

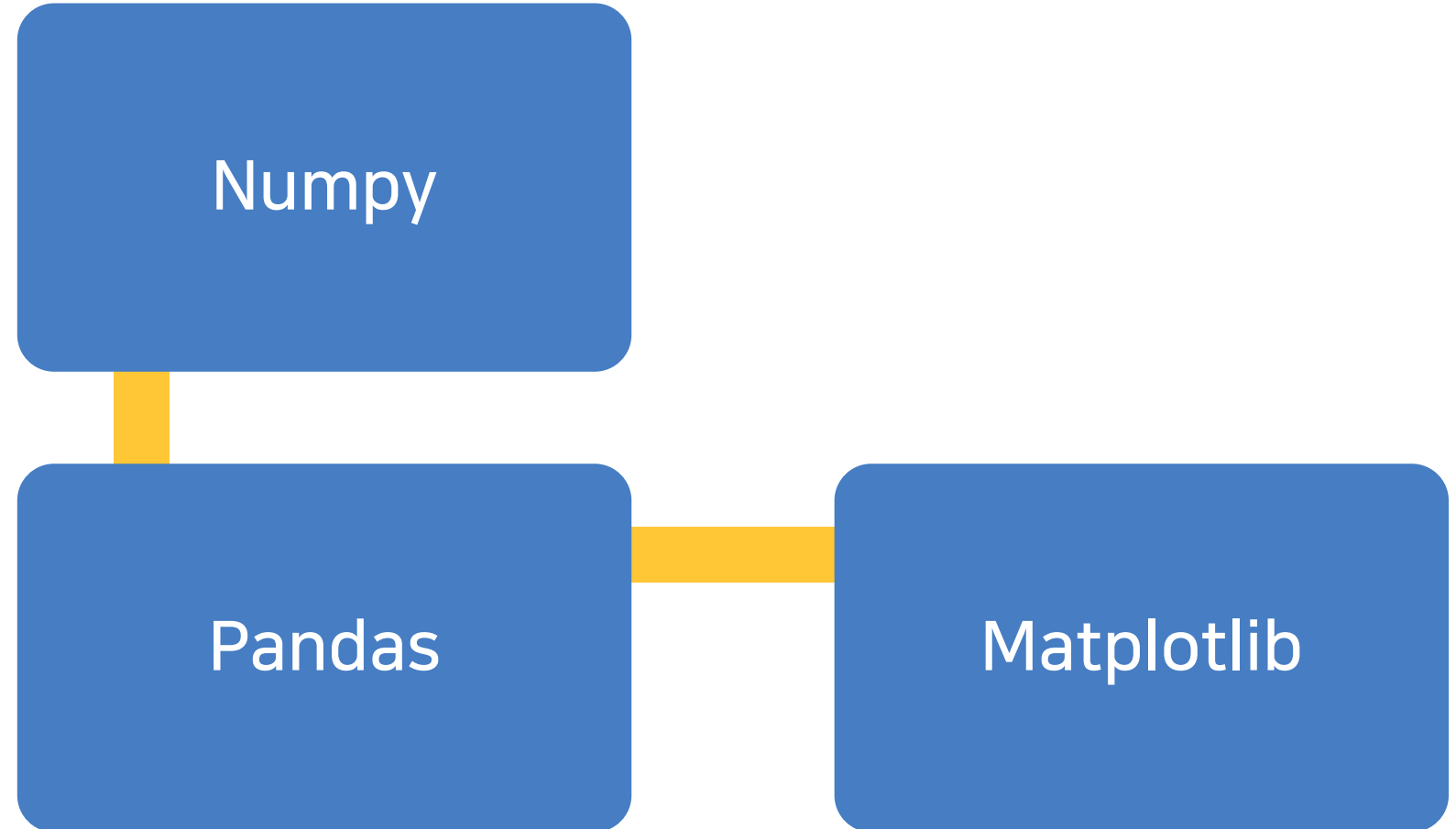


스마트인재개발원
Smart Human Resources Development

| 이상준 연구원



수업 진행방향

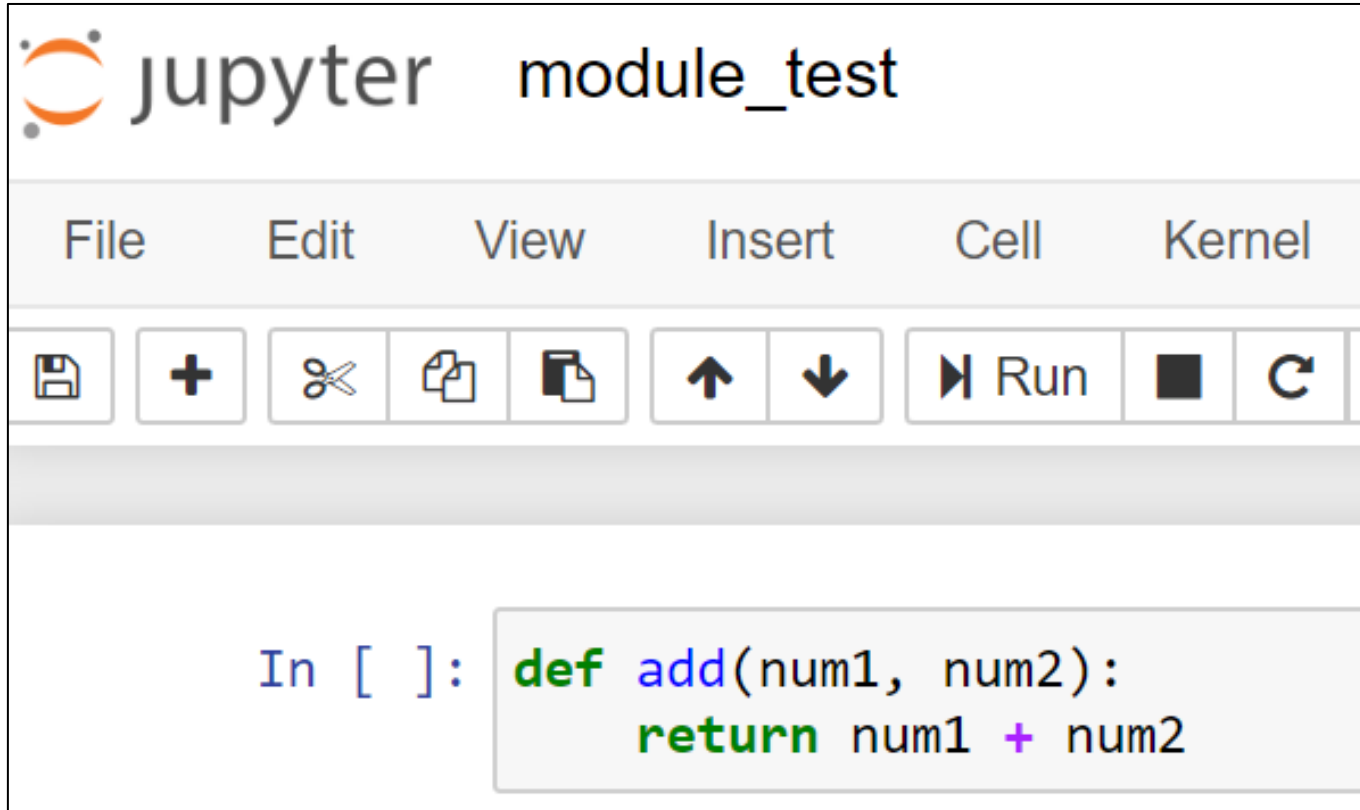




학습목표

- 모듈의 개념에 대해서 이해한다.
- Numpy라이브러리를 활용할 수 있다.

- 필요한 코드를 재사용하기 위해 변수나 함수 또는 클래스를 모아 놓은 **파일**
- 다른 파이썬 프로그램에서 불러와 사용할 수 있게끔 만든 파이썬 파일
- 모듈은 다른 사람이 이미 만들어 놓은 모듈을 사용할 수도 있고 직접 만들어서 사용할 수도 있다.
- 파이썬에서 사용할 수 있는 모듈은 **확장자가 .py파일** 이다.



The image shows a Jupyter Notebook window titled "module_test". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", and "Kernel". Below the menu bar is a toolbar with icons for saving, adding new cells, cutting, copying, pasting, moving cells up and down, running the cell, and refreshing. The main area displays a code cell with the following Python code:

```
In [ ]: def add(num1, num2):  
        return num1 + num2
```

- File → Download as → Python (.py) → 같은 경로로 옮기기

The image illustrates the process of downloading a Jupyter Notebook as a Python file in four steps:

- Step 1:** Click the **File** menu in the top-left corner of the Jupyter interface.
- Step 2:** Click **Download as** in the dropdown menu.
- Step 3:** Select **Python (.py)** from the list of file formats.
- Step 4:** The file **module_test.py** is saved in the same directory as the notebook, alongside **09.모듈.ipynb** and **module_test.ipynb**.

import 모듈이름

```
import module_test
```

```
module_test.add(10, 20)
```

30

from 모듈이름 import 함수(or클래스)

```
from module_test import add
```

```
add(20, 50)
```

70

분석에 특화된 모듈(라이브러리)

