

데이터분석입문

Lecture 10. 대중교통 데이터 분석하기

동양미래대학교
인공지능소프트웨어학과
강 환수

- ❖ 01. 대중교통 데이터 시각화하기
- ❖ 02. 지하철 시간대별 데이터 시각화하기

01. 대중교통 데이터 시각화하기

02. 지하철 시간대별 데이터 시각화하기

❖ ① 지하철 유무임별 이용현황 데이터 정제하기 (1/2)

- subwayfee.csv 파일을 읽고, 데이터 출력해보기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

for row in data:
    print(row)

f.close()
```

헤더가 존재하고 있습니다.

```
['사용월', '호선명', '역 ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']
['2021-11', '1호선', '1', '서울역', '1,086,374', '1,044,750', '180,495', '170,842']
['2021-11', '1호선', '10', '동묘앞', '119,396', '128,531', '137,406', '138,097']
['2021-11', '1호선', '2', '시청', '516,353', '528,021', '72,373', '72,182']
['2021-11', '1호선', '3', '종각', '797,960', '781,329', '118,815', '110,536']
['2021-11', '1호선', '4', '종로3가', '433,793', '397,349', '272,242', '254,421']
['2021-11', '1호선', '5', '종로5가', '362,377', '370,057', '230,840', '220,847']
['2021-11', '1호선', '6', '동대문', '210,503', '195,301', '112,597', '110,925']
['2021-11', '1호선', '7', '신설동', '271,006', '262,798', '108,130', '103,695']
['2021-11', '1호선', '8', '제기동', '222,630', '214,314', '247,332', '263,992']
['2021-11', '1호선', '9', '청량리(서울시립대입구)', '315,789', '317,666', '232,661', '237,208']
['2021-11', '1호선', '11', '시청', '547,821', '521,220', '40,250', '45,066']
```

인원수 데이터가 문자열 자료형으로 저장되어 있습니다.
그리고 쉼표(,)도 존재합니다.

❖ ① 지하철 유무임별 이용현황 데이터 정제하기 (2/2)

- 인원수 데이터를 정수 자료형으로 바꾸고, 데이터 출력하기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(', ', ''))

    print(row)

f.close()
```

['2021-11', '1호선', '1', '서울역', 1086374, 1044750, 180495, 170842]
['2021-11', '1호선', '10', '동묘앞', 119396, 128531, 137406, 138097]
['2021-11', '1호선', '2', '시청', 516353, 528021, 72373, 72182]
['2021-11', '1호선', '3', '종각', 797960, 781329, 118815, 110536]
['2021-11', '1호선', '4', '종로3가', 433793, 397349, 272242, 254421]
['2021-11', '1호선', '5', '종로5가', 362377, 370057, 230840, 220847]
['2021-11', '1호선', '6', '동대문', 210503, 195301, 112597, 110925]
['2021-11', '1호선', '7', '신설동', 271006, 262798, 108130, 103695]
['2021-11', '1호선', '8', '제기동', 222630, 214314, 247332, 263992]
['2021-11', '1호선', '9', '청량리(서울시립대입구)', 315789, 317666, 232661, 237208]
['2021-11', '2호선', '11', '시청', 547821, 531320, 40350, 450661]

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (1/13)

- 유임 승차 비율이 가장 높은 역은 어디일까요?

알고리즘(Algorithm)으로 생각하기

- ✓ Step 1) 데이터를 읽어온다.
- ✓ Step 2) 모든 역의 데이터를 바탕으로 각 역의 유임 승차 비율(Rate)을 계산한다.
- ✓ Step 3) 비율이 가장 높은 역을 찾는다.
- ✓ Step 4) 비율이 가장 높은 역이 어디인지, 그 비율이 얼마인지 출력한다.

- 유임(有賃): 값을 치름
- 무임(無賃): 값을 치르지 않음

$$\text{Rate} = \text{유임 승차 인원} / \text{무임 승차 인원}$$

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (2/13)

- 유임 승차 인원/무임 승차 인원 비율의 최대값 찾기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

rate = 0
max_rate = 0

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',', ''))

    rate = row[4] / row[6]
    if rate > max_rate:
        max_rate = rate

f.close()
print(max_rate)
```

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (3/13)

- 유임 승차 인원/무임 승차 인원 비율의 최대값 찾기(실행결과)

◆ 오류 메시지를 읽고, 오류 원인 생각해 보기

['사용월', '호선명', '역 ID', '지하철역', '유임 승차', '유임 하차', '무임 승차', '무임 하차']

```
-----  
ZeroDivisionError                                Traceback (most recent call last)  
~\AppData\Local\Temp\ipykernel_14536\2052947337.py in <module>  
    14         row[i] = int(row[i].replace(',', ''))  
    15  
--> 16         rate = row[4] / row[6]  
    17         if rate > max_rate:  
    18             max_rate = rate
```

ZeroDivisionError: division by zero

오류 문구를 확인해 보니, 0으로 값을 나누려고 했다고 하네요.
row[6]의 무임 승차 인원 데이터에 0값이 존재하는 것 같네요.

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (4/13)

- row[6]가 0인 데이터 출력하기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

rate = 0
max_rate = 0

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',',''))

    if row[6] == 0:
        print(row)

f.close()
```

row[6]가 0인 역이 존재하네요.

```
['사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']
['2022-08', '경원선', '1022', '창동', 10, 0, 0, 0]
['2022-08', '분당선', '1031', '복정', 13, 0, 0, 0]
['2022-08', '경의선', '1297', '검암', 2, 0, 0, 0]
['2022-08', '7호선', '2756', '신중동', 1, 0, 0, 0]
['2022-08', '7호선', '2758', '상동', 2, 0, 0, 0]
['2022-08', '7호선', '2759', '삼산체육관', 1, 0, 0, 0]
['2022-08', '7호선', '2760', '굴포천', 2, 0, 0, 0]
['2022-08', '7호선', '2761', '부평구청', 3, 0, 0, 0]
```

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (5/13)

- row[6]가 0인 역은 제외하고, 유임 승차 인원/무임 승차 인원 비율의 최대값 찾기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

rate = 0
max_rate = 0

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',', ''))

    if row[6] != 0:
        rate = row[4] / row[6]
        if rate > max_rate:
            max_rate = rate
            print(row, max_rate)

f.close()
```

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (6/13)

- row[6]가 0인 역은 제외하고, 유임 승차 인원/무임 승차 인원 비율의 최대값 찾기(실행결과)

```
['사용월', '호선명', '역 ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']  
['2022-08', '1호선', '0150', '서울역', 1167657, 1130901, 180626, 172177] 6.464501234595241  
['2022-08', '2호선', '0201', '시청', 568964, 547537, 51980, 47237] 10.945825317429781  
['2022-08', '2호선', '0202', '을지로입구', 1030364, 1059452, 91971, 85574] 11.203140120255298  
['2022-08', '2호선', '0209', '한양대', 177378, 188876, 11994, 12537] 14.788894447223612  
['2022-08', '2호선', '0239', '홍대입구', 1717050, 1851268, 90063, 86905] 19.064987841844044  
['2022-08', '6호선', '2615', '연신내', 52, 0, 2, 0] 26.0
```

max_rate 값의 소수점 아래 수가 너무 길어서
보기가 힘드네요.

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (7/13)

- 유임 승차 인원/무임 승차 인원 비율의 최대값을 소수 둘째 자리 까지만 표현하기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

rate = 0
max_rate = 0

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',', ''))

    if row[6] != 0:
        rate = row[4] / row[6]
        if rate > max_rate:
            max_rate = rate
        print(row, round(max_rate, 2))

f.close()
```

round(max_rate, 2)는 max_rate 값을 소수점 셋째 자리에서 반올림하여 둘째 자리까지 표현하는 명령어입니다.

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (8/13)

- 유임 승차 인원/무임 승차 인원 비율의 최대값을 소수 둘째 자리 까지만 표현하기(실행결과)

```
['사용월', '호선명', '역 ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']  
['2022-08', '1호선', '0150', '서울역', 1167657, 1130901, 180626, 172177] 6.46  
['2022-08', '2호선', '0201', '시청', 568964, 547537, 51980, 47237] 10.95  
['2022-08', '2호선', '0202', '을지로입구', 1030364, 1059452, 91971, 85574] 11.2  
['2022-08', '2호선', '0209', '한양대', 177378, 188876, 11994, 12537] 14.79  
['2022-08', '2호선', '0239', '홍대입구', 1717050, 1851268, 90063, 86905] 19.06  
['2022-08', '6호선', '2615', '연신내', 52, 0, 2, 0] 26.0
```

비율의 최대값은 26.0으로 해당 역은 "연신내"역입니다.
그런데 "연신내"역의 정보가 이상한 것 같습니다.

