	0.030421	-0.156421	-0.167243	-0.226853	1.000000	-0.175318	0.838612	-0.021147
고령자	0.163905	-0.072251	0.936737	0.936155	-0.175318	1.000000	-0.620300	-0.348840
외국인비율	-0.045956	-0.047102	-0.601076	-0.645463	0.838612	-0.620300	1.000000	0.242816
고령자비율	-0.267841	0.190396	-0.637414	-0.628360	-0.021147	-0.348840	0.242816	1.000000

- 데이터의 관계를 찾을 때, 최소한의 근거가 있어야 해당 데이터를 비교하는 의미가 존재
- 상관계수를 조사해서 0.2 이상의 데이터를 비교하는 것은 의미가 있다

	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
소계	1.000000	-0.264378	0.232555	0.227852	0.030421	0.163905	-0.045956	-0.267841
최근증가율	-0.264378	1.000000	-0.097165	-0.086341	-0.156421	-0.072251	-0.047102	0.190396
인구수	0.232555	-0.097165	1.000000	0.998151	-0.167243	0.936737	-0.601076	-0.637414
한국인	0.227852	-0.086341	0.998151	1.000000	-0.226853	0.936155	-0.645463	-0.628360
외국인	0.030421	-0.156421	-0.167243	-0.226853	1.000000	-0.175318	0.838612	-0.021147
고령자	0.163905	-0.072251	0.936737	0.936155	-0.175318	1.000000	-0.620300	-0.348840
외국인비율	-0.045956	-0.047102	-0.601076	-0.645463	0.838612	-0.620300	1.000000	0.242816
고령자비율	-0.267841	0.190396	-0.637414	-0.628360	-0.021147	-0.348840	0.242816	1.000000

- CCTV 전체 수(소계)와 가장 상관관계가 있는 데이터는 인구수이다

그러므로

구별 인구대비 CCTV 현황을 분석해서
상대적으로 CCTV가 적거나 많은 구를 찾는 것은
의미를 가진다

In [65]:

```
data_result[ "CCTV비율" ] = data_result[ "소계" ] / data_result[ "인구수" ]
data_result[ "CCTV비율" ] = data_result[ "CCTV비율" ] * 100
data_result.sort_values(by="CCTV비율", ascending=False).head(5)
```

Out [65]:

	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율	CCTV비율
구별									
종로구	1619	248.922414	164257	154770	9487	26182	5.775705	15.939656	0.985651
용산구	2096	53.216374	244444	229161	15283	36882	6.252148	15.088118	0.857456
중구	1023	147.699758	134593	125709	8884	21384	6.600640	15.887899	0.760069
강남구	3238	150.619195	561052	556164	4888	65060	0.871220	11.596073	0.577130
금천구	1348	100.000000	253491	235154	18337	34170	7.233787	13.479769	0.531774

- CCTV 비율을 만들어 CCTV 비율이 높은 구를 보자

In [66]: `data_result.sort_values(by="CCTV비율", ascending=True).head(5)`

Out[66]:

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율	CCTV비율
----	----	-------	-----	-----	-----	-----	-------	-------	--------

강서구	911	134.793814	608255	601691	6564	76032	1.079153	12.500021	0.149773
송파구	1081	104.347826	671173	664496	6677	76582	0.994825	11.410173	0.161061
중랑구	916	79.960707	412780	408226	4554	59262	1.103251	14.356800	0.221910
강동구	1010	166.490765	440359	436223	4136	56161	0.939234	12.753458	0.229358
광진구	878	53.228621	372298	357703	14595	43953	3.920247	11.805865	0.235833

- 인구대비 CCTV 비율이 낮은 구를 보자

MATPLOTLIB 기초



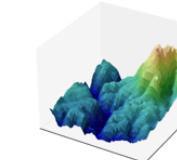
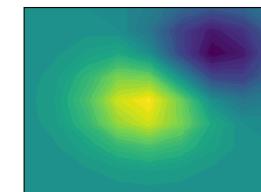
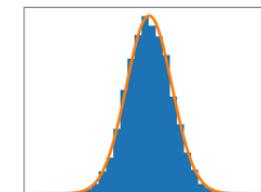
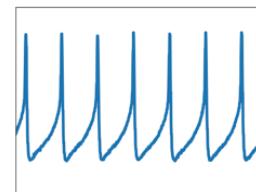
출처: <https://matplotlib.org/>

Version 3.4.3

[Installation](#) [Documentation](#) [Examples](#) [Tutorials](#) [Contributing](#)[home](#) | [contents](#) » Matplotlib: Python plotting

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.



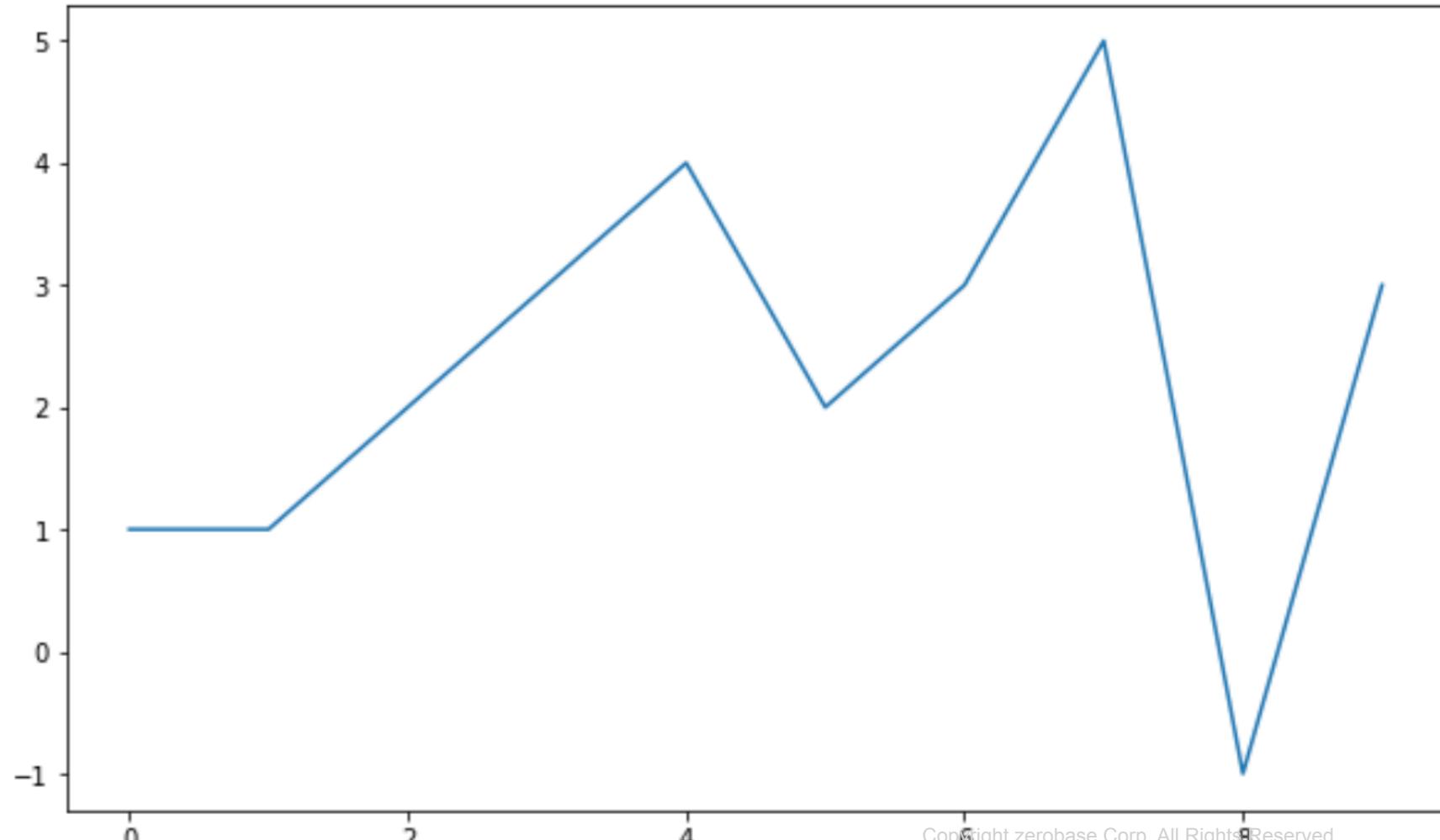
- 파이썬의 대표 시각화 도구
- matplotlib은 plt로 많이 naming하여 사용한다
- Jupyter Notebook 유저의 경우 matplotlib의 결과가 out session에 나타나는 것이 유리하므로
- %matplotlib inline 옵션을 사용한다

```
In [67]: import matplotlib.pyplot as plt
# %matplotlib inline
get_ipython().run_line_magic("matplotlib", "inline")
# black 권고 : 아마 정식 코드로 호출해서 사용하라는 의미
```

matplotlib의 기초

zero-base /

```
In [68]: plt.figure(figsize=(10, 6))
plt.plot([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [1, 1, 2, 3, 4, 2, 3, 5, -1, 3])
plt.show()
```



삼각함수 그리기

- numpy와도 사실 친해져야하는데… 그건 다음에
- numpy의 sin 함수를 가져오자
 - **np.arange(a, b, s)** : a부터 b까지의 s의 간격
 - **np.sin(value)**

