
Introduction To Data Analysis
데이터분석입문
Lecture 21. 서울 지하철 승객 시간별 데이터 시각화
인공지능소프트웨어학과 강환수 교수

학습개요

- 목록의 항목을 변환하는 내장 함수 map() 활용
- 지하철 월별 역 승하차 인원 수 데이터 분석

학습목표

- 내장 함수 map()을 활용해 목록의 항목을 정수로 변환할 수 있다.
- 지하철 출근시간 데이터를 시각화할 수 있다.

LESSON 01

매핑(mapping) 개요와 활용

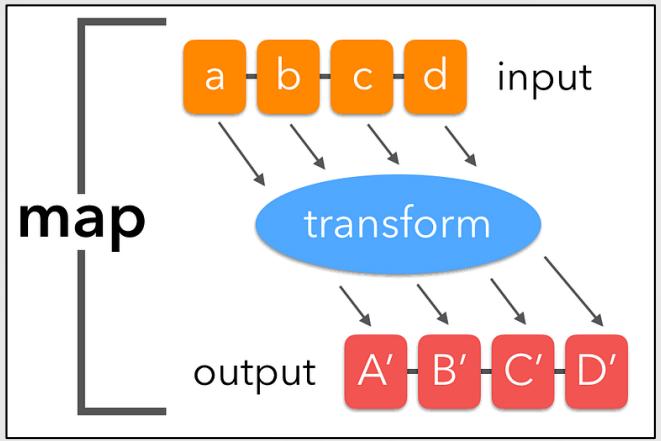
매핑 개요

매핑(mapping)

- 반복적인 개체의 각 항목에 변환 함수를 적용해 새로운 반복가능 개체로 변환해야 할 때 유용
- 내장 함수 map() 사용

내장 함수 map()

- 반복 구문을 사용하지 않고 반복가능(iterable) 컨테이너(container)의 모든 항목(item)을 특정 함수의 인자로 처리할 수 있는 함수



모든 항목에 10 더하기

함수 addsome

```
def addsome(value):
    return value + 10

lst = [10, 23, 4, 7]
result = map(addsome, lst)
list(result)
```

✓ 0.0s Python

```
[20, 33, 14, 17]
```

항목에서 콤마 제거

함수 myreplace()

```python
def myreplace(s):
    return s.replace(',', '')

lst = ['34,000', '1,999', '2,786', '1,000,000']
result = map(myreplace, lst)
list(result)
```

Python

```
['34000', '1999', '2786', '1000000']
```

모든 항목을 정수로 변환

함수 myreplace()

```python
lst = ['3', '4', '10']
result = map(int, lst)
list(result)
```

Python

```
[3, 4, 10]
```

모든 항목을 콤마를 제거하고 정수로 변환

map()을 2 번 사용

```
def myreplace(s):
    return s.replace(',', '')

lst = ['34,000', '1,999', '2,786', '1,000,000']
result = map(int, map(myreplace, lst))
list(result)
```

✓ 0.0s Python

```
[34000, 1999, 2786, 1000000]
```

다양한 map() 활용

```
int('101', 2)
```

✓ 0.0s Python

```
5
```

```
list(map(int, [3, 5, 11]))
```

✓ 0.0s Python

```
[3, 5, 11]
```

```
list(map(int, ['3', '5', 11]))
```

✓ 0.0s Python

```
[3, 5, 11]
```

```
list(map(int, ['10', '17', '1f'], [2, 8, 16]))
```

✓ 0.0s Python

```
[2, 15, 31]
```

익명 함수 사용

람다 함수

```
(lambda x, y: x + y)(30, 4)
```

✓ 0.0s Python

```
34
```

```
# def myreplace(s):
#     return s.replace(',', '')

lst = ['34,000', '1,999', '2,786', '1,000,000']
result = map(int, map(lambda s: s.replace(',', ''), lst))
list(result)
```

✓ 0.0s Python

```
[34000, 1999, 2786, 1000000]
```

LESSON 02

지하철 출근시간 데이터 시각화

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', '']
['', '', '', '', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차', '승차', '하차']
['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']
```

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
```