데이터에 따르면, 유임승차 인원수가 52명이고
유임하차, 무임하차 인원수각 0명이네요.

이런 "연신내"역을 유임 승차 비율이
가장 높은 역이라고 부르는 것이 적절할까요?

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (9/13)

- 새로운 기준으로 유임 승차 비율 최대값 찾기

◆ 기존 유임 승차 비율(Rate) 계산 방법

$$\text{Rate} = \text{유임 승차 인원} / \text{무임 승차 인원}$$

◆ 새로운 유임 승차 비율(Rate) 계산 방법

- 유임 + 무임승차 인원을 합해서 100,000명 이상인 경우로만 한정

$$\text{Rate} = \text{유임 승차 인원} / \text{전체 (유임 + 무임승차) 인원}$$

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (10/13)

- 새로운 기준으로 유임 승차 비율 최대값 찾기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

rate = 0
max_rate = 0

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',', ''))

    if (row[6] != 0) and (row[4] + row[6] > 100000):
        rate = row[4] / (row[4] + row[6])
        if rate > max_rate:
            max_rate = rate
            print(row, round(max_rate, 2))

f.close()
```

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (11/13)

● 새로운 기준으로 유임 승차 비율 최대값 찾기(실행결과)

```
['사용월', '호선명', '역ID', '지하철역', '유임승차', '유임하차', '무임승차', '무임하차']  
['2022-08', '1호선', '0150', '서울역', 1167657, 1130901, 180626, 172177] 0.87  
['2022-08', '2호선', '0201', '시청', 568964, 547537, 51980, 47237] 0.92  
['2022-08', '2호선', '0202', '을지로입구', 1030364, 1059452, 91971, 85574] 0.92  
['2022-08', '2호선', '0209', '한양대', 177378, 188876, 11994, 12537] 0.94  
['2022-08', '2호선', '0239', '홍대입구', 1717050, 1851268, 90063, 86905] 0.95  
['2022-08', '공항철도 1호선', '4203', '홍대입구', 334381, 344582, 17164, 16166] 0.95
```

“연신대”역이 제외되었습니다.
새로운 기준으로 유임 승차 비율 최대값을 찾은 결과
최대값은 0.95이고, 해당 역은 “홍대입구” 역입니다.

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (12/13)

- rate > 0.92로 수정 후, 코드 실행해 보기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

rate = 0
max_rate = 0

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',', ''))

    if (row[6] != 0) and (row[4] + row[6] > 100000):
        rate = row[4] / (row[4] + row[6])
        if rate > 0.92:
            max_rate = rate
            print(row, round(max_rate, 2))

f.close()
```

유임 승차 비율이 0.92 이상인
역을 살펴볼까요?

❖ ③ 유임 승차 비율이 가장 높은 역은 어디일까 (13/13)

- rate > 0.92로 수정 후, 코드 실행해 보기(실행결과)

```
['사용월', '호선명', '역 ID', '지하철역', '유임 승차', '유임 하차', '무임 승차', '무임 하차']  
['2022-08', '2호선', '0209', '한양대', 177378, 188876, 11994, 12537] 0.94  
['2022-08', '2호선', '0219', '삼성(무역센터)', 1360753, 1373810, 111605, 103224] 0.92  
['2022-08', '2호선', '0222', '강남', 2073808, 2018576, 146482, 127762] 0.93  
['2022-08', '2호선', '0238', '합정', 841083, 921865, 62256, 62839] 0.93  
['2022-08', '2호선', '0239', '홍대입구', 1717050, 1851268, 90063, 86905] 0.95  
['2022-08', '3호선', '0327', '신사', 728935, 769927, 62606, 61011] 0.92  
['2022-08', '경부선', '1001', '서울역', 246839, 70773, 13293, 2816] 0.95  
['2022-08', '경부선', '1702', '가산디지털단지', 410976, 487813, 32364, 35714] 0.93  
['2022-08', '분당선', '1848', '압구정로데오', 502688, 575961, 43710, 41839] 0.92  
['2022-08', '5호선', '2527', '여의도', 710781, 755045, 50386, 47769] 0.93  
['2022-08', '5호선', '2528', '여의나루', 333874, 381345, 28279, 27372] 0.92  
['2022-08', '6호선', '2624', '상수', 241040, 285596, 20003, 20534] 0.92  
['2022-08', '6호선', '2632', '한강진', 224575, 266059, 14949, 14544] 0.94  
['2022-08', '7호선', '2748', '가산디지털단지', 1011630, 1004540, 80508, 75856] 0.93  
['2022-08', '9호선', '4115', '여의도', 721152, 680330, 46281, 42069] 0.94  
['2022-08', '9호선', '4125', '신논현', 790971, 787147, 48569, 46859] 0.94  
['2022-08', '공항철도 1호선', '4203', '홍대입구', 334381, 344582, 17164, 16166] 0.95  
['2022-08', '공항철도 1호선', '4204', '디지털미디어시티', 338524, 313402, 24556, 23008] 0.93  
['2022-08', '공항철도 1호선', '4206', '마곡나루(서울식물원)', 216283, 206531, 12852, 12660] 0.94  
['2022-08', '공항철도 1호선', '4207', '김포공항', 337165, 182448, 22741, 14639] 0.94  
['2022-08', '공항철도 1호선', '4210', '청라국제도시', 178960, 168163, 12695, 12606] 0.93
```

재미있는 결과가 나왔습니다.

❖ ④ 유무임 승하차 인원이 가장 많은 역은 어디일까 (1/4)

● 이번에는 유무임 승하차 인원이 가장 많은 역이 어디인지 찾아보겠습니다.

◆ 유임 승차 인원이 가장 많은 역은?

◆ 유임 하차 인원이 가장 많은 역은?

◆ 무임 승차 인원이 가장 많은 역은?

◆ 무임 하차 인원이 가장 많은 역은?

알고리즘(Algorithm)으로 생각하기

- ✓ Step 1) 데이터를 읽어온다.
- ✓ Step 2) 모든 역의 데이터를 바탕으로 유임 승차, 유임 하차, 무임 승차, 무임 하차 인원이 가장 많은 역을 각각 찾는다.
- ✓ Step 3) 각각의 인원이 가장 많은 역을 출력한다.

❖ ④ 유무임 승하차 인원이 가장 많은 역은 어디일까 (2/4)

- 인원수를 저장할 리스트의 이름을 max_people이라고 하겠습니다.
- max_people 리스트를 초기화 하는 방법을 살펴보겠습니다.

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)

# 초기화하는 방법1
max_people = [0, 0, 0, 0]
print(max_people)

# 초기화하는 방법2
max_people = [0] * 4
print(max_people)

# 초기화하는 방법3
max_people = []
for i in range(4):
    max_people.append(0)

print(max_people)

f.close()
```

3가지 방법 모두
초기화 방법으로 사용할 수 있습니다.

실행결과

```
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]
```

❖ ④ 유무임 승하차 인원이 가장 많은 역은 어디일까 (3/4)

- 역이름을 저장할 리스트의 이름을 max_station이라고 하겠습니다.
- max_station 리스트를 초기화 하는 방법을 살펴보겠습니다.

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)

max_people = [0] * 4

# 초기화하는 방법1
max_station = ['', '', '', '']
print(max_station)

# 초기화하는 방법2
max_station = [''] * 4
print(max_station)

# 초기화하는 방법3
max_station = []
for i in range(4):
    max_station.append('')

print(max_station)

f.close()
```

3가지 방법 모두
초기화 방법으로 사용할 수 있습니다.

