python™

Numpy

Numpy 라이브러리

- 2005년에 Travis Oliphant가 발표한 수치해석용 Python 패키지
- C로 구현된 CPython에서만 사용할 수 있으며 Jython, IronPython, PyPy등의 Python 구현에서 사용할 수 없음
- C로 구현된 내부 반복문을 사용하므로 Python 반복문과 비교해 속도가 빠름
- 숫자를 다루는 금융, 선형대수를 위한 패키지
- 수학식, 행렬, 벡터

```
1 import numpy as np
2 np.__version__
```

```
'1.21.5'
```

Numpy 라이브러리

- 리스트는 수학 계산이 안됨

```
1  a = [1,2,3]
2  b = [9,8,7]
3
4  a * b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [9], in <cell line: 4>()
      1 a = [1,2,3]
      2 b = [9,8,7]
----> 4 a * b

TypeError: can't multiply sequence by non-int of type 'list'
```

```
1  a = [1,2,3]
2  b = [9,8,7]
3
4  np_a = np.array(a)
5  np_b = np.array(b)
6
7  np_a * np_b
```

```
array([ 9, 16, 21])
```

- 넘파이의 array는 계산 가능

Numpy 라이브러리

- ndarray 생성

```
1  mylist2 = [[1, 2, 3, 4],
2             [5, 6, 7, 8]]
3
4  print(mylist2)
5  arr2 = np.array(mylist2)
6  print(arr2)
```

```
[[1, 2, 3, 4], [5, 6, 7, 8]]
[[1 2 3 4]
 [5 6 7 8]]
```

- ndarray shape

```
1  arr2.shape
```

```
(2, 4)
```

Numpy 라이브러리

- ndarray 데이터 타입

```
1  np.array([리스트], dtype=자료형)
```

```
1  arr = np.array([1, 2, 3, 3.14], dtype=int)
2  arr
```

```
array([1, 2, 3, 3])
```

```
1  arr = np.array([1, 2, 3, 3.14], dtype=float)
2  arr
```

```
array([1.  , 2.  , 3.  , 3.14])
```

```
1  arr = np.array([1, 2, 'a', '3.14'], dtype=int)
2  arr
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [19], in <cell line: 1>()
----> 1 arr = np.array([1, 2, 'a', '3.14'], dtype=int)
      2 arr

ValueError: invalid literal for int() with base 10: 'a'
```

Numpy 라이브러리

- ndarray 인덱싱, 리스트와 같다

```
1 arr2d = np.array([[1, 2, 3, 4],
2                   [5, 6, 7, 8],
3                   [9, 10, 11, 12]])
```

```
1 arr2d[0, :]
```

```
array([1, 2, 3, 4])
```

```
1 arr2d[:, 2]
```

```
array([ 3,  7, 11])
```

```
1 arr2d[:2, :]
```

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```
1 arr2d[:2, 2:]
```

```
array([[3, 4],
       [7, 8]])
```

```
1 arr2d[[0, 2], :]
```

```
array([[ 1,  2,  3,  4],
       [ 9, 10, 11, 12]])
```

```
1 arr2d[:, [1, 2, 3]]
```

```
array([[ 2,  3,  4],
       [ 6,  7,  8],
       [10, 11, 12]])
```

```
1 arr2d[ arr2d > 5]
```

```
array([ 6,  7,  8,  9, 10, 11, 12])
```

Numpy 라이브러리

- np.arrange(start, stop, step) 배열 생성

```
1  mylist = []
2
3  for i in range(1, 11, 2):
4      mylist.append(i)
5
6  print(mylist)
```

```
[1, 3, 5, 7, 9]
```

```
1  arr = np.arange(1, 11, 2)
2  print(arr)
```

```
[1 3 5 7 9]
```

Numpy 라이브러리

- np.sort(array) 순정렬, 역정렬

```
1 arr = np.array([1, 10, 5, 8, 2, 4, 3, 6, 8, 7, 9])
2 print(np.sort(arr)[::1])
3 print(np.sort(arr)[::-1])
```

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

- 정렬 된 값은 유지가 안됨