지하철 시간대별 이용 현황 데이터 정제하기 (3/3)

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

```
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, 62027, 59146, 58165, 78671, 61463, 88156, 65725, 136669, 75650, 198548, 82448, 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, 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, 28214, 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, 2639, 3960, 3158, 20401, 5614, 89885, 9209, 225883, 10921, 131732, 15255, 53438, 23410, 48848, 27245, 42754, 34266, 41537,
```

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(서	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	신당	18	0	5,746	1,188	10,743	8,016	26,628	16,735	43,549	31,151

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)
```

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, 225
33, 50866, 66999, 76037, 35469, 113682, 51500, 14920, 14026, 19738, 14694, 37772, 24831, 16398, 21777, 73756, 79730, 122277, 78413, 1
84073, 85469, 131269, 64768, 126331, 41387, 36682, 42911, 41829, 59887, 35765, 24155, 27299, 14870, 7971, 6552, 4176, 3304, 30755, 47
631, 4976, 23154, 82016, 124350, 49220, 67128, 61010, 13149, 14171, 13217, 5722, 1723, 2133, 0, 5761, 24966, 24980, 21425, 15013, 105
86, 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, 4548
5, 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, 2701
0, 54503, 47928, 24004, 17140, 28229, 13365, 17469, 21433, 8379, 4298, 5587, 3725, 52625, 18168, 16478, 66199, 30653, 56443, 29720, 2
1180, 20677, 7155, 18568, 29713, 17470, 16874, 1897, 1069, 26790, 21087, 73160, 55186, 86301, 4772, 20658, 9506, 3116, 8715, 14959, 2
3426, 0, 22687, 12675, 5287, 6491, 4308, 17618, 42871, 50675, 51510, 33284, 18127, 27732, 25385, 29186, 13686, 8381, 40777, 20924, 18
527, 36309, 19088, 17797, 16609, 51869, 19718, 29762, 10868, 42461, 0, 81195, 42617, 50817, 9379, 21187, 15711, 8498, 27681, 58696, 5
5553, 22248, 20760, 29422, 13411, 56458, 57360, 1766, 24237, 12234, 1585, 1565, 6018, 407, 1917, 1679, 941, 6413, 802, 3282, 427, 197
1, 5514, 4890, 2758, 2964, 7703, 2373, 733, 5100, 3470, 20330, 3456, 3445, 17680, 38941, 15344, 1907, 34216, 22955, 46034, 36991, 292
88, 29918, 17316, 19584, 2366, 2509, 15877, 0, 3163, 4395, 1463, 10497, 1, 0, 7719, 16450, 13052, 18538, 9911, 4354, 26417, 7566, 136
69, 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, 2050
5, 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, 5165
0, 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, 3597
2, 21637, 34618, 36170, 51394, 55319, 53969, 23165, 30804, 35432, 19227, 11594, 21379, 10361, 6702, 8387, 8401, 7055, 10741, 17801, 2
5537, 37258, 35755, 32455, 25516, 51050, 18113, 39744, 14799, 30129, 21470, 79889, 75912, 50532, 46436, 0, 0, 0, 0, 0, 0, 78741, 3895
```