실행결과

```
['', '', '', '']
['', '', '', '']
['', '', '', '']
```

❖ ④ 유무임 승하차 인원이 가장 많은 역은 어디일까 (4/4)

- 유무임 승하차 인원이 가장 많은 역 찾기

```
import csv

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)

max_people = [0] * 4
max_station = [''] * 4
label = ['(유임승차)', '(유임하차)', '(무임승차)', '(무임하차)']

for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(', ', ''))
        if row[i] > max_people[i-4]:
            max_people[i-4] = row[i]
            max_station[i-4] = row[3] + ' ' + row[1]

f.close()

for i in range(4):
    print(label[i] + ' ' + max_station[i] + ':', max_people[i], '명')
```

실행결과

(유임승차) 강남 2호선: 2073808 명
(유임하차) 강남 2호선: 2018576 명
(무임승차) 종로3가 1호선: 277589 명
(무임하차) 제기동 1호선: 282283 명

❖ ⑤ 모든 역의 유무임 승하차 비율은 어떻게 될까 (1/4)

- 데이터가 있는 모든 역에 대한 유무임 승하차 비율을 표현해 보겠습니다.
- 역마다 4개의 정보가 표현될 테니 원그래프(Pie Chart)를 이용하겠습니다.
- 유무임 승하차 비율 원그래프로 시각화하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
label = ['유임승차', '유임하차', '무임승차', '무임하차']
color = ['skyblue', 'palegreen', 'pink', 'beige']

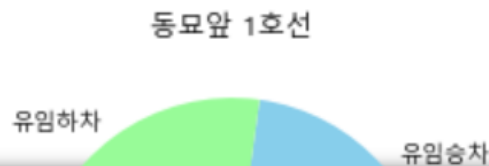
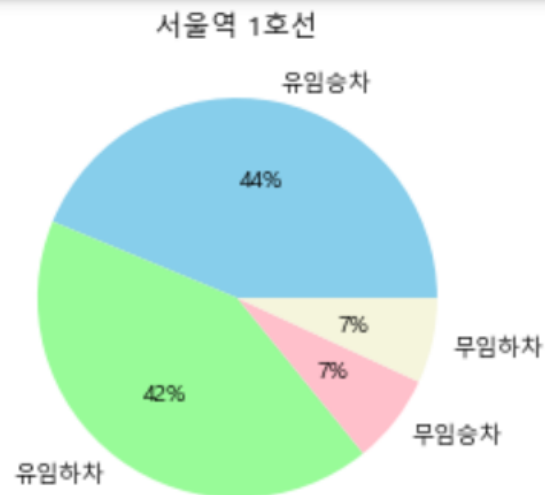
plt.rc('font', family='Malgun Gothic')
for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',', ''))

    plt.title(row[3] + ' ' + row[1])
    plt.pie(row[4:8], labels=label, colors=color, autopct='%1.1f%%')
    plt.show()

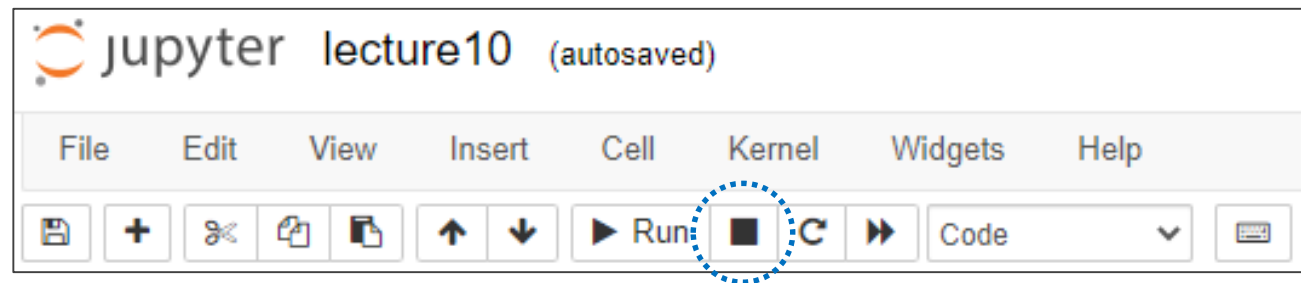
f.close()
```

❖ ⑤ 모든 역의 유무임 승하차 비율은 어떻게 될까 (2/4)

- 유무임 승하차 비율 원그래프로 시각화하기(실행결과)



모든 역에 대해서 수행되다 보니
엄청나게 많은 그래프가 생성됩니다.
그래프를 하나씩 저장하기에는 너무 번거롭습니다.



■ 버튼을 클릭하여 실행을 멈춥니다.

❖ ⑤ 모든 역의 유무임 승하차 비율은 어떻게 될까 (3/4)

- 각 그래프를 이미지 파일로 저장하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwayfee.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
label = ['유임 승차', '유임 하차', '무임 승차', '무임 하차']
color = ['skyblue', 'palegreen', 'pink', 'beige']

plt.rc('font', family='Malgun Gothic')
for row in data:
    for i in range(4, 8):
        row[i] = int(row[i].replace(',', ''))

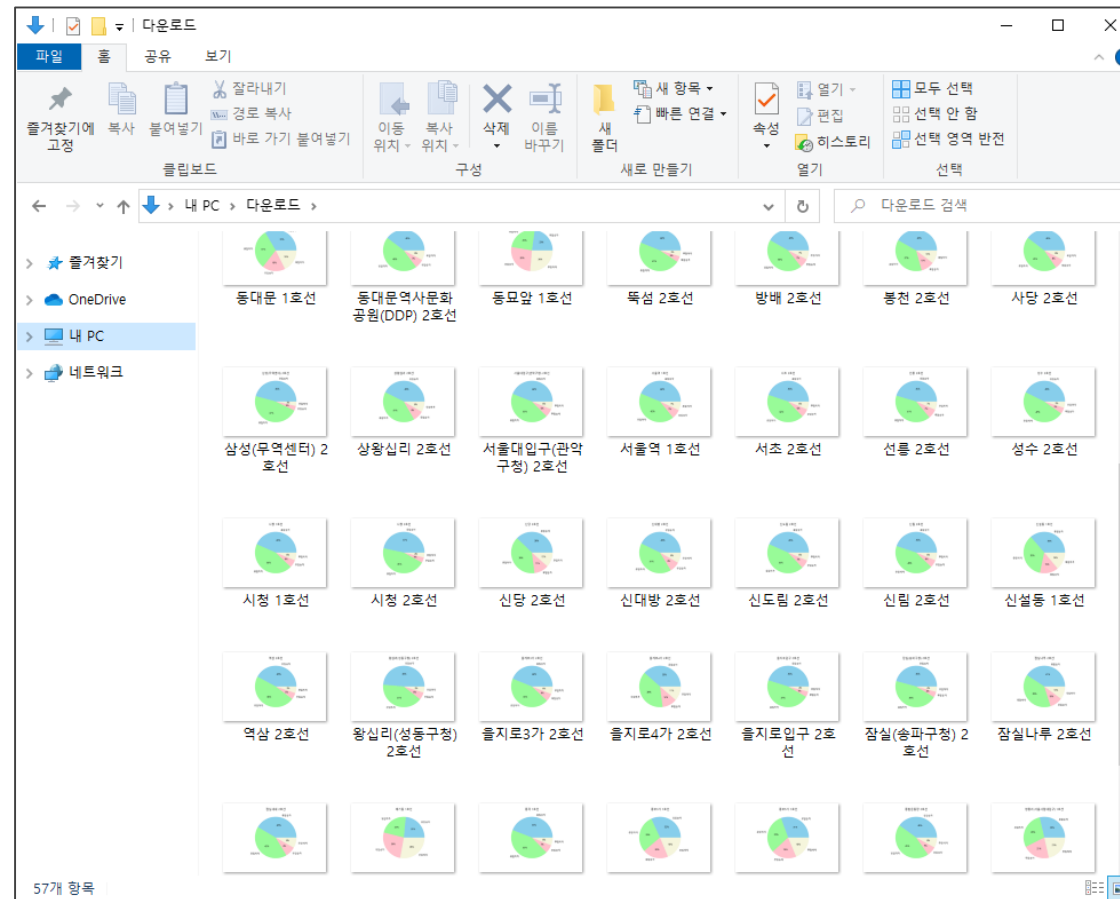
    plt.title(row[3] + ' ' + row[1])
    plt.pie(row[4:8], labels=label, colors=color, autopct='%1.f%%')
    plt.savefig(row[3] + ' ' + row[1] + '.png')
    plt.show()

f.close()
```

savefig 함수를 이용하면 그래프를 저장할 수 있습니다.
함수의 입력 값으로는 저장할 파일의 이름을 적어주면 됩니다.