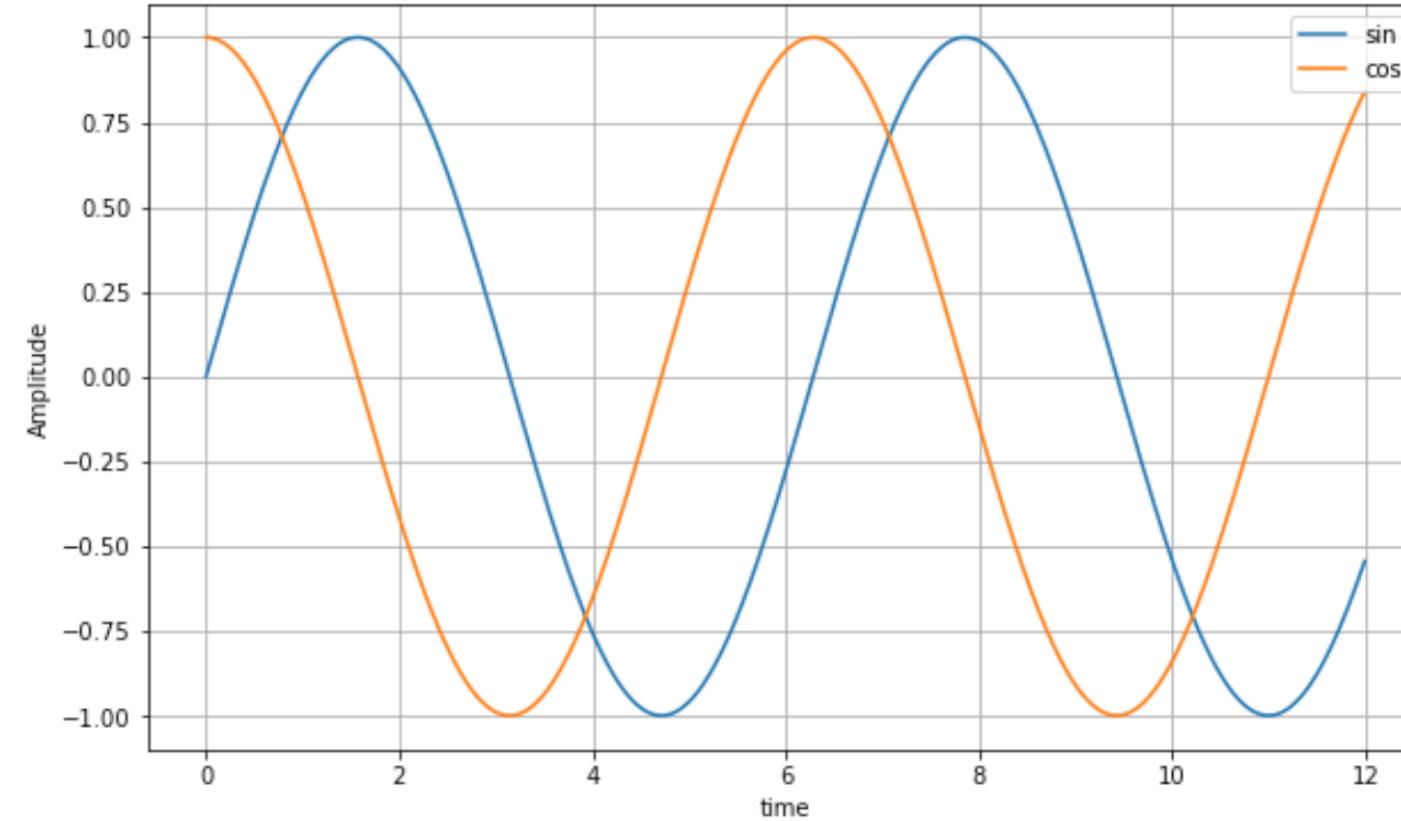
```
In [69]: import numpy as np  
  
t = np.arange(0, 12, 0.01)  
y = np.sin(t)
```

- 우리 수업에서는 그래프의 결과가 중요한 경우
- 그래프를 그리는 코드를 **def()**로 작성한다
- 그러면 나중에 별도의 셀에서 그림만 나타낼 수 있기 때문

```
In [71]: def drawGraph():
    plt.figure(figsize=(10, 6))
    plt.plot(t, np.sin(t), label="sin")
    plt.plot(t, np.cos(t), label="cos")
    plt.grid()
    plt.legend()
    plt.xlabel("time")
    plt.ylabel("Amplitude")
    plt.title("Example of sinewave")
    plt.show()
```

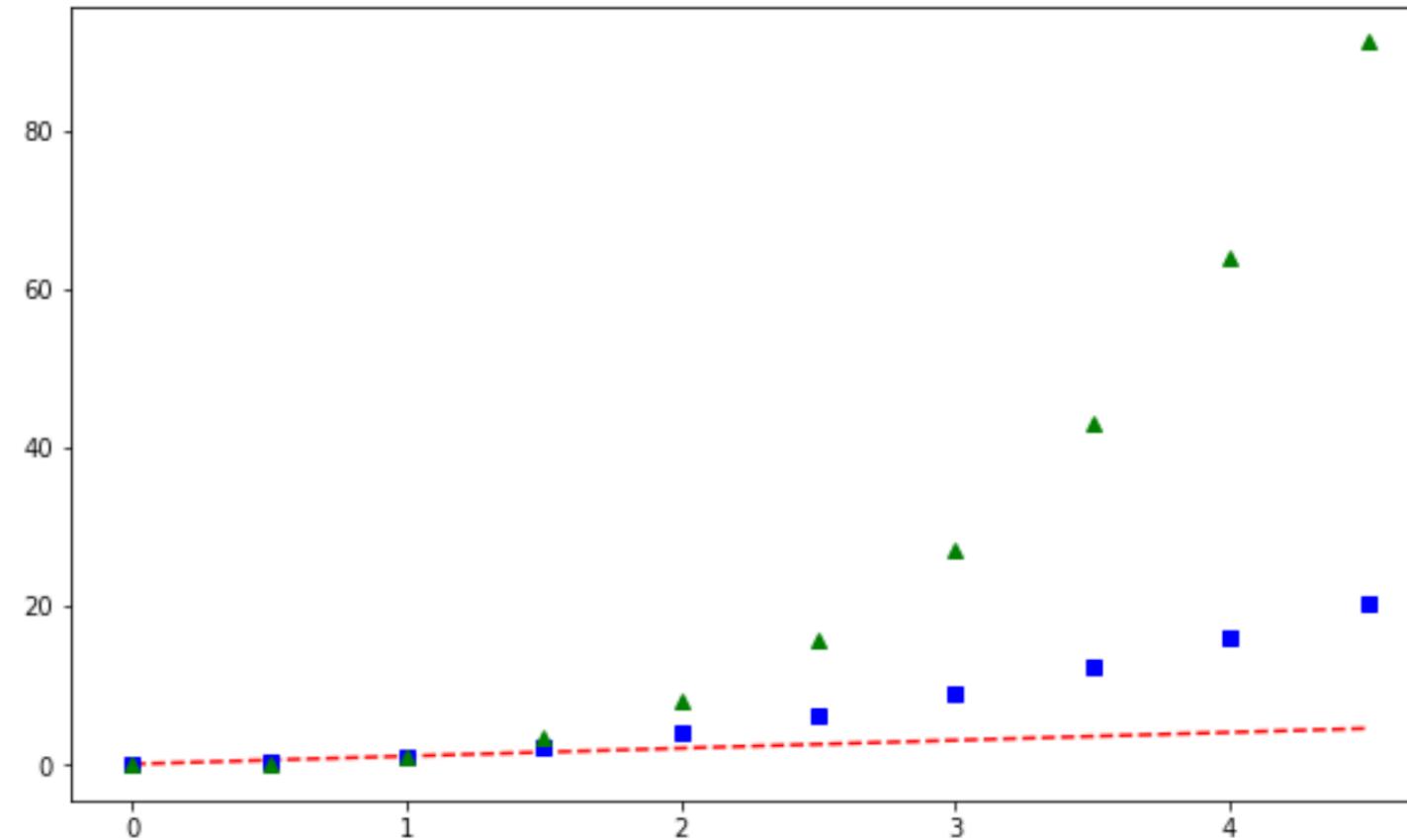
In [72]: `drawGraph()`

Example of sinewave



```
In [73]: t = np.arange(0, 5, 0.5)

def drawGraph():
    plt.figure(figsize=(10, 6))
    plt.plot(t, t, "r--")
    plt.plot(t, t ** 2, "bs")
    plt.plot(t, t ** 3, "g^")
    plt.show()
```

