

# 대중 교통 데이터

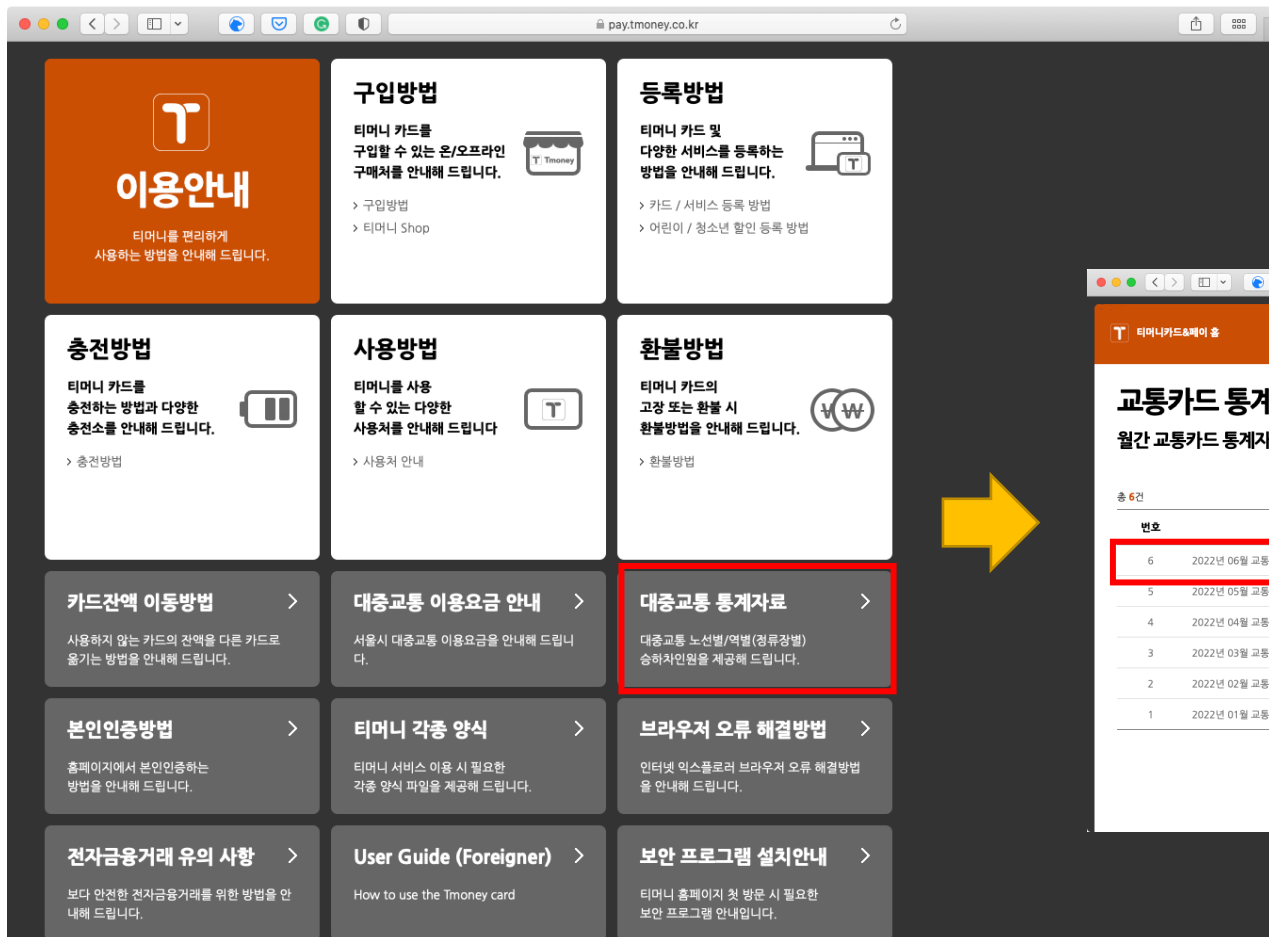
---

공공데이터 분석

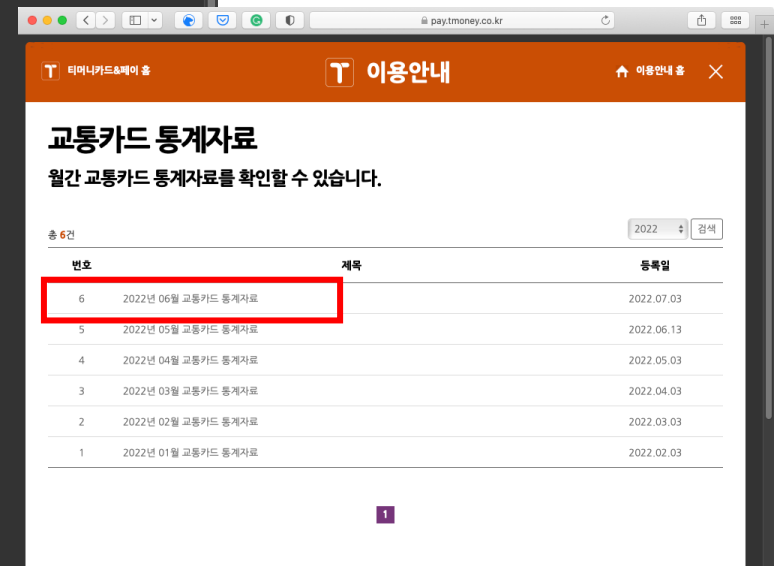
# 대중교통 데이터 내려받기

## ■ 대중교통 데이터: t-money

- <https://pay.tmoney.co.kr/ncs/pct/ugd/ReadUgdMainGd.dev>
- 이용안내 화면 > 대중교통 통계자료 선택 > 2022년 06월 교통카드 통계자료



