

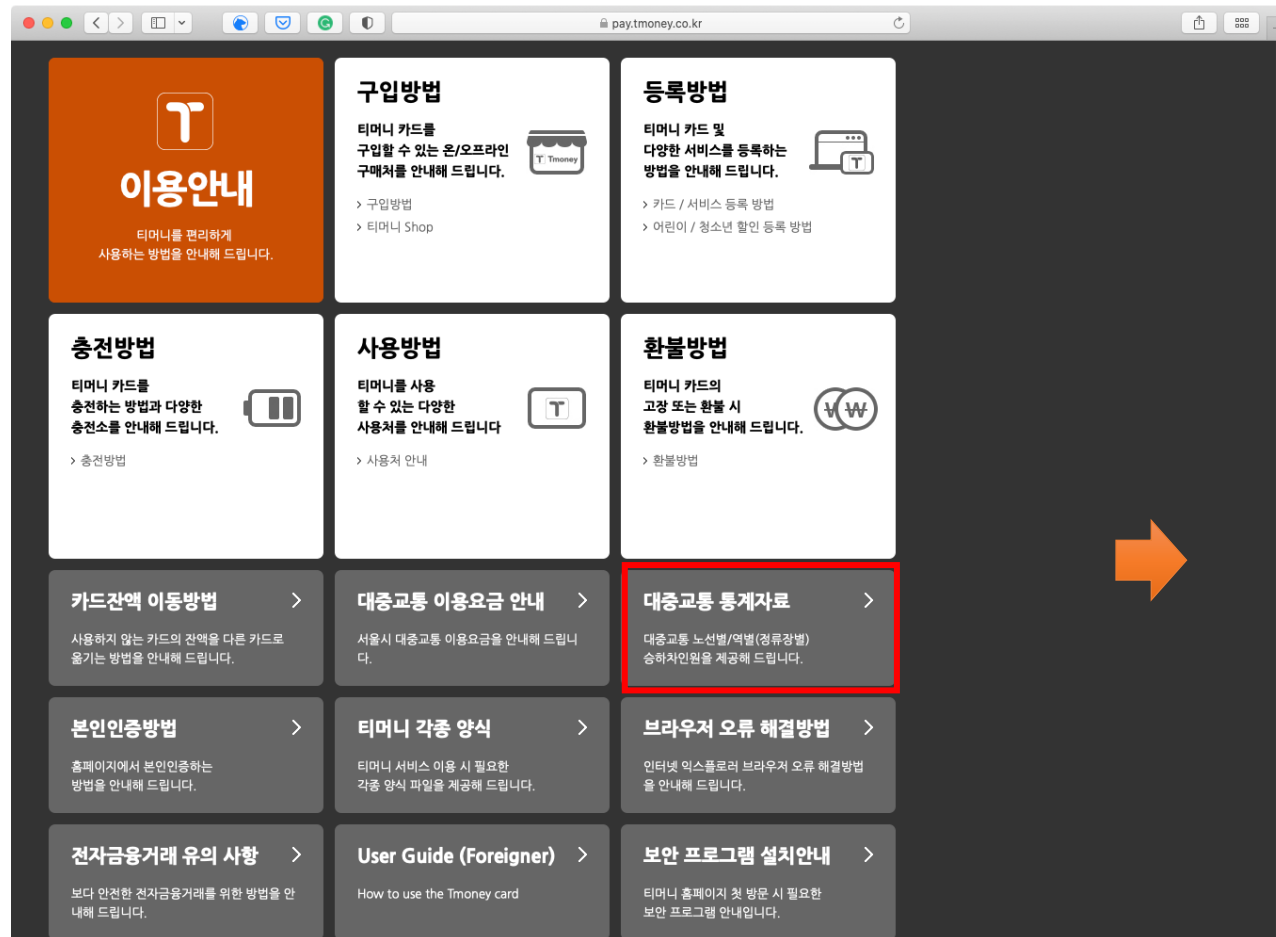
대중 교통 데이터

공공데이터 분석

대중교통 데이터 내려받기

■ 대중교통 데이터: t-money

- <https://pay.tmoney.co.kr/ncs/pct/ugd/ReadUgdMainGd.dev>
- 이용안내 화면 > 대중교통 통계자료 선택 > 2023년 12월 교통카드 통계자료



subway.xls 로 저장

이용안내		
교통카드 통계자료		
월간 교통카드 통계자료를 확인할 수 있습니다.		
총 12건		
번호	제목	등록일
12	2023년 12월 교통카드 통계자료	2024.01.03
11	2023년 11월 교통카드 통계자료	2023.12.03
10	2023년 10월 교통카드 통계자료	2023.11.03
9	2023년 09월 교통카드 통계자료	2023.10.03
8	2023년 08월 교통카드 통계자료	2023.09.03
7	2023년 07월 교통카드 통계자료	2023.08.03
6	2023년 06월 교통카드 통계자료	2023.07.03
5	2023년 05월 교통카드 통계자료	2023.06.03
4	2023년 04월 교통카드 통계자료	2023.05.03
3	2023년 03월 교통카드 통계자료	2023.04.03

교통카드 통계자료

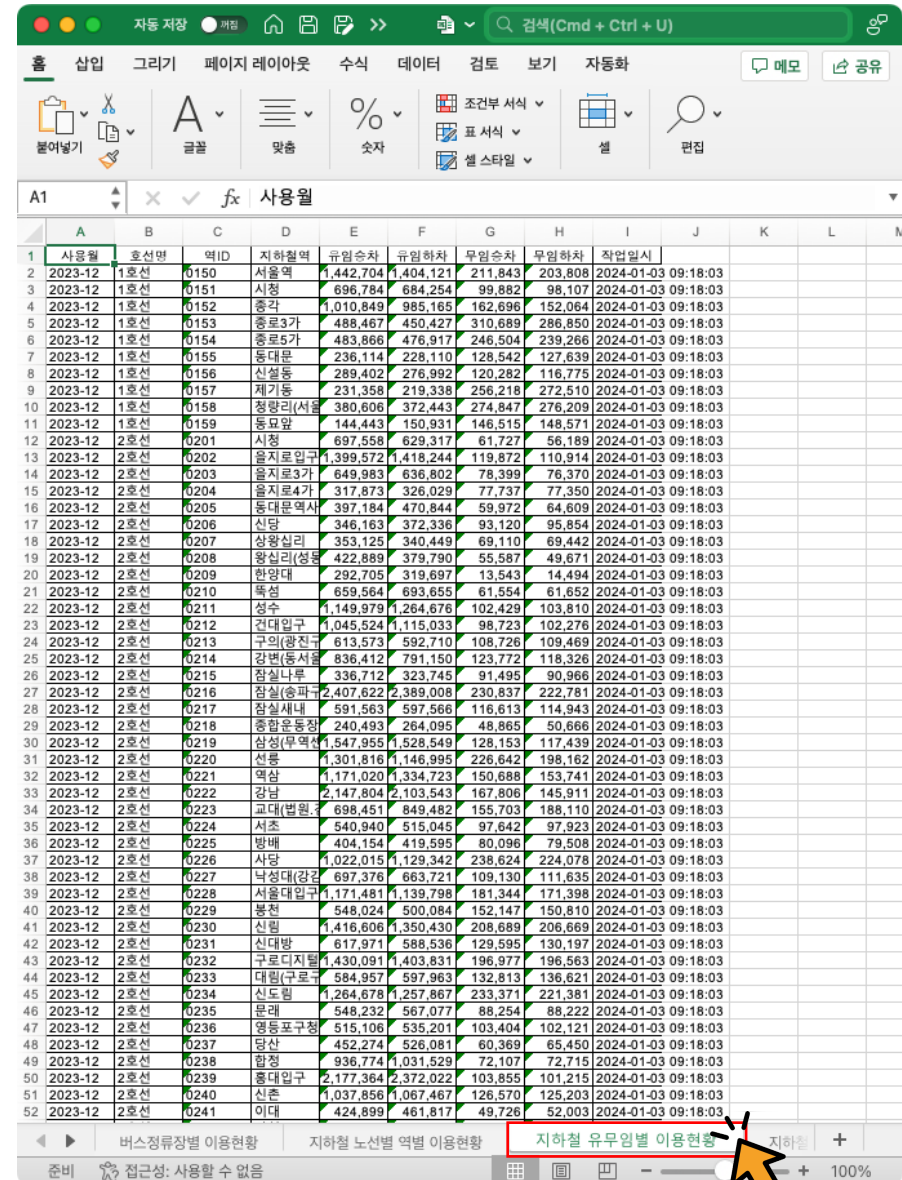
■ 내용: 4개의 탭으로 구성

- 버스정류장별 이용현황
- 지하철 노선별 역별 이용현황
- 지하철 유무임별 이용현황
- 지하철 시간대별 이용현황

■ csv 파일로 저장

- [지하철 유무임별 이용현황] 탭 선택
- 다른 이름으로 저장
 - 파일 형식: CSV UTF-8(쉼표로 분리)
 - 파일 이름: subwayfee.csv

파일 형식: CSV UTF-8(쉼표로 분리) (.csv)