❖ ⑤ 모든 역의 유무임 승하차 비율은 어떻게 될까 (4/4)

- 각 그래프를 이미지 파일로 저장하기(실행결과)



02. 지하철 시간대별 데이터 시각화하기

02. 지하철 시간대별 데이터 시각화하기

❖ ① 지하철 시간대별 이용 현황 데이터 정제하기 (1/3)

- subwaytime.csv 파일을 읽고, 데이터 출력해보기

```
import csv
```

```
f = open('subwaytime.csv', encoding='cp949')
```

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

```
for row in data:
    print(row)
```

```
f.close()
```

헤더가 2개의 행으로 이루어 졌습니다.

인원수 데이터가 문자열 자료형으로
저장되어 있습니다.
그리고 쉼표(,)도 존재합니다.

['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', '', '11:00:00~11:59:59', '', '12:00:00~12:59:59', '', '13:00:00~13:59:59', '', '14:00:00~14:59:59', '', '15:00:00~15:59:59', '', '16:00:00~16:59:59', '', '17:00:00~17:59:59', '', '18:00:00~18:59:59', '', '19:00:00~19:59:59', '', '20:00:00~20:59:59', '', '21:00:00~21:59:59', '', '22:00:00~22:59:59', '', '23:00:00~23:59:59', '', '00:00:00~00:59:59', '', '01:00:00~01:59:59', '', '02:00:00~02:59:59', '', '03:00:00~03:59:59', '']

[illegible]

['Nov-21', '1호선', '1', '서울역', '630', '11', '8,985', '7,058', '12,028', '40,803', '36,492', '93,181', '61,857', '196,998', '48,518', '131,911', '49,785', '67,104', '57,296', '57,503', '67,212', '64,828', '68,923', '62,027', '59,146', '58,165', '78,671', '61,463', '88,156', '65,725', '136,669', '75,650', '198,548', '82,448', '93,839', '51,787', '65,417', '34,209', '70,807', '29,311', '46,511', '21,788', '17,275', '12,360', '101', '1,257', '1', '3', '2', '2', '0', '0']

['Nov-21', '1호선', '10', '동묘앞', '141', '1', '2,570', '907', '3,387', '4,074', '5,711', '7,976', '9,492', '20,213', '7,707', '16,905', '8,586', '18,424', '12,140', '24,629', '18,157', '27,618', '24,695', '30,966', '29,042', '28,726', '32,042', '24,195', '34,642', '17,024', '29,570', '11,763', '17,697', '8,988', '7,597', '6,173', '4,948', '4,434', '4,545', '4,032', '3,070', '4,587', '1,055', '2,974', '8', '2,017', '0', '2', '0', '0', '0', '0']

[illegible]

❖ ① 지하철 시간대별 이용 현황 데이터 정제하기 (2/3)

- 쉼표(,) 없애기

```
import csv

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')
    print(row[4:])

f.close()
```

```
['630', '11', '8985', '7058', '12028', '40803', '36492', '93181', '61857', '196998', '48518', '131911', '49785', '67104', '57296',
'57503', '67212', '64828', '68923', '62027', '59146', '58165', '78671', '61463', '88156', '65725', '136669', '75650', '198548', '82
448', '93839', '51787', '65417', '34209', '70807', '29311', '46511', '21788', '17275', '12360', '101', '1257', '1', '3', '2', '2',
'0', '0']
['141', '1', '2570', '907', '3387', '4074', '5711', '7976', '9492', '20213', '7707', '16905', '8586', '18424', '12140', '24629', '1
8157', '27618', '24695', '30966', '29042', '28726', '32042', '24195', '34642', '17024', '29570', '11763', '17697', '8988', '7597',
'6173', '4948', '4434', '4545', '4032', '3070', '4587', '1055', '2974', '8', '2017', '0', '2', '0', '0', '0', '0']
['30', '0', '2006', '4859', '2980', '19785', '6504', '57521', '8275', '173717', '8576', '83121', '9894', '35008', '15611', '34889',
'17230', '30807', '20617', '27847', '28214', '25240', '37716', '21714', '44129', '20124', '75744', '20612', '142785', '20226', '513
05', '9055', '42181', '5305', '40637', '4545', '26627', '3323', '7600', '2196', '65', '309', '0', '0', '0', '0', '0', '0']
['118', '1', '2639', '3960', '3158', '20401', '5614', '89885', '9209', '225883', '10921', '131732', '15255', '53438', '23410', '488
```

❖ ① 지하철 시간대별 이용 현황 데이터 정제하기 (3/3)

- map() 함수를 이용하여 문자열 자료형의 데이터를 한꺼번에 정수형 자료형 데이터로 변환하기

```
import csv

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

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

f.close()
```

map 함수는 일괄적으로 데이터에 특정 함수를 적용할 수 있습니다.
→ map(일괄 적용할 함수 이름, 적용할 데이터)

```
[630, 11, 8985, 7058, 12028, 40803, 36492, 93181, 61857, 196998, 48518, 131911, 49785, 67104, 57296, 57503, 67212, 64828, 68923, 62
027, 59146, 58165, 78671, 61463, 88156, 65725, 136669, 75650, 198548, 82448, 93839, 51787, 65417, 34209, 70807, 29311, 46511, 2178
8, 17275, 12360, 101, 1257, 1, 3, 2, 2, 0, 0]
[141, 1, 2570, 907, 3387, 4074, 5711, 7976, 9492, 20213, 7707, 16905, 8586, 18424, 12140, 24629, 18157, 27618, 24695, 30966, 29042,
28726, 32042, 24195, 34642, 17024, 29570, 11763, 17697, 8988, 7597, 6173, 4948, 4434, 4545, 4032, 3070, 4587, 1055, 2974, 8, 2017,
0, 2, 0, 0, 0, 0]
[30, 0, 2006, 4859, 2980, 19785, 6504, 57521, 8275, 173717, 8576, 83121, 9894, 35008, 15611, 34889, 17230, 30807, 20617, 27847, 282
14, 25240, 37716, 21714, 44129, 20124, 75744, 20612, 142785, 20226, 51305, 9055, 42181, 5305, 40637, 4545, 26627, 3323, 7600, 2196,
65, 309, 0, 0, 0, 0, 0, 0]
[118, 1, 2620, 3060, 3158, 20401, 5614, 80885, 0200, 22588, 10021, 131732, 15255, 52438, 22410, 48848, 27245, 42754, 24266, 41537
```

02. 지하철 시간대별 데이터 시각화하기

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (1/11)

- 출근 시간대를 오전 7시라고 가정해 보겠습니다.

오전 7시 승차 데이터의 위치를 확인하니 10번 인덱스(Index)에 저장되어 있습니다.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	...
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
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		08:00:00~08:59:59	
2					승차	하차	승차	하차	승차	하차	승차	하차	승차	하차
3	Aug-22	1호선	150	서울역	573	19	8,638	8,274	12,332	45,706	39,560	102,779	63,523	200,999
4	Aug-22	1호선	151	시청	39	0	2,005	4,665	3,404	23,606	6,430	65,621	8,401	181,920
5	Aug-22	1호선	152	종각	54	4	3,356	4,382	3,765	22,971	5,801	98,968	9,571	243,599
6	Aug-22	1호선	153	종로3가	118	10	3,367	3,149	3,409	13,161	4,642	25,201	8,037	69,020
7	Aug-22	1호선	154	종로5가	38	2	1,632	3,635	2,766	15,329	5,251	40,866	8,560	93,100
8	Aug-22	1호선	155	동대문	561	16	9,859	1,842	8,375	6,305	13,390	11,046	17,632	20,315
9	Aug-22	1호선	156	신설동	309	22	8,586	2,260	8,758	9,028	18,458	22,614	26,047	54,554
10	Aug-22	1호선	157	제기동	357	4	5,001	2,038	8,276	8,838	21,335	19,703	31,333	40,232
11	Aug-22	1호선	158	청량리(서울)	915	17	10,286	4,451	15,174	21,761	34,968	17,224	44,626	34,255
12	Aug-22	1호선	159	동묘앞	145	1	2,799	1,039	3,456	4,571	5,920	8,160	10,055	17,264
13	Aug-22	2호선	201	시청	49	0	1,009	1,639	1,930	17,329	5,094	61,499	7,682	203,301
14	Aug-22	2호선	202	을지로입구	64	0	2,287	2,681	3,816	27,574	9,428	120,481	15,319	314,501
15	Aug-22	2호선	203	을지로3가	19	0	1,140	1,642	2,243	18,943	5,103	68,599	10,147	175,176
16	Aug-22	2호선	204	을지로4가	5	0	922	1,527	2,083	14,176	4,264	35,174	8,723	75,030
17	Aug-22	2호선	205	동대문역사	195	15	4,633	1,210	4,190	7,735	6,124	19,963	10,802	44,774
18	Aug-22	2호선	206	시당	18	0	5,746	1,188	10,743	8,016	26,628	16,735	43,540	31,151