subway.xls 로 저장



# 교통카드 통계자료

## ■ 내용: 4개의 탭으로 구성

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

## ■ csv 파일로 저장: subwayfee.csv

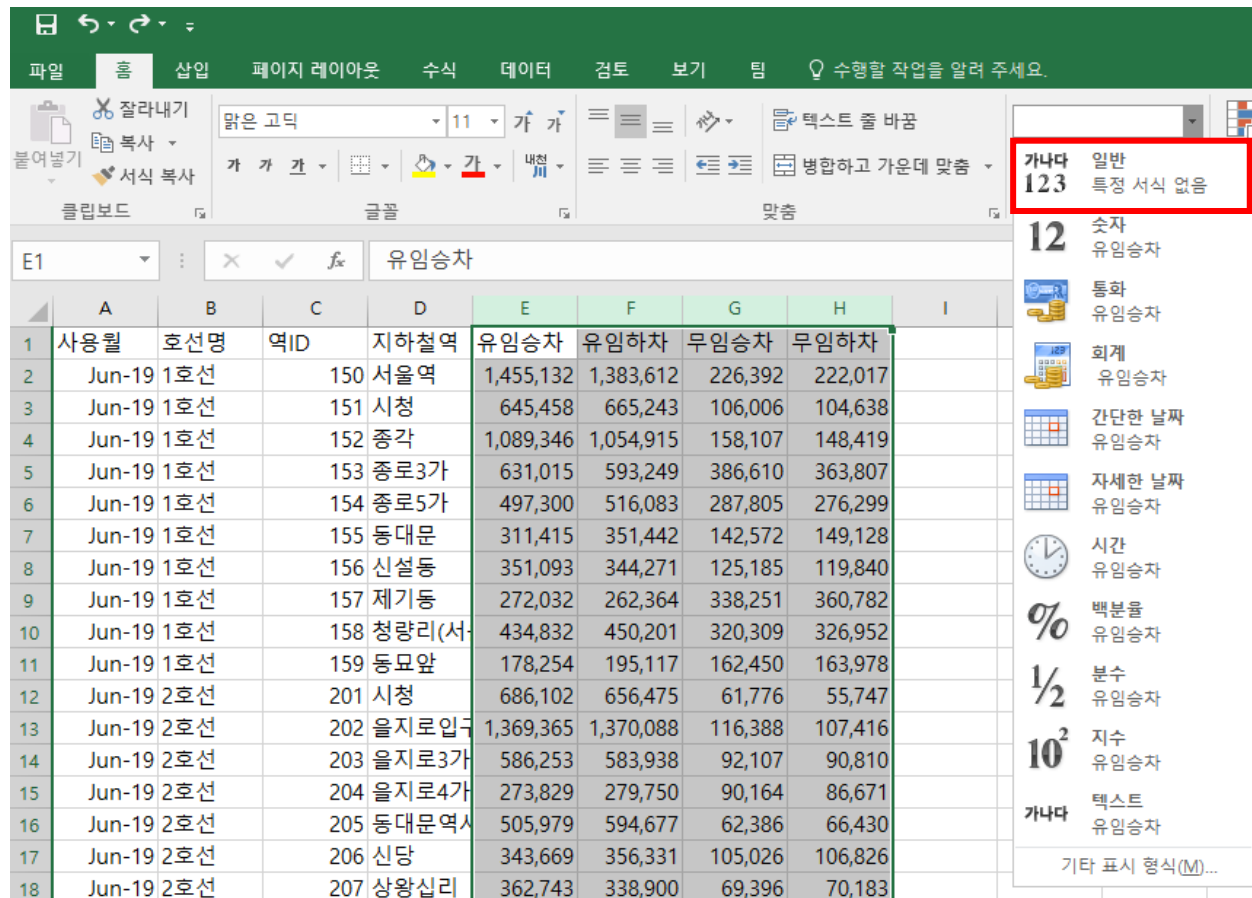
- 지하철 유무임별 이용현황 탭 선택 후 다른 이름으로 저장
- 파일 형식

파일 형식: CSV UTF-8(쉼표로 분리) (.csv)