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	사용월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차	작업일시				
2	2023-12	1호선	0150	서울역	1,442,704	1,404,121	211,843	203,808	2024-01-03 09:18:03				
3	2023-12	1호선	0151	시정	696,784	684,254	99,882	98,107	2024-01-03 09:18:03				
4	2023-12	1호선	0152	중각	1,010,849	985,165	162,696	152,064	2024-01-03 09:18:03				
5	2023-12	1호선	0153	중로3가	488,467	450,427	310,689	286,850	2024-01-03 09:18:03				
6	2023-12	1호선	0154	중로5가	483,866	476,917	246,504	239,266	2024-01-03 09:18:03				
7	2023-12	1호선	0155	동대문	236,114	228,110	128,542	127,639	2024-01-03 09:18:03				
8	2023-12	1호선	0156	신설동	289,402	276,992	120,282	116,775	2024-01-03 09:18:03				
9	2023-12	1호선	0157	재기동	231,358	219,338	256,218	272,510	2024-01-03 09:18:03				
10	2023-12	1호선	0158	청량리(서울	380,606	372,443	274,847	276,209	2024-01-03 09:18:03				
11	2023-12	1호선	0159	동묘암	144,443	150,931	146,515	148,571	2024-01-03 09:18:03				
12	2023-12	2호선	0201	시정	697,558	629,317	61,727	56,189	2024-01-03 09:18:03				
13	2023-12	2호선	0202	을지로입구	1,399,572	1,418,244	119,872	110,914	2024-01-03 09:18:03				
14	2023-12	2호선	0203	을지로3가	649,983	636,802	78,399	76,370	2024-01-03 09:18:03				
15	2023-12	2호선	0204	을지로4가	317,873	326,029	77,737	77,350	2024-01-03 09:18:03				
16	2023-12	2호선	0205	동대문역사	397,184	470,844	59,972	64,609	2024-01-03 09:18:03				
17	2023-12	2호선	0206	신당	346,183	372,336	93,120	95,854	2024-01-03 09:18:03				
18	2023-12	2호선	0207	상왕십리	353,125	340,449	69,110	69,442	2024-01-03 09:18:03				
19	2023-12	2호선	0208	왕십리(성북	422,889	379,790	55,587	49,671	2024-01-03 09:18:03				
20	2023-12	2호선	0209	한양대	292,705	319,697	13,543	14,494	2024-01-03 09:18:03				
21	2023-12	2호선	0210	북성문	659,564	693,655	61,554	61,652	2024-01-03 09:18:03				
22	2023-12	2호선	0211	성수	1,149,979	1,264,676	102,429	103,810	2024-01-03 09:18:03				
23	2023-12	2호선	0212	건대입구	1,045,524	1,115,033	98,723	102,276	2024-01-03 09:18:03				
24	2023-12	2호선	0213	구의(광진구	613,573	592,710	108,726	109,469	2024-01-03 09:18:03				
25	2023-12	2호선	0214	강변(동서로	836,412	791,150	123,772	118,326	2024-01-03 09:18:03				
26	2023-12	2호선	0215	잠실나루	336,712	323,745	91,495	90,966	2024-01-03 09:18:03				
27	2023-12	2호선	0216	잠실(송파구	2,407,622	2,389,008	230,837	222,781	2024-01-03 09:18:03				
28	2023-12	2호선	0217	잠실새내	591,563	597,566	116,613	114,943	2024-01-03 09:18:03				
29	2023-12	2호선	0218	중합운동장	240,493	264,095	48,865	50,666	2024-01-03 09:18:03				
30	2023-12	2호선	0219	삼성(무역사	1,547,955	1,528,549	128,153	117,439	2024-01-03 09:18:03				
31	2023-12	2호선	0220	선릉	1,301,816	1,146,995	226,642	198,162	2024-01-03 09:18:03				
32	2023-12	2호선	0221	역삼	1,171,020	1,334,723	150,688	153,741	2024-01-03 09:18:03				
33	2023-12	2호선	0222	강남	2,147,804	2,103,543	167,806	145,911	2024-01-03 09:18:03				
34	2023-12	2호선	0223	교대(법원)	698,451	849,482	155,703	188,110	2024-01-03 09:18:03				
35	2023-12	2호선	0224	서초	540,940	515,045	97,642	97,923	2024-01-03 09:18:03				
36	2023-12	2호선	0225	방배	404,154	419,595	80,096	79,508	2024-01-03 09:18:03				
37	2023-12	2호선	0226	사당	1,022,015	1,129,342	238,624	224,078	2024-01-03 09:18:03				
38	2023-12	2호선	0227	낙성대(강남	697,376	663,721	109,130	111,635	2024-01-03 09:18:03				
39	2023-12	2호선	0228	서울대입구	1,171,481	1,139,798	181,344	171,398	2024-01-03 09:18:03				
40	2023-12	2호선	0229	북성문	548,024	500,084	152,147	150,810	2024-01-03 09:18:03				
41	2023-12	2호선	0230	신림	1,416,606	1,350,430	208,689	206,669	2024-01-03 09:18:03				
42	2023-12	2호선	0231	신대방	617,971	588,536	129,595	130,197	2024-01-03 09:18:03				
43	2023-12	2호선	0232	구로디지털	1,430,091	1,403,831	196,977	196,563	2024-01-03 09:18:03				
44	2023-12	2호선	0233	대림(구로구	584,957	597,963	132,813	136,621	2024-01-03 09:18:03				
45	2023-12	2호선	0234	신도림	1,264,678	1,257,867	233,371	221,381	2024-01-03 09:18:03				
46	2023-12	2호선	0235	문래	548,232	567,077	88,254	88,222	2024-01-03 09:18:03				
47	2023-12	2호선	0236	영등포구청	515,106	535,201	103,404	102,121	2024-01-03 09:18:03				
48	2023-12	2호선	0237	당산	452,274	526,081	60,369	65,450	2024-01-03 09:18:03				
49	2023-12	2호선	0238	합정	936,774	1,031,529	72,107	72,715	2024-01-03 09:18:03				
50	2023-12	2호선	0239	홍대입구	2,177,364	2,372,022	103,855	101,215	2024-01-03 09:18:03				
51	2023-12	2호선	0240	신촌	1,037,856	1,067,467	126,570	125,203	2024-01-03 09:18:03				
52	2023-12	2호선	0241	이대	424,899	461,817	49,726	52,003	2024-01-03 09:18:03				

CSV 파일 데이터 정리: 자리수 콤마 제거

- 맨 오른쪽에 있는 **작업일시** 컬럼 제거
- CSV 파일에서 숫자에 포함된 **자리수 콤마(,)** 제거
 - 컬럼 선택 후 > **숫자로 변경**
 - 셀 서식: **1000** 단위 구분 기호(,) 사용 해제

The screenshot shows an Excel spreadsheet with a table of subway fare data. The 'Format Cells' dialog box is open, and the 'Number' tab is selected. The 'Use 1000 separator' checkbox is unchecked, which is the step to remove commas from large numbers.

	A	B	C	D	E	F	G	H	I
1	사용월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차	
2	Dec.23	1호선	0150	서울역	1442704	1404121	211843	203808	
3	Dec.23	1호선	0151	시청	696784	684254	99882	98107	
4	Dec.23	1호선	0152	종각	1010849	985165	162696	152064	
5	Dec.23	1호선	0153	종로3가	488467	450427	310689	286850	
6	Dec.23	1호선	0154	종로5가	483866	476917	246504	239266	
7	Dec.23	1호선	0155	동대문	236114	228110	128542	127639	
8	Dec.23	1호선	0156	신설동	289402	276992	120282	116775	
9	Dec.23	1호선	0157	제기동	231358	219338	256218	272510	
10	Dec.23	1호선	0158	청량리(서울시)	380606	372443	274847	276209	
11	Dec.23	1호선	0159	동묘앞	144443	150931	146515	148571	
12	Dec.23	2호선	0201	시청	697558	629317	61727	56189	
13	Dec.23	2호선	0202	율지로입구	1399572	1418244	119872	110914	
14	Dec.23	2호선	0203	율지로3가	649983	636802	78399	76370	
15	Dec.23	2호선	0204	율지로4가	317873	326029	77737	77350	
16	Dec.23	2호선	0205	동대문역사문화	397184	470844	59972	64609	
17	Dec.23	2호선	0206	신당	346163	372336	93120	95854	
18	Dec.23	2호선	0207	상왕십리	353125	340449	69110	69442	
19	Dec.23	2호선	0208	왕십리(성동구)	422889	379790	55587	49671	
20	Dec.23	2호선	0209	한양대	292705	319697	13543	14494	
21	Dec.23	2호선	0210	독성	659564	693655	61554	61652	
22	Dec.23	2호선	0211	성수	1149979	1264676	102429	103810	
23	Dec.23	2호선	0212	건대입구	1045524	1115033	98723	102276	
24	Dec.23	2호선	0213	구의(광진구청)	613573	592710	108726	109469	
25	Dec.23	2호선	0214	강변(동서울터)	836412	791150	123772	118326	
26	Dec.23	2호선	0215	잠실나루	336712	323745	91495	90966	
27	Dec.23	2호선	0216	잠실(송파구청)	2407622	2389008	230837	222781	
28	Dec.23	2호선	0217	잠실새내	591563	597566	116613	114943	
29	Dec.23	2호선	0218	종합운동장	240493	264095	48865	50666	
30	Dec.23	2호선	0219	삼성(무역센터)	1547955	1528549	128153	117439	
31	Dec.23	2호선	0220	선릉	1301816	1146995	226642	198162	
32	Dec.23	2호선	0221	역삼	1171020	1334723	150688	153741	
33	Dec.23	2호선	0222	강남	2147804	2103543	167806	145911	
34	Dec.23	2호선	0223	고덕(비와역)	694851	840482	155702	188110	



The screenshot shows the same Excel spreadsheet, but now the 'Format Cells' dialog box is open with the 'Number' tab selected. The 'Use 1000 separator' checkbox is now checked, which will add commas to large numbers.