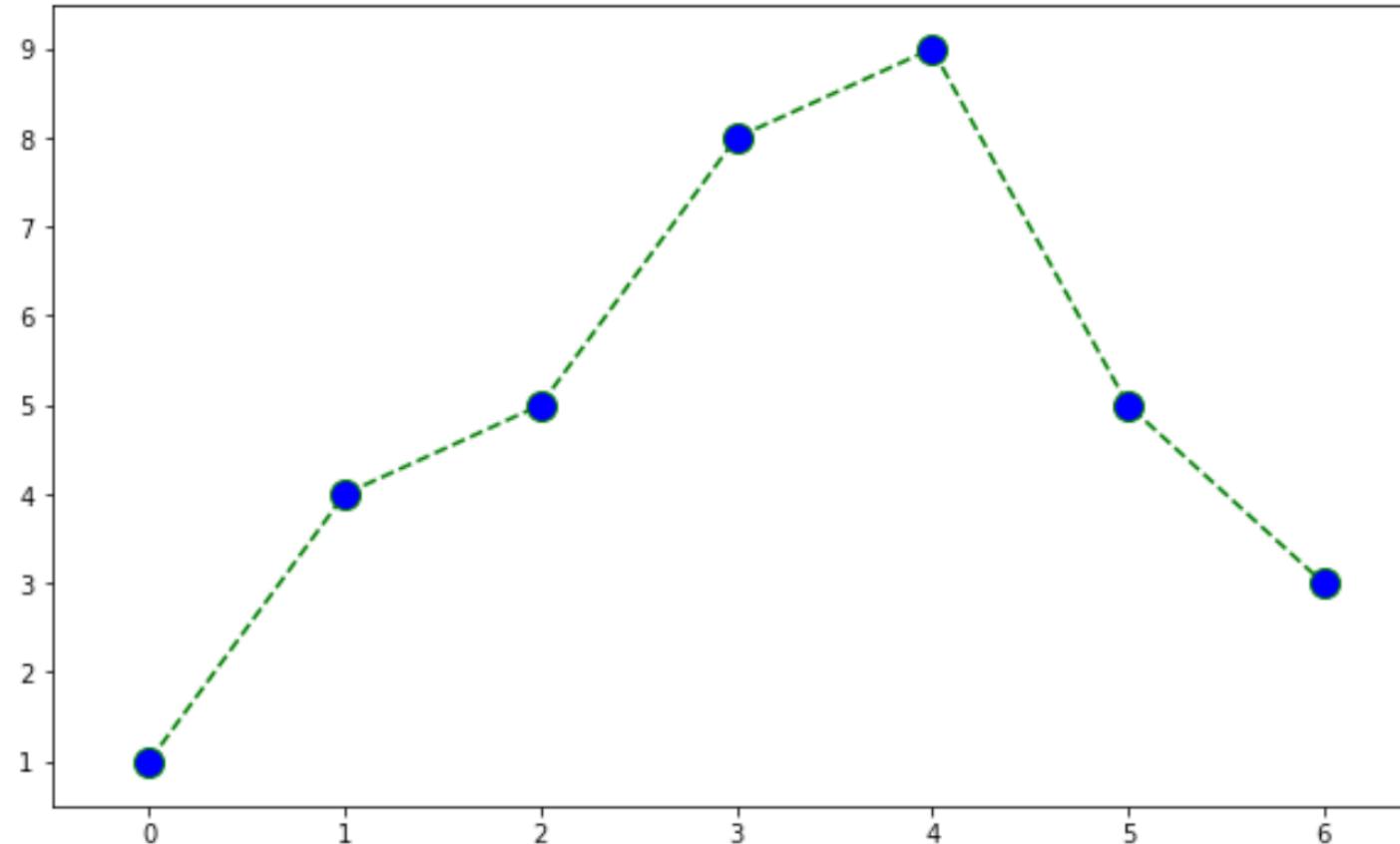
In [74]: `drawGraph()`

```
In [75]: t = [0, 1, 2, 3, 4, 5, 6]
y = [1, 4, 5, 8, 9, 5, 3]

# 다양한 스타일을 지정할 수 있다
def drawGraph():
    plt.figure(figsize=(10, 6))
    plt.plot(
        t,
        y,
        color="green",
        linestyle="dashed",
        marker="o",
        markerfacecolor="blue",
        markersize=12,
    )

    plt.xlim([-0.5, 6.5])
    plt.ylim([0.5, 9.5])
    plt.show()
```

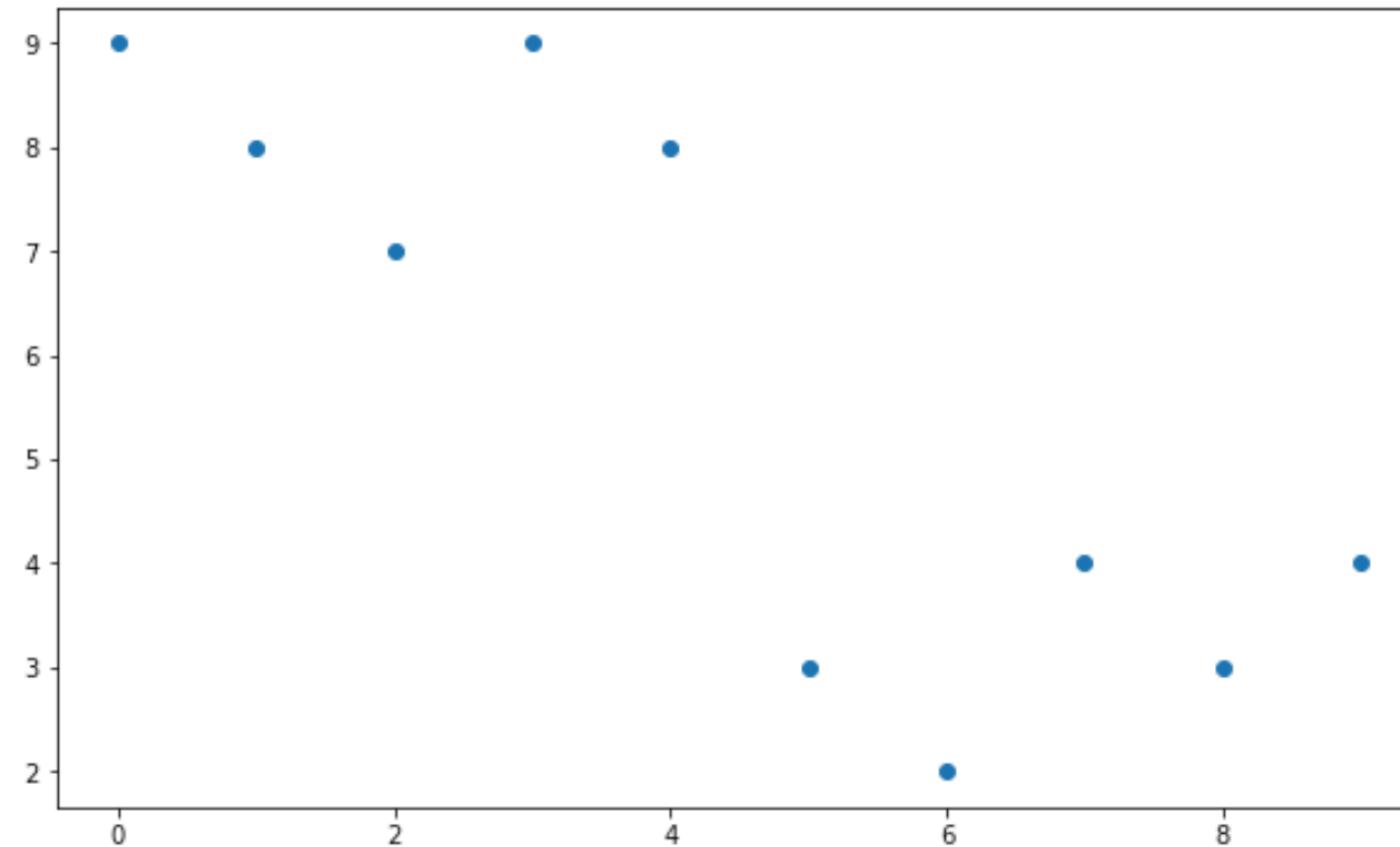
```
In [76]: drawGraph()
```



```
In [77]: t = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([9, 8, 7, 9, 8, 3, 2, 4, 3, 4])
```