지하철	유무임차	유임승차	유임하차	무임승차	무임하차	직원일시
2019-06 1호선 0151	서동	1,455,132	383,612	226,382	222,017	2019-07-03 12:03:28
2019-06 1호선 0152	중각	645,458	665,243	106,006	104,638	2019-07-03 12:03:28
2019-06 1호선 0153	중로3가	1,089,346	1,054,915	158,107	148,419	2019-07-03 12:03:28
2019-06 1호선 0154	중로5가	631,015	593,249	386,810	363,807	2019-07-03 12:03:28
2019-06 1호선 0155	충정로	497,300	516,083	287,805	276,299	2019-07-03 12:03:28
2019-06 1호선 0156	신정동	311,415	351,442	142,572	149,128	2019-07-03 12:03:28
2019-06 1호선 0157	계기동	351,093	344,271	125,185	119,840	2019-07-03 12:03:28
2019-06 1호선 0158	정왕리(서동)	272,032	262,364	338,251	360,782	2019-07-03 12:03:28
2019-06 1호선 0159	정왕리(서동)	434,832	450,201	320,309	326,952	2019-07-03 12:03:28
2019-06 2호선 0201	신정	178,254	195,117	162,450	163,978	2019-07-03 12:03:28
2019-06 2호선 0202	충정로	886,102	856,475	61,776	55,747	2019-07-03 12:03:28
2019-06 2호선 0203	충정로3가	1,369,365	1,370,088	116,368	107,416	2019-07-03 12:03:28
2019-06 2호선 0204	충정로4가	586,253	583,938	92,107	90,810	2019-07-03 12:03:28
2019-06 2호선 0205	충정로5가	273,829	279,750	90,164	86,671	2019-07-03 12:03:28
2019-06 2호선 0206	신정	505,970	594,677	62,386	66,430	2019-07-03 12:03:28
2019-06 2호선 0207	신정	343,669	356,331	105,026	106,826	2019-07-03 12:03:28
2019-06 2호선 0208	신정	362,743	338,900	69,396	70,183	2019-07-03 12:03:28
2019-06 2호선 0209	신정	533,116	432,024	61,682	51,527	2019-07-03 12:03:28
2019-06 2호선 0210	신정	325,829	371,702	15,741	16,953	2019-07-03 12:03:28
2019-06 2호선 0211	신정	500,962	533,742	64,803	65,088	2019-07-03 12:03:28
2019-06 2호선 0212	신정	730,281	804,360	108,181	109,528	2019-07-03 12:03:28
2019-06 2호선 0213	신정	123,400	134,701	100,909	105,058	2019-07-03 12:03:28
2019-06 2호선 0214	신정	606,392	576,245	111,048	111,700	2019-07-03 12:03:28
2019-06 2호선 0215	신정	1,194,678	1,180,114	150,896	147,023	2019-07-03 12:03:28
2019-06 2호선 0216	신정	385,324	373,879	81,590	92,592	2019-07-03 12:03:28
2019-06 2호선 0217	신정	2,274,859	2,191,037	222,644	211,606	2019-07-03 12:03:28
2019-06 2호선 0218	신정	620,946	628,593	115,122	114,943	2019-07-03 12:03:28
2019-06 2호선 0219	신정	378,318	429,546	57,628	61,320	2019-07-03 12:03:28
2019-06 2호선 0220	신정	1,587,016	1,647,028	126,975	120,776	2019-07-03 12:03:28
2019-06 2호선 0221	신정	1,352,293	1,205,628	216,778	191,086	2019-07-03 12:03:28
2019-06 2호선 0222	신정	1,167,896	1,324,223	164,900	167,909	2019-07-03 12:03:28
2019-06 2호선 0223	신정	2,716,424	2,802,762	181,864	163,127	2019-07-03 12:03:28
2019-06 2호선 0224	신정	1,887,781	1,903,008	168,563	190,630	2019-07-03 12:03:28
2019-06 2호선 0225	신정	591,516	586,967	96,293	96,104	2019-07-03 12:03:28
2019-06 2호선 0226	신정	499,443	514,817	84,525	83,821	2019-07-03 12:03:28
2019-06 2호선 0227	신정	1,090,379	1,225,257	228,223	215,555	2019-07-03 12:03:28
2019-06 2호선 0228	신정	776,344	737,304	117,701	119,407	2019-07-03 12:03:28
2019-06 2호선 0229	신정	1,370,334	1,326,684	206,549	198,878	2019-07-03 12:03:28
2019-06 2호선 0230	신정	591,394	531,900	165,397	165,873	2019-07-03 12:03:28
2019-06 2호선 0231	신정	1,849,900	1,782,312	254,521	258,325	2019-07-03 12:03:28
2019-06 2호선 0232	신정	728,841	688,917	140,041	141,006	2019-07-03 12:03:28

# CSV 파일 데이터 정리: 자리수 콤마 제거

- CSV 파일에서 숫자에 포함된 자리수 콤마(,) 제거
  - 컬럼 선택 후 > 일반 (특정 서식 없음) 선택 후 csv 파일로 저장
- 맨 오른쪽에 있는 작업일시 컬럼 제거



The screenshot shows the Microsoft Excel interface. The 'Format Cells' dialog box is open, and the 'General' tab is selected, which is highlighted with a red rectangle. The dialog box shows the number '123' and the text '일반 (특정 서식 없음)'. The background spreadsheet shows a table with columns A through I. The data in the table is as follows:

	A	B	C	D	E	F	G	H	I
1	사용월	호선명	역ID	지하철역	유임승차	유임하차	무임승차	무임하차	
2	Jun-19	1호선	150	서울역	1,455,132	1,383,612	226,392	222,017	
3	Jun-19	1호선	151	시청	645,458	665,243	106,006	104,638	
4	Jun-19	1호선	152	종각	1,089,346	1,054,915	158,107	148,419	
5	Jun-19	1호선	153	종로3가	631,015	593,249	386,610	363,807	
6	Jun-19	1호선	154	종로5가	497,300	516,083	287,805	276,299	
7	Jun-19	1호선	155	동대문	311,415	351,442	142,572	149,128	
8	Jun-19	1호선	156	신설동	351,093	344,271	125,185	119,840	
9	Jun-19	1호선	157	제기동	272,032	262,364	338,251	360,782	
10	Jun-19	1호선	158	청량리(서)	434,832	450,201	320,309	326,952	
11	Jun-19	1호선	159	동묘앞	178,254	195,117	162,450	163,978	
12	Jun-19	2호선	201	시청	686,102	656,475	61,776	55,747	
13	Jun-19	2호선	202	을지로입구	1,369,365	1,370,088	116,388	107,416	
14	Jun-19	2호선	203	을지로3가	586,253	583,938	92,107	90,810	
15	Jun-19	2호선	204	을지로4가	273,829	279,750	90,164	86,671	
16	Jun-19	2호선	205	동대문역사	505,979	594,677	62,386	66,430	
17	Jun-19	2호선	206	신당	343,669	356,331	105,026	106,826	
18	Jun-19	2호선	207	상왕십리	362,743	338,900	69,396	70,183	

# 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 = 0
for row in data:
    print(row)
    if i > 5:
        break
    i += 1
f.close()
```

```
['사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']
['Jun.22', '1호선', '150', '서울역', '1150754', '1123193', '194717', '187155']
['Jun.22', '1호선', '151', '시청', '537067', '543569', '80671', '78279']
['Jun.22', '1호선', '152', '종각', '834173', '815704', '131987', '121717']
['Jun.22', '1호선', '153', '종로3가', '450317', '404106', '299617', '278151']
['Jun.22', '1호선', '154', '종로5가', '394420', '404105', '254236', '244555']
['Jun.22', '1호선', '155', '동대문', '212719', '198987', '117499', '118515']
['Jun.22', '1호선', '156', '신설동', '264825', '252339', '112362', '106855']
```

# csv 파일 \ufeff 제거

- 한글이 포함된 csv 파일(UTF-8 인코딩)을 읽을 때 발생
  - 바이트 순서 표시 코드: BOM(Byte Order Mark)
  - 일부 Windows 프로그램에서 UTF-8 파일 생성시 자동으로 BOM 추가
  - Linux나 Mac환경에서 문제 발생
- 해결책
  - encoding 방식 변경: 'utf-8-sig' 사용

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