	A	B	C	D	E	F	G	H	I
1	사용월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차	
2	23.Jul	1호선		150 서울역	1373670	1301690	195005	185973	
3	23.Jul	1호선		151 시청	637798	641026	86072	84194	
4	23.Jul	1호선		152 종각				131264	
5	23.Jul	1호선						263176	
6	23.Jul	1호선						236244	
7	23.Jul	1호선						119156	
8	23.Jul	1호선						110263	
9	23.Jul	1호선						272805	
10	23.Jul	1호선						253163	
11	23.Jul	1호선						139701	
12	23.Jul	2호선						49944	
13	23.Jul	2호선						93155	
14	23.Jul	2호선						71133	
15	23.Jul	2호선						73757	
16	23.Jul	2호선						60402	
17	23.Jul	2호선						93523	
18	23.Jul	2호선						68017	
19	23.Jul	2호선						45299	
20	23.Jul	2호선						12987	
21	23.Jul	2호선						57743	
22	23.Jul	2호선						100958	
23	23.Jul	2호선						91309	

대중교통 데이터 읽어오기

- 데이터 헤더

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

```
import csv

f = open('subwayfee.csv', encoding = 'utf-8-sig')
data = csv.reader(f)
header = next(data)
print(header)
i = 1
for row in data:
    print(row)
    if i > 5:
        break
    i += 1
f.close()
```

<day2_subwayfee_01.py>

```
['사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']
['23.Dec', '1호선', '150', '서울역', '1442704', '1404121', '211843', '203808']
['23.Dec', '1호선', '151', '시청', '696784', '684254', '99882', '98107']
['23.Dec', '1호선', '152', '종각', '1010849', '985165', '162696', '152064']
['23.Dec', '1호선', '153', '종로3가', '488467', '450427', '310689', '286850']
['23.Dec', '1호선', '154', '종로5가', '483866', '476917', '246504', '239266']
['23.Dec', '1호선', '155', '동대문', '236114', '228110', '128542', '127639']
```

전체 탑승 인원 대비 유임 승차 비율이 가장 높은 역은?

- 유임 승차 대 무임 승차 비율 (rate) 계산

- $$\text{rate} = \frac{\text{유임 승차 인원}}{\text{무임 승차 인원}}$$

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

```
import csv
```

<day2_subwayfee_02.py>

```
f = open('subwayfee.csv', encoding='utf-8-sig')
```

```
data = csv.reader(f)
```

```
header = next(data)
```

```
max_rate = 0
```

```
rate = 0
```

```
for row in data:
```

```
    for i in range(4, 8):
```

```
        row[i] = int(row[i]) # 4, 5, 6, 7 컬럼 값을 정수로 변환
```

```
        rate = row[4] / row[6] # [6]컬럼의 값이 0인 행 확인 용도
```

```
        if rate > max_rate:
```

```
            max_rate = rate
```

```
print(max_rate)
```

```
f.close()
```

row[6]의 값이 0인
역이 존재하는지 확인

Traceback (most recent call last):

File "day2_subwayfee_02.py", line 11, in <module>

rate = row[4] / row[6]

ZeroDivisionError: division by zero

무임승차 인원이 0인 역 찾기 #1

```
import csv
```

<day2_subwayfee_03.py>

```
f = open('subwayfee.csv', encoding='utf-8-sig')
```

```
data = csv.reader(f)
```

```
header = next(data)
```

```
rate = 0
```

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

```
for row in data:
```

```
    for i in range(4, 8):
```

```
        row[i] = int(row[i]) # 4, 5, 6, 7 컬럼 값을 정수로 변환
```

```
        rate = row[4] / (row[4] + row[6])
```

```
    if row[6] == 0: # 무임승차 인원[6]이 없는 역 출력
        print(row)
```

$$\text{rate} = \frac{\text{유임 승차 인원}}{\text{전체 탑승인원(유임승차+무임승차)}}$$

```
f.close()
```

```
['23.Dec', '일산선', '1949', '지축', 8, 0, 0, 0]
['23.Dec', '경의선', '1295', '김포공항', 1, 0, 0, 0]
['23.Dec', '6호선', '2649', '신내', 10, 0, 0, 0]
['23.Dec', '7호선', '2755', '춘의', 1, 0, 0, 0]
['23.Dec', '7호선', '2756', '신중동', 1, 0, 0, 0]
['23.Dec', '7호선', '2760', '굴포천', 1, 0, 0, 0]
```

최대 무임 승차 비율 확인

```
import csv
```

<day2_subwayfee_04.py>

```
f = open('subwayfee.csv', encoding='utf-8-sig')
```

```
data = csv.reader(f)
```

```
header = next(data)
```

```
max_rate = 0
```

```
for row in data:
```

```
    for i in range(4, 8):
```

```
        row[i] = int(row[i]) # 4, 5, 6, 7 컬럼 값을 정수로 변환
```

```
    if row[6] != 0:
```

```
        # 무임 승차 (%) = (무임 승차 수 x 100) / (유임 승차 수 + 무임 승차 수)
```

```
        rate = (row[6] * 100) / (row[4] + row[6])
```

```
        if rate > max_rate:
```

```
            max_rate = rate
```

```
            print(row, round(rate, 2), '%')
```

```
f.close()
```

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

```
['23.Dec', '1호선', '150', '서울역', 1442704, 1404121, 211843, 203808] 12.8 %  
['23.Dec', '1호선', '152', '종각', 1010849, 985165, 162696, 152064] 13.86 %  
['23.Dec', '1호선', '153', '종로3가', 488467, 450427, 310689, 286850] 38.88 %  
['23.Dec', '1호선', '157', '제기동', 231358, 219338, 256218, 272510] 52.55 %  
['23.Dec', '경원선', '1916', '소요산', 22856, 19642, 40555, 37554] 63.96 %  
['23.Dec', '경원선', '1919', '연천', 10626, 10601, 20838, 21191] 66.23 %
```

최대 무임 승차 비율을
찾아가는 과정

최대 유임 승차 인원이 있는 역은? #1

- 10만명이 넘게 승·하차 하는 역에서 유임 승차 비율이 제일 높은 역은?
 - 유임승차비율 = 유임승차인원 / 전체승차인원(유임+무임)
 - 유동 인구가 많은 지하철 역중에서 비교

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

호선명: 1호선, 역이름: 서울역, 전체 인원: 1,654,547명, 유임승차인원: 1,442,704명, 유임승차 비율: 87.2%
호선명: 1호선, 역이름: 시청, 전체 인원: 796,666명, 유임승차인원: 696,784명, 유임승차 비율: 87.46%
호선명: 2호선, 역이름: 시청, 전체 인원: 759,285명, 유임승차인원: 697,558명, 유임승차 비율: 91.87%
호선명: 2호선, 역이름: 을지로입구, 전체 인원: 1,519,444명, 유임승차인원: 1,399,572명, 유임승차 비율: 92.11%
호선명: 2호선, 역이름: 한양대, 전체 인원: 306,248명, 유임승차인원: 292,705명, 유임승차 비율: 95.58%
호선명: 공항철도 1호선, 역이름: 홍대입구, 전체 인원: 464,662명, 유임승차인원: 444,277명, 유임승차 비율: 95.61%

호선명: 공항철도 1호선, 역이름: 홍대입구, 전체 인원: 464,662명, 유임승차인원: 444,277명, 유임승차 비율: 95.61%

최대 유임 승차 인원이 있는 역은? #2

```
import csv
```

<day2_subwayfee_05.py>

```
f = open('subwayfee.csv', encoding='utf-8-sig')
data = csv.reader(f)
next(data)
```

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

```
max_rate = 0
max_row = []
max_total_num = 0
```

```
for row in data :
    for i in range(4,8) :
        row[i] = int(row[i])
    total_count = row[4] + row[6] # 유임승차수 + 무임승차수
    if (row[6] !=0) and (total_count >100000):
        rate = row[4] / total_count
```

```
        if rate > max_rate :
            max_rate = rate
            max_row = row
            max_total_num = total_count
```

```
print()
print(f"호선명: {max_row[1]}, 역이름: {max_row[3]}, 전체 인원: {max_total_num:,}명, "
      f"유임승차인원: {max_row[4]:,}명, 유임승차 비율: {round(max_rate * 100, 2):,}%")
```

```
f.close()
```

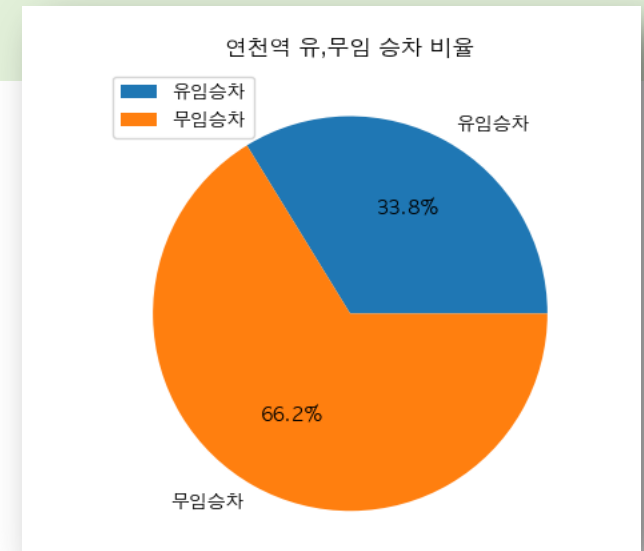
실습: 유임 승차 비율이 50% 이하인 역

- 서울 지하철 노선에서 유임 승차 비율이 50% 이하이고
- 총 승차 인원이 10,000명 이상을 모두 출력
- 유임 승차 비율이 가장 낮은 역의 비율을 파이 차트로 표시하시오.

```
['사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']  
['23.Dec', '1호선', '157', '제기동', 231358, '219338', 256218, '272510'] 0.47  
['23.Dec', '1호선', '159', '동묘앞', 144443, '150931', 146515, '148571'] 0.5  
['23.Dec', '경원선', '1916', '소요산', 22856, '19642', 40555, '37554'] 0.36  
['23.Dec', '경원선', '1919', '연천', 10626, '10601', 20838, '21191'] 0.34  
['23.Dec', '중앙선', '1218', '원덕', 4951, '4673', 5471, '5273'] 0.48
```