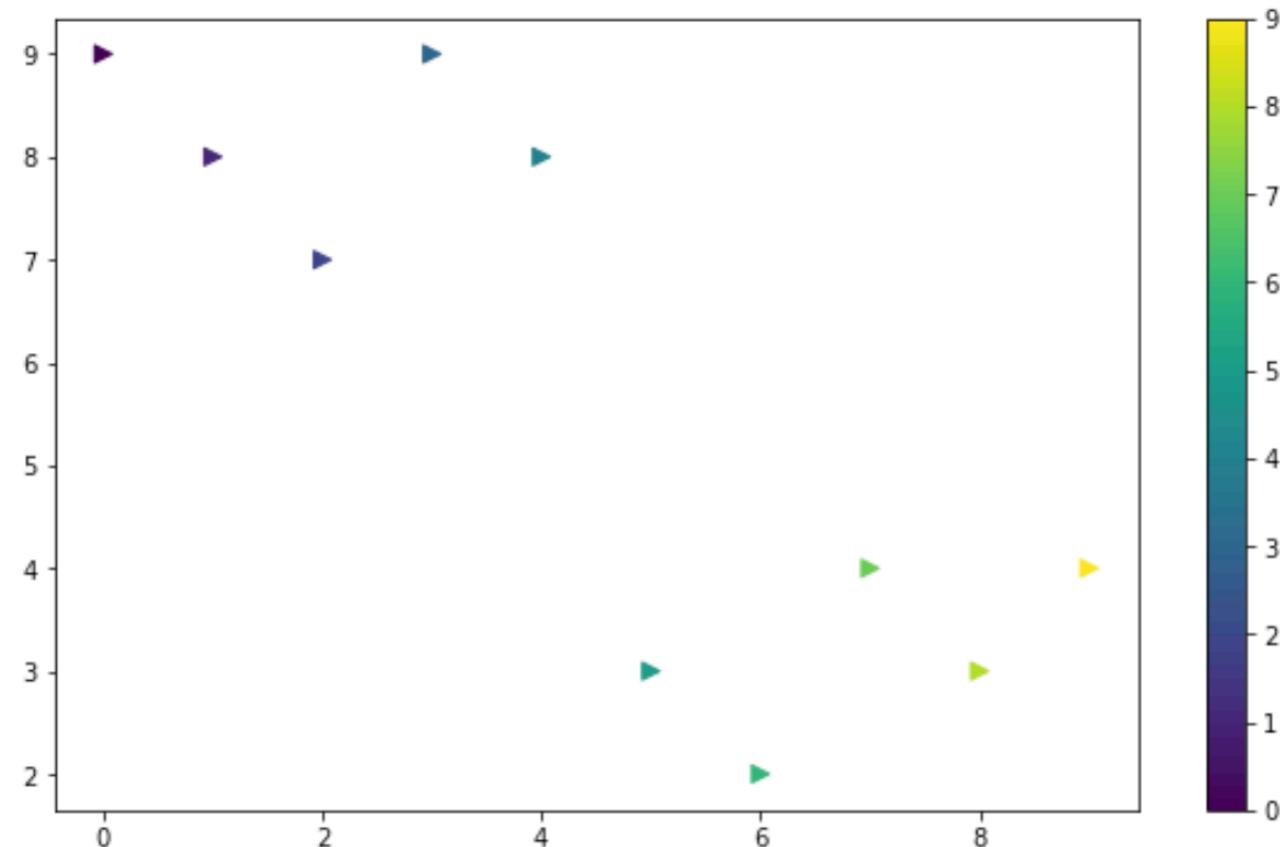
```
# scatter는 점을 뿌리듯이 그리는 그림
def drawGraph():
    plt.figure(figsize=(10, 6))
    plt.scatter(t, y)
    plt.show()
```

In [78]: drawGraph()

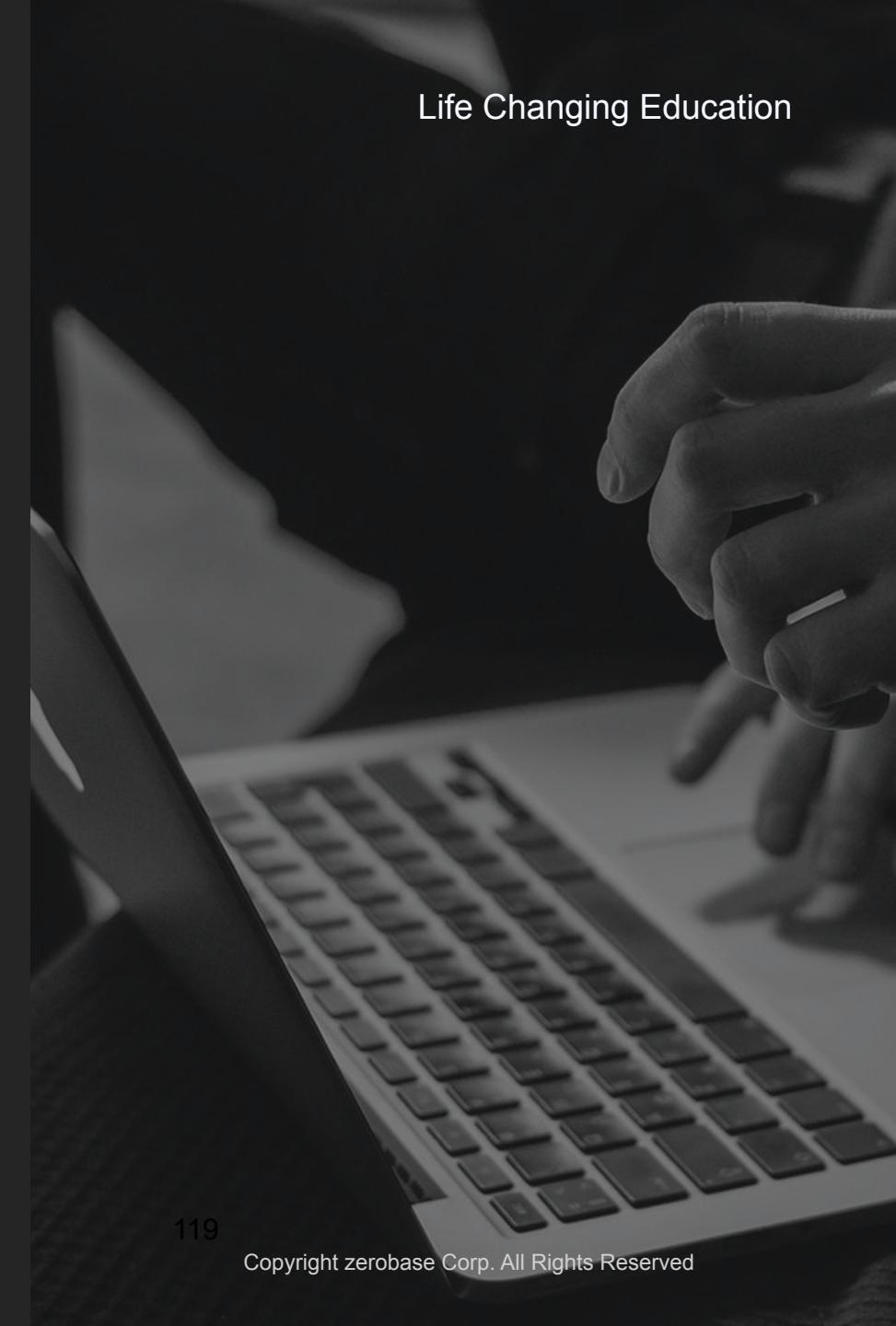


```
In [79]: colormap = t
```

```
# colormap을 적용할 수 있다
def drawGraph():
    plt.figure(figsize=(10, 6))
    plt.scatter(t, y, s=50, c=colormap, marker=">")
    plt.colorbar()
    plt.show()
```

In [80]: `drawGraph()`

데이터 시각화



표로만 확인하는 것은
아무래도 정보나 주장을 전달하는 데 한계가 있다
가자~ Visualization !!!

```
In [103]: from matplotlib import rc  
  
plt.rcParams[ "axes.unicode_minus" ] = False  
  
rc( "font" , family="Arial Unicode MS" )
```

```
In [104]: data_result.head()
```

