Chapter 13

Numpy 데이터 분석

Numpy란?

- Numpy: ‘Numerical Python’

  파이썬을 이용한 데이터 분석과 산술 연산의 가장 기본적인 패키지

- 배열의 구조, 다차원 배열, 배열 간 연산, 배열의 정렬 등의 많은 편리한

  기능 제공

- Ndarray : 배열 객체, 리스트와 유사, 배열을 다루는 편리한 기능 제공

예제 13-1. ndarray 객체 생성

ex13.ipynb

```
import numpy as np


data = np.array([1, 2, 3, 4, 5])

print(data)

print(type(data))

print(data.dtype)
```

: : 실행 결과

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
int32
```

예제 13-2. 2차원 배열 객체 생성

ex13.ipynb

```
import numpy as np


data = np.random.randn(2,3)

print(data)

print(data.shape)

print(data.dtype)
```

: : 실행 결과

```
[[-0.44892817 0.55240635 1.81518668]
 [-0.33471118 1.1305588 -0.31145892]]
(2, 3)
float64
```

예제 13-3. ndarray 0으로 초기화하기

ex13.ipynb

```
import numpy as np

data1 = np.zeros(10)

print(data1)

print(data1.dtype)

data2 = np.zeros((2, 3))

print(data2)

data3 = np.zeros((2, 3), dtype=np.int32)

print(data3)
```

: : 실행 결과

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
float64
[[0. 0. 0.]
 [0. 0. 0.]]
[[0 0 0]
 [0 0 0]]
```

예제 13-4. ndarray 1로 초기화하기

ex13.ipynb

```
import numpy as np


data1 = np.ones(8)

print(data1)

data2 = np.ones((3, 4), dtype=np.int32)

print(data2)
```

: : 실행 결과

```
[1. 1. 1. 1. 1. 1. 1. 1.]
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]
```

예제 13-5. arrange() 메소드

ex13.ipynb

```
data = np.arange(10, 121, 10)

print(data)

print(data[2])

print(data[5:8])

data[7:10] = 800

print(data)

data2 = data.reshape(2, 6)

print(data2)
```

: : 실행 결과

```
[ 10 20 30 40 50 60 70 80 90 100 110 120]
30
[60 70 80]
[ 10 20 30 40 50 60 70 800 800 800 110 120]
[[ 10 20 30 40 50 60]
 [ 70 800 800 800 110 120]]
```

표 13-1. ndarray 생성 메소드

메소드	설명
array()	매개 변수로 사용되는 리스트, 튜플, 배열 형의 데이터를 이용하여 ndarray 객체를 생성한다.
random.randn()	가우시안 정규 분포를 갖는 랜덤 수를 생성한다.
zeros()	ndarray 객체를 생성하고 0으로 초기화한다.
ones()	ndarray 객체를 생성하고 1로 초기화한다.
arange()	내장 함수 range()와 동일한 기능, ndarray 객체를 반환한다.
reshape()	ndarray 객체의 차원을 재구성한다. 예를 들어 reshape(3,5)은 배열의 구조를 3행 5열의 2차원으로 변환한다.

표 13-2. ndarray 속성

속성	설명
dtype	ndarray 객체의 데이터 형을 나타낸다. int32는 32비트 정수형, float64는 64비트 실수형을 의미한다.
shape	ndarray 객체의 행과 열을 튜플로 나타낸다.

예제 13-6. 2차원 배열의 요소 추출(계속)

: : 실행 결과

```
[[ 1 2 3 4 5]
 [ 6 7 8 9 10]
 [11 12 13 14 15]]
14
[2 3 4 5]
[1 2 3 4 5]
[[ 6 7 8 9 10]
 [11 12 13 14 15]]
```

```
[[ 1 2 3 4 5]
 [100 100 100 100 100]
 [ 11 12 13 14 15]]
[[200 200 200 200 200]
 [200 200 200 200 200]
 [200 200 200 200 200]]
```

