03_Numpy 파일 입출력

Numpy 파일 입출력

- 파일 처리 .npy 넘파이배열(np) 파일 확장자

  - ndarray의 정보를 파일로 저장할 수 있다.
  - 일반적인 파일 확장자는 npy를 사용한다.

> 파일 저장(tofile) 저장하고 가져올때 데이터만 가져가고 속성값은 안가져감 그래서 shape 정보가 다름

```
n = np.random.randn(3, 5)
print(n)
n.tofile('n.npy')
```

Out : [[-0.47256261 0.52953411 0.3417467 -0.71718817 1.11659064]
[0.69278475 -0.99542026 -0.30128399 -0.61580567 1.22173985]
[0.44158276 0.83449537 1.73090983 -0.44021898 1.39393169]]

Numpy 파일 입출력

- 파일 처리

> 파일 불러오기(fromfile)

```
m = np.fromfile('n.npy')
print(m)
```

Out : [-0.47256261 0.52953411 0.3417467 -0.71718817
1.11659064 0.69278475
 -0.99542026 -0.30128399 -0.61580567 1.22173985 0.44158276
0.83449537
  1.73090983 -0.44021898 1.39393169]

Numpy 파일 입출력

- 파일 처리

> 파일 저장, 불러오기(np.save, np.load)

```
np.save('n1.npy',n)
m1 = np.load('n1.npy')
print(m1)
```

원 shape 정보대로 저장하고 가져오기

```
Out : [[-0.47256261  0.52953411  0.3417467  -0.71718817
1.11659064]
       [ 0.69278475 -0.99542026 -0.30128399 -0.61580567
1.22173985]
       [ 0.44158276  0.83449537  1.73090983 -0.44021898
1.39393169]]
```

Numpy 파일 입출력

- 파일 처리

tofile/fromefile 짝으로 쓰고
save/losd 짝으로 써야 안깨짐

> 주의해야할 점

```
m = np.load('n.npy')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-107-8b496dd6e9b6> in <module>
----> 1 m = np.load('n.npy')

~₩anaconda3₩lib₩site-packages₩numpy₩lib₩npyio.py in load(file, mmap_mode, allow_pickle, fix_imports, encoding)
    442             # Try a pickle
    443             if not allow_pickle:
--> 444                 raise ValueError("Cannot load file containing pickled data "
    445                                  "when allow_pickle=False")
    446             try:

ValueError: Cannot load file containing pickled data when allow_pickle=False
```

```
np.fromfile('n1.npy')
```

Out : array([1.87585069e-309, 1.17119999e+171, 5.22741680e-037,

8.44740097e+252, 2.65141232e+180, 9.92152605e+247,

...

Numpy 파일 입출력

- 파일 처리

> txt 파일 저장(np.savetxt)

```
np.savetxt('n.txt',n)
```

> txt 파일 불러오기(np.loadtxt)

```
np.loadtxt('n.txt')
```

Out : array([[-0.47256261, 0.52953411, 0.3417467 , -0.71718817,
1.11659064],

       [0.69278475, -0.99542026, -0.30128399, -0.61580567,
1.22173985],

       [0.44158276, 0.83449537, 1.73090983, -0.44021898,
1.39393169]])

Numpy 파일 입출력

- 파일 처리

> csv 파일 저장, 불러오기

```
np.savetxt('n.csv',n)  # 확장자만 csv로 바꾸면 됨
```

```
np.savetxt('n_1.csv',n,delimiter=',')
```

콤마로 구분하여 csv로 저장함

> csv 파일 불러오기

```
np.loadtxt('n_1.csv',delimiter=',')
```

```
np.loadtxt('n.csv')
```

```
Out : array([[-0.47256261,  0.52953411,  0.3417467 , -0.71718817,
1.11659064],
       [ 0.69278475, -0.99542026, -0.30128399, -0.61580567,
1.22173985],
       [ 0.44158276,  0.83449537,  1.73090983, -0.44021898,
1.39393169]])
```

불러올때 type을 바꾸어서 불러올 수 있음

```
np.loadtxt('n_1.csv',delimiter=',', dtype=int, converters=float)
```

Numpy 파일 입출력

연습문제

- 파일 처리

> [문제1]
1~25 범위의 5x5인 int8 자료형의 2차원 행렬을 생성하고
구분자(delimiter)는 ,(콤마)로 지정해서 data.csv에
저장해 봅시다.

> [문제2]
06_data1.txt 파일을 읽어와서 dtype = str으로 변수 arr1에
저장하고 출력해 봅시다.