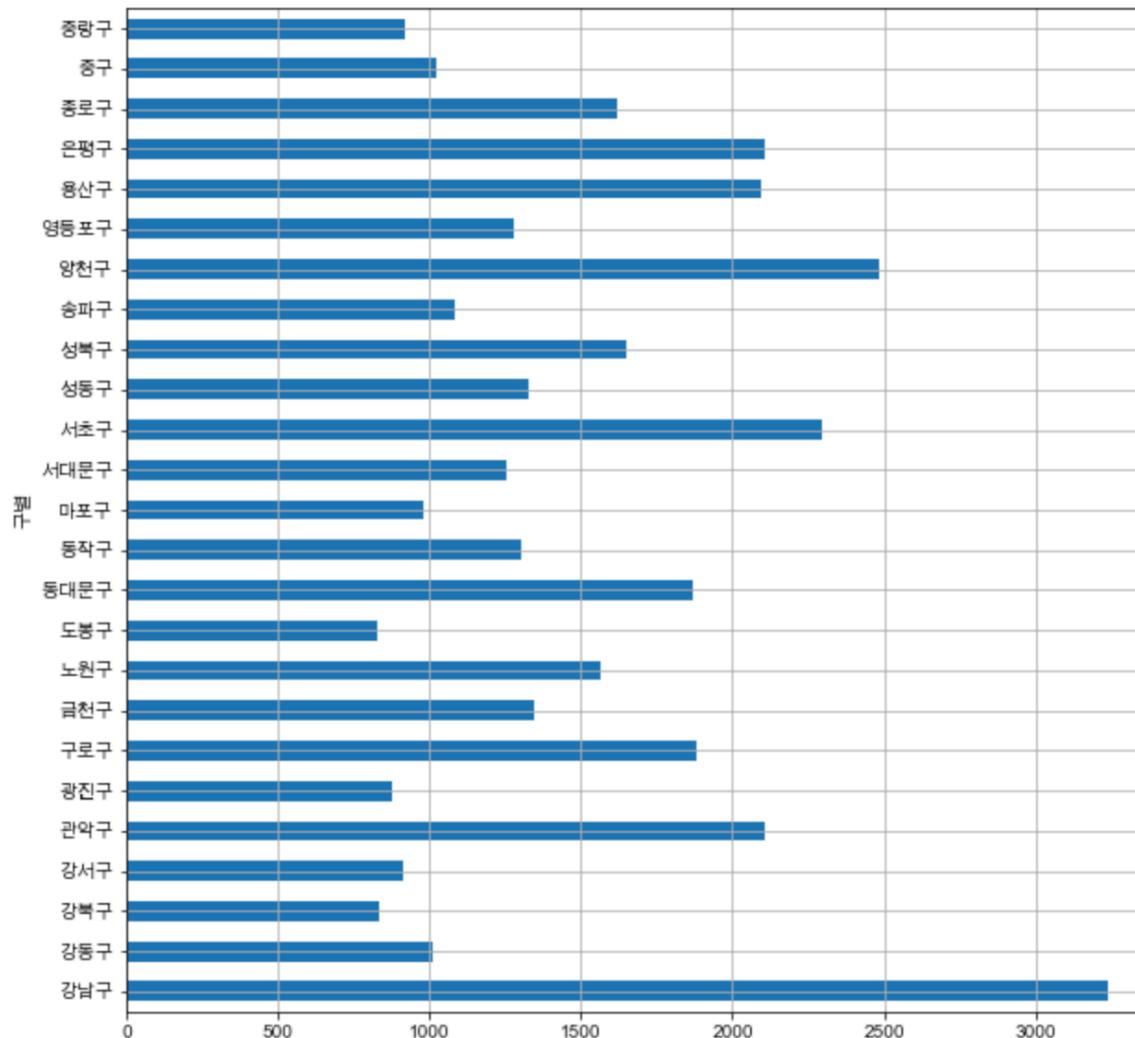
```
Out[104]:
```

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율	CCTV비율
강남구	3238	150.619195	561052	556164	4888	65060	0.871220	11.596073	0.577130
강동구	1010	166.490765	440359	436223	4136	56161	0.939234	12.753458	0.229358
강북구	831	125.203252	328002	324479	3523	56530	1.074079	17.234651	0.253352
강서구	911	134.793814	608255	601691	6564	76032	1.079153	12.500021	0.149773
관악구	2109	149.290780	520929	503297	17632	70046	3.384722	13.446362	0.404854

```
In [105]: data_result["소계"].plot(kind="barh", grid=True, figsize=(10, 10));
```

- Pandas DataFrame은 데이터 변수에서 바로 plot() 명령을 사용할 수 있다
- 그리고 데이터(컬럼)가 많은 경우 정렬한 후 그리는 것이 효과적일 때가 많다

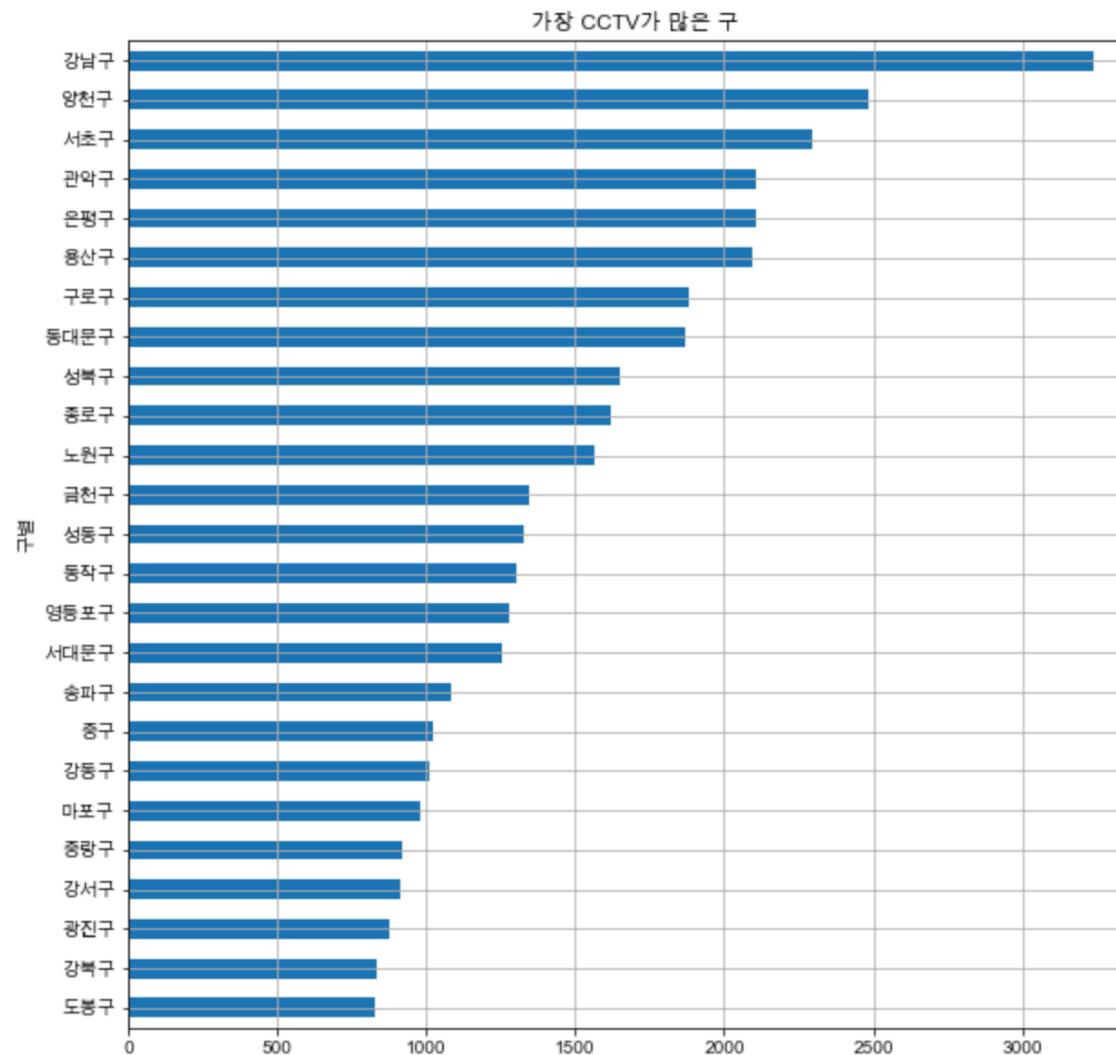
bar 그리기



- 짠~~~~~
- 그런데... 이게 뭐???
- 강남구가 CCTV가 제일 많네...
- 그 이외의 정보는 잘 안 보인다

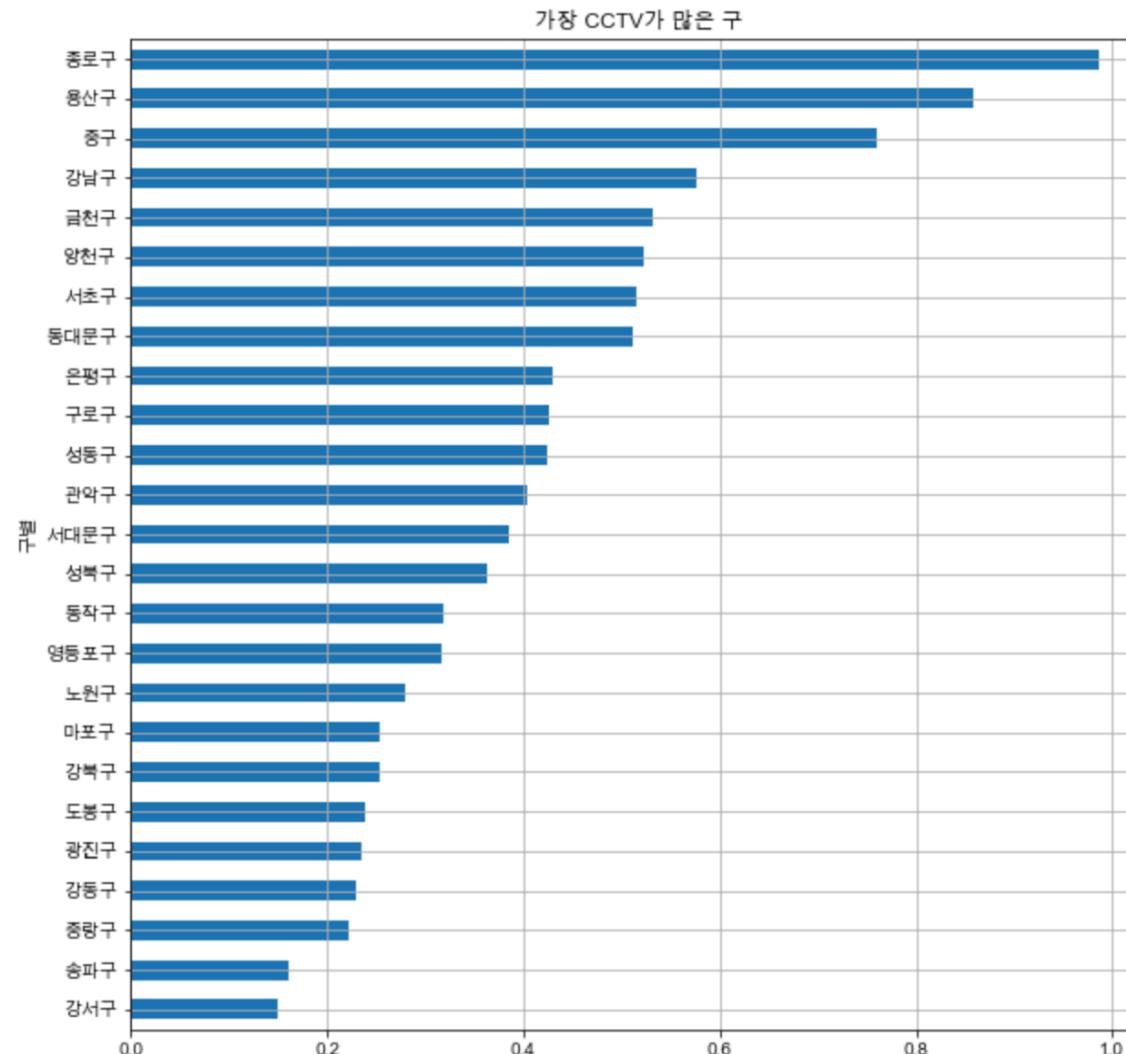
```
In [109]: def drawGraph():
    data_result[ "소계" ].sort_values().plot(
        kind="barh",
        grid=True,
        title="가장 CCTV가 많은 구",
        figsize=(10, 10)
    )
```