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (2/11)

- 오전 7시 승차 데이터 개수 및 인원수 출력하기

```
import csv

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

result = []
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    result.append(row[10])    # 오전 7시 승차 데이터

f.close()

print(len(result))
print(result)
```


❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (3/11)

● 오전 7시 승차 데이터 개수 및 인원수 출력하기(실행결과)

617

[39560, 6430, 5801, 4642, 5251, 13390, 18458, 21335, 34968, 5920, 5094, 9428, 5103, 4264, 6124, 26628, 38761, 25807, 5181, 17677, 22533, 50866, 66999, 76037, 35469, 113682, 51500, 14920, 14026, 19738, 14694, 37772, 24831, 16398, 21777, 73756, 79730, 122277, 78413, 184073, 85469, 131269, 64768, 126331, 41387, 36682, 42911, 41829, 59887, 35765, 24155, 27299, 14870, 7971, 6552, 4176, 3304, 30755, 47631, 4976, 23154, 82016, 124350, 49220, 67128, 61010, 13149, 14171, 13217, 5722, 1723, 2133, 0, 5761, 24966, 24980, 21425, 15013, 10586, 10771, 23787, 8660, 27262, 32013, 15798, 8335, 22440, 3832, 25842, 20850, 32557, 18820, 21670, 24834, 36927, 84271, 48791, 82798, 118109, 109614, 44728, 90295, 83533, 44671, 29089, 17071, 4689, 6451, 8566, 5294, 4462, 11582, 16692, 6982, 11201, 12074, 1845, 45485, 65778, 2126, 9734, 12061, 20737, 16250, 24790, 38548, 9202, 23742, 36569, 14912, 43889, 36004, 31404, 82237, 25758, 84384, 18302, 26700, 64767, 40887, 67777, 44808, 14243, 48493, 9578, 11419, 22805, 2813, 12263, 14654, 9568, 25043, 5941, 1705, 14493, 9810, 25332, 9878, 11811, 86398, 48887, 113340, 102427, 104323, 78392, 26801, 59218, 60458, 37744, 37401, 5962, 25392, 31034, 38494, 22133, 7719, 22309, 43116, 16581, 8765, 3371, 4522, 65, 6868, 25707, 27755, 18765, 34304, 24931, 29934, 24635, 21826, 0, 32834, 19201, 20764, 27010, 54503, 47928, 24004, 17140, 28229, 13365, 17469, 21433, 8379, 4298, 5587, 3725, 52625, 18168, 16478, 66199, 30653, 56443, 29720, 21180, 20677, 7155, 18568, 29713, 17470, 16874, 1897, 1069, 26790, 21087, 73160, 55186, 86301, 4772, 20658, 9506, 3116, 8715, 14959, 23426, 0, 22687, 12675, 5287, 6491, 4308, 17618, 42871, 50675, 51510, 33284, 18127, 27732, 25385, 29186, 13686, 8381, 40777, 20924, 18527, 36309, 19088, 17797, 16609, 51869, 19718, 29762, 10868, 42461, 0, 81195, 42617, 50817, 9379, 21187, 15711, 8498, 27681, 58696, 5553, 22248, 20760, 29422, 13411, 56458, 57360, 1766, 24237, 12234, 1585, 1565, 6018, 407, 1917, 1679, 941, 6413, 802, 3282, 427, 1971, 5514, 4890, 2758, 2964, 7703, 2373, 733, 5100, 3470, 20330, 3456, 3445, 17680, 38941, 15344, 1907, 34216, 22955, 46034, 36991, 29288, 29918, 17316, 19584, 2366, 2509, 15877, 0, 3163, 4395, 1463, 10497, 1, 0, 7719, 16450, 13052, 18538, 9911, 4354, 26417, 7566, 13669, 1608, 4150, 654, 3115, 180, 64, 439, 498, 4102, 1595, 15005, 9331, 1529, 1207, 21484, 87, 10000, 15531, 13796, 7032, 1191, 13244, 13958, 8153, 21637, 12161, 2740, 2250, 4773, 4132, 12229, 40334, 14158, 6648, 6080, 8399, 6481, 1311, 6514, 26968, 21520, 4553, 20505, 24217, 25736, 55003, 112875, 122511, 62167, 57700, 48129, 24132, 7424, 21705, 14108, 9593, 4722, 25442, 20916, 17453, 6502, 17400, 8934, 3093, 1652, 1351, 6631, 20704, 31053, 5902, 15491, 53569, 52909, 32010, 41672, 38657, 26971, 52781, 21234, 34503, 34324, 28601, 76651, 22907, 10335, 17778, 7809, 22214, 46065, 37584, 18993, 64544, 10455, 24727, 21104, 68678, 16122, 10425, 13064, 32, 29518, 51650, 37784, 45152, 7972, 38239, 25337, 12585, 9611, 18369, 14470, 20190, 15750, 8119, 4749, 10095, 2988, 4640, 4428, 6922, 5781, 9058, 13986, 21059, 9024, 21312, 38834, 23688, 37139, 48416, 18819, 57690, 47120, 0, 15449, 49292, 47395, 61753, 37225, 60161, 67709, 35972, 21637, 34618, 36170, 51394, 55319, 53969, 23165, 30804, 35432, 19227, 11594, 21379, 10361, 6702, 8387, 8401, 7055, 10741, 17801, 25537, 37258, 35755, 32455, 25516, 51050, 18113, 39744, 14799, 30129, 21470, 79889, 75912, 50532, 46436, 0, 0, 0, 0, 0, 0, 0, 78741, 3895

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (4/11)

- 오전 7시 승차 데이터를 막대그래프로 시각화하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

result = []
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

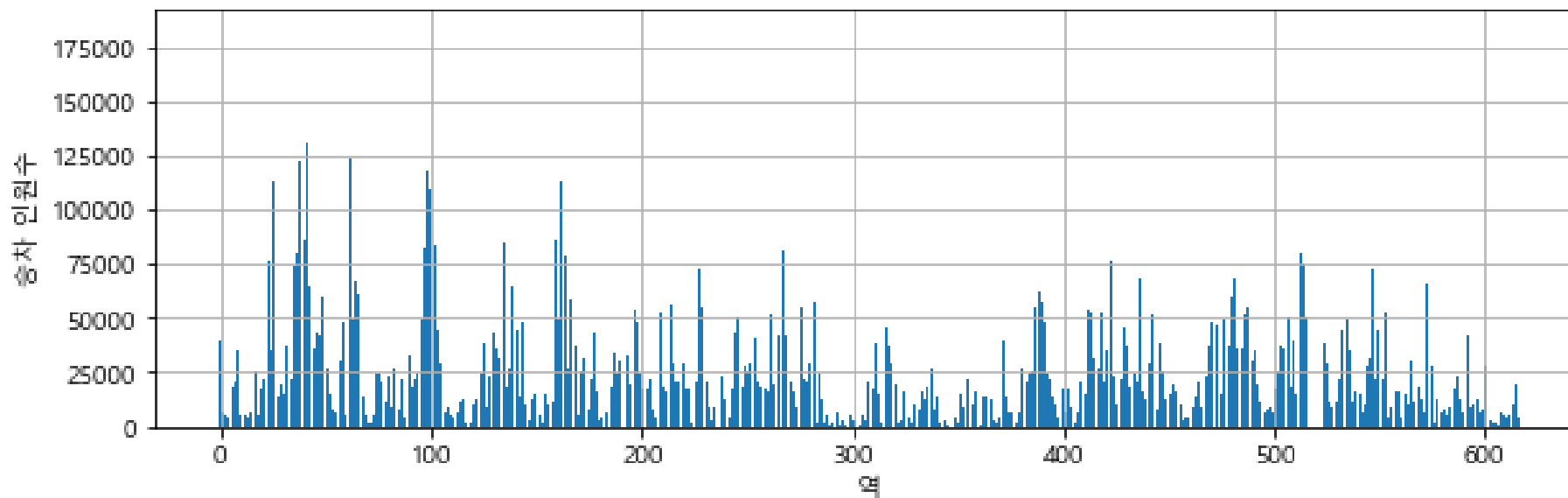
    row[4:] = map(int, row[4:])
    result.append(row[10])    # 오전 7시 승차 데이터