유임 승차 비율이
50% 이하인 역

유임 승차 비율이 가장 낮은 역: 연천
전체 인원:31,464명, 유임승차인원:10,626명, 유임승차비율:33.8%



실습: 유임 승차 비율이 50% 이하인 역

```
import csv
import matplotlib.pyplot as plt
import platform

f = open('subwayfee.csv', encoding='utf-8-sig')
data = csv.reader(f)
header = next(data)
print(header)

min_rate = 100
min_row = []
min_total_count = 0

for row in data:
    for i in [4,6]:
        row[i] = int(row[i])
        total_count = row[4] + row[6]
        # 무임승차 인원이 없고, 총 승차인원이 1만명 이상
        if (row[6] != 0) and (total_count >= 10000):
            rate = row[4] / total_count
            if rate <= 0.5:
                print(row, round(rate, 2))
                if rate < min_rate:
                    min_rate = rate
                    min_row = row
                    min_total_count = total_count
```

<day2_subwayfee_06.py>

유임승차, 무임 승차
데이터만 가져옴

```
f.close()

print()
print(f'유임 승차 비율이 가장 낮은 역: {min_row[3]}')
print(f'전체 인원:{min_total_count:,}명, '
      f'유임승차인원:{min_row[4]:,}명, '
      f'유임승차비율:{round(min_rate*100, 1)}%')

if platform.system() == 'Windows':
    plt.rc('font', family='Malgun Gothic')
else:
    plt.rc('font', family='AppleGothic')

plt.title(min_row[3] + "역 유,무임 승차 비율")
label = ['유임승차', '무임승차']
values = [min_row[4], min_row[6]]
plt.pie(values, labels=label, autopct='%.1f%%')
plt.legend(loc=2)
plt.show()
```

승·하차 인원이 가장 많은 역은?

- 모든 역의 유임 승차, 유임 하차, 무임 승차, 무임 하차 인원 분석

<day2_subwayfee_07.py>

```
import csv
max = [0] * 4 # [0]: 최대 유임승차, [1]: 최대 유임하차, [2]: 최대 무임승차, [3]: 최대 무임하차
max_station = [''] * 4
label = ['유임승차', '유임하차', '무임승차', '무임하차']

# with 구문: 자동으로 파일을 close()시킴
with open('subwayfee.csv', encoding='utf-8-sig') as f:
    data = csv.reader(f)
    next(data)

    for row in data:
        for i in range(4, 8):
            row[i] = int(row[i])
            if row[i] > max[i-4]: # 원본데이터의 컬럼 (인덱스-4) -> max리스트의 인덱스
                max[i-4] = row[i]
            max_station[i-4] = row[3] + ' ' + row[1] # '역이름 지하철노선' 추가

for i in range(4):
    print(f'{label[i]}: {max_station[i]} {max[i]:,}명')
```

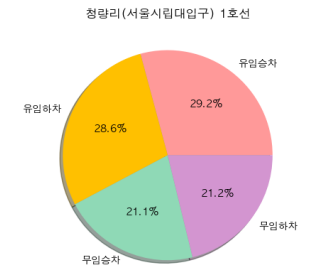
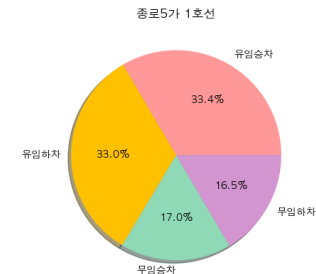
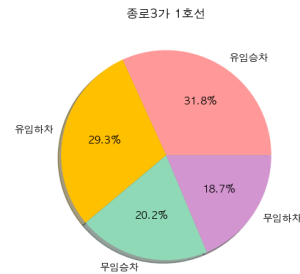
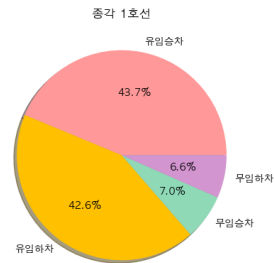
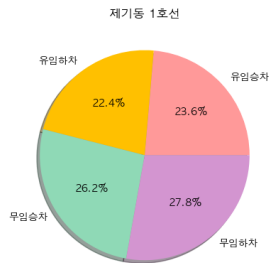
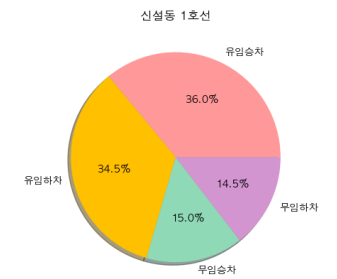
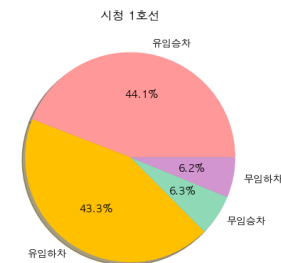
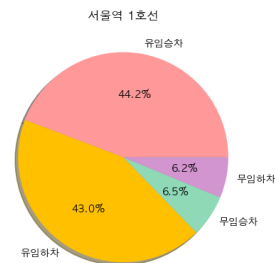
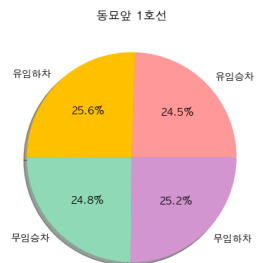
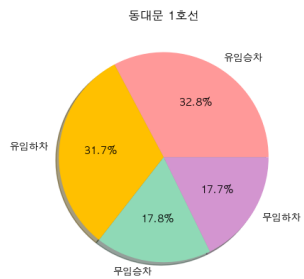
max[0]	max[1]	max[2]	max[3]

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

유임승차: 잠실(송파구청) 2호선 2,407,622명
유임하차: 잠실(송파구청) 2호선 2,389,008명
무임승차: 종로3가 1호선 310,689명
무임하차: 영등포 경부선 307,586명

전체 지하철역 승·하차 인원 분석 및 저장

- 파일 저장: `savefig('파일 이름', dpi)`
 - 총 616개 지하철역의 승·하차 정보가 파일로 저장됨



전체 지하철 역 파이차트 분석

<day2_subwayfee_08.py>

```
import csv
import matplotlib.pyplot as plt
import platform

label = ['유임승차', '유임하차', '무임승차', '무임하차']
color_list = ['#ff9999', '#ffc000', '#8fd9b6', '#d395d0'] # 파이 차트 컬러 값
pic_count = 0
with open('subwayfee.csv', encoding='utf-8-sig') as f:
    data = csv.reader(f)
    next(data)

    if(platform.system() == 'Windows'):
        plt.rc('font', family='Malgun Gothic')
    else:
        plt.rc('font', family='AppleGothic')

    for row in data:
        for i in range(4, 8):
            row[i] = int(row[i])
            print(row)
            plt.figure(dpi=100) # 저장할 그림파일의 dpi 설정
            plt.title(row[3] + ' ' + row[1])
            plt.pie(row[4:8], labels=label, colors=color_list, autopct = '%.1f%%', shadow=True)
            plt.savefig('img/' + row[3] + ' ' + row[1] + '.png')
            plt.close() # 파일 닫기

            pic_count += 1
            if pic_count >= 10:
                break
```

사용 월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

4개 항목에 대한 파이 차트 작성

img/지하철역이름 + 호선번호.png

10개 역의
파이차트만 저장함

지하철 시간대별 데이터 시각화

■ 지하철 시간대별 데이터 활용

- 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까?
- 지하철 시간대별로 가장 많은 사람이 승·하차 하는 역은 어디일까?

■ [지하철 시간대별 이용현황] 데이터

- subwaytime.csv 파일로 저장
 - CSV UTF-8 파일 형식으로 저장

- 데이터에 있는 1000자리 콤마를 제거
 - 데이터 속성: 숫자

- 마지막 ‘작업일시’ 컬럼 제거

The screenshot shows an Excel spreadsheet titled 'subwaytime'. The data is organized in columns representing different time intervals. A dropdown menu is open, showing options like '간단한 날짜' and '자세한 날짜'. The data is organized in a grid with multiple columns for different time periods.