bar 그리기

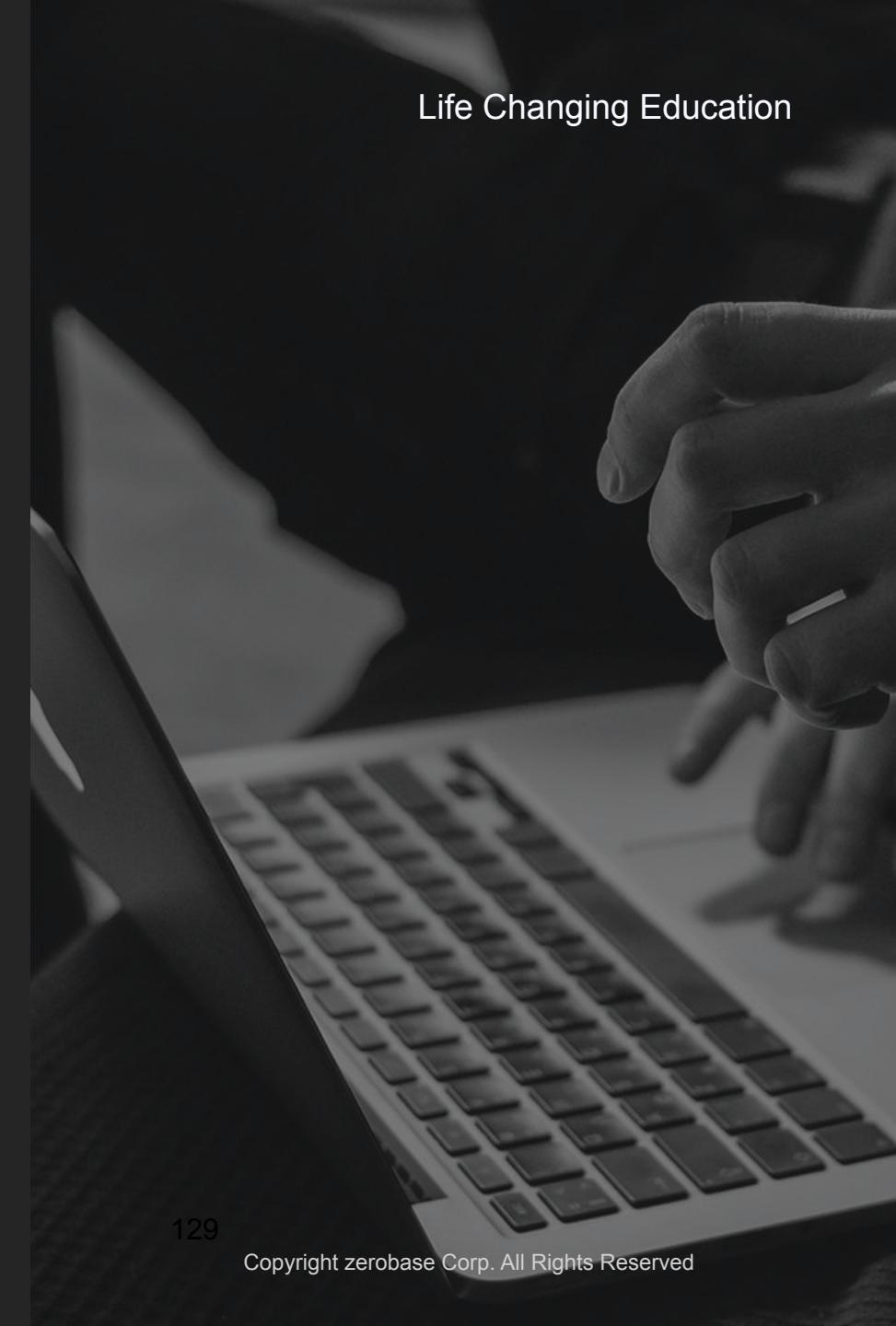
In [110]: `drawGraph()`

```
In [111]: def drawGraph():
    data_result["CCTV비율"].sort_values().plot(
        kind="barh",
        grid=True,
        title="가장 CCTV가 많은 구",
        figsize=(10, 10)
    );
```

bar 그리기

In [112]: `drawGraph()`

데이터의 경향을 표시해보자



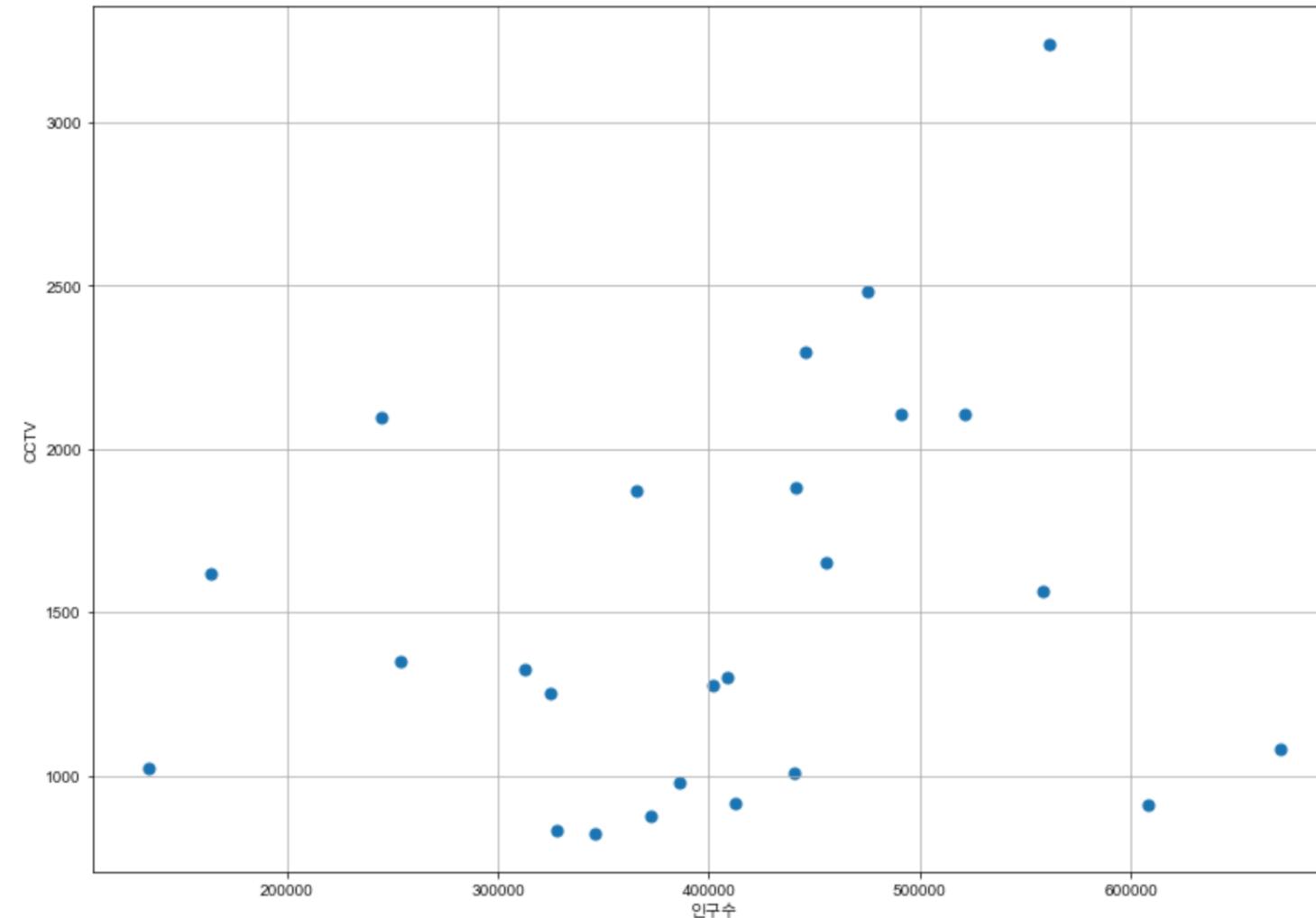
경향을 파악할 필요

- 단순 CCTV 수와 인구대비 CCTV 비율을 볼 때
- CCTV 많은 구는 강남, 양천, 서초, 관악, 은평, 용산
- CCTV 비율이 높은 구는 종로, 용산, 중구가 1위 그룹
- 비율로 데이터를 보아도
- 전체 경향과 함께 보지 않으면 데이터를 제대로 이해시키기 어려울 듯 하다

```
In [113]: def drawGraph():
    plt.figure(figsize=(14, 10))
    plt.scatter(data_result[ "인구수" ], data_result[ "소계" ], s=50)
    plt.xlabel( "인구수" )
    plt.ylabel( "CCTV" )
    plt.grid()
    plt.show()
```

scatter

In [114]: drawGraph()