예제 13-7. ndarray의 산술 연산

ex13.ipynb

```
a = np.array([[10, 7, -8, 2],
              [-2, 2, 8, 3],
              [6, -8, -5, 3]])
b = a * 2
print(b)
c = a * a
print(c)
print(a > b)
```

: : 실행 결과

```
[[ 20 14 -16 4]
 [ -4 4 16 6]
 [ 12 -16 -10 6]]
[[100 49 64 4]
 [ 4 4 64 9]
 [ 36 64 25 9]]
[[False False True False]
 [ True False False False]
 [False True True False]]
```

예제 13-8. 배열의 합계, 평균, 최댓값, 최솟값

ex13.ipynb

```
data = np.array([[80, 78, 90, 93],
                 [65, 87, 88, 75],
                 [98, 100, 68, 80]])
print(data.sum())
print(data.mean())
print(data.max())
print(data.min())
print()
```

```
print(data.max(axis=0))
print(data.max(axis=1))
print()


index1 = np.argmax(data, axis=0)
index2 = np.argmin(data, axis=1)
print(index1, index2)
```

예제 13-8. 배열의 합계, 평균, 최댓값, 최솟값(계속)

: : 실행 결과

```
1002
83.5
100
65

[ 98 100 90 93]
[ 93 88 100]

[2 2 0 0] [1 0 2]
```

예제 13-9. 배열에 조건식 사용

ex13.ipynb

```
data = np.random.randn(3,4)
print(data)


print(data > 0)


total = (data < 0).sum()
print(total)
```

```
data2 = np.where(data > 0, 1, -1)
print(data2)


data3 = np.where(data > 0, 5, data)
print(data3)
```

예제 13-9. 배열에 조건식 사용(계속)

: : 실행 결과

```
[[ 0.823293 0.292996 -2.59991429 0.9087267 ]
 [-0.28825592 0.3996413 0.49922846 -0.32000343]
 [ 1.12672336 -0.51248337 0.99202783 0.77582767]]
[[ True True False True] [False True True False]
 [ True False True True]]
4
[[ 1 1 -1 1]
 [-1 1 1 -1]
 [ 1 -1 1 1]]
[[ 5. 5. -2.59991429 5. ]
 [-0.28825592 5. 5. -0.32000343]
 [ 5. -0.51248337 5. 5. ]]
```

예제 13-10. 배열의 요소 정렬

ex13.ipynb

```
data = np.array([[13, 22, 17, 2],
                 [-2, 20, 8, 3],
                 [-16, 10, -5, 33]])
data.sort(0)
print(data)
print()
data.sort(1)
print(data)
```

: : 실행 결과

```
[[-16 10 -5 2]
 [ -2 20 8 3]
 [ 13 22 17 33]]

[[-16 -5 2 10]
 [ -2 3 8 20]
 [ 13 17 22 33]]
```

예제 13-11. 배열에 열과 행 삽입

ex13.ipynb

```
a = np.arange(10)
print(a)


b = np.insert(a, 3, 10)
print(b)
x = np.array([[1, 1, 1], [2, 2, 2], [3, 3, 3]])
print(x)
```

```
y = np.insert(x, 1, 10, axis=0)
print(y)


y = np.insert(x, 1, 10, axis=1)
print(y)
```

예제 13-11. 배열에 열과 행 삽입(계속)

: : 실행 결과

```
[0 1 2 3 4 5 6 7 8 9][ 0 1 2 10 3 4 5 6 7 8 9]
[[1 1 1]
 [2 2 2]
 [3 3 3]]
[[ 1 1 1]
 [10 10 10]
 [ 2 2 2]
 [ 3 3 3]]
[[ 1 10 1 1]
[ 2 10 2 2]
[ 3 10 3 3]]
```

표 13-3. ndarray 객체의 메소드

메소드	설명
sum()	배열의 모든 요소들의 합계를 반환한다.
mean()	배열의 모든 요소들의 평균 값을 반환한다.
max()	배열의 요소들중에서 최댓값을 반환한다. 옵션 axis=0 은 각 열의 최댓값, axis=1은 각 행의 최댓값을 구하는 데 사용한다.
min()	배열의 요소들중에서 최솟값을 반환한다.
argmax()	열 또는 행의 요소들 중 최댓값을 가지는 요소의 인덱스를 반환한다.