f.close()

plt.rc('font', family='Malgun Gothic')
plt.figure(figsize=(10, 3))
plt.bar(range(len(result)), result)
plt.xlabel('역')
plt.ylabel('승차 인원수')
plt.grid(True)
plt.show()
```

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (5/11)

- 오전 7시 승차 데이터를 막대그래프로 시각화하기(실행결과)



승차 인원수의 편차가 매우 크네요.
오름차순으로 정렬해 보겠습니다.

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (6/11)

- 오전 7시 승차 데이터를 오름차순으로 정렬하고, 막대그래프로 시각화하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

result = []
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    result.append(row[10])    # 오전 7시 승차 데이터

f.close()

result.sort()    # 오름차순으로 정렬

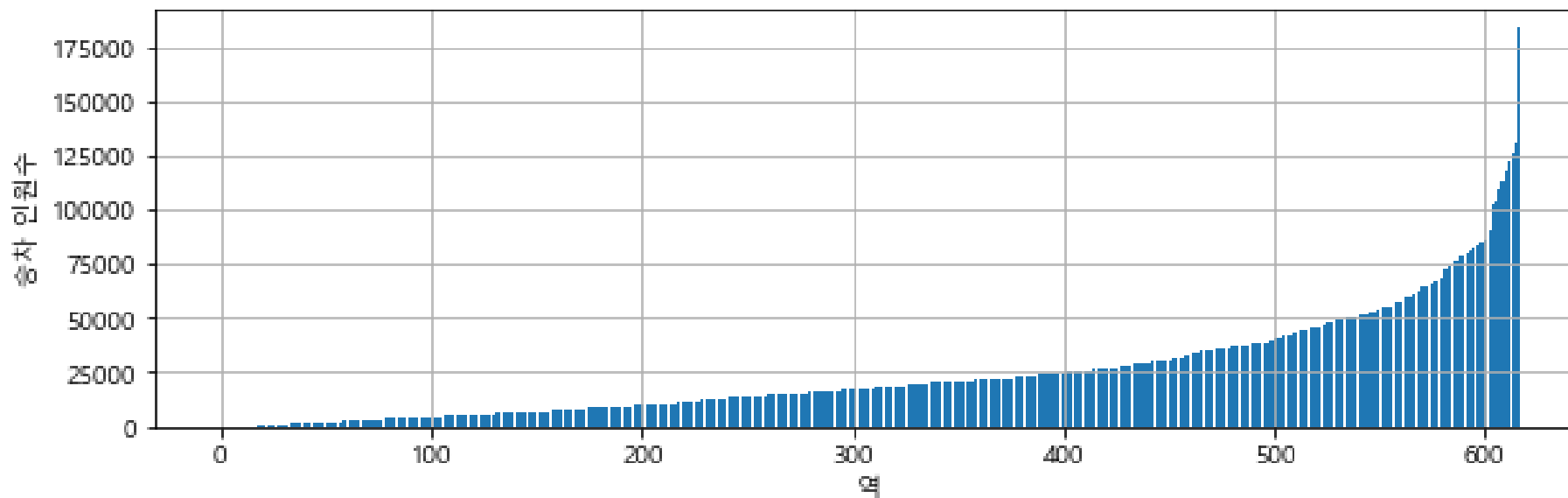
plt.rc('font', family='Malgun Gothic')
plt.figure(figsize=(10, 3))
plt.bar(range(len(result)), result)
plt.xlabel('역')
plt.ylabel('승차 인원수')
plt.grid(True)
plt.show()
```

sort 함수를 이용하면 데이터의 순서를
오름차순으로 정렬할 수 있습니다.

[Tip] 내림차순으로 정렬하고 싶다면
sort(reverse=True)라고 적으면 됩니다.

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (7/11)

- 오전 7시 승차 데이터를 오름차순으로 정렬하고, 막대그래프로 시각화하기(실행결과)



❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (8/11)

- 오전 7~9시 승차 데이터를 오름차순으로 정렬하고, 막대그래프로 시각화하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

result = []
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    result.append(sum(row[10:15:2]))    # 오전 7~9시 승차 데이터

f.close()

result.sort()    # 오름차순으로 정렬

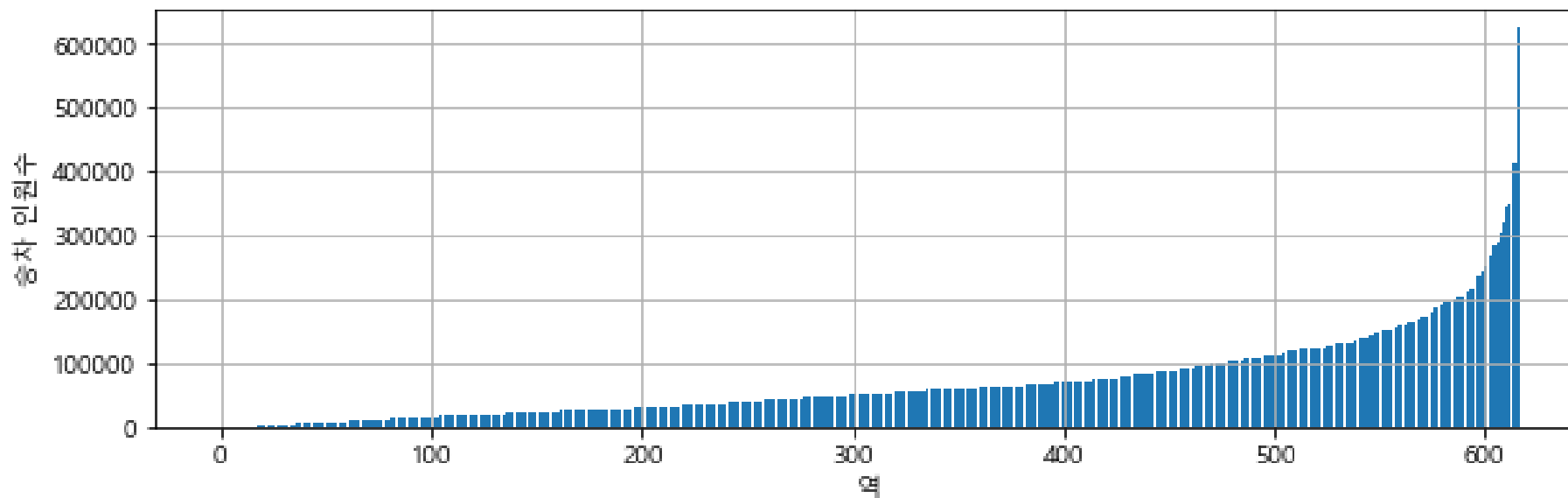
plt.rc('font', family='Malgun Gothic')
plt.figure(figsize=(10, 3))
plt.bar(range(len(result)), result)
plt.xlabel('역')
plt.ylabel('승차 인원수')
plt.grid(True)
plt.show()
```

오전 7시 승차 데이터: row[10]
오전 8시 승차 데이터: row[12]
오전 9시 승차 데이터: row[14]

- ① row[10] + row[12] + row[14]
- ② sum([row[10], row[12], row[14]])
- ③ sum(row[10:15:2])

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (9/11)

- 오전 7~9시 승차 데이터를 오름차순으로 정렬하고, 막대그래프로 시각화하기(실행결과)



❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (10/11)

- 오전 7~9시 승차 인원이 최대인 역 찾기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

max_people = 0
max_station = ''
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    if sum(row[10:15:2]) > max_people:
        max_people = sum(row[10:15:2])
        max_station = row[3] + '(' + row[1] + ')'

f.close()

print(max_station, ': ', max_people, '명')
```

신림(2호선) : 624717 명

❖ ② 출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (11/11)

