18강

numpy

18-1 numpy 개요

- numpy는 numerical python의 줄임 말로 수치 데이터를 다루기 위한 라이브러리이다.
- numpy는 1995년 numeric으로 발표했다가 2006년도에 지금의 이름인 numpy로 발표가 되었다.
- 다차원 배열 자료구조인 ndarray를 지원한다.
- 수치 데이터의 벡터 연산과 여러 수치 연산 함수를 지원한다.
- 데이터의 연산을 간결하게 할 수 있는 구조를 제공한다.
- 선형대수를 지원하면, 각종 난수를 발생할 수 있는 함수를 제공한다.

18-1 numpy 개요

- numpy의 핵심 기능인 N차원의 배열 객체 ndarray는 대규모 데이터 집합을 담을 수 있는 빠르고 유연한 자료 구조다.
- 배열은 스칼라 원소 간의 연산에 사용하는 문법과 비슷한 방식을 사용해서 전체 데이터 블록에 수학적인 연산을 수행한다.
- ndarray는 같은 종류의 데이터를 담을 수 있는 포괄적인 다차원 배열이며 ndarray의 모든 원소는 같은 자료형이다.
- 데이터 분석을 위한 중요 기능은 아래와 같다.
  - ✓ 벡터 배열 상에서 데이터 개조, 정제, 부분 집합, 필터링
  - ✓ 정렬, 유일 원소 찾기, 집합 연산 같은 일반적인 배열 처리 알고리즘
  - ✓ 통계의 효과적인 표현과 데이터의 수집과 요약

18-2 ndarray 생성

- ndarray 객체 생성 방법
  - ✓ numy.array()를 이용하여 ndarray(배열화 객체)를 생성할 수 있다.
  - ✓ ndarray의 크기와 데이터타입은 shape와 dtype을 통해 알 수 있다.

```
import numpy as np

data1 = [6, 7.5, 8, 0, 1]
arr1 = np.array(data1)
print(arr1)
print(data1)

data2 = [[1,2,3,4], [5,6,7,8]]
arr2 = np.array(data2)
print(arr2)

print(arr2.shape)
print(arr2.dtype)
```

[결과]

```
[6.  7.5 8.  0.  1. ]
[6, 7.5, 8, 0, 1]
[[1 2 3 4]
 [5 6 7 8]]
(2, 4)
int32
```

18-2 ndarray 생성

- ndarray 객체 생성 함수
  - ✓ Ndarray는 아래의 생성 함수를 이용해서도 생성할 수 있다.

함수	설명
array	입력데이터(리스트, 튜플, 배열 또는 순차형 데이터)를 ndarray로 변환. dtype이 명시되지 않으면 자료형을 추정하고 입력데이터는 복사해서 생성
arange	내장 range함수와 유사하나 리스트 대신 ndarray 반환
linspace	주어진 데이터범위를 일정한 간격으로 분할해서 배열을 만든 후 튜플로 반환
ones	주어진 dtype으로 배열을 생성하고 내용을 모두1로 초기화
zeros	ones와 같지만 내용을 0으로 채움
empty	메모리를 할당하여 새로운 배열을 생성하지만 값을 초기화하지 않음

```
import numpy as np

arr1 = np.arange(1,10,2)
print(arr1)

arr2 = np.linspace(-2,2,4)
print(arr2)

arr3 = np.ones([2,3])
print(arr3)

arr4 = np.zeros([2,3])
print(arr4)
```

[결과]

```
[1 3 5 7 9]
[-2.         -0.66666667  0.66666667  2.        ]
[[1. 1. 1.]
 [1. 1. 1.]]
[[0. 0. 0.]
 [0. 0. 0.]]
```

18-2 ndarray 생성

- 스칼라연산과 ndarray연산
  - ✓ for 반복문을 작성하지 않고 데이터를 일괄 처리할 수 있다.
  - ✓ 스칼라 연산을 ndarray에 적용하면 모든 요소에 연산이 이루어진다.
  - ✓ 같은 크기의 배열 간 산술 연산은 배열의 각 요소 단위로 적용된다.

```
import numpy as np

arr = np.array([[1., 2., 3.], [4., 5., 6.]])
print(arr)
print(arr/2)
print(arr * arr)
print(arr - arr)

arr2 = np.array([[0., 4., 1.], [7., 2., 12.]])
print(arr2 > arr)
```

[결과]

```
[[1. 2. 3.]
 [4. 5. 6.]]

[[0.5 1.  1.5]
 [2.  2.5 3. ]]

[[ 1.  4.  9.]
 [16. 25. 36.]]

[[0. 0. 0.]
 [0. 0. 0.]]

[[False  True False]
 [ True False  True]]
```

