Numpy

Python

저녁이 있는 프로젝트
오상훈
6 Hours, 1 Month

Intro

- It stands for '**Numerical Python**'. It is a library consisting of **multidimensional array objects** and a collection of routines for processing of array

- 불러오기

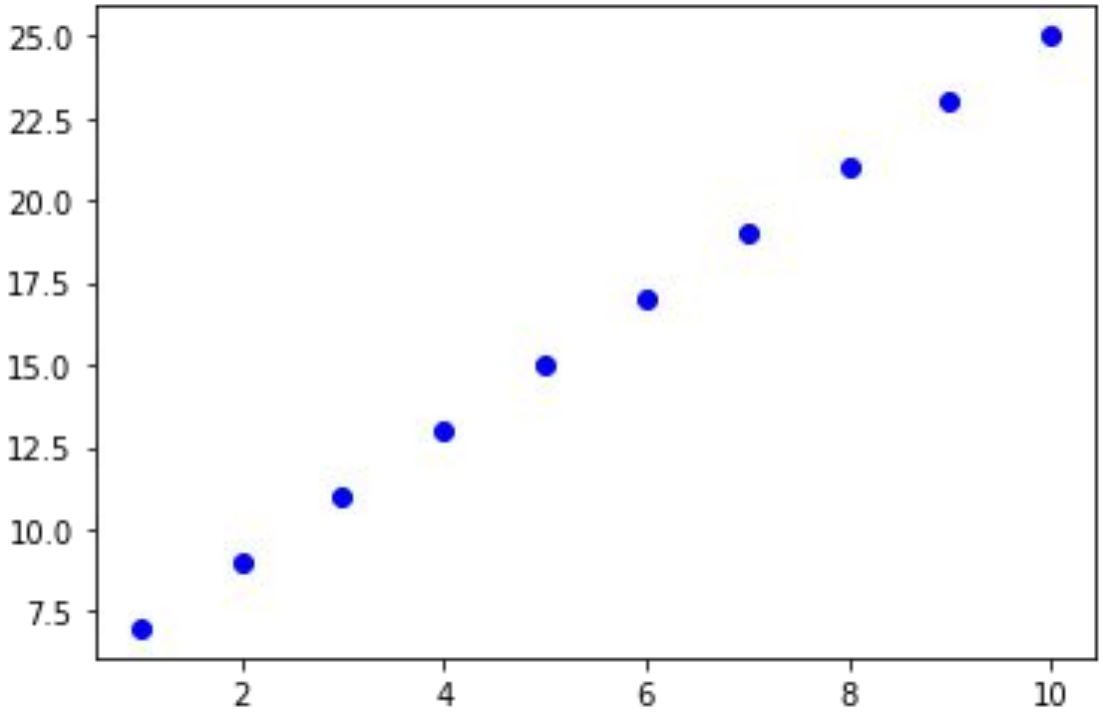
  import numpy **as np**

- 같이 하기

  from matplotlib import pyplot as plt

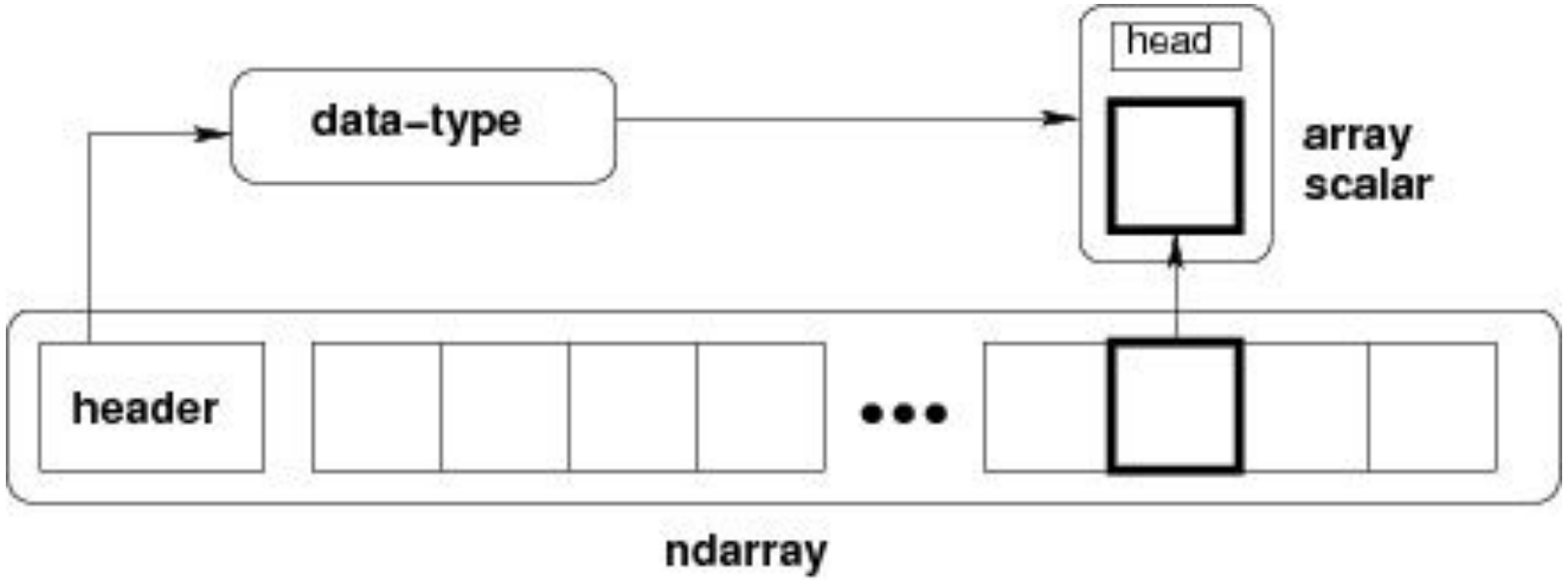
```
x = np.arange(1,11)
y = 2 * x + 5

plt.plot(x,y)
plt.show()
```



Ndarray Object

- NumPy is an **N-dimensional array type** called ndarray.
- numpy.**array**(**object**, **dtype** = None, copy = True, order = None, **ndmin** = 0)
  arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
  - **object** : Any object exposing the array interface method returns an array, or any (nested) sequence.
  - **dtype** : Desired data type of array, optional
    'b'−boolean, 'i'−(signed) integer, 'u'−unsigned integer, 'f'−floating-point
    'c'−complex-floating point, 'm'−timedelta, 'M'−datetime, 'O'−(Python) objects
    'S', 'a'−(byte-)string, 'U'−Unicode, 'V'−raw data (void)
  - **ndmin** : Specifies minimum dimensions of resultant array



Temporary Array Creation

- numpy.empty(shape, dtype = float, order = 'C')

  x = np.empty([3,2], dtype = int)
- numpy.zeros(shape, dtype = float, order = 'C')

  x = np.zeros((2,2), dtype = [('x', 'i4'), ('y', 'i4')])
- numpy.ones(shape, dtype = None, order = 'C')

  x = np.ones([2,2], dtype = int)
- numpy.asarray(list, dtype = None, order = None)

  x = (1,2,3)

  a = np.asarray(x)
- numpy.fromiter(iterable, dtype, count = -1)