```
['\ufeff사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']
```

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

```
['사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']
```

# 유임 승차 vs 무임 승차 비율이 가장 높은 역은?

## ■ 유임 승차 대 무임 승차 비율 (rate) 계산

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

```
import csv
f = open('subwayfee.csv')
data = csv.reader(f)
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 컬럼값을 정수로 변환
    rate = row[4] / row[6]
    if rate > max_rate:
        max_rate = rate
print(max_rate)
```

row[6]의 값이 0인  
역이 존재함

```
ZeroDivisionError                                Traceback (most recent call last)
<ipython-input-14-6068e259cc1a> in <module>
      9     for i in range(4, 8):
     10         row[i] = int(row[i]) # 4, 5, 6, 7 컬럼값을 정수로 변환
--> 11     rate = row[4] / row[6]
     12     if rate > max_rate:
     13         max_rate = rate
```

# 무임승차 인원이 0인 역 찾기 #1

```
import csv
f = open('subwayfee.csv', encoding='utf-8-sig')
data = csv.reader(f)
header = next(data)
max_rate = 0
rate = 0
i = 0
for row in data:
    for i in range(4, 8):
        row[i] = int(row[i]) # 4, 5, 6, 7 컬럼값을 정수로 변환
        rate = row[4] / (row[4] + row[6])

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

f.close()
```

```
['Jun.22', '일산선', '1949', '지축', 10, 0, 0, 0]
['Jun.22', '경의선', '1296', '계양', 10, 0, 0, 0]
['Jun.22', '경의선', '1297', '검암', 2, 0, 0, 0]
['Jun.22', '6호선', '2615', '연신내', 31, 0, 0, 0]
['Jun.22', '6호선', '2649', '신내', 4, 0, 0, 0]
['Jun.22', '7호선', '2753', '까치울', 1, 0, 0, 0]
['Jun.22', '7호선', '2758', '상동', 1, 0, 0, 0]
['Jun.22', '7호선', '2761', '부평구청', 1, 0, 0, 0]
```



# 최대 무임 승차 비율 확인

```
'''
    [0]      [1]      [2]      [3]      [4]      [5]      [6]      [7]
    ['사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']
'''
import csv

f = open('subwayfee.csv')
data = csv.reader(f)
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 컬럼값을 정수로 변환

    if row[6] != 0:
        rate = (row[6]*100) / (row[4]+row[6]) # 무임승차수 / (유임승차수+무임승차수)
        if rate > max_rate:
            max_rate = rate
            print(row, round(rate, 2), '%')
```

```
['Jun.22', '1호선', '150', '서울역', 1150754, 1123193, 194717, 187155] 14.47 %
['Jun.22', '1호선', '153', '종로3가', 450317, 404106, 299617, 278151] 39.95 %
['Jun.22', '1호선', '157', '제기동', 218782, 209109, 263173, 281985] 54.61 %
['Jun.22', '경원선', '1916', '소요산', 32659, 25998, 62990, 56806] 65.86 %
['Jun.22', '7호선', '2756', '신중동', 0, 0, 1, 0] 100.0 %
```

# 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

['Jun.22', '2호선', '209', '한양대', 269240, 295829, 12481, 13227]  
역이름: 한양대, 전체 인원: 281,721, 유임승차인원: 269,240, 유임승차 비율: 0.96

# 최대 유임 승차 인원이 있는 역은? #2

```
import csv
f = open('subwayfee.csv', encoding='utf-8-sig')
data = csv.reader(f)
next(data)
max_rate = 0
rate = 0
max_row = []
total_count = 0
max_total_num = 0

for row in data :
    for i in range(4,8) :
        row[i] = int(row[i]) # 문자열을 정수로 변환
    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(max_row)
print("역이름: {0}, 전체 인원: {1:,}, 유임승차인원: {2:,}, 유임승차 비율: {3:,} ".format(max_row[3], max_total_num, max_row[4], round(max_rate, 2)))

f.close()
```

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

# 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']  
['Jun.22', '1호선', '157', '제기동', 218782, '209109', 263173, '281985'] 0.45  
['Jun.22', '1호선', '159', '동묘앞', 122460, '131299', 146455, '147539'] 0.46  
['Jun.22', '경원선', '1916', '소요산', 32659, '25998', 62990, '56806'] 0.34  
['Jun.22', '중앙선', '1218', '원덕', 5854, '5749', 6959, '6739'] 0.46  
['Jun.22', '중앙선', '1219', '용문', 34168, '34477', 38467, '38202'] 0.47  
유임 승차 비율이 가장 낮은 역: 소요산, 전체 인원:95,649, 유임승차인원:32,659, 유임승차비율:0.34
```

# 실습: 유임 승차 비율이 50% 이하인 역 #1

```
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
rate = 0
min_row = []
total_count = 0
min_total_count = 0

for row in data:
    for i in [4,6]:
        row[i] = int(row[i])
        total_count = row[4] + row[6]
        if (row[6] != 0) and (total_count >= 10000): # 무임승차 인원이 없고, 총 승차인원이 1만명 이상
            rate = row[4] / total_count
            if rate <= 0.5:
                print(row, round(rate, 2))
                if rate < min_rate: # 유임 승차 비율이 가장 낮은 역 찾기
                    min_rate = rate
                    min_row = row
                    min_total_count = total_count
print('유임 승차 비율이 가장 낮은 역: {0}, 전체 인원:{1:}, 유임승차인원:{2:}, 유임승차비율:{3:}'.
      format(min_row[3], min_total_count, min_row[4], round(min_rate, 2)))

f.close()
```