Linear Regression

선형회귀

Trend 파악

(feat. Numpy)

Numpy를 이용한 1차 직선 만들기

- numpy가 제공하는 간단한 함수를 이용해서 1차 직선을 만들어 그래프로 비교하자
- 절차
 - np.polyfit
 - 직선을 구성하기 위한 계수 계산
 - np.poly1d
 - polyfit으로 찾은 계수로 python에서 사용할 함수로 만들어 줌

```
In [112]: import numpy as np

fp1 = np.polyfit(data_result["인구수"], data_result["소계"], 1)
fp1

Out[112]: array([ 1.11155868e-03,  1.06515745e+03])
```

```
In [116]: f1 = np.poly1d(fp1)
```

- polyfit에서 찾은 계수를 넣어서 함수 완성

```
In [117]: f1(400000)
```

```
Out[117]: 1509.7809252413335
```

- 인구 400000인 구에서 서울시의 전체 경향에 맞는 적당한 CCTV 수를 알고 싶다면?

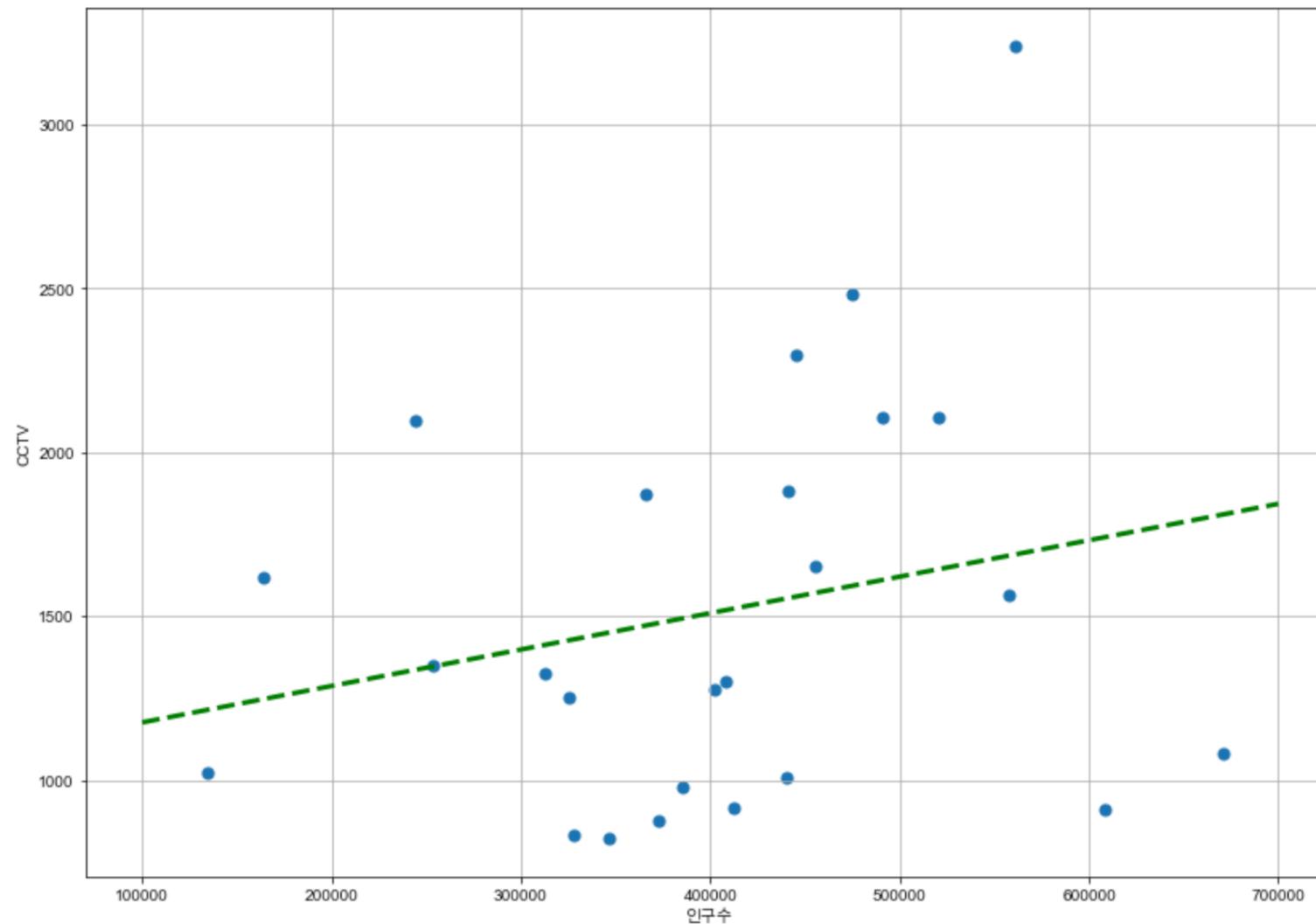
```
In [115]: fx = np.linspace(100000, 700000, 100)
```

- 경향선을 그리기 위해 X 데이터 생성
- `np.linspace(a, b, n)` : a부터 b까지 n개의 등간격 데이터 생성

```
In [119]: def drawGraph():
    plt.figure(figsize=(14, 10))
    plt.scatter(data_result["인구수"], data_result["소계"], s=50)
    plt.plot(fx, f1(fx), ls="dashed", lw=3, color="g")
    plt.xlabel("인구수")
    plt.ylabel("CCTV")
    plt.grid()
    plt.show()
```

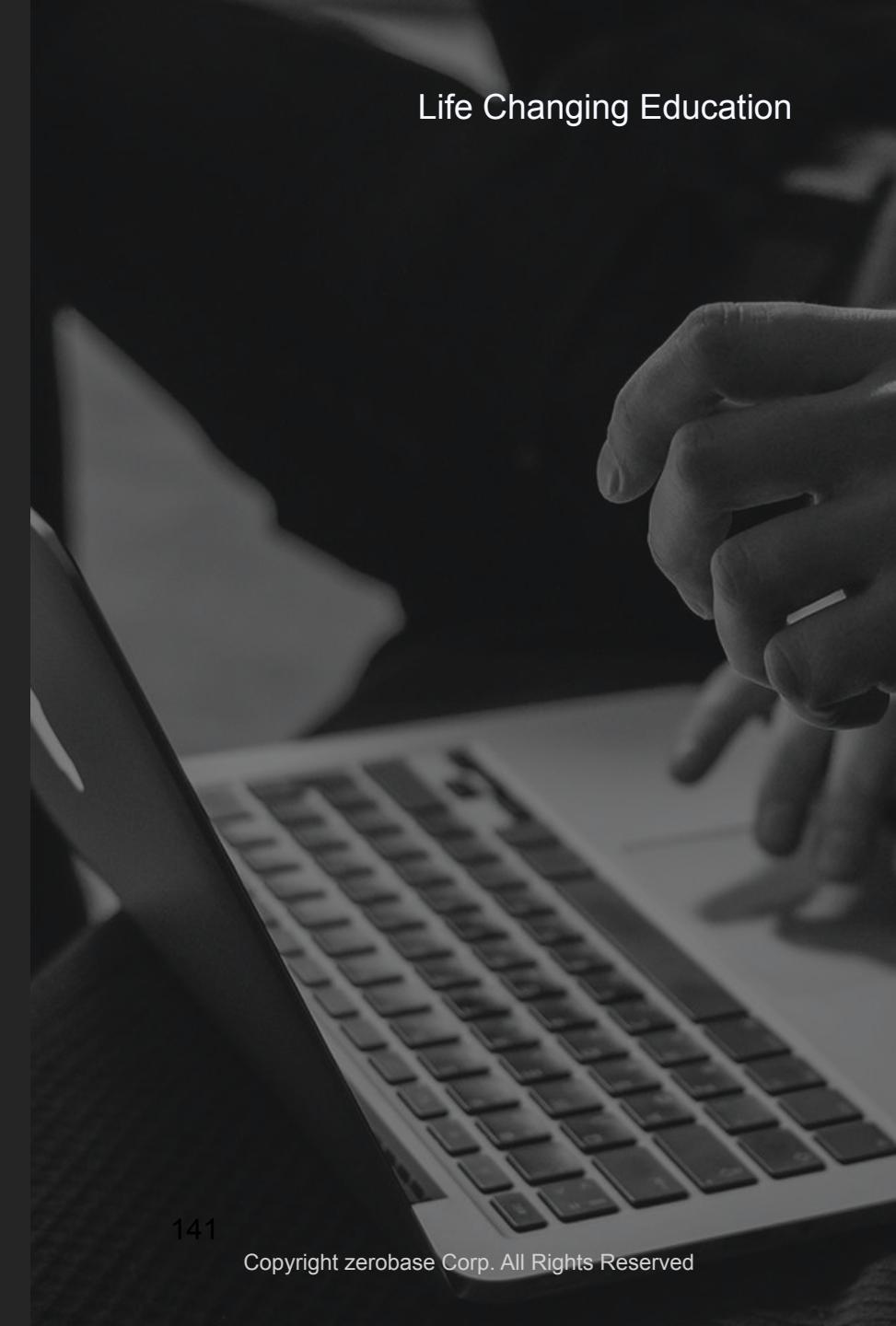
데이터의 경향을 직선으로 표현하기

In [120]: drawGraph()