사용일	호선명	역ID	지하철역	04:00:00-04:59:59	05:00:00-05:59:59	06:00:00-06:59:59	07:00:00-07:59:59	08:00:00-08:59:59	09:00:00-09:59:59	10:00:00-10:59:59	11:00:00-11:59:59
Dec 23	1호선	0150	서울역	700	35	7812	8436	12190	하차	하차	하차
Dec 23	1호선	0151	시정	73	1	2208	4356	3731	하차	하차	하차
Dec 23	1호선	0152	중각	167	1	4280	4932	4329	하차	하차	하차
Dec 23	1호선	0153	종로3가	230	16	4174	2538	3621	하차	하차	하차
Dec 23	1호선	0154	종로5가	40	2	1862	3023	3027	하차	하차	하차
Dec 23	1호선	0155	동대문	876	24	11117	2085	8840	하차	하차	하차
Dec 23	1호선	0156	신설동	416	30	8628	1943	9192	하차	하차	하차
Dec 23	1호선	0157	제기동	398	5	4926	2045	7896	하차	하차	하차
Dec 23	1호선	0158	청량리(서울시)	988	51	10364	2548	16198	하차	하차	하차
Dec 23	1호선	0159	동묘앞	186	2	2785	956	3330	하차	하차	하차
Dec 23	2호선	0201	시정	58	0	832	1697	2266	하차	하차	하차
Dec 23	2호선	0202	을지로입구	68	1	2461	2722	4386	하차	하차	하차
Dec 23	2호선	0203	을지로3가	21	0	1335	1815	2782	하차	하차	하차
Dec 23	2호선	0204	을지로4가	18	1	894	1092	1961	하차	하차	하차
Dec 23	2호선	0205	동대문역사문화	274	4	5222	1124	4033	하차	하차	하차
Dec 23	2호선	0206	신당	39	2	6218	1188	10878	하차	하차	하차
Dec 23	2호선	0207	상왕십리	60	1	5657	830	12622	하차	하차	하차
Dec 23	2호선	0208	왕십리(성동구)	691	9	6482	908	9478	하차	하차	하차
Dec 23	2호선	0209	한양대	3	0	1266	567	2370	하차	하차	하차
Dec 23	2호선	0210	북성	4	0	2948	2320	7579	하차	하차	하차
Dec 23	2호선	0211	성수	63	0	5786	3390	8980	하차	하차	하차
Dec 23	2호선	0212	건대입구	324	6	15464	1959	20985	하차	하차	하차
Dec 23	2호선	0213	구의(광진구청)	46	3	13033	1495	24625	하차	하차	하차
Dec 23	2호선	0214	강변(동서울터)	33	1	8316	1994	25176	하차	하차	하차
Dec 23	2호선	0215	잠실나루	22	1	3194	2907	11370	하차	하차	하차
Dec 23	2호선	0216	잠실(송파구청)	118	1	12952	4902	48144	하차	하차	하차
Dec 23	2호선	0217	잠실새내	45	0	5757	1083	16425	하차	하차	하차
Dec 23	2호선	0218	종합운동장	22	0	1452	948	5695	하차	하차	하차
Dec 23	2호선	0219	삼성(무역센터)	158	2	4565	6282	6487	하차	하차	하차
Dec 23	2호선	0220	선릉	154	2	4947	5003	10959	하차	하차	하차
Dec 23	2호선	0221	역삼	78	2	3628	6224	8911	하차	하차	하차
Dec 23	2호선	0222	강남	162	12	10077	10718	18160	하차	하차	하차

지하철 시간대별 자료

- 데이터 내용 (총 지하철 역의 시간대별 승·하차 인원수)
 - 승차시간: 교통카드를 찍고 들어오는 시각
 - 환승 인원은 확인 할 수 없음
 - 두 줄의 헤더 정보를 포함하고 있음

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	...	[50]	[51]
사용 월	호선명	역ID	지하철역	04:00~04:59:59		05:00~05:59:59		06:00~06:59:59		...	03:00~03:59:59	
				승차	하차	승차	하차	승차	하차		승차	하차
23.Dec	1호선	1	서울역	700	35	7812	8436	12190	50415		0	0

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
사용월	호선명	역ID	지하철역	04:00:00~04:59:59		05:00:00~05:59:59		06:00:00~06:59:59		07:00:00~07:59:59		08:00:00~08:59:59		09:00:00~09:59:59		10:00:00~10:59:59	
				승차	하차	승차	하차	승차	하차	승차	하차	승차	하차	승차	하차	승차	하차
23.Dec	1호선	0150	서울역	700	35	7812	8436	12190	50415	37075	105313	68020	218007	67218	160369	65860	86649
23.Dec	1호선	0151	시청	73	1	2208	4356	3731	21903	7341	63969	9896	182565	13154	92045	16154	46463
23.Dec	1호선	0152	종각	167	1	4280	4932	4329	25201	6459	98556	10114	244858	12834	148416	18502	66602
23.Dec	1호선	0153	종로3가	230	16	4174	2538	3621	11378	5454	23309	8475	61562	12940	61089	19368	54571
23.Dec	1호선	0154	종로5가	40	2	1862	3023	3027	14468	5753	38726	9099	90539	13585	64668	22511	57653
23.Dec	1호선	0155	동대문	876	24	11117	2085	8840	5685	14001	9896	19766	17228	19131	20927	17494	22928
23.Dec	1호선	0156	신설동	416	30	8628	1943	9192	8105	19443	20328	29568	51992	20702	31823	18772	21435
23.Dec	1호선	0157	제기동	398	5	4926	2045	7896	8073	19163	18337	30564	33073	22514	30452	23631	33402
23.Dec	1호선	0158	청량리(서울시)	988	51	10364	2548	16198	11446	42147	16179	53214	34154	35344	33452	31854	36786
23.Dec	1호선	0159	동묘앞	186	2	2785	956	3330	4326	7069	7749	11703	18470	10199	15954	11962	19840

데이터 정수 변환

■ map() 함수

- 리스트의 요소를 지정된 함수로 처리함
- map 객체를 리턴
- map(function, iterable)
 - 첫 번째 인자: 데이터에 적용할 함수 이름 입력
 - 두 번째 인자: 그 함수를 적용할 데이터 입력

```
def fun_square(x):  
    return x**2
```

```
a = [1, 2, 3, 4]
```

```
a = list(map(fun_square, a)) # 각 숫자의 제곱  
print(a)
```

```
data = ['1', '2', '3', '4']
```

```
data = list(map(int, data)) # int() 함수를 이용하여 문자열(data)을 정수로 변환  
print(data)
```

```
[1, 4, 9, 16]  
[1, 2, 3, 4]
```

시간대별 지하철 이용 인원 수

■ 새벽 4시 지하철 승차 전체 인원

<day2_subwaytime_01.py>

```
import csv
```

```
result = []
```

```
total_number = 0
```

```
with open('subwaytime.csv') as f:
```

```
    data = csv.reader(f)
```

```
    next(data) # 2줄의 헤더 정보를 건너뛴
```

```
    next(data)
```

```
    for row in data:
```

```
        row[4:] = map(int, row[4:]) # 문자열을 숫자로 변경
```

```
        total_number += row[4]
```

```
        result.append(row[4])
```

```
print(f'총 지하철 역의 수: {len(result)}')
```

```
print(f'새벽 4시 승차인원: {total_number:,}')
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	...	[50]	[51]
사용 월	호선명	역ID	지하철역	04:00~04:59:59		05:00~05:59:59		06:00~06:59:59		...	03:00~03:59:59	
				승차	하차	승차	하차	승차	하차		승차	하차
23.Dec	1호선	1	서울역	700	35	7812	8436	12190	50415		0	0

row[4:]: 인덱스 4부터 끝까지

총 지하철 역의 수: 621

새벽 4시 승차인원: 133,185

새벽4시 지하철 이용 인원 수 (그래프)

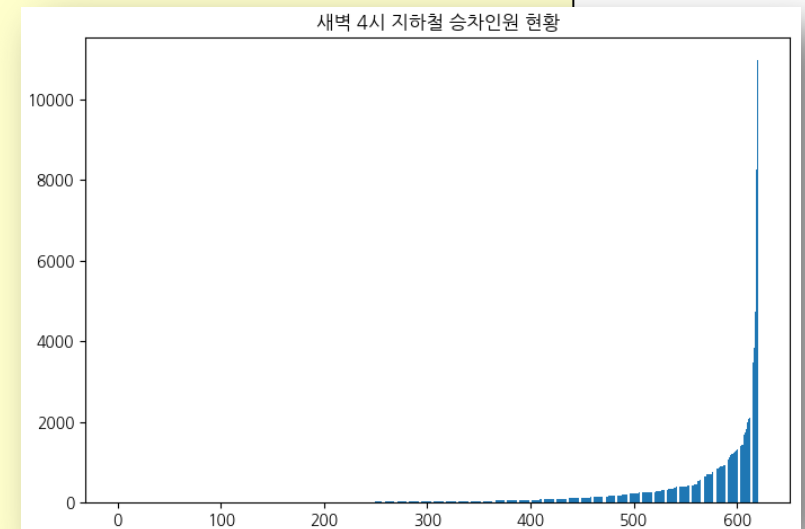
```
import csv
import matplotlib.pyplot as plt
import koreanize_matplotlib

with open('subwaytime.csv') as f:
    data = csv.reader(f)
    next(data) # 2줄의 헤더 정보 건너뛰기
    next(data)
    result = []
    total_number = 0
    max_num = -1
    max_station = ''

    for row in data:
        row[4:] = map(int, row[4:])
        total_number += row[4]
        result.append(row[4])
        if row[4] > max_num:
            max_num = row[4]
            max_station = row[3]