유임승차, 무임 승차  
데이터만 가져옴

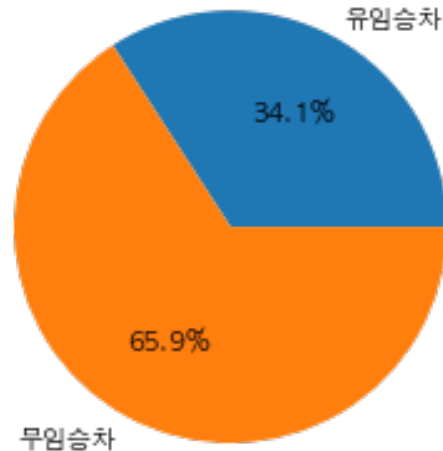
# 실습: 유임 승차 비율이 60% 이하인 역 #2

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

plt.title(min_row[3] + " 유,무임 승차 비율")
label = ['유임승차', '무임승차']
values = [min_row[4], min_row[6]]

plt.pie(values, labels=label, autopct='%.1f%%')
plt.show()
```

소요산 유,무임 승차 비율



# 승·하차 인원이 가장 많은 역은?

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

```
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

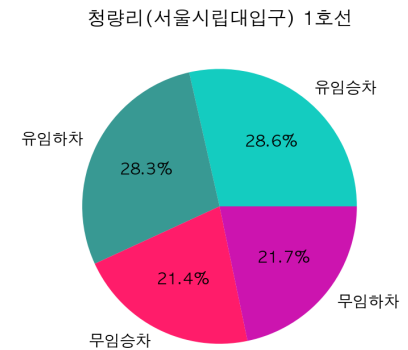
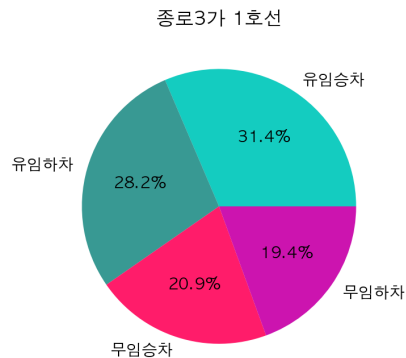
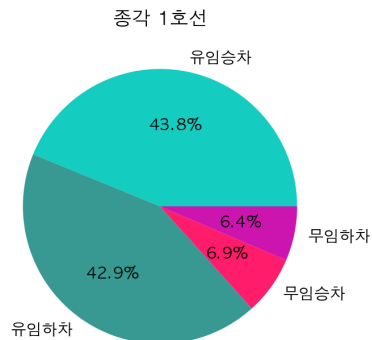
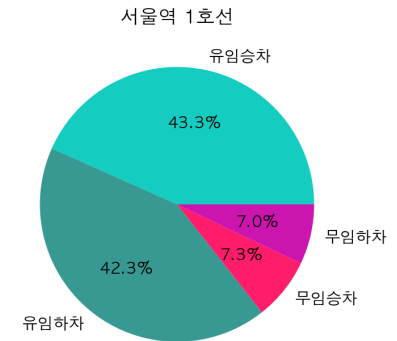
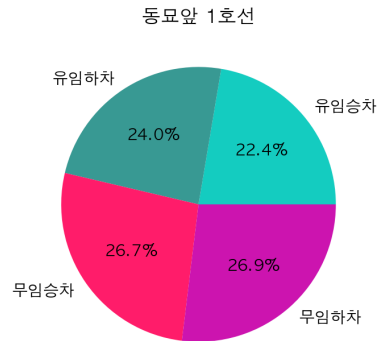
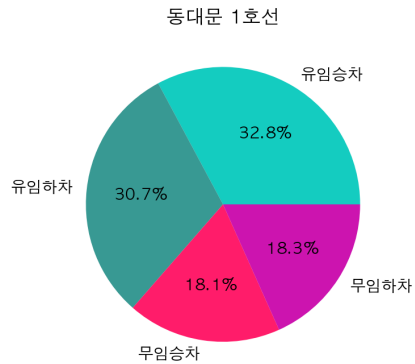
for i in range(4):
    print('{0}: {1} {2:,}'.format(label[i], max_station[i], max[i]))
```

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

유임승차: 강남 2호선 2,055,521  
유임하차: 강남 2호선 2,039,847  
무임승차: 종로3가 1호선 299,617  
무임하차: 제기동 1호선 281,985

# 전체 지하철역 승·하차 인원 분석 및 저장

- 파일 저장: `savefig('파일 이름', dpi)`
  - 총 615개 지하철역의 승·하차 정보가 파일로 저장됨





# 전체 지하철 역 파이차트 분석

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

label = ['유임승차', '유임하차', '무임승차', '무임하차']
c = ['#14CCC0', '#389993', '#FF1C6A', '#CC14AF'] # 파이차트 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])

            plt.figure(dpi=200) # 저장할 그림파일의 dpi 설정
            plt.title(row[3] + ' ' + row[1])
            plt.pie(row[4:8], labels=label, colors=c, autopct = '%.1f%%')
            plt.savefig(row[3] + ' ' + row[1] + '.png')
            plt.close() # 파일 닫기

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

4개 항목에 대한 파이 차트 작성

지하철역 이름 + 호선  
번호.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

파일 형식: CSV UTF-8(쉼표로 분리) (.csv)

- 데이터에 있는 1000자리 콤마를 제거
  - 데이터 속성: 일반 (특정 서식 없음)
- 마지막 '작업일시' 컬럼 제거

가나다 123	일반 특정 서식 없음
12	숫자 유일승차
	통화 유일승차
	회계 유일승차
	간단한 날짜 유일승차
	자세한 날짜 유일승차
	시간 유일승차
%	백분율 유일승차
1/2	분수 유일승차
10 <sup>2</sup>	지수 유일승차
가나다	텍스트 유일승차
기타 표시 형식(M)...	

# 지하철 시간대별 자료

## ■ 데이터 내용 (총 615개 지하철 역)

- 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	호선 명	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
21.Apr	1호선	1	서울역	746	16	9151	6038	11560	33958		0	0

	A	B	C	D	E	F	G	H	I	J	K	L
1	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	
2					승차	하차	승차	하차	승차	하차	승차	하차
3	22.Jun	1호선	150	서울역	646	24	8433	7694	12608	46963	39233	100397
4	22.Jun	1호선	151	시청	271	7	3133	4802	3321	23621	6647	62427
5	22.Jun	1호선	152	종각	91	9	4007	3960	3682	22533	5814	95376
6	22.Jun	1호선	153	종로3가	190	12	3882	3261	3570	14038	4746	25469
7	22.Jun	1호선	154	종로5가	34	0	1730	3909	2986	16406	5020	41304
8	22.Jun	1호선	155	동대문	890	31	11300	2056	8821	7698	14163	12675

# 데이터 정수 변환

---

## ■ map()함수 사용

- 리스트의 요소를 지정된 함수로 처리함
- 원본 리스트를 변경하지 않고 새 리스트를 생성함
- map(function, iterable)
  - 첫 번째 인자: 데이터에 적용할 함수 이름 입력
  - 두 번째 인자: 그 함수를 적용할 데이터 입력