  list = range(5)

  it = iter(list)

  x = np.fromiter(it, dtype = float)
- numpy.arange(start, stop, step, dtype)

  x = np.arange(10,20,2)
- numpy.logspace(start, stop, num, endpoint, base, dtype)

  a = np.logspace(1,10,num = 10, base = 2)

What different is List and Ndarray

```
>>> data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> result = data * 2
>>> result, type(result)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x = np.array(data)
>>> x, type(x)
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]), numpy.ndarray)
>>> result = x + 2
>>> result, type(result)
(array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11]), numpy.ndarray)
>>> x * 2, x // 2
(array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18]),
 array([0, 0, 1, 1, 2, 2, 3, 3, 4, 4]))
>>> a = np.array([1, 2, 3])
>>> b = np.array([10, 20, 30])
>>> result = 2 * a + b
>>> result, type(result), result.ndim
(array([12, 24, 36]), numpy.ndarray, 1)
```

N-dimension and reshape

❖ N-dimension

```
>>> c = np.array([[0, 1, 2], [3, 4, 5]])
>>> c, len(c), len(c[0]), c.shape, type(c)
>>> d = np.array([[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
                  [[11, 12, 13, 14], [15, 16, 17, 18], [19, 20, 21, 22]]])
>>> d, d.shape, len(d), len(d[0]), len(d[0][0])
```

❖ reshape : 배열 크기 변환

```
>>> a = np.arange(12)
>>> a, a.shape, type(a)
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]), numpy.ndarray)
>>> b = a.reshape(3, 4)
>>> b, b.shape, type(b)
(array([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]]), (3, 4), numpy.ndarray)
```