- 오전 7~9시 하차 인원이 최대인 역 찾기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

max_people = 0
max_station = ''
for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    if sum(row[11:16:2]) > max_people:
        max_people = sum(row[11:16:2])
        max_station = row[3] + '(' + row[1] + ')'

f.close()

print(max_station, ':', max_people, '명')
```

역삼(2호선) : 802638 명

02. 지하철 시간대별 데이터 시각화하기

❖ ③ 밤 11시에 사람들이 가장 많이 타는 역은 어디일까 (1/2)

- 승차 시간 t와 인덱스 i 사이의 관계식 찾기

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	...
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
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	08:00:00~08:59:59					
2					승차	하차	승차	하차	승차	하차	승차	하차	승차	하차
3	Aug-22	1호선	150	서울역	573	19	8,638	8,274	12,332	45,706	39,560	102,779	63,523	200,999
4	Aug-22	1호선	151	시청	39	0	2,005	4,665	3,404	23,606	6,430	65,621	8,401	181,920
5	Aug-22	1호선	152	종각	54	4	3,356	4,382	3,765	22,971	5,801	98,968	9,571	243,599

승차 시간 t	인덱스 i	관계식
4	4	
5	6	
6	8	
7	10	$i = 4 + (t - 4) * 2$
8	12	$= 2t - 4$
...	...	
23	?	

❖ ③ 밤 11시에 사람들이 가장 많이 타는 역은 어디일까 (2/2)

- 밤 11시에 사람들이 가장 많이 타는 역 찾기

```
import csv

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

max_people = 0
max_station = ''
t = int(input('몇 시의 승차 인원이 가장 많은 역이 궁금하세요?: '))

for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    people = row[2 * t - 4]
    if people > max_people:
        max_people = people
        max_station = row[3] + '(' + row[1] + ')'

f.close()

print(max_station, ': ', max_people, '명')
```

몇 시의 승차 인원이 가장 많은 역이 궁금하세요?: 23
홍대입구(2호선) : 81164 명

02. 지하철 시간대별 데이터 시각화하기

❖ ④ 시간대별로 사람들이 가장 많이 타고 내리는 역은 어디일까 (1/6)

시간대별 최대 승차 역 이름 및 승차 인원 출력하기

- ✓ 새벽 4시~새벽 3시 → 24시간을 1시간 단위로 구분 → 24개 → for 반복문 사용
- ✓ 변수 j와 인덱스 i 사이의 관계식 찾기

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	...
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
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		08:00:00~08:59:59	
2					승차	하차	승차	하차	승차	하차	승차	하차	승차	하차
3	Aug-22	1호선	150	서울역	573	19	8,638	8,274	12,332	45,706	39,560	102,779	63,523	200,999

변수 j	인덱스 i	관계식
0	4	
1	6	
2	8	
...	...	
22	48	
23	50	

$$i = 2 * j + 4$$

❖ ④ 시간대별로 사람들이 가장 많이 타고 내리는 역은 어디일까 (2/6)

- 시간대별 최대 승차 역 이름 및 승차 인원 출력하기

```
import csv

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

max_people = [0] * 24
max_station = [''] * 24

for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    for j in range(24):
        people = row[2 * j + 4]
        if people > max_people[j]:
            max_people[j] = people
            max_station[j] = row[3]

f.close()

print(max_station)
print(max_people)
```

❖ ④ 시간대별로 사람들이 가장 많이 타고 내리는 역은 어디일까 (3/6)

- 시간대별 최대 승차 역 이름 및 승차 인원 출력하기(실행결과)

```
['구로', '신림', '신림', '신림', '신림', '신림', '신림', '신림', '잠실(송파구청)', '잠실(송파구청)', '강남', '강남', '강남', '강남', '강남',  
'강남', '강남', '잠실(송파구청)', '강남', '강남', '홍대입구', '강남', '금정', '동암', '용산']  
[11898, 38345, 71039, 184073, 274665, 165979, 88153, 76852, 84468, 98834, 106471, 131677, 154492, 241721, 321759, 197282, 153516, 177  
679, 167850, 81164, 14129, 29, 2, 1]
```

❖ ④ 시간대별로 사람들이 가장 많이 타고 내리는 역은 어디일까 (4/6)

- 시간대별 최대 승차 역 이름 및 승차 인원 막대그래프로 시각화하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

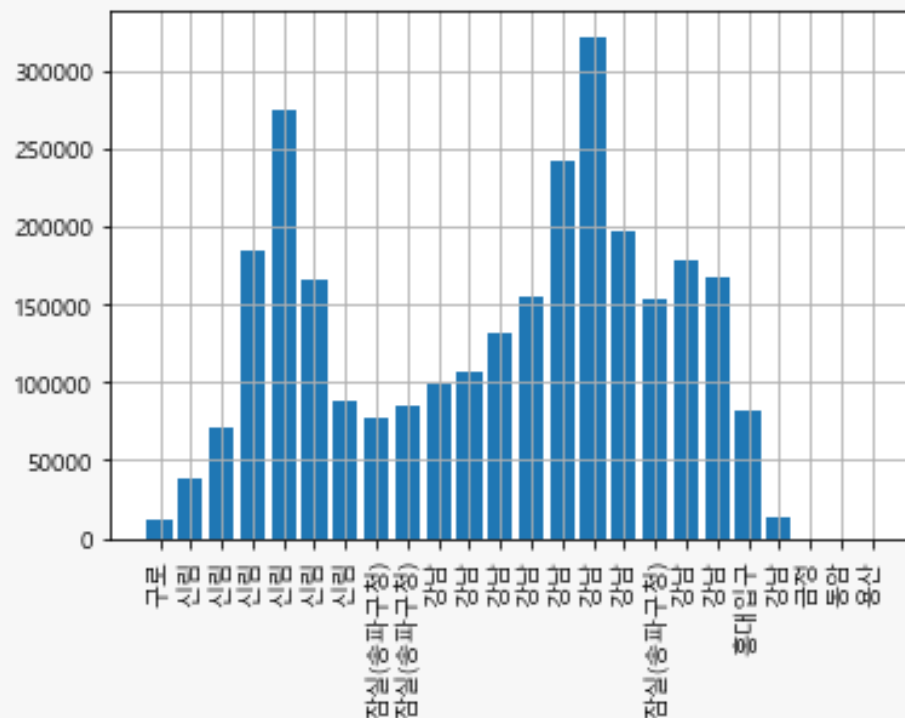
max_people = [0] * 24
max_station = [''] * 24

for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    for j in range(24):
        people = row[2 * j + 4]
        if people > max_people[j]:
            max_people[j] = people
            max_station[j] = row[3]

f.close()

plt.rc('font', family='Malgun Gothic')
plt.bar(range(24), max_people)
plt.xticks(range(24), max_station, rotation=90)
plt.grid(True)
plt.show()
```



x축에 몇 시인지 추가하면 좋을 것 같습니다.

❖ ④ 시간대별로 사람들이 가장 많이 타고 내리는 역은 어디일까 (5/6)

● 시간 정보 추가하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

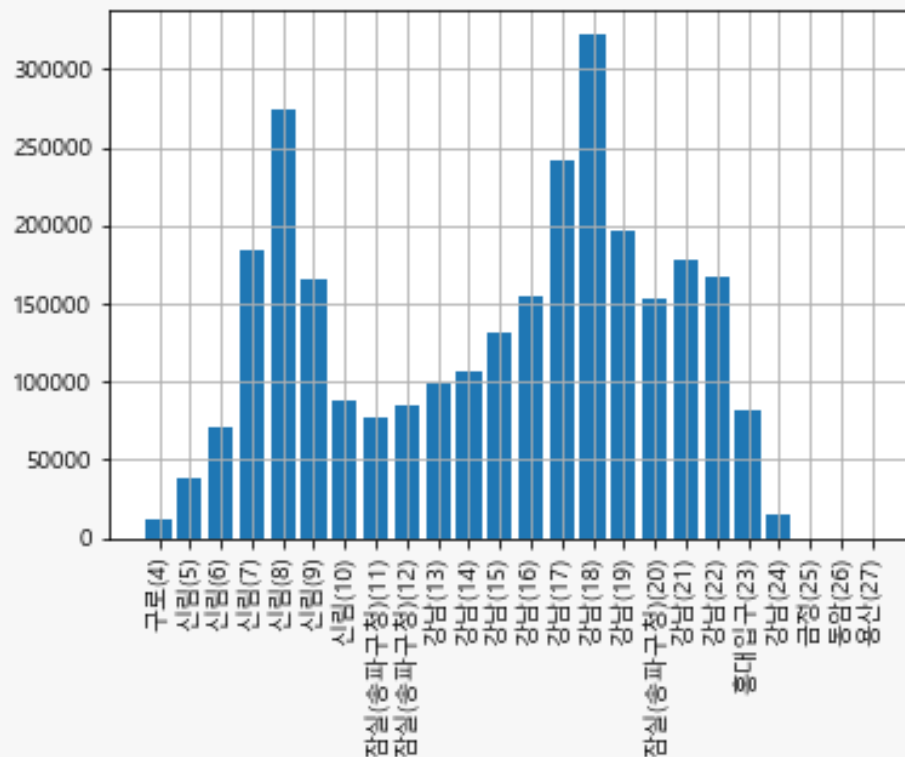
max_people = [0] * 24
max_station = [''] * 24