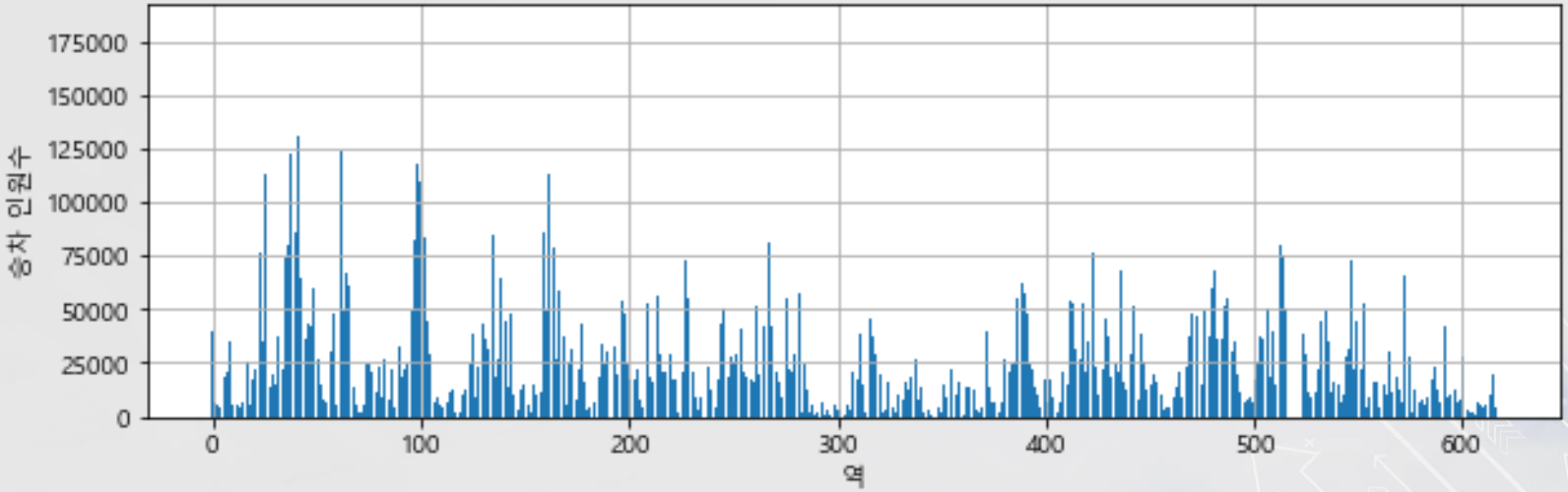
출근 시간대 사람들이 가장 많이 타고 내리는 역은 어디일까 (4/11)

모든 역의 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()
```

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

모든 역의 오전 7시 승차 데이터를 막대그래프로 시각화하기(실행결과)



- 승차 인원수의 편차가 매우 크네요.
  오름차순으로 정렬해 보겠습니다.

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

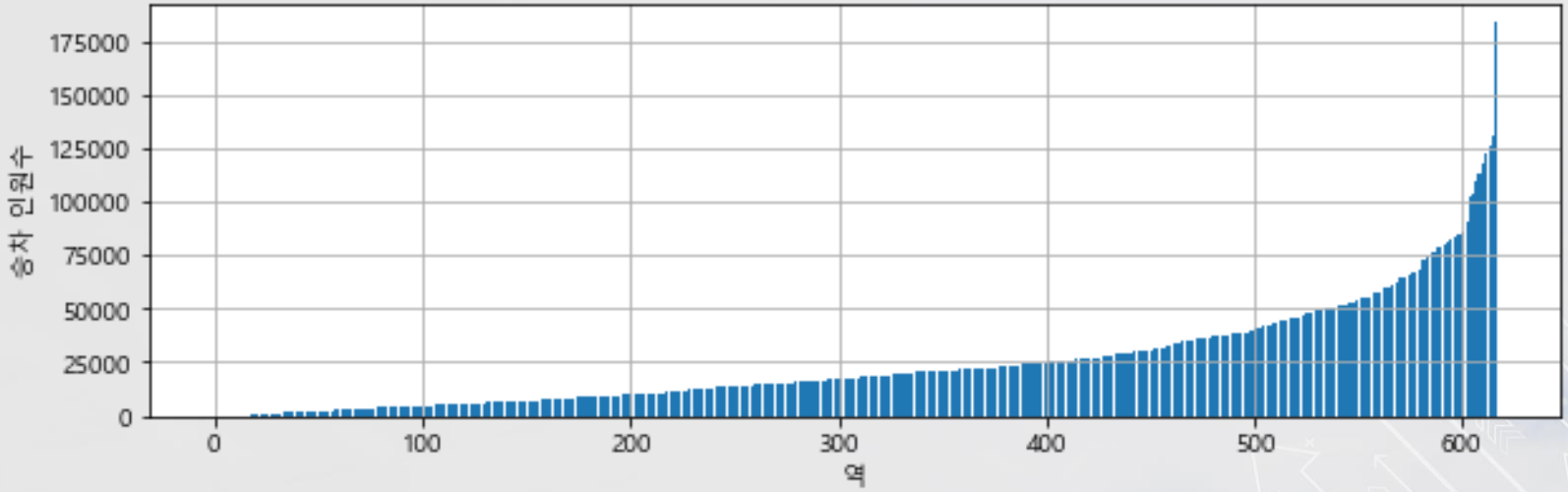
오전 7시 승차 데이터를 오름차순으로 정렬하고, 막대그래프로 시각화하기