print('새벽 4시 승차 인원수: {0:,}'.format(total_number))
print('최대 승차역: {0}, 인원수:{1:,}'.format(max_station, max_num))
result.sort() # 오름 차순으로 정렬 result.sort(reverse=True)
plt.figure(dpi=100)
plt.bar(range(len(result)), result)
plt.title('새벽 4시 지하철 승차인원 현황')
plt.show()
```

<day2_subwaytime_02.py>

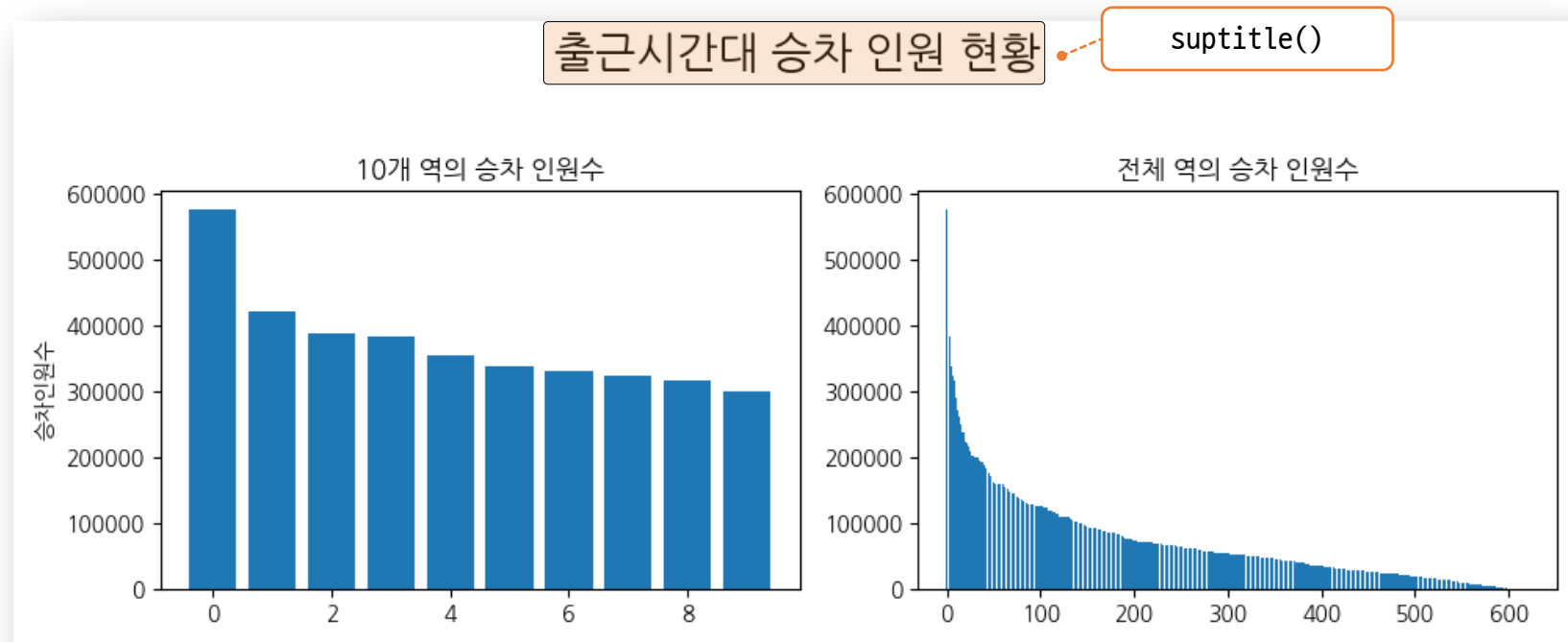


새벽 4시 승차 인원수: 133,185
최대 승차역: 구로, 인원수:10,986

출근 시간대 지하철 이용 현황 #1

- 출근 시간대(7~9시까지) 모든 역의 **승차 인원**을 계산하고 내림차순으로 10개 역의 승차 인원을 막대 그래프로 출력 하시오.
 - 7시, 8시, 9시 승차: index=10, 12, 14

최대 승차 인원역: 신림(2호선) 576,994



출근 시간대 지하철 이용 현황 #2

<day2_subwaytime_03.py>

```
import csv
import matplotlib.pyplot as plt
import koreanize_matplotlib

with open('subwaytime.csv') as f:
    data = csv.reader(f)
    next(data) # 2줄의 헤더 정보 건너뛰
    next(data)
    result = []
    total_number = 0
    max_num = -1
    max_station = ''

    for row in data:
        row[4:] = map(int, row[4:])
        row_sum = sum(row[10:15:2]) # index 10, 12, 14
        #row_sum = row[10] + row[12] + row[14]
        result.append(row_sum)
        if row_sum > max_num:
            max_num = row_sum
            max_station = row[3] + '(' + row[1] + ')'
```

row[10], [12], [14]:
오전 7시, 8시, 9시 승차

```
print(f'최대 승차 인원역: {max_station} {max_num:,}')
result.sort(reverse=True)
```

```
# 1행, 2열의 그래프 그리기
plt.figure(figsize=(10, 4))
ax1 = plt.subplot(1, 2, 1) # (행의 수, 열의 수, 인덱스)
plt.title('10개 역의 승차 인원수', size=12)
plt.bar(range(10), result[0:10])
plt.ylabel('승차인원수')
```

```
ax2 = plt.subplot(1, 2, 2, sharey=ax1)
plt.title('전체 역의 승차 인원수', size=12)
plt.bar(range(len(result)), result)
```

sharey: y축 label
공유

```
plt.suptitle('출근시간대 승차 인원 현황\n', size=20)
plt.tight_layout()
plt.show()
```

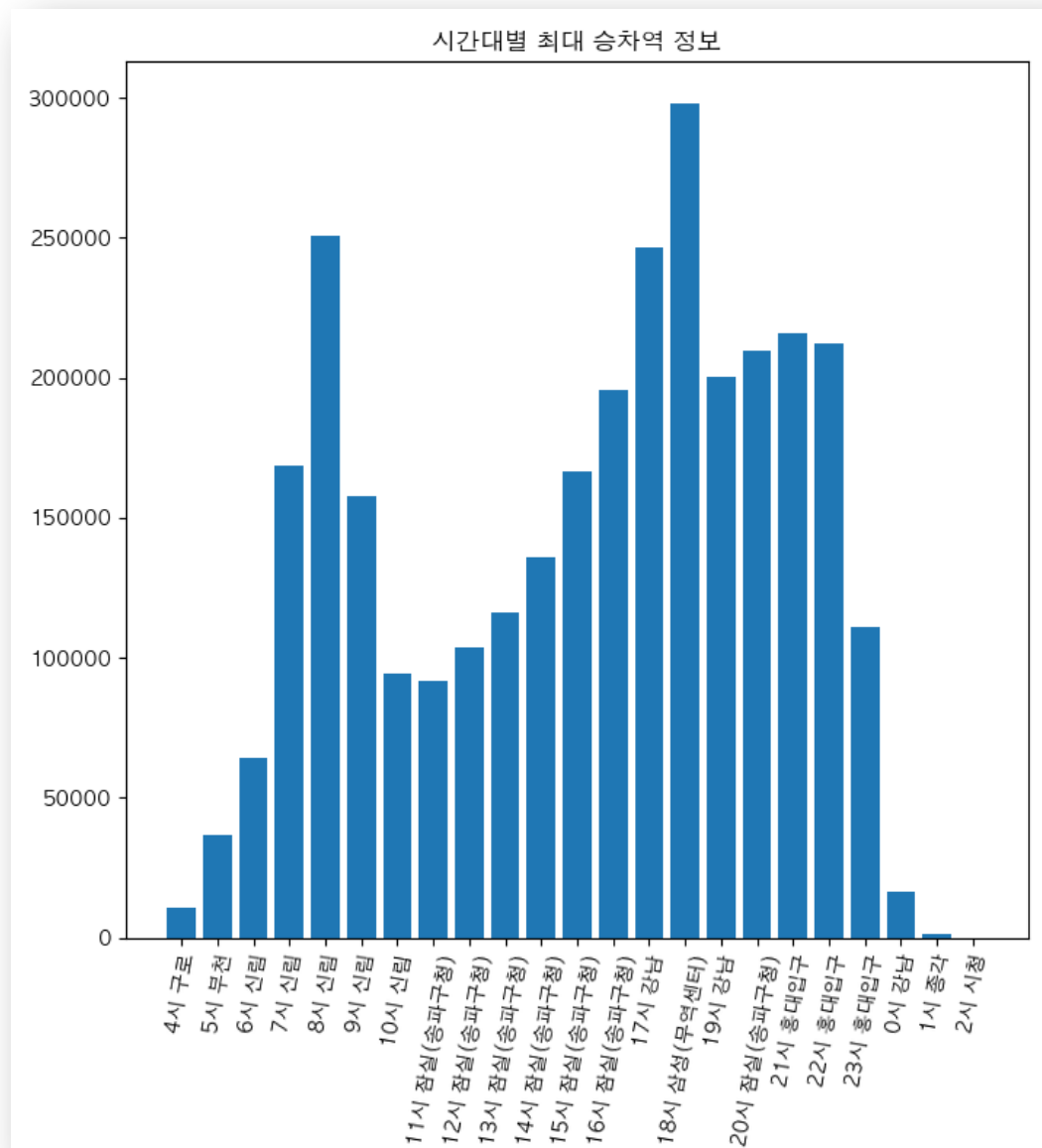
suptitle():
subplot들의 전체 부모
타이틀

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	...	[50]	[51]
사용 월	호선 명	역ID	역이 름	04:00~04:59:59		05:00~05:59:59		06:00~06:59:59		...	03:00~03:59:59	
				승차	하차	승차	하차	승차	하차		승차	하차
21.Apr	1호선	1	서울역	746	16	9151	6038	11560	33958		0	0

시간대별 가장 많이 승차하는 역 정보 분석

- 시간대: 새벽 4시 ~ 다음날 새벽2시
 - 총 23개의 데이터
 - 새벽 3시는 지하철 운행 안함

[4시 구로]: 10,986
[5시 부천]: 36,908
[6시 신림]: 64,151
[7시 신림]: 168,672
[8시 신림]: 250,796
[9시 신림]: 157,526
[10시 신림]: 94,494
[11시 잠실(송파구청)]: 92,035
[12시 잠실(송파구청)]: 103,606
[13시 잠실(송파구청)]: 116,176
[14시 잠실(송파구청)]: 135,787
[15시 잠실(송파구청)]: 166,370
[16시 잠실(송파구청)]: 195,705
[17시 강남]: 246,610
[18시 삼성(무역센터)]: 298,236
[19시 강남]: 200,483
[20시 잠실(송파구청)]: 209,945
[21시 홍대입구]: 216,017
[22시 홍대입구]: 212,376
[23시 홍대입구]: 110,986
[0시 강남]: 16,352
[1시 종각]: 1,638
[2시 시청]: 13



시간대별 가장 많이 승차하는 역 정보 분석 #1

```
import csv
import matplotlib.pyplot as plt
import platform
import koreanize_matplotlib
```

<day2_subwaytime_04.py>