> [문제3]
06_data2.txt 파일을 읽어와서 변수 arr2에 저장하고
출력해 봅시다.

- 파일 처리

> [문제1]

1~25 범위의 5x5인 int8 자료형의 2차원 행렬을 생성하고
구분자(delimiter)는 ,(콤마)로 지정해서 data.csv에
저장해 봅시다.

```
arr = np.arange(1,26, dtype = np.int8).reshape(5, 5)
np.savetxt('data.csv', arr, delimiter=',')
```

```
np.loadtxt('data.csv', delimiter=',')
```

Out : array([[0., 1., 2., 3., 4.],
[5., 6., 7., 8., 9.],
[10., 11., 12., 13., 14.],
[15., 16., 17., 18., 19.],
[20., 21., 22., 23., 24.]])

- 파일 처리

> [문제2]
06_data1.txt 파일을 읽어와서 dtype = str으로 변수 arr1에 저장하고 출력해 봅시다.

```
arr1 = np.loadtxt('06_data1.txt', dtype = str)
print(arr1)
```

Out : [['1' '15' '14' '4']
['12' '6' '7' '9']
['8' '10' '11' '5']
['13' '3' '2' '16']]

- 파일 처리

> [문제3]
06_data2.txt 파일을 읽어와서 변수 arr2에 저장하고
출력해 봅시다.

```
arr2 = np.loadtxt('06_data2.txt', delimiter=',')
print(arr2)
```

Out : [[1. 15. 14. 4.]
[12. 6. 7. 9.]
[8. 10. 11. 5.]
[13. 3. 2. 16.]]

- 파일 처리

> 파일 ex1_지역별 전기요금 데이터 현황

지역별 전기요금 데이터 현황

```
# 지역구분,대상 가구수(호),가구당 평균 전력사용량(kWh),가구당 평균 전기요금(원)
# 강원1,경기2,경남3,경북4,광주5,대구6,대전7,부산8,서울9
# 세종10,울산11,인천12,전남13,전북14,제주15,충남16,충북17
1,565082,273,27177
2,2827233,444,45111
3,1144913,289,29043
4,1131034,230,22436
5,380425,391,38020
6,668623,364,35245
7,425659,344,32331
8,947300,366,36274
9,2737725,361,36569
10,49394,580,57091
11,323151,353,35470
12,734723,398,40049
13,726699,249,25120
14,634696,285,28556
15,258605,221,22180
16,742285,282,27901
17,584500,268,26899
```

- 파일 처리

> 01. 파일 읽어오기

```
data = np.loadtxt('ex1_지역별전기요금.csv', delimiter=',',
                  dtype=np.int)
print(data[:5, :])  # 상위 5개 데이터 출력
```

```
Out : [[      1  565082     273   27177]
       [      2 2827233     444   45111]
       [      3 1144913     289   29043]
       [      4 1131034     230   22436]
       [      5  380425     391   38020]]
```

- 파일 처리

> 02. 데이터 확인하기

```
# 데이터의 모양 확인
print(data.shape)
```

Out : (17, 4)

```
# 중복 지역 확인
print(np.unique(data[:,0]))
```

Out : [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17]

- 파일 처리

> 02. 데이터 확인하기

```
# 데이터의 크기순 정렬 (argsort 사용)
print(data[:,1].argsort())
```

Out : [9 14 10 4 6 0 16 13 5 12 11 15 7 3 2 8 1]

```
# fancy index
print(data[data[:,1].argsort()])
```

Out : [[10   49394   580   57091]
 [15 258605   221   22180]
 [11 323151   353   35470]
 [5 380425   391   38020]
 ...

- 파일 처리

> 03. 데이터 통계

```
# 전국 대상 가구수의 합
print(np.sum(data[:, 1]))
```

Out : 14882047

```
# 가구당 평균 전력 사용량의 전국 평균값
print(round(data[:, 2].mean(), 2))
```

Out : 335.18

- 파일 처리

> 03. 데이터 통계

```
# 가구당 평균 전력 사용량이 평균 이상인 지역과 대상 가구수 추출
avg_ = data[:, 2].mean()  # 가구당 평균 전력 사용량의 전국 평균
print(data[data[:, 2] > avg_, 0:2])
```

```
Out : [[     2 2827233]
       [     5  380425]
       [     6  668623]
       [     7  425659]
       [     8  947300]
       [     9 2737725]
       [    10   49394]
       [    11  323151]
       [    12  734723]]
```