❖ transpose : 전치 연산(행과 열

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

Indexing

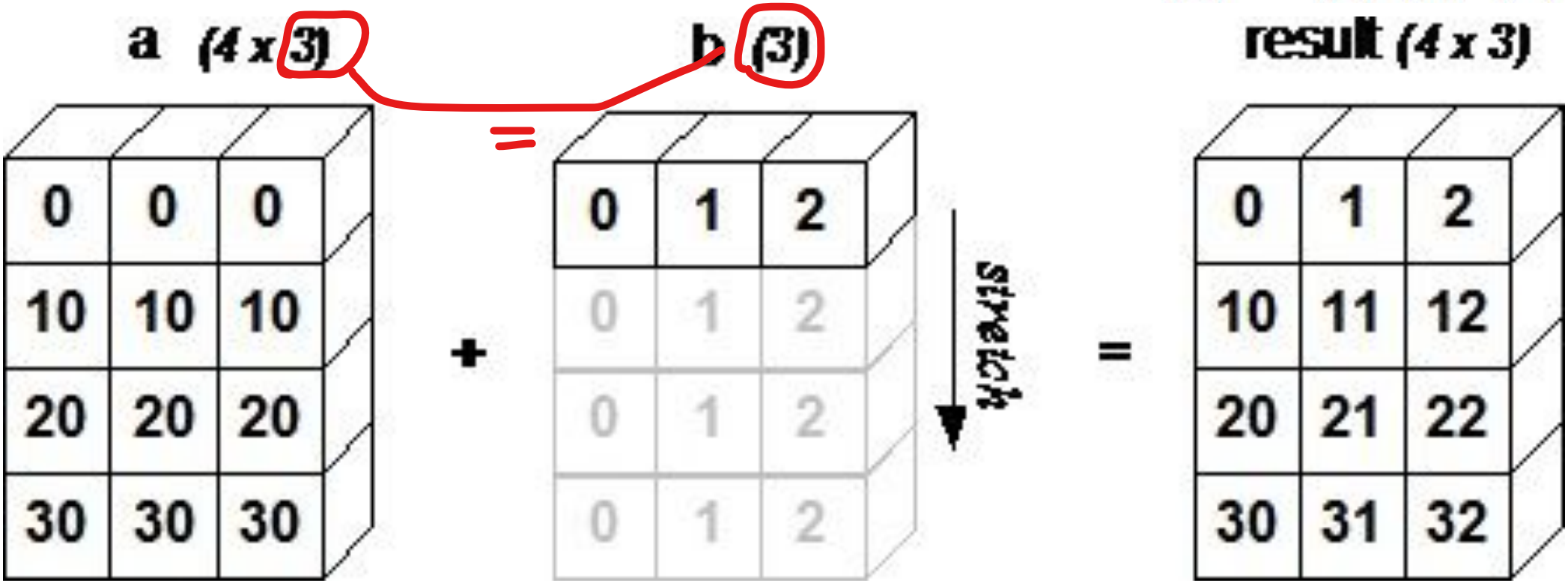
```
>>> b = np.array([[0, 1, 2], [3, 4, 5]])
>>> b, b[0][0], b[0, 0], b[0, 1], b[0][1], b[-1, -1], b[-1][-1]
>>> x = np.array([[1, 2], [3, 4], [5, 6]])
>>> y = x[[0,1,2], [0,1,0]]
>>> x, x.shape, y, y.shape
(array([[1, 2], [3, 4], [5, 6]]), (3, 2),
array([1, 4, 5]), (3,))
>>> y = x[1:4,[1,2]]
>>> x, x.shape, y, y.shape
(array([[ 0,  1,  2], [ 3,  4,  5], [ 6,  7,  8], [ 9, 10, 11]]), (4, 3),
array([[ 4,  5], [ 7,  8], [10, 11]]), (3, 2))
>>> a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> idx = np.array([True, False, True, False, True, False, True, False, True, False])
>>> a[idx], a[a % 2 == 0]    # Take True array
(array([0, 2, 4, 6, 8]), array([0, 2, 4, 6, 8]))
>>> x = np.array([[ 0,  1,  2],[ np.nan,  4,  5],[ 6,  7,  8],[ 9, 10, 11]])
>>> x, x[x > 5], x[np.isnan(x)]
(array([[ 0.,  1.,  2.], [nan,  4.,  5.], [ 6.,  7.,  8.],  [ 9., 10., 11.]]),
array([ 6.,  7.,  8.,  9., 10., 11.]),
array([nan]))
```