for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    for j in range(24):
        people = row[2 * j + 4]
        if people > max_people[j]:
            max_people[j] = people
            max_station[j] = row[3] + '(' + str(j + 4) + ')'

f.close()

plt.rc('font', family='Malgun Gothic')
plt.bar(range(24), max_people)
plt.xticks(range(24), max_station, rotation=90)
plt.grid(True)
plt.show()
```



❖ ④ 시간대별로 사람들이 가장 많이 타고 내리는 역은 어디일까 (6/6)

- 시간대별 최대 하차 역 이름 및 하차 인원 막대그래프로 시각화하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)
header = next(data)
next(data)
# print(header)

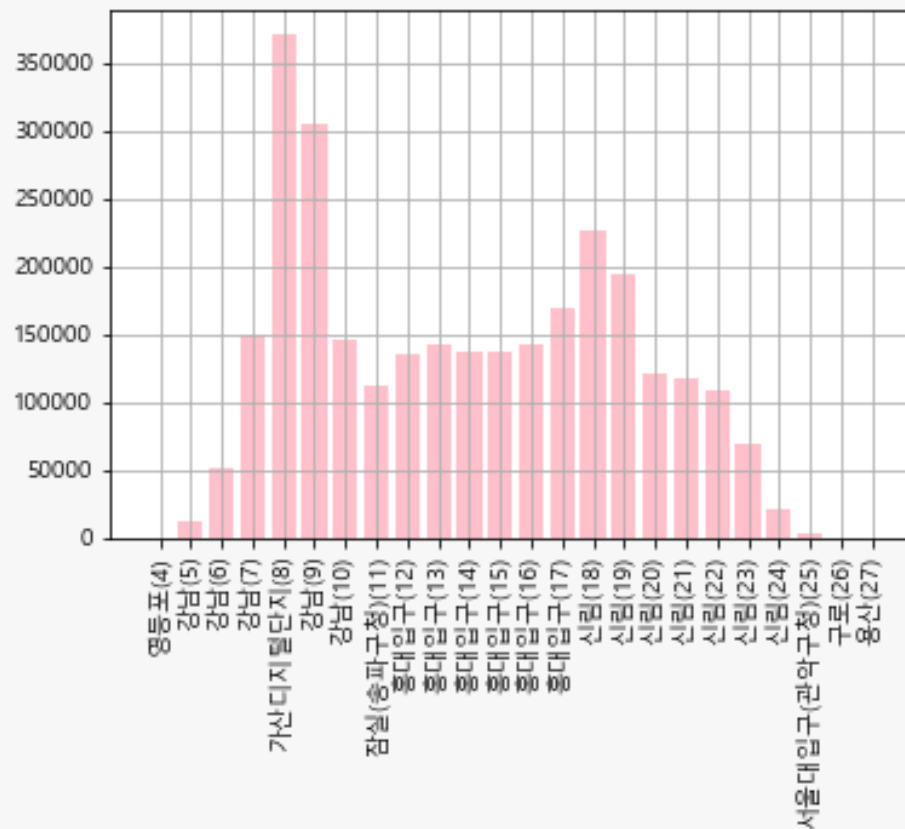
max_people = [0] * 24
max_station = [''] * 24

for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

    row[4:] = map(int, row[4:])
    for j in range(24):
        people = row[2 * j + 5]
        if people > max_people[j]:
            max_people[j] = people
            max_station[j] = row[3] + '(' + str(j + 4) + ')'

f.close()

plt.rc('font', family='Malgun Gothic')
plt.bar(range(24), max_people, color='pink')
plt.xticks(range(24), max_station, rotation=90)
plt.grid(True)
plt.show()
```



❖ ⑤ 모든 지하철역에서 시간대별 승하차 인원을 모두 더하면 (1/3)

알고리즘(Algorithm) 설계하기

- ✓ Step 1) 데이터를 읽어온다.
- ✓ Step 2) 모든 역에 대해 시간대별 승차 인원과 하차 인원을 누적해서 더한다.
- ✓ Step 3) 시간대별 승차 인원과 하차 인원을 그래프로 표현한다.

❖ ⑤ 모든 지하철역에서 시간대별 승하차 인원을 모두 더하면 (2/3)

- 시간대별 승하차 인원을 저장할 리스트 만들기

```
import csv

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
next(data)

s_in = [0] * 24      # 승차 인원을 저장할 리스트 초기화
s_out = [0] * 24     # 하차 인원을 저장할 리스트 초기화

for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

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

f.close()
```

❖ ⑤ 모든 지하철역에서 시간대별 승하차 인원을 모두 더하면 (3/3)

- 모든 지하철역의 시간대별 승하차 인원 추이 시각화하기

```
import csv
import matplotlib.pyplot as plt

f = open('subwaytime.csv', encoding='cp949')
data = csv.reader(f)

header = next(data)
next(data)

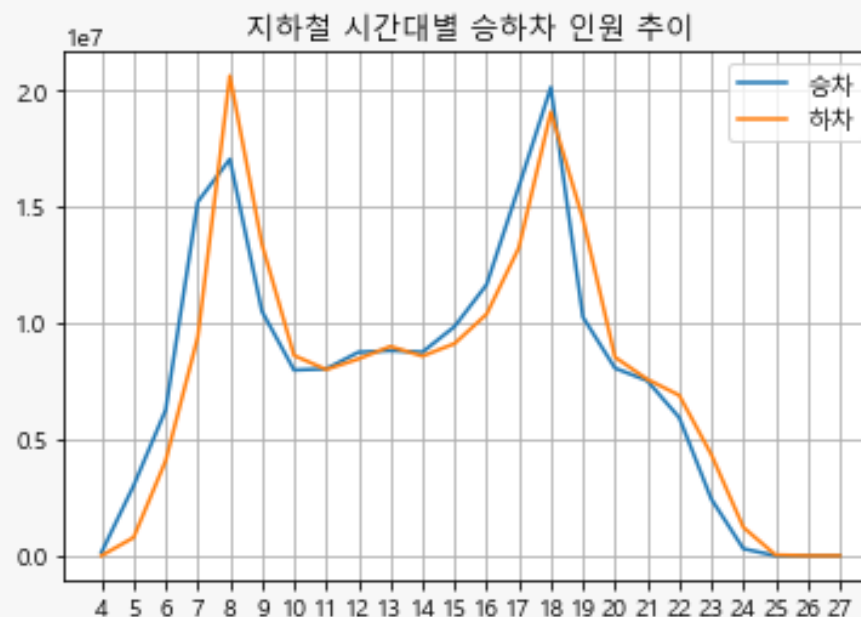
s_in = [0] * 24    # 승차 인원을 저장할 리스트 초기화
s_out = [0] * 24   # 하차 인원을 저장할 리스트 초기화

for row in data:
    for i in range(4, 52):
        row[i] = row[i].replace(',', '')

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

f.close()

plt.rc('font', family='Malgun Gothic')
plt.title('지하철 시간대별 승하차 인원 추이')
plt.plot(s_in, label='승차')
plt.plot(s_out, label='하차')
plt.legend()
plt.grid(True)
plt.xticks(range(24), range(4, 28))
plt.show()
```



- ❖ 01. 대중교통 데이터 시각화하기
- ❖ 02. 지하철 시간대별 데이터 시각화하기

THANK YOU!

Q & A

- Name: 강환수
- Office: 동양미래대학교 2호관 706호 (02-2610-1941)
- E-mail: hsknag@dongyang.ac.kr
- Homepage: <https://github.com/ai7dnn/2023-DA>