IPA 주관 인공지능센터 기본(fundamental) 과정

- **GitHub link: here**
- **E-Mail: windkyle7@gmail.com**

NumPy Part 3

In [1]:

```
import numpy as np
```

행과 열의 개념을 `np.newaxis`를 통해 다시 한번 살펴보고자 한다.

In [2]:

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

In [3]:

```
a
```

Out[3]:

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

In [4]:

```
a[np.newaxis]
```

Out[4]:

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

In [5]:

```
a[:, np.newaxis]
```

Out[5]:

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

In [6]:

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

In [7]:

```
b
```

Out[7]:

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

In [8]:

```
b[np.newaxis]
```

Out[8]:

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

In [9]:

```
b[:, np.newaxis]
```

Out[9]:

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

       [[4, 5, 6]]])
```

In [10]:

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

In [11]:

```
c
```

Out[11]:

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

In [12]:

```
c[np.newaxis]
```

Out[12]:

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

In [13]:

```
c[:, np.newaxis]
```

Out[13]:

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

       [[3, 4]],

       [[5, 6]]])
```

In [14]:

```
c[:, :, np.newaxis]
```

Out[14]:

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

       [[3],
        [4]],

       [[5],
        [6]]])
```

In [15]:

```
c[np.newaxis, :]
```

Out[15]:

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

In [16]:

```
c[np.newaxis, :, :]
```

Out[16]:

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

스칼라 (Scalar)

스칼라는 크기만 있고 방향을 가지지 않는 양을 의미한다.

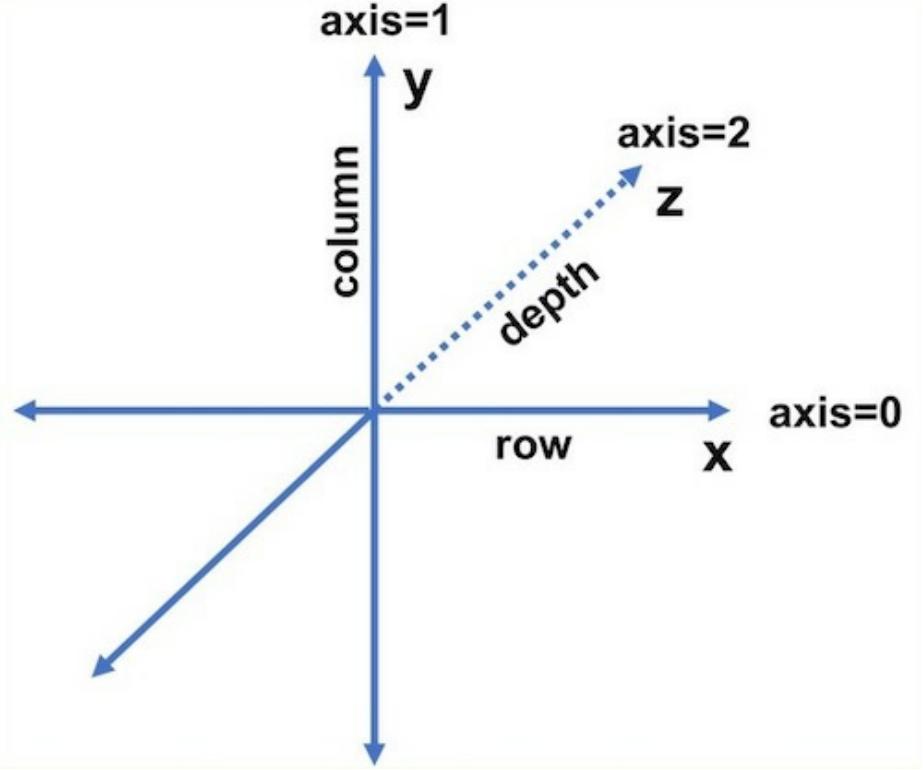
벡터 (Vector)

벡터는 방향과 크기를 가지는 것을 의미한다.

행렬 (Matrix)

행렬은 복수의 차원을 가지는 데이터가 다시 여러 개 있는 경우, 데이터를 합쳐서 표기한 것이다. 2차원 배열을 행렬이라고 부르며, 3차원 이상 배열은 텐서(tensor)라고 한다.

다차원 배열의 축(axis)



`벡터`는 x 축만을 갖는 자료형이다. 1차원 배열에 해당하는 벡터의 각 요소는 그 자체가 `열(row)`이다.

2차원 배열 형태의 `행렬`은 x축의 행과 y축의 `행(column)`을 갖는다. 2차원 배열 행렬은 개념적으로 `깊이(depth)`가 1이라고 생각할 수 있다.

3차원 배열 형태의 `텐서`는 행과 열을 갖고 각 컬럼은 벡터 형태를 갖는다. 이러한 벡터를 깊이 `(depth)`로 표현한다.

In [17]:

```
a = np.array([[[0, 1, 2], [3, 4, 5]], [[6, 7, 8], [9, 10, 11]]])
```

In [18]:

```
a
```

Out[18]:

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

위와 같이 **3**차원 배열이 있다고 하였을때, 각 의미를 풀이해보면 다음과 같다.

```
[
    [ # 열(Row)
        [0,  1,  2], # 행(Column)
      #  -   -   -     깊이(Depth) 1 ~ 3
        [3,  4,  5]  # 행(Column)
      #  -   -   -     깊이(Depth) 1 ~ 3
    ],
    [ # 열(Row)
        [6,  7,  8], # 행(Column)
      #  -   -   -     깊이(Depth) 1 ~ 3
        [9, 10, 11]  # 행(Column)
      #  -   -   -     깊이(Depth) 1 ~ 3
    ]
]
```

`np.genfromtxt`를 통해 `csv` 파일을 불러온다. 이 외에 불러오는 방식이 여러 가지가 있으며, 각각 차이가 있기 때문에 불러오는 옵션을 체크하지 않고 불러오면 잦은 에러를 발생시킬 수 있다.

In [19]:

```
%%writefile test.csv
1,2,3
4,5,6
```

```
Writing test.csv
```

In [20]:

```
x = np.genfromtxt('test.csv')
```

In [21]:

```
x
```

Out[21]:

```
array([nan, nan])
```

`np.fr` 까지 입력한 후 자동 완성키 `TAB` 키를 눌러보면 여러 방식으로 불러올 수 있는 것을 확인할 수 있다.

In [22]:

```
y = np.fromfile('test.csv')
```

In [23]:

```
y
```

Out[23]:

```
array([9.38200607e-96])
```

In [24]:

```
y = np.fromfile('test.csv', sep=',')
```

In [25]:

```
y
```

Out[25]:

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

In [27]:

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

ndarray 타입을 그대로 `npy` 파일로 저장하고 불러올 수 있다.

In [31]:

```
np.save('test', z)
```

In [34]:

```
z = np.load('test.npy')
```

In [35]:

```
z
```

Out[35]:

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

pickle

pickle 모듈은 파이썬 객체 구조의 직렬화와 역 직렬화를 위한 바이너리 프로토콜을 구현한다.

`피클링(pickling)`은 파이썬 객체 계층 구조가 바이트 스트림으로 변환되는 절차이며, `역 피클링(unpickling)`은 반대 연산으로, (바이너리 파일 이나 바이트열류 객체로부터의) 바이트 스트림을 객체 계층 구조로 복원한다. 피클링(그리고 역 피클링)은 `직렬화(serialization)`, `마샬링(marshalling)` 또는 `평탄화(flattening)` 라고도 한다. 그러나 혼란을 피하고자, 여기에서 사용된 용어는 `피클링`과 `역 피클링`이다.

In [36]:

```
import pickle
```

In [37]:

```
with open('pickle.ss', 'wb') as file:
    pickle.dump(z, file)
```

In [38]:

```
with open('pickle.ss', 'rb') as file:
    my_z = pickle.load(file)
```

In [39]:

```
my_z
```

Out[39]:

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

여기까지 **NumPy**를 간단히 살펴보았다. 위의 코드를 보면 알겠지만, 파일을 불러오는 데 불편한 점을 느낄 수 있다.

따라서 이제는 데이터를 불러올 때 **판다스(Pandas)** 패키지를 사용하여 데이터를 불러온 후 `탐색적 데이터 분석(EDA)` 및 `데이터 전처리(Pre-processing)`, `시각화(Visualization)`를 하는 방법에 대해 다루고자 한다.

Pandas & Scikit-learn

파이썬 수업이 끝이났으므로 다음 링크를 통해 업로드됩니다.

https://wind-kyle.github.io/ai-course-fundamentals/