Slicing & Broadcasting

- [0:n] 0번째부터 n-1번째까지, [-1] 제일 끝에 있는 배열값 반환
  ex) [:5] 0번째부터 4번째까지,5번 비포함, [2:] 2번째부터 끝까지
  a = np.array([[0, 1, 2, 3], [4, 5, 6, 7]])
  a, a[0], a[0, :], a[0][:], a[:2, :2], a[:2][:2]

- Broadcasting (10.NumPy_Python.ipynb > '배열 연산' 참조

$$\begin{bmatrix}0\\1\\2\\3\end{bmatrix} + 1 = \begin{bmatrix}0\\1\\2\\3\end{bmatrix} + \begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\2\\3\\4\end{bmatrix}$$



차원 축소(Dimension Reduction)

- 최대/최소: min, max, argmin, argmax
- 통계: sum, mean(평균), median(중간값), std(표준편차), var(분산)

```
x = np.array([1, 2, 3, 1])
x.sum(), x.max(), x.argmin(), x.argmax(), x.mean(), np.median(x), x.std()
x = np.array([[0, 1, 2], [7,8,9], [2, 3, 4], [4, 5, 6]])
x.sum(), x.max(), x.argmin(), x.argmax(), x.mean(), np.median(x), x.std()
```

Try - Ndarray 다루기

❖ 해 보기

```
# {'num':'1', 'name':'김철수','kor':90, 'eng':80, 'math':85, 'total':0, 'avg':0.0}
student_score = [
        [1, 90, 80, 85, 0, 0.0, 0],
        [2, 90, 85, 90, 0, 0.0, 0],
        [3, 80, 80, 80, 0, 0.0, 0]
]
ex) ndarray[:, 4] = np.sum(ndarray[:,1:4],axis=0)
```

➢ 각 구성원 total, avg 필드 각 생성과 값 구하기

➢ 학급 평균 구하기

➢ 과목별 평균 구하기

❖ 해보기

➢ 아래와 같은 dictionary type을 위와 같은 결과 출력

```
students_score = [
      {'num':'1', 'name':'김철수','kor':90, 'eng':80, 'math':85},
      {'num':'2', 'name':'박제동','kor':90, 'eng':85, 'math':90},
      {'num':'3', 'name':'홍길동','kor':80,' eng':80, 'math':80}
]
```

Array Manipulation(1)

- numpy.concatenate((a1, a2, ...), axis) : Joins a sequence of arrays along an existing axis

```
>>> a = np.array([[1,2],[3,4]])
>>> b = np.array([[5,6],[7,8]])
>>> np.concatenate((a,b)), np.concatenate((a,b),axis=0), np.concatenate((a,b),axis=1)
(array([[1, 2], [3, 4], [5, 6], [7, 8]]),
array([[1, 2], [3, 4], [5, 6], [7, 8]]),
array([[1, 2, 5, 6], [3, 4, 7, 8]]))
```

- numpy.split(ary, indices_or_sections, axis)

```
>>> a = np.arange(9)
>>> b = np.split(a,3)   # case 2 - TypeError: object of type 'int' has no len()
>>> a, type(a), b, type(b)
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]), numpy.ndarray,
 [array([0, 1, 2]), array([3, 4, 5]), array([6, 7, 8])], list)
```

Array Manipulation(2)

❖ numpy.resize(arr, shape)

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

❖ numpy.insert(arr, obj, values, axis)

```
>>> a = np.array([[1,2],[3,4],[5,6]])
(array([ 1,  2,  3, 11, 12,  4,  5,  6]),
array([[ 1,  2], [11, 11], [ 3,  4], [ 5,  6]]),
array([[ 1, 11,  2], [ 3, 11,  4],  [ 5, 11,  6]]))
```

❖ Numpy.delete(arr, obj, axis)

```
>>> a = np.arange(12).reshape(3,4)
>>> a, np.delete(a,5), np.delete(a,1,axis = 1)
(array([[ 0,  1,  2,  3],  [ 4,  5,  6,  7], [ 8,  9, 10, 11]]),
 array([ 0,  1,  2,  3,  4,  6,  7,  8,  9, 10, 11]),
 array([[ 0,  2,  3], [ 4,  6,  7], [ 8, 10, 11]]))
```


수고하셨습니다.