```
def func(x):  
    return x**2  
  
a = [1, 2, 3, 4]  
a = list(map(func, a)) # 각 숫자의 제곱  
print(a)  
  
data = ['1', '2', '3', '4']  
data = list(map(int, data)) # 문자를 정수로 변환  
print(data)
```

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

# 시간대별 지하철 이용 인원 수

## ■ 새벽 4시 지하철 승차 전체 인원

[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

```
import csv

result = []
total_number = 0

with open('subwaytime.csv') as f:
    data = csv.reader(f)
    next(data) #2줄의 헤더 정보를 건너뛴다
    next(data)

    for row in data:
        row[4:] = map(int, row[4:]) # 문자열을 숫자로 변경(천단위 콤마 제거)
        total_number += row[4]
        result.append(row[4])

print('총 지하철 역의 수:', len(result))
print('새벽 4시 승차인원: {:,}'.format(total_number))
```

row[4:]: 인덱스 4부터 끝까지

총 지하철 역의 수: 615  
새벽 4시 승차인원: 135,398

# 새벽4시 지하철 이용 인원 수 (그래프)

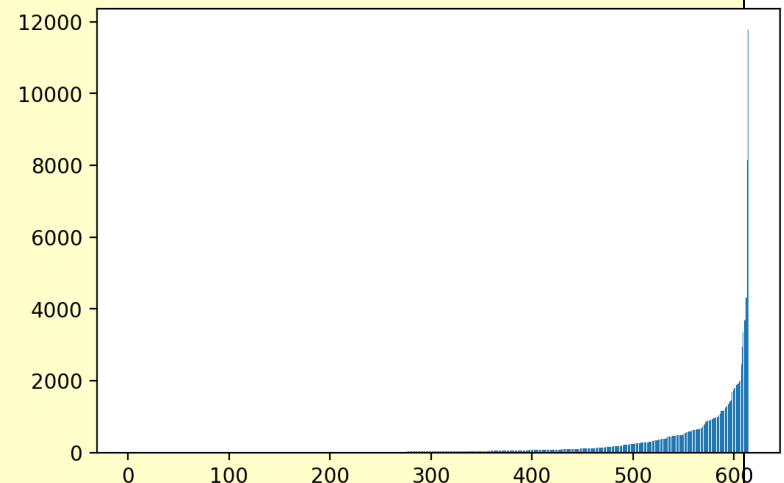
```
import csv
import matplotlib.pyplot as plt

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

새벽 4시 승차 인원수: 135,398  
최대 승차역: 구로, 인원수:11,784

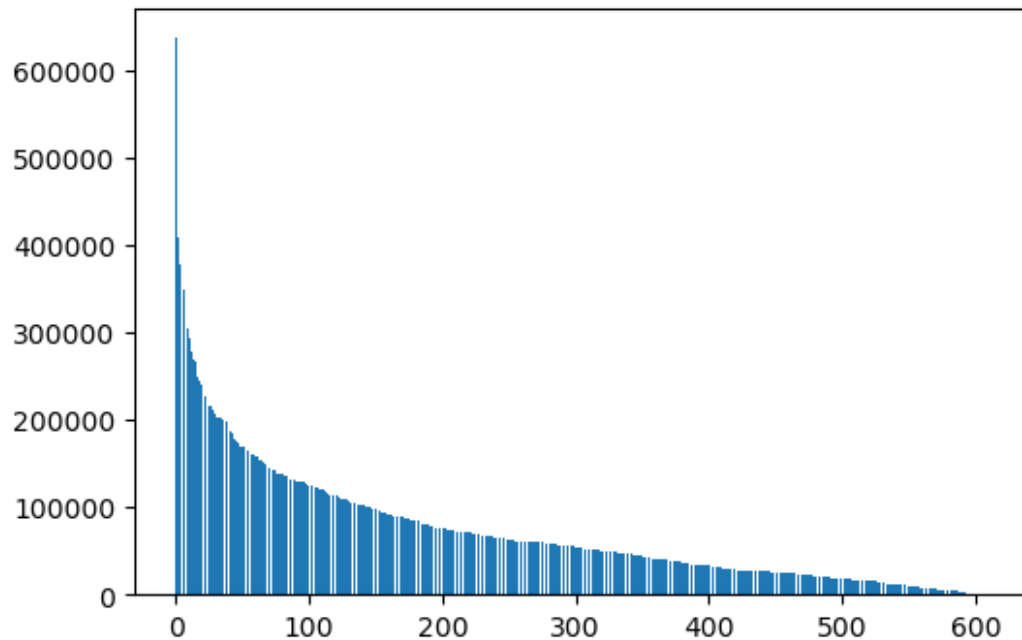


```
result.sort() # 오름 차순으로 정렬, 내림차순: result.sort(reverse=True)
plt.figure(dpi=100)
plt.bar(range(len(result)), result)
```