18-2 ndarray 생성

- ndarray 인덱싱
  - ✓ 1차원 ndarray의 인덱싱은 파이썬 시퀀스 객체의 인덱싱과 유사하다.
  - ✓ 다차원 ndarray는 배열은 [I,j], 형태로 인덱싱할 수 있다.
  - ✓ ndarray는 불리언 색인을 지원한다.

```
import numpy as np

arr = np.arange(10)
print(arr[5])
print(arr[5:8])
arr[5:8] = 12
print(arr)


arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr[:2])
print(arr[:2, 1:])

arr2 = np.array([[True,False,False],[True,False,True],[False,True,False]])
print(arr[arr2])
```

```
[결과]
5

[5 6 7]

[ 0  1  2  3  4 12 12 12  8  9]

[[1 2 3]
 [4 5 6]]

[[2 3]
 [5 6]]

[1 4 6 8]
```

18-3 ndarray 함수

- **재형성**
  - ✓ ndarray는 데이터를 복사하지 않고 다른 모양으로 바꿀 수 있는 reshape()를 제공한다.

```
import numpy as np

arr = np.arange(8)
arr2 = arr.reshape((4, 2))
print(arr2)

arr3 = arr.reshape((4, 2)).reshape((2, 4))
print(arr3)
```

```
[결과]
[[0 1]
 [2 3]
 [4 5]
 [6 7]]

[[0 1 2 3]
 [4 5 6 7]]
```

- **정렬**
  - ✓ 파이썬의 내장 리스트형처럼 numpy에서도 sort 함수를 이용하여 정렬 할 수 있다.

```
import numpy as np

arr = np.array([2,5,9,1,3,7])
arr.sort()
print(arr)
```

```
[결과]
[1 2 3 5 7 9]
```

18-3 ndarray 함수

- 난수 생성
  - ✓ numpy.random 모듈은 다양한 종류의 확률분포로부터 효과적으로 표본 값을 생성한다.
  - ✓ numpy.random 모듈은 파이썬 내장 random 모듈에 비해 더 효율적으로 값을 생성하여 파이썬 내장 모듈보다 수십 배 이상 빠른 속도를 자랑한다.

함수	설명
seed	난수 발생기의 seed를 지정한다.
permutation	임의의 순열을 반환한다.
shuffle	리스트나 배열의 순서를 뒤섞는다.
rand	균등분포에서 표본을 추출한다.
randint	주어진 최소/최대 범위 안에서 임의의 난수를 추출한다.
randn	표준편차가 1이고 평균값이 0인 정규분포에서 표본을 추출한다.
binomial	이항분포에서 표본을 추출한다.
beta	베타분포에서 표본을 추출한다.
chisquare	카이제곱 분포에서 표본을 추출한다.
gamma	감마분포에서 표본을 추출한다.
uniform	균등(0, 1)에서 표본을 추출한다.

18-3 ndarray 함수

- 통계함수
  - ✓ ndarray의 통계 값을 추출하기 위해 사용하는 함수 이다.
  - ✓ ndarray 전체 혹은 한 축의 자료를 통계 함수에 적용하여 계산 할 수 있다.

sum(), mean() : 배열 전체 합, 평균
cumsum(), cumprod() : 누적 합, 누적 곱
std(), var() : 표준편차, 분산
min(), max() : 최소값, 최대값
argmin(), argmax() : 최소 원소의 색인 값, 최대 원소의 색인 값

18-3 ndarray 함수

- 통계함수
  - ✓ 적용 코드

```
import numpy as np

arr = np.random.randn(5, 4)
print(arr)
print('mean',np.mean(arr))
print('sum:',arr.sum())

print(arr.mean(axis=1))
print(arr.sum(axis=0))

arr = np.array([0, 1, 2, 3, 4, 5, 6, 7])
print(arr.cumsum())

arr = np.array([[0, 1, 2],
                [3, 4, 5],
                [6, 7, 8]])

print(arr.cumsum(axis=0))
print(arr.cumprod(axis=1))
```

[결과]

```
[[-0.29012688 -0.33110504  0.03470932 -1.41980985]
 [-1.10256146 -1.04342239 -1.06138052  1.50809397]
 [ 0.55348689  0.98198936  1.20125565  1.80516035]
 [ 1.17875836 -0.77161277 -0.70690687 -0.25478024]
 [-2.04398141 -0.03122163  1.69248503 -0.91239554]]

mean -0.05066828381493975

sum: -1.013365676298795

[-0.50158311 -0.4248176   1.13547306 -0.13863538 -0.32377839]

[-1.70442451 -1.19537246  1.1601626   0.72626869]

[ 0  1  3  6 10 15 21 28]

[[ 0  1  2]
 [ 3  5  7]
 [ 9 12 15]]

[[  0   0   0]
 [  3  12  60]
 [  6  42 336]]
```