표 13-3. ndarray 객체의 메소드(계속)

argmin()	열 또는 행의 요소들 중 최솟값을 가지는 요소의 인덱스를 반환한다.
where()	조건식에 따라 배열의 요소 값을 특정 값으로 변경한다.
sort()	배열을 오름차순으로 정렬한다. 2차원 배열인 경우에 sort(0)는 열을 중심으로 요소들을 정렬하고, sort(1)은 행을 중심으로 요소들을 정렬한다.
insert()	배열에 행 또는 열을 삽입한다.

전국 초등학교 학생 데이터 분석

- CSV 파일의 구조

```
지역,학교명,학급수,학생수,교사수
서울특별시 서초구,서울교육대학교부설초등학교,28,616,32
서울특별시 종로구,서울대학교사범대학부설초등학교,31,632,35
서울특별시 강남구,서울개일초등학교,31,837,38
서울특별시 강남구,서울구룡초등학교,25,492,30
서울특별시 강남구,서울논현초등학교,19,339,22
...
```

예제 13-12. CSV => 2차원 리스트

ex13.ipynb

```
import csv
f = open('school_2019.csv', 'r',
encoding='utf-8')
lines = csv.reader(f)


header = next(lines)
print(header)
```

```
list_data = []
for line in lines :
    list_data.append(line[:])


print(list_data)
f.close()
```

예제 13-12. CSV => 2차원 리스트(계속)

: : 실행 결과

```
['지역', '학교명', '학급수', '학생수', '교사수']
[['서울특별시 서초구', '서울교육대학교부설초등학교', '28', '616', '32'], ['서울특별시 종로구', '서울대학교사범대학부설초등학교', '31', '632', '35'], ['서울특별시 강남구', '서울개일초등학교', '31', '837', '38'], ['서울특별시 강남구', '서울구룡초등학교', '25', '492', '30'], ['서울특별시 강남구', '서울논현초등학교', '19', '339', '22'], ['서울특별시 강남구', '서울대곡초등학교', '44', '1226', '54'], ['서울특별시 강남구', '서울대도초등학교', '62', '2157', '73'], ['서울특별시 강남구', '서울대모초등학교', '32', '1084', '43'], ['서울특별시 강남구', '서울대왕초등학교', '39', '904', '48'], ...
```

예제 13-13. 2차원 리스트 => ndarray

ex13.ipynb

```
import csv
import numpy as np

f = open('school_2019.csv', 'r',
encoding='utf-8')

lines = csv.reader(f)
header = next(lines)

list_data = []
```

```
for line in lines :
    list_data.append(line[:])

length = len(list_data)
print(length)
data = np.zeros((length, 3), dtype='int32')

for i in range(length) :
    for j in range(3) :
        data[i][j] = list_data[i][j+2]
print(data)
f.close()
```

예제 13-13. 2차원 리스트 => ndarray(계속)

: : 실행 결과

```
6264
[[ 28 616 32]
 [ 31 632 35]
 [ 31 837 38]
 ...
 [ 36 985 42]
 [ 34 862 41]
 [ 26 603 31]]
```

예제 13-14. 전국 최대 학생 수의 학교

ex13.ipynb

```
import csv
import numpy as np

f = open('school_2019.csv', 'r',
encoding='utf-8')
lines = csv.reader(f)
header = next(lines)

list_data = []
for line in lines :
    list_data.append(line[:])
```

```
length = len(list_data)

data = np.zeros((length, 3), dtype='int32')

for i in range(length) :
    for j in range(3) :
        data[i][j] = list_data[i][j+2]

max_index = np.argmax(data, axis=0)
print(max_index)
```