```
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.

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

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



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

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

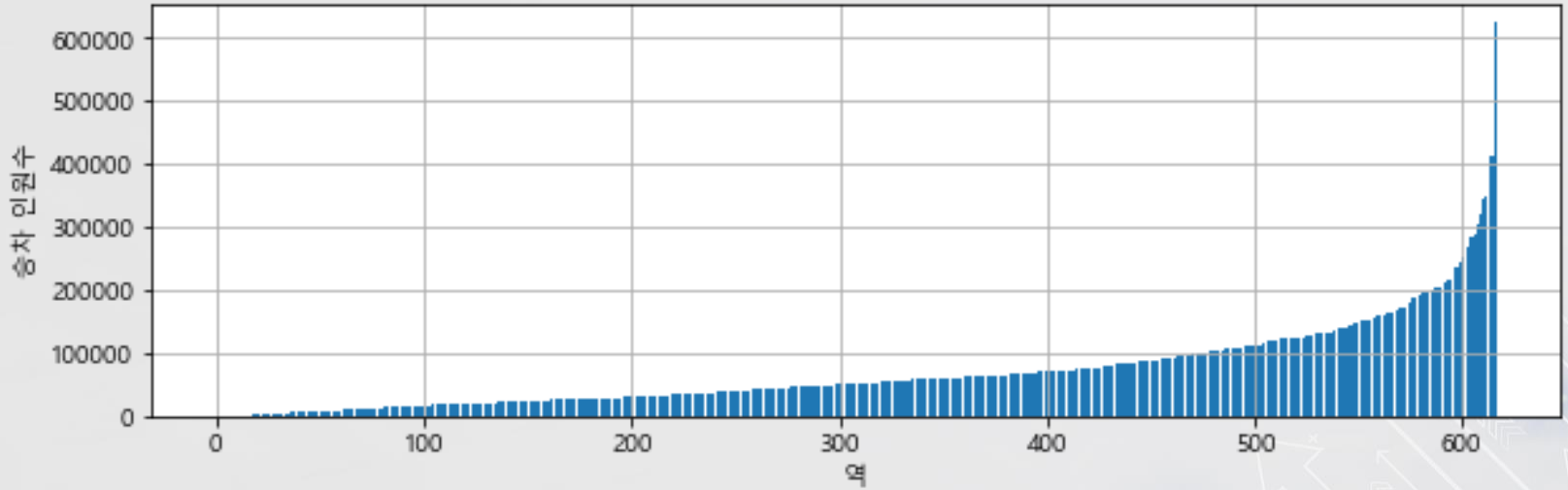
```
import csv
import matplotlib.pyplot as plt

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])

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

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



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
```

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
```

SUMMARY

학습정리

내장 함수 map 활용

```
list(map(int, [3.1, 5.8, 11.3]))
```

✓ 0.0s

```
[3, 5, 11]
```

지하철 월별 역 승하차 인원 수 데이터 분석

- 사용월, 호선명, 역ID, 지하철역, 4시승차, 4시하차, 5시승차, 5시하차, …

서울 지하철 모든 역 오전 7시 승차인원 정렬

- row[10]

서울 지하철 모든 역 오전 7시~9시 승차인원 정렬

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