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

전년대비 시도별 교통사망사고 데이터 현황

```
# 발생년도,구분,발생건수,구성비(%),증감수,증감률
# 강원1,경기2,경남3,경북4,광주5,대구6,대전7,부산8,서울9
# 세종10,울산11,인천12,전남13,전북14,제주15,충남16,충북17
2014,1,218,4.8,-9,-4
2014,2,872,19,-45,-4.9
2014,3,401,8.7,-45,-10.1
2014,4,464,10.1,-41,-8.1
2014,5,99,2.2,-5,-4.8
2014,6,182,4,19,11.7
2014,7,99,2.2,8,8.8
2014,8,163,3.6,-47,-22.4
2014,9,386,8.4,14,3.8
2014,10,20,0.4,0,0
2014,11,103,2.2,-16,-13.4
2014,12,144,3.1,-6,-4
2014,13,400,8.7,-35,-8
2014,14,317,6.9,-35,-9.9
2014,15,86,1.9,-14,-14
2014,16,387,8.4,-5,-1.3
2014,17,242,5.3,-31,-11.4
```

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제1]
위 파일을 읽어오고 상위 3개의 (발생년도, 발생건수, 구성비)의 데이터를 추출해 봅시다.

> [문제2]
위 데이터에서 세종에서 발생한 교통사망사고 모든 현황을 추출해 봅시다.

> [문제3]
위 데이터에서 전국 사고 발생건수의 합계와 전국 평균을 계산해 봅시다.

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제4]
위 데이터에서 교통사고 증감수가 가장 큰 지역의 모든 현황을 추출해 봅시다.

> [문제5]
위 데이터에서 교통사고 증감률이 가장 작은 지역의 모든 현황을 추출해 봅시다.

> [문제6]
위 데이터에서 사고 발생 건수가 400건 이상인 곳의 지역과 교통사고 발생건수, 증감수, 증감률을 추출해 봅시다.

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제1]
위 파일을 읽어오고 상위 3개의 (발생년도, 발생건수, 구성비)의 데이터를 추출해 봅시다.

```
data = np.loadtxt('ex2_전년대비_시도별_교통사망사고.csv',
                  delimiter=',', dtype=np.float)
print(data[:3, [0, 2, 3]])  # 상위 3개, [0, 2, 3] 열 데이터 출력
```

Out : [[2014. 218.   4.8]
 [2014. 872.   19.]
 [2014. 401.   8.7]]

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제2]
위 데이터에서 세종(10)에서 발생한 교통사망사고 모든 현황을 추출해 봅시다.

```
print(data[data[:, 1] == 10]) # 세종에 해당하는 데이터 추출
```

Out : [[2014. 10. 20. 0.4 0. 0.]]

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제3]
위 데이터에서 전국 사고 발생건수의 합계와 전국 평균을 계산해 봅시다.

```
print(np.sum(data[:, 2]))    # 발생건수의 총합
```

Out : 4583.0

```
print(round(np.sum(data[:, 2]) / 17, 2))
# 발생건수의 전국 평균
```

Out : 269.59

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제4]
위 데이터에서 교통사고 증감수가 가장 큰 지역의 모든 현황을 추출해 봅시다.

```
# 증감수 컬럼에서 가장 큰 값
print(np.max(data[:,4]))
```

Out : 19.0

```
print(data[np.argmax(data[:,4])]) # 증감수가 가장 큰 지역
```

Out : [2014. 6. 182. 4. 19. 11.7]

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제5]
위 데이터에서 교통사고 증감률이 가장 작은 지역의 모든 현황을 추출해 봅시다.

```
# 증감률 컬럼에서 가장 작은 값
print(np.min(data[:,5]))
```

Out : -22.4

```
print(data[np.argmin(data[:,5])]) # 증감률이 가장 작은 지역
```

Out : [2014. 8. 163. 3.6 -47. -22.4]

- 파일 처리

> 파일 ex2_전년대비_시도별_교통사망사고 데이터 현황

> [문제6]
위 데이터에서 사고 발생 건수가 400건 이상인 곳의 지역과
교통사고 발생건수, 증감수, 증감률을 추출해 봅시다.

```
# 사고 발생건수가 400건 이상인 데이터 추출
data2 = data[data[:,2] >= 400,:]
print(data2[:, [1, 2, 4, 5]])
```

Out : [[2. 872. -45. -4.9]
[3. 401. -45. -10.1]
[4. 464. -41. -8.1]
[13. 400. -35. -8.]]