예제 13-14. 전국 최대 학생 수의 학교(계속)

```
max_class = list_data[max_index[0]][1]
num_class = list_data[max_index[0]][2]

max_student =
list_data[max_index[1]][1]
num_student =
list_data[max_index[1]][3]

max_teacher =
list_data[max_index[2]][1]
num_teacher =
list_data[max_index[2]][4]
```

```
print('최대 학급수의 초등학교 : %s, 학급
수 : %s개 ' % (max_class, num_class))

print('최대 학생수의 초등학교 : %s, 학생
수 : %s명' % (max_student, num_student))

print('최대 교사수의 초등학교 : %s, 교사
수 : %s명' % (max_teacher, num_teacher))

f.close()
```

예제 13-14. 전국 최대 학생 수의 학교(계속)

: : 실행 결과

```
[ 138 1999 138]
최대 학급수의 초등학교 : 서울신정초등학교, 학급수 : 75개
최대 학생수의 초등학교 : 탄벌초등학교, 학생수 : 2178명
최대 교사수의 초등학교 : 서울신정초등학교, 교사수 : 91명
```

예제 13-15. 학생수와 교사수 비교

ex13.ipynb

```
import csv
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
rc('font', family='Malgun Gothic')

f = open('school_2019.csv', 'r',
encoding='utf-8')
lines = csv.reader(f)

header = next(lines)
```

```
schools = {}

for line in lines:
    if ('광주' in line[0]) and ('마재' in line[1]) :
        schools.update({'마재':[line[2], line[3], line[4]]})
    if ('울산' in line[0]) and ('약사' in line[1]) :
        schools.update({'약사':[line[2], line[3], line[4]]})
    if ('용인' in line[0]) and ('정평' in line[1]) :
        schools.update({'정평':[line[2], line[3], line[4]]})
    if ('제주' in line[0]) and ('도평' in line[1]) :
        schools.update({'도평':[line[2], line[3], line[4]]})
print(schools)
```

예제 13-15. 학생수와 교사수 비교(계속)

```
data = np.zeros((4,3), dtype='int32')

school = list(schools.values())

for i in range(len(schools)) :
    for j in range(3) :
        data[i][j] = school[i][j]

print(data)
```

```
data = np.insert(data, 2, 0, axis=1)
data = np.insert(data, 4, 0, axis=1)
print(data)

row = data.shape[0]
for i in range(row) :
    data[i][2] = round(data[i][1]/data[i][0])
    data[i][4] = round(data[i][1]/data[i][3])

print(data)
```

예제 13-15. 학생수와 교사수 비교(계속)

```
xdata = ['마재', '약사', '정평', '도평']
plt.plot(xdata, data[:, 1], label='학생수',  color='red', linestyle='--', marker='x')
plt.plot(xdata, data[:, 3], label='교사수', color='blue', linestyle=':', marker='d')
plt.title('마재/약사/정평/도평 초등학교 학생수와 교사수')
plt.legend(loc='best')
plt.show()

plt.plot(xdata, data[:, 2], label='학급당 학생수',  color='magenta', linestyle='-.', marker='o')
plt.plot(xdata, data[:, 4], label='교사 1인당 학생수', color='green', linestyle='--', marker='x')
plt.title('마재/약사/정평/도평 초등학교 학급당 학생수 및 교사 1인당 학생수')
plt.legend(loc='best')
plt.show()
f.close()
```

예제 13-15. 학생수와 교사수 비교(계속)

: : 실행 결과

```
{'마재': ['24', '481', '28'], '약사': ['29', '685', '33'], '정평': ['36', '945', '43'], '도평': ['14', '300', '17']}
[[ 24 481 28]
 [ 29 685 33]
 [ 36 945 43]
 [ 14 300 17]]
```

```
[[ 24 481 0 28 0]
 [ 29 685 0 33 0]
 [ 36 945 0 43 0]
 [ 14 300 0 17 0]]
[[ 24 481 20 28 17]
 [ 29 685 24 33 21]
 [ 36 945 26 43 22]
 [ 14 300 21 17 18]]
```

예제 13-15. 학생수와 교사수 비교(계속)