인구대비 CCTV 개수도 그리고
경향(Trend)도 표현했지만,
그래프 자체로는 아직 부족하다

강조하고 싶은 데이터를 시각화해보자



그래프 다듬기

경향과의 오차를 만들자

```
data_result['오차'] = data_result['소계'] - f1(data_result['인구수'])
```

- 경향(trend)과의 오차를 만들자
- 경향은 f1 함수에 해당 인구를 입력 : f1(data_result['인구수'])
- 현재값 : data_result['소계']

```
In [118]: fp1 = np.polyfit(data_result["인구수"], data_result["소계"], 1)

f1 = np.poly1d(fp1)
fx = np.linspace(100000, 700000, 100)

data_result["오차"] = data_result["소계"] - f1(data_result["인구수"])

# 경향과 비교해서 데이터의 오차가 너무 나는 데이터를 계산
df_sort_f = data_result.sort_values(by="오차", ascending=False)
df_sort_t = data_result.sort_values(by="오차", ascending=True)
```

```
In [119]: df_sort_f.head()
```

```
Out[119]:
```

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율	CCTV비율	오차
강남구	3238	150.619195	561052	556164	4888	65060	0.871220	11.596073	0.577130	1549.200326
양천구	2482	34.671731	475018	471154	3864	55234	0.813443	11.627770	0.522507	888.832166
용산구	2096	53.216374	244444	229161	15283	36882	6.252148	15.088118	0.857456	759.128697
서초구	2297	63.371266	445401	441102	4299	53205	0.965198	11.945415	0.515715	736.753199
은평구	2108	85.237258	491202	486794	4408	74559	0.897390	15.178888	0.429151	496.842700

- 경향 대비 CCTV를 많이 가진 구

In [120]: df_sort_t.head()

Out[120]:

	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율	CCTV비율	오차
구별										
강서구	911	134.793814	608255	601691	6564	76032	1.079153	12.500021	0.149773	-830.268578
송파구	1081	104.347826	671173	664496	6677	76582	0.994825	11.410173	0.161061	-730.205628
도봉구	825	246.638655	346234	344166	2068	53488	0.597284	15.448512	0.238278	-625.016861
중랑구	916	79.960707	412780	408226	4554	59262	1.103251	14.356800	0.221910	-607.986645
광진구	878	53.228621	372298	357703	14595	43953	3.920247	11.805865	0.235833	-600.988527

- 경향 대비 CCTV를 적게 가진 구

```
In [121]: from matplotlib.colors import ListedColormap  
  
# color map을 사용자 정의(user define)로 세팅  
color_step = ["#e74c3c", "#2ecc71", "#95a5a6", "#2ecc71", "#3498db", "#3498db"]  
my_cmap = ListedColormap(color_step)
```

```
In [122]: def drawGraph():
    plt.figure(figsize=(14, 10))
    plt.scatter(data_result["인구수"], data_result["소계"], c=data_result["오차"], s=50, cmap=my_cmap)
    plt.plot(fx, f1(fx), ls="dashed", lw=3, color="grey")

    for n in range(5):
        plt.text(
            df_sort_f["인구수"][n] * 1.02,
            df_sort_f["소계"][n] * 0.98,
            df_sort_f.index[n],
            fontsize=15,
        )

        plt.text(
            df_sort_t["인구수"][n] * 1.02,
            df_sort_t["소계"][n] * 0.98,
            df_sort_t.index[n],
            fontsize=15,
        )

    plt.xlabel("인구수")
    plt.ylabel("CCTV")
    plt.colorbar()
    plt.grid()
    plt.show()
```

```
plt.scatter(data_result[ '인구수' ], data_result[ '소계' ],  
            c=data_result[ '오차' ], s=50, cmap=my_cmap)
```

- s : 마커의 크기
- c : color 세팅에 방금 계산한 경향과의 오차를 적용
- cmap : 사용자 정의한 맵을 적용

```
for n in range(5):
    plt.text(df_sort_f['인구수'][n]*1.02, df_sort_f['소계'][n]*0.98,
              df_sort_f.index[n], fontsize=15)

plt.text(df_sort_t['인구수'][n]*1.02, df_sort_t['소계'][n]*0.98,
          df_sort_t.index[n], fontsize=15)
```

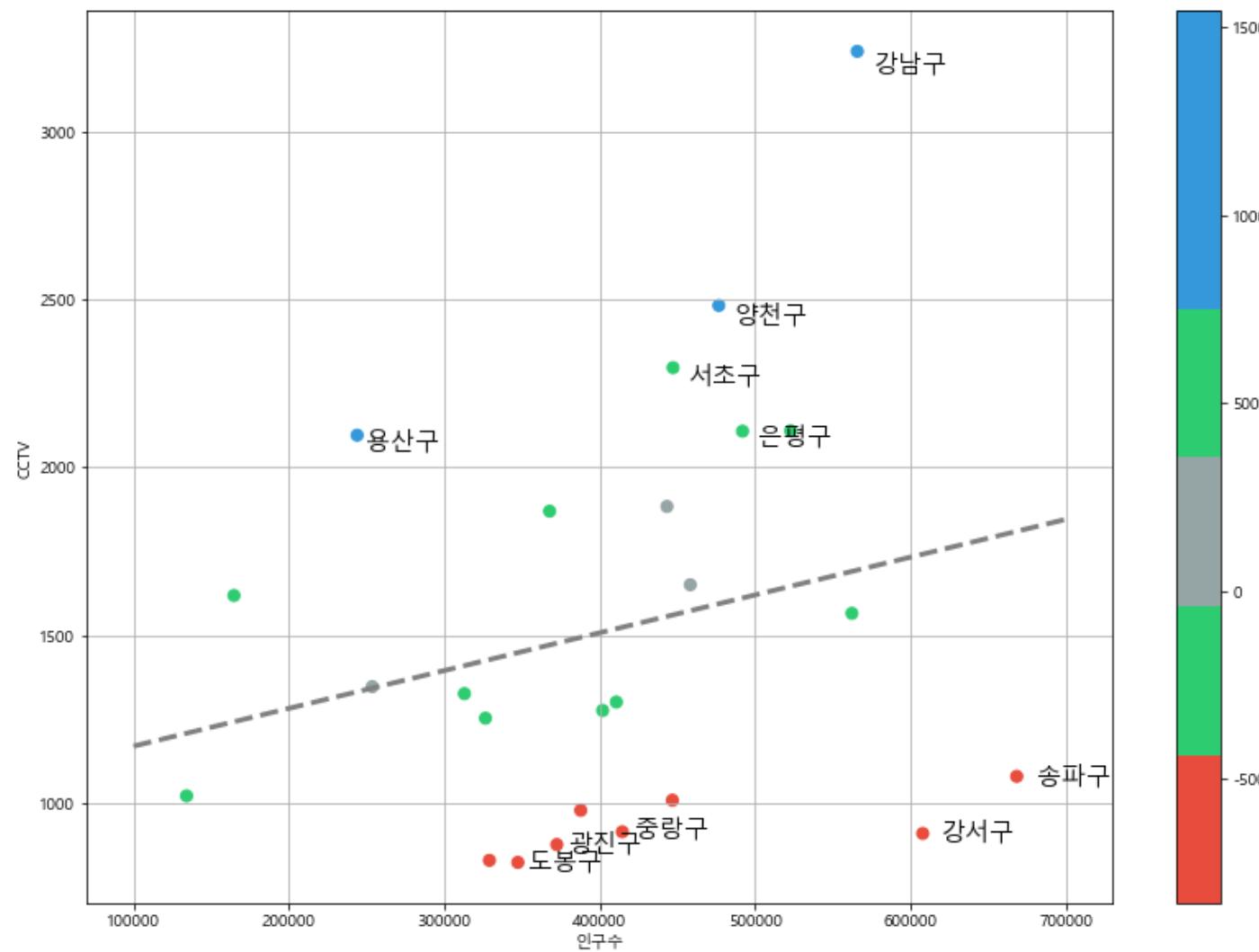
- 오차가 큰 데이터 아래 위로 5개씩만 특별히 마커 옆에 구 이름을 명시

```
plt.text(df_sort_f['인구수'][n]*1.02, df_sort_f['소계'][n]*0.98,  
        df_sort_f.index[n], fontsize=15)
```

- text : 그래프에 글자를 그리는 명령
- plt.text(X, Y, Text, 설정)
- x, y 데이터에 1.02, 0.98을 곱한 이유는?
- 구 이름이 마커에 겹치지 않도록 살짝 거리를 두게 하는 의도

경향에서 벗어난 데이터 강조하기

zero-base /



```
In [124]: data_result.to_csv("../data/01. CCTV_result.csv", sep=",", encoding="utf-8")
```

- 소중한 데이터를 저장하자