```
1 print(arr)
2 arr = np.sort(arr)
3 print(arr)
```

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

```
1 print(arr)
2 arr.sort()
3 print(arr)
```

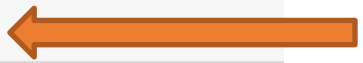
```
[ 1 10  5  8  2  4  3  6  8  7  9]
[ 1  2  3  4  5  6  7  8  8  9 10]
```

Numpy 라이브러리

- np.sort(array) 2차원 정렬

```
1 arr2d = np.array([[3, 4, 7, 8],
2                   [5, 6, 2, 1],
3                   [10, 9, 12, 11]])
```

```
1 np.sort(arr2d, axis=1)
```



가로 방향, 같은 행 안에서 정렬

```
array([[ 3,  4,  7,  8],
       [ 1,  2,  5,  6],
       [ 9, 10, 11, 12]])
```

```
1 np.sort(arr2d, axis=0)
```

세로 방향, 같은 열 끼리 정렬

```
array([[ 3,  4,  2,  1],
       [ 5,  6,  7,  8],
       [10,  9, 12, 11]])
```

Numpy 라이브러리

- 행렬 덧셈

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([[3, 4, 5],
5                [1, 2, 3]])
6
7  a + b
```

```
array([[4, 6, 8],
       [3, 5, 7]])
```

- 행렬 뺄셈

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([[3, 4, 5],
5                [1, 2, 3]])
6
7  a - b
```

```
array([[-2, -2, -2],
       [ 1,  1,  1]])
```

Numpy 라이브러리

- 행렬 모양이 다르면 계산 할 수 없다

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([[1, 2],
5                [3, 4],
6                [5, 6]])
7
8  a + b
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [63], in <cell line: 8>()
      1 a = np.array([[1, 2, 3],
      2                [2, 3, 4]])
      4 b = np.array([[1, 2],
      5                [3, 4],
      6                [5, 6]])
----> 8 a + b

ValueError: operands could not be broadcast together with shapes (2,3) (3,2)
```

Numpy 라이브러리

- array.reshape(행, 열) 배열의 모양을 바꿔준다

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([[1, 2],
5                [3, 4],
6                [5, 6]])
7
8  b = b.reshape(2,3)
9  print(b)
10 print(a + b)
```

```
[[1 2 3]
 [4 5 6]]
[[ 2  4  6]
 [ 6  8 10]]
```

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3  a.reshape(3,-1)
```

```
array([[1, 2],
       [3, 2],
       [3, 4]])
```

- reshape() 의 인수에서 -1 은 지정하지 않음의 의미.

Numpy 라이브러리

- 행렬 곱셈

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([[3, 4, 5],
5                [1, 2, 3]])
6
7  a * b
```

```
array([[ 3,  8, 15],
       [ 2,  6, 12]])
```

- 행렬곱연산

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([[1, 2],
5                [3, 4],
6                [5, 6]])
7
8  np.dot(a, b)
```

```
array([[22, 28],
       [31, 40]])
```

$$A \times B = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix}$$

(출처) https://mathbang.net/562

Numpy 라이브러리

- Broadcasting

모양이 서로 다른 배열들 간에, 어떤 조건을 만족했을 때
계산이 가능해지도록 자동적으로 모양을 변환 하는 것

01 맴버가 하나인 배열은 가능

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([1])
5
6  a + b
```

```
array([[2, 3, 4],
       [3, 4, 5]])
```

Numpy 라이브러리

- Broadcasting

02 하나의 배열의 차원이 1인 경우

```
1  a = np.array([[1, 2, 3],
2                [2, 3, 4]])
3
4  b = np.array([1, 2, 3])
5
6  a + b
```

```
array([[2, 4, 6],
       [3, 5, 7]])
```

Numpy 라이브러리

- Broadcasting

03 차원의 짝이 맞는 경우

```
1  a = np.array([1, 2, 3])
2
3  b = np.array([[1],
4                [2],
5                [3]])
6
7  a + b
```

```
array([[2, 3, 4],
       [3, 4, 5],
       [4, 5, 6]])
```