```
with open('subwaytime.csv') as f:
```

```
    data = csv.reader(f)
```

```
    next(data)
```

```
    next(data)
```

```
    max = [0] * 23 # 새벽 3시는 지하철 운행 안함
```

```
    max_station = [''] * 23
```

```
    xtick_list = []
```

```
    for i in range(4, 27):
```

```
        n = i % 24
```

```
        xtick_list.append(str(n))
```

x축의 tick을 4, 5, 6, ... 23, 0, 1, 2시로
표시하기 위함 (24 % 24=0)

```
    for row in data:
```

```
        row[4:] = map(int, row[4:])
```

```
        for j in range(23):
```

```
            a = row[j * 2 + 4] # j=0: data[0 * 2 + 4]의 값을 max[0]에 저장하기 위함
```

```
            if a > max[j]:
```

```
                max[j] = a
```

```
            max_station[j] = xtick_list[j] + '사:' + row[3] # 4사: 구로
```

```
    for i in range(len(max)):
```

```
        print(f'[{max_station[i]}]: {max[i]:,}')
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	...	[50]	[51]
사용 월	호선 명	역ID	역 이 름	04:00~04:59:59		05:00~05:59:59		06:00~06:59:59		...	03:00~03:59:59	
				승차	하차	승차	하차	승차	하차		승차	하차
21.Apr	1호선	1	서울역	746	16	9151	6038	11560	33958		0	0

시간대별 가장 많이 승차하는 역 정보 분석 #2

<day2_subwaytime_04.py>

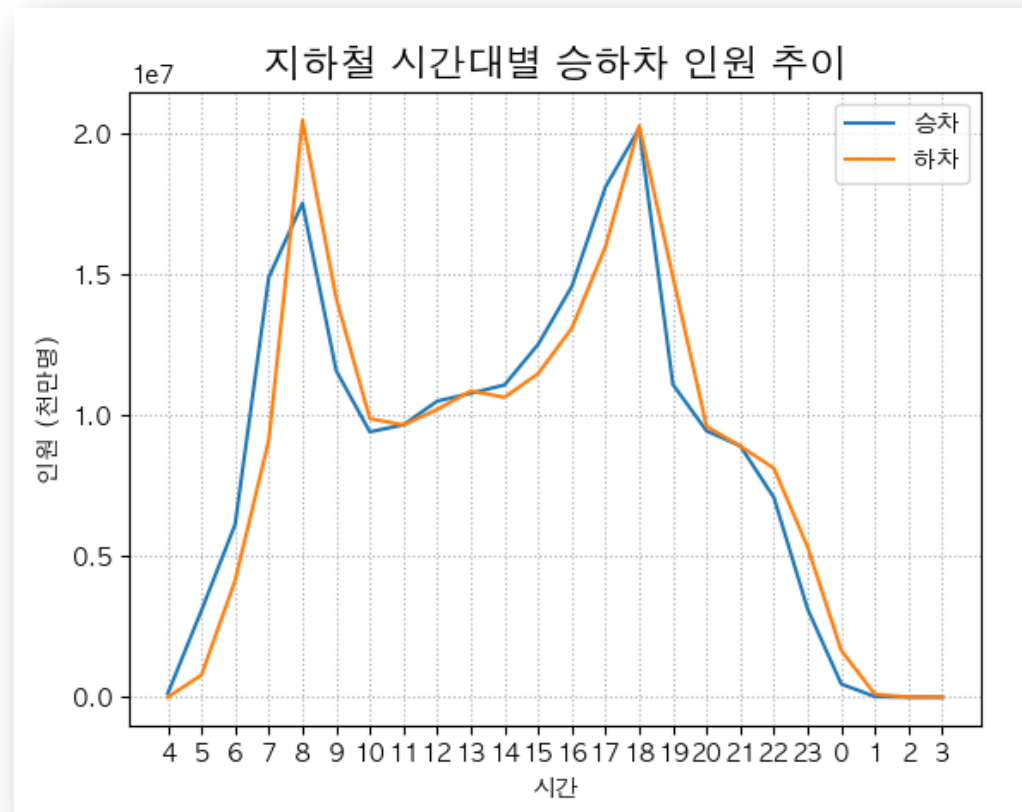
```
if(platform.system() == 'Windows'):
    plt.rc('font', family = 'Malgun Gothic')
else:
    plt.rc('font', family = 'AppleGothic')

plt.figure(figsize=(10, 10))
plt.title('시간대별 최대 승차역 정보')
plt.bar(range(23), max)
plt.xticks(range(23), labels=max_station, rotation=80)
plt.tight_layout()
plt.show()
```

xtick 문자열을
80도 회전

모든 지하철역에서 시간대별 승하차 인원 #1

- 시간대별 전체 지하철역의 승차, 하차 인원 분포
 - 전체 역의 시간대별 승차, 하차 인원 누적 계산
 - y축: $1e7$ (1×10^7), 천 만명 단위



모든 지하철역에서 시간대별 승하차 인원 #2

<day2_subwaytime_05.py>

```
import csv
import matplotlib.pyplot as plt
import koreanize_matplotlib

with open('subwaytime.csv') as f:
    data = csv.reader(f)
    next(data)
    next(data)
    subway_in = [0] * 24 # 승차 인원 저장 리스트
    subway_out = [0] * 24 # 하차 인원 저장 리스트

    for row in data:
        row[4:] = map(int, row[4:])
        for i in range(24):
            subway_in[i] += row[4+i*2]
            subway_out[i] += row[5+i*2]

    if(platform.system() == 'Windows'):
        plt.rc('font', family='Malgun Gothic')
    else:
        plt.rc('font', family='AppleGothic')
```

```
xtick_list = []
for i in range(4, 28):
    n = i % 24
    xtick_list.append(str(n))

plt.figure(dpi=100)
plt.title('지하철 시간대별 승하차 인원 추이', size=16)
plt.grid(linestyle=':') # 그리드 라인 표시
plt.plot(subway_in, label='승차')
plt.plot(subway_out, label='하차')
plt.legend()

plt.xticks(range(24), labels=xtick_list)
plt.xlabel('시간')
plt.ylabel('인원 (천만명)')
plt.show()
```

lambda와 operator를 사용한 dictionary 정렬

- lambda를 사용한 정렬

```
import operator

names = {'Mary':10999, 'Sams':2111, 'Aimy':9778, 'Tom':20245, 'Michale':27115, 'Bob':5887, 'Kelly':7855}

# Key를 기준으로 정렬 (기본: 오름차순)
print("[lambda] dict 정렬: key 기준 오름차순")
res = sorted(names.items(), key=(lambda x: x[0]))
print(res)
print()

# Value를 기준으로 정렬, 내림차순: reverse=True
print("[lambda] dict 정렬: value 기준, 내림차순")
res = sorted(names.items(), key=(lambda x: x[1]), reverse=True)
print(res)
```

x[0]: key 기준

x[1]: value 기준

```
[lambda] dict 정렬: key 기준 오름차순
[('Aimy', 9778), ('Bob', 5887), ('Kelly', 7855), ('Mary', 10999), ('Michale', 27115), ('Sams', 2111), ('Tom', 20245)]

[lambda] dict 정렬: value 기준, 내림차순
[('Michale', 27115), ('Tom', 20245), ('Mary', 10999), ('Aimy', 9778), ('Kelly', 7855), ('Bob', 5887), ('Sams', 2111)]
```

lambda와 operator를 사용한 Dictionary 정렬

■ operator 모듈

- 파이썬 내장 연산자에 대한 많은 함수들을 포함
 - add(), lt(), le(), itemgetter(), attrgetter() 등

```
import operator

names = {'Mary':10999, 'Sams':2111, 'Aimy':9778, 'Tom':20245, 'Michale':27115, 'Bob':5887, 'Kelly':7855}

# key를 기준으로 정렬 (오름차순)
sorted_x = sorted(names.items(), key=operator.itemgetter(0))
print("[operator] dict정렬: key 기준, 오름차순")
print(sorted_x)

print()
# value를 기준으로 정렬 (내림차순)
sorted_x = sorted(names.items(), key=operator.itemgetter(1), reverse=True)
print("[operator] dict정렬: value 기준, 내림차순")
print(sorted_x)
```

```
[operator] dict 정렬: key 기준, 오름차순
[('Aimy', 9778), ('Bob', 5887), ('Kelly', 7855), ('Mary', 10999), ('Michale', 27115), ('Sams', 2111), ('Tom', 20245)]
```

```
[operator] dict 정렬: value 기준, 내림차순
[('Michale', 27115), ('Tom', 20245), ('Mary', 10999), ('Aimy', 9778), ('Kelly', 7855), ('Bob', 5887), ('Sams', 2111)]
```

Excel 파일 열기

- xlrd 라이브러리 설치
 - Excel 파일 포맷(xls 또는xlsx)을 읽기 위한 패키지

```
conda install xlrd
```

```
(base) changsu@Changsuui-MacBookAir ~ % conda install xlrd
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/changsu/opt/anaconda3

  added / updated specs:
    - xlrd

The following packages will be downloaded:



| package        | build      |        |
|----------------|------------|--------|
| openssl-3.0.10 | h1a28f6b_0 | 4.3 MB |
| Total:         |            | 4.3 MB |



The following NEW packages will be INSTALLED:

xlrd                pkgs/main/noarch::xlrd-2.0.1-pyhd3eb1b0_1
```



```
Last login: Tue Jan 23 10:26:52 on console
(base) changsu@Changsuui-MacBookAir ~ % conda list xlrd
# packages in environment at /Users/changsu/opt/anaconda3:
#
# Name          Version      Build    Channel
xlrd            2.0.1        pyhd3eb1b0_1

(base) changsu@Changsuui-MacBookAir ~ %
```