# 실습: 출근 시간대 지하철 이용 현황 #1

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

최대 승차 인원역: 신림(2호선) 638,753



# 실습: 출근 시간대 지하철 이용 현황 #2

```
import csv
import matplotlib.pyplot as plt

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

print('최대 승차 인원역: {0} {1:,}'.format(max_station, max_num))
result.sort(reverse=True)
plt.figure(dpi=100)
plt.bar(range(len(result)), result)
plt.show()
```

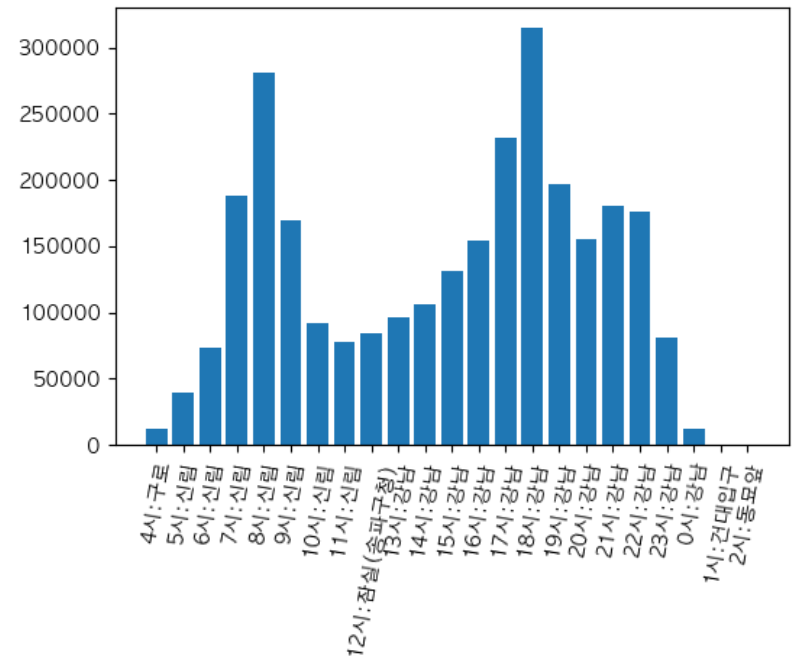
row[10], [12], [14]:  
오전 7시, 8시, 9시



# 시간대별 가장 많이 승차하는 역 정보 분석

## ■ 시간대: 새벽 4시 ~ 다음날 새벽2시

- 총 23개의 데이터
- 새벽 3시는 지하철 운행 안함



4시:구로: 11,784, 5시:신림: 38,809, 6시:신림: 73,523, 7시:신림: 188,110,  
8시:신림: 281,017, 9시:신림: 169,626, 10시:신림: 91,989, 11시:신림: 77,496,  
12시:잠실(송파구청): 84,383, 13시:강남: 95,697, 14시:강남: 106,394, 15시:강남: 131,625,  
16시:강남: 154,526, 17시:강남: 231,839, 18시:강남: 314,775, 19시:강남: 196,425,  
20시:강남: 155,347, 21시:강남: 180,108, 22시:강남: 176,473, 23시:강남: 81,016,  
0시:강남: 12,407, 1시:건대입구: 8, 2시:동묘앞: 1,

# 시간대별 가장 많이 승차하는 역 정보 분석

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

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 # 4, 5, 6, ... 23, 0, 1, 2시로 표시
        xtick_list.append(str(n))

    for row in data:
        row[4:] = map(int, row[4:])
        for j in range(23):
            a = row[j * 2 + 4] # j=0: data[j*2+4]의 값을 max[0]에 저장하기 위함
            if a > max[j]:
                max[j] = a
                max_station[j] = xtick_list[j] + '시:' + row[3] # 4시: 구로

    for i in range(len(max)):
        print('{0}: {1:,}'.format(max_station[i], max[i]), end=', ')
        if (i+1) % 4 == 0: # 한 줄에 6개씩 출력
            print()
```

# 시간대별 가장 많이 승차하는 역 정보 분석

---

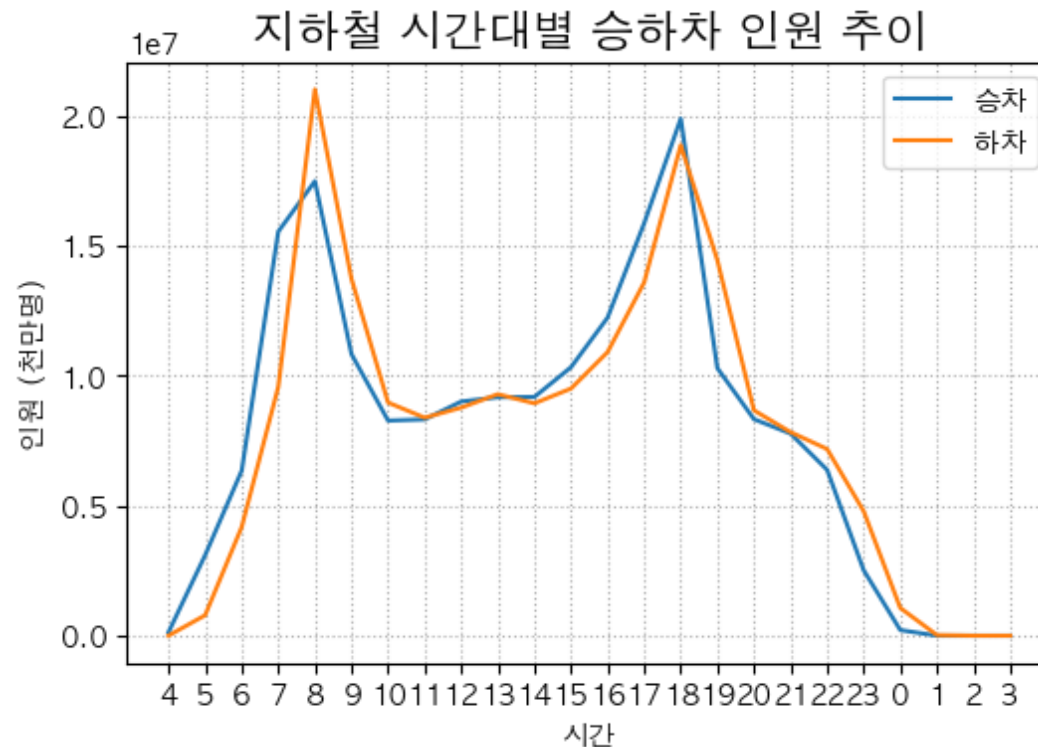
```
if(platform.system() == 'Windows'):
    plt.rc('font', family = 'Malgun Gothic')
else:
    plt.rc('font', family = 'AppleGothic')

plt.figure(dpi=100)
plt.bar(range(23), max)
plt.xticks(range(23), max_station, rotation=80)
plt.show()
```

xtick 문자열을  
80도 회전

# 모든 지하철역에서 시간대별 승하차 인원?

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



# 소스 코드

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

f = open('subwaytime.csv')
data = csv.reader(f)
next(data)
next(data)
s_in = [0] * 24 # 승차 인원 저장 리스트
s_out = [0] * 24 # 하차 인원 저장 리스트

for row in data:
    row[4:] = map(int, row[4:])
    for i in range(24):
        s_in[i] += row[4+i*2]
        s_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))
print(xtick_list)

plt.figure(dpi=100)
plt.title('지하철 시간대별 승하차 인원 추이', size=16)
plt.grid(linestyle=':') # 그리드 라인 표시

plt.plot(s_in, label='승차')
plt.plot(s_out, label='하차')

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

# Dictionary 정렬

- lambda 사용
- operator 모듈 사용

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

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

lambda 사용

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

```
print()
# key를 기준으로 정렬 (오름차순)
sorted_x = sorted(names.items(), key=operator.itemgetter(0))
print(sorted_x)
```

operator 모듈  
사용

```
print()
# value를 기준으로 정렬 (내림차순)
sorted_x = sorted(names.items(), key=operator.itemgetter(1), reverse=True)
print(sorted_x)
```

# Pandas 활용: Jupyter Notebook #1

## ■ 출퇴근 시간대 이용 현황

[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

```
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
```

header[0]

header[1]

	사용월	호선명	역ID	지하철역	:
	Unnamed: 0_level_1	Unnamed: 1_level_1	Unnamed: 2_level_1	Unnamed: 3_	
0	2022-06	1호선		150	
1	2022-06	1호선		151	
2	2022-06	1호선		152	
3	2022-06	1호선		153	

## • Pandas에서 엑셀 파일 읽기

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

# Pandas 활용 (Jupyter Notebook)

- 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’, ‘두 번째 행’)]`

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

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



# Pandas 활용: Jupyter Notebook #2

- 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 Unnamed: 1_level_1	지하철역 Unnamed: 3_level_1	07:00:00~07:59:59 승차	08:00:00~08:59:59 승차	09:00:00~09:59:59 승차
0	1호선	서울역	39,233	65,106	54,833
1	1호선	시청	6,647	8,294	9,211
2	1호선	종각	5,814	9,612	11,847
3	1호선	종로3가	4,746	8,589	13,179
4	1호선	종로5가	5,020	8,550	11,925

# Pandas 활용: Jupyter Notebook #3

- 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 Unnamed: 1_level_1	지하철역 Unnamed: 3_level_1	07:00:00~07:59:59 승차	08:00:00~08:59:59 승차	09:00:00~09:59:59 승차
0	1호선	서울역	39233	65106	54833
1	1호선	시청	6647	8294	9211
2	1호선	종각	5814	9612	11847

# Pandas 활용: Jupyter Notebook #4

- 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

정수형으로 변경

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

```
row_sum_df = commute_time_df.sum(axis=1)
passenger_number_list = row_sum_df.to_list() # DataFrame을 리스트로 변환
```

	data
0	159172
1	24152
2	27273
3	26514

# Pandas 활용: Jupyter Notebook #5

---

- 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
```

638753

```
max_index = row_sum_df.idxmax()
max_line, max_station = df.iloc[max_index, [1, 3]] # [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 638,753명

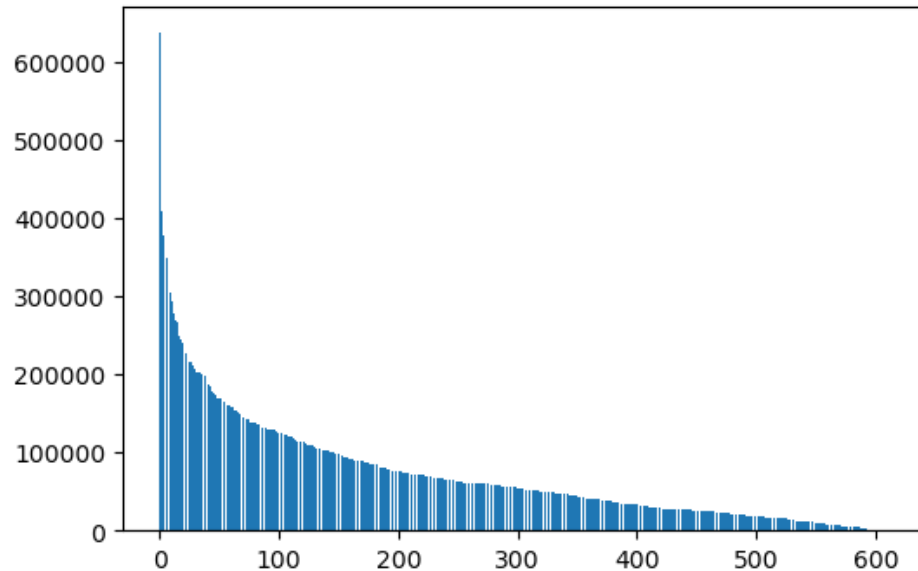
# Pandas 활용: Jupyter Notebook #6

---

- 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?