지하철 시간대별 이용 현황: 엑셀 파일 및 Pandas 활용 #1

■ Pandas에서 엑셀 파일 읽기

- `pd.read_excel('파일이름', sheet_name='엑셀시트이름', header=[0,1])`

■ 출퇴근 시간대 이용 현황

<day2_subwaytime_excel_01.py>

```
import pandas as pd
# 지하철 시간대별 이용현황
```

Multi-index 형태
- header=[0, 1] 추가

```
df = pd.read_excel('subway.xls', sheet_name='지하철 시간대별 이용현황', header=[0, 1])
df.head()
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	...	[50]	[51]
사용월	호선명	역ID	역이름	04:00~04:59:59		05:00~05:59:59		06:00~06:59:59		...	03:00~03:59:59	
				승차	하차	승차	하차	승차	하차		승차	하차
21.Apr	1호선	1	서울역	746	16	9151	6038	11560	33958		0	0

header[0]

header[1]

	사용월	호선명	역ID	지하철역	04:00:00~04:59:59	05:00:00~05:59:59	06:00:00~06:59:59	...	23:00:00~23:59:59	00:00:00~00:59:59				
	Unnamed: 0_level_1	Unnamed: 1_level_1	Unnamed: 2_level_1	Unnamed: 3_level_1	승차	하차	승차	하차	승차	하차	...	하차	승차	하차
0	2023-12	1호선	150	서울역	700	35	7,812	8,436	12,190	50,415	...	15,733	3,170	4,088
1	2023-12	1호선	151	시청	73	1	2,208	4,356	3,731	21,903	...	5,965	4,290	1,523
2	2023-12	1호선	152	종각	167	1	4,280	4,932	4,329	25,201	...	5,317	2,621	1,402
3	2023-12	1호선	153	종로3가	230	16	4,174	2,538	3,621	11,378	...	7,655	7,239	2,844

지하철 시간대별 이용 현황: 엑셀 파일 및 Pandas 활용 #2

- 모든 컬럼 내용 확인

```
df.columns
```

```
MultiIndex([(          '사용월', 'Unnamed: 0_level_1'),  
            (          '호선명', 'Unnamed: 1_level_1'),  
            (          '역ID', 'Unnamed: 2_level_1'),  
            (          '지하철역', 'Unnamed: 3_level_1'),  
            ('04:00:00~04:59:59', '승차'),  
            ('04:00:00~04:59:59', '하차'),  
            ('05:00:00~05:59:59', '승차'),  
            ('05:00:00~05:59:59', '하차')])
```

- 특정 컬럼 데이터 가져오기: 호선명
 - MultiIndex의 경우, 튜플 형식으로 접근
 - df[(‘첫 번째 행’, ‘두번째 행’)]

```
df[(‘호선명’, ‘Unnamed: 1_level_1’)]
```

```
0      1호선  
1      1호선  
2      1호선  
3      1호선  
4      1호선  
...  
610    신림선  
611    신림선  
620    신림선
```


지하철 시간대별 이용 현황: 엑셀 파일 및 Pandas 활용 #3

- 특정 컬럼 데이터 가져오기: 지하철역

```
df[('지하철역', 'Unnamed: 3_level_1')]
```

```
0    서울역
1    시청
2    종각
3    종로3가
4    종로5가
```

- DataFrame에서 여러 컬럼 선택
 - `iloc[row_index, col_index]` (`iloc`: integer location)
 - `iloc[:, [1, 3, 10, 12, 14]]` : 모든 행과 1, 3, 10, 12, 14 열 선택

```
commute_time_df = df.iloc[:, [1, 3, 10, 12, 14]]
commute_time_df.head()
```

	호선명	지하철역	07:00:00~07:59:59	08:00:00~08:59:59	09:00:00~09:59:59
	Unnamed: 1_level_1	Unnamed: 3_level_1	승차	승차	승차
0	1호선	서울역	37,075	68,020	67,218
1	1호선	시청	7,341	9,896	13,154
2	1호선	종각	6,459	10,114	12,834
3	1호선	종로3가	5,454	8,475	12,940
4	1호선	종로5가	5,753	9,099	13,585

지하철 시간대별 이용 현황: 엑셀 파일 및 Pandas 활용 #4

- 모든 컬럼의 데이터 타입 확인

```
commute_time_df.dtypes
```

호선명	Unnamed: 1_level_1	object
지하철역	Unnamed: 3_level_1	object
07:00:00~07:59:59	승차	object
08:00:00~08:59:59	승차	object
09:00:00~09:59:59	승차	object

문자열 형태

- 천 단위 콤마 제거
 - `apply(lambda x : x.replace(',', ''))`

```
commute_time_df[('07:00:00~07:59:59', '승차')] = commute_time_df[('07:00:00~07:59:59', '승차')].apply(lambda x : x.replace(',', ''))
commute_time_df[('08:00:00~08:59:59', '승차')] = commute_time_df[('08:00:00~08:59:59', '승차')].apply(lambda x : x.replace(',', ''))
commute_time_df[('09:00:00~09:59:59', '승차')] = commute_time_df[('09:00:00~09:59:59', '승차')].apply(lambda x : x.replace(',', ''))
commute_time_df
```

	호선명	지하철역	07:00:00~07:59:59	08:00:00~08:59:59	09:00:00~09:59:59
	Unnamed: 1_level_1	Unnamed: 3_level_1	승차	승차	승차
0	1호선	서울역	38148	66885	57091
1	1호선	시청	7195	9565	11529
2	1호선	종각	6369	10271	12959
3	1호선	종로3가	5176	8742	13883

지하철 시간대별 이용 현황: 엑셀 파일 및 Pandas 활용 #5

- 데이터 타입 변경: object에서 int64로 변경
 - `df.astype({'컬럼명' : '변경타입'})`

```
commute_time_df = commute_time_df.astype({'07:00:00~07:59:59', '승차': 'int64'})
commute_time_df = commute_time_df.astype({'08:00:00~08:59:59', '승차': 'int64'})
commute_time_df = commute_time_df.astype({'09:00:00~09:59:59', '승차': 'int64'})
commute_time_df.dtypes
```

호선명	Unnamed: 1_level_1	object
지하철역	Unnamed: 3_level_1	object
07:00:00~07:59:59	승차	int64
08:00:00~08:59:59	승차	int64
09:00:00~09:59:59	승차	int64

dtype: object

- 각 행(지하철 역)의 승차 인원 수 합 계산
 - 행(row)의 합: `df.sum(axis=1)`
 - 열(column)의 합: `df.sum(axis=0)`

```
row_sum_df = commute_time_df.sum(axis=1, numeric_only=True)
passenger_number_list = row_sum_df.to_list()
row_sum_df
```

```
0      172313
1       30391
2       29407
3       26869
4       28437
...
616     42803
617     14506
618     43910
619     96672
620     14947
Length: 621, dtype: int64
```

지하철 시간대별 이용 현황: 엑셀 파일 및 Pandas 활용 #6

- 최대값 및 최대값 인덱스 찾기
 - 최대 승차 수를 가지는 지하철 역 찾기
 - 최대값 계산: `df.max(axis=0)`
 - 최대값 인덱스: `df.idxmax()`

```
max_number = row_sum_df.max(axis=0) # 해당 열에서 최대값 찾기
max_number
```

576994

```
max_index = row_sum_df.idxmax()
max_line, max_station = df.iloc[max_index, [1, 3]] # 최대값의 [1]: 호선, [3]: 지하철역명
print('출근 시간대 최대 승차 인원역: {0} {1} {2:},}명'.format(max_line, max_station,
                                                             max_number))
```

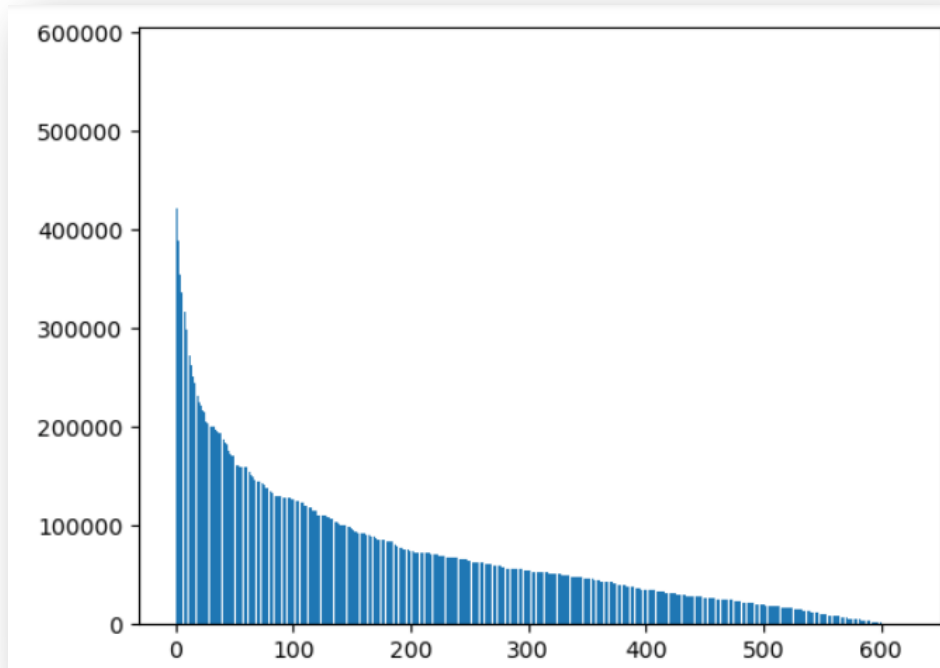
출근 시간대 최대 승차 인원역: 2호선 신림 576,994명

지하철 시간대별 이용 현황: 엑셀 파일 및 Pandas 활용 #7

- bar-chart 그리기

```
import matplotlib.pyplot as plt

passenger_number_list.sort(reverse=True)
plt.figure(dpi=100)
plt.bar(range(len(passenger_number_list)), passenger_number_list)
plt.show()
```





Questions?