Numpy

- 고성능 과학계산을 위한 데이터분석 라이브러리

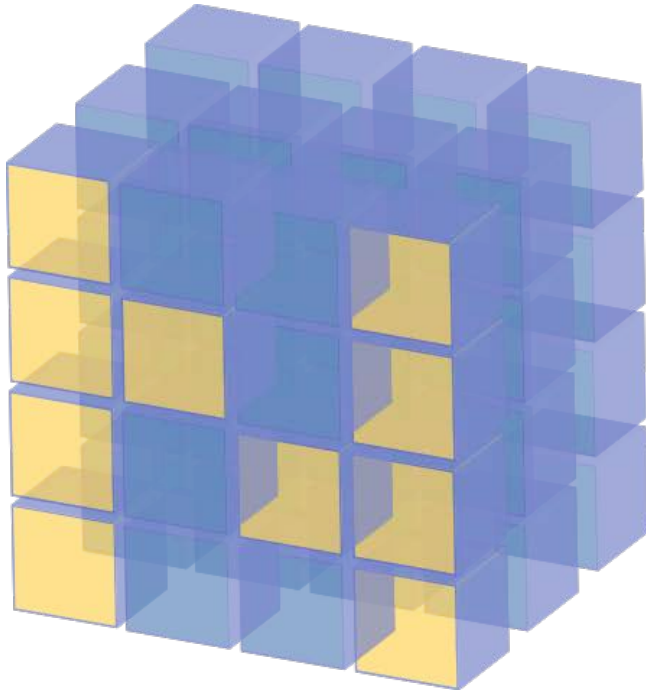
Pandas

- 행과 열로 구성된 표 형식의 데이터를 지원하는 라이브러리

Matplotlib

- 2D 그래프로 시각화가 가능한 라이브러리

Numpy 기초



NumPy

numerical python 약자

Numpy의 주요 기능

- 빠르고 효율적인 벡터 산술연산을 제공하는 다차원배열 제공 (`ndarray` 클래스)
- 반복문 없이 전체 데이터 배열 연산이 가능한 표준 수학 함수 (`sum()`, `sqrt()`, `mean()`)
- 선형대수, 난수(`random`수) 생성, 푸리에 변환

모듈(라이브러리) 사용하기

```
1 import numpy as np
```

- numpy모듈(라이브러리)를 import하고 앞으로 np라는 이름으로 부르겠다.

numpy.ndarray 클래스

- 동일한 자료형을 가지는 값들이 배열 형태로 존재함.
- N차원 형태로 구성이 가능하다.
- 각 값들은 양의 정수로 색인(index)이 부여되어 있다.
- numpy에서 차원(dimension)을 rank, axis라고 부르기도 한다.
- ndarray를 줄여서 array로 표현한다.

ndarray생성하기 : 1차원

```
list1 = [1,2,3,4,5]  
list1
```

```
[1, 2, 3, 4, 5]
```

```
arr = np.array(list1)  
arr
```

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

or

```
arr1 = np.array([1,2,3,4,5])  
arr1
```

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

ndarray생성하기 : 2차원

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

```
2 arr2
```

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

배열의 크기 확인하기

```
arr1
```

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

```
arr1.shape
```

```
(5,)
```

```
arr2
```

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

```
arr2.shape
```

```
(2, 3)
```

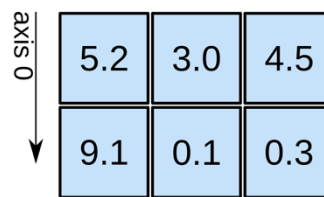

배열의 크기 확인하기

1D array



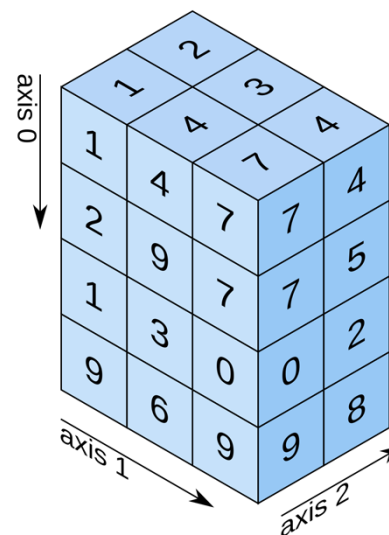
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)

배열의 전체 요소 개수 확인하기

```
print(arr1)  
print(arr1.size)
```

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

```
print(arr2)  
print(arr2.size)
```

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

배열의 타입 확인

```
print(arr1)  
print(arr1.dtype)
```

```
[1 2 3 4 5]  
int32
```

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

```
[[1 2 3]  
 [4 5 6]]  
int32
```

배열의 차원(Dimension) 확인

```
print(arr1)  
print(arr1.ndim)
```

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

```
print(arr2)  
print(arr2.ndim)
```

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

numpy를 이용해 다음과 같은 3차원 배열을 만들고
배열의 크기, 차원, 전체 요소의 개수를 확인해봅시다.

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

배열의 크기 : (2, 2, 2)
배열의 차원 : 3
배열의 개수 : 8

Array Creation

특정한 값으로 배열 생성하기

모든 값 **0**으로 초기화

```
1 arr_zeros = np.zeros((3,4))
2 arr_zeros

array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

모든 값 **1**로 초기화

```
1 arr_ones = np.ones((3,4))
2 arr_ones

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

특정한 값으로 배열 생성하기

```
1 arr_full = np.full((5,5),7)
2 arr_full
```

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


1,2,3,...,50 이 담긴 리스트 생성하기

```
1 list = []  
2 for i in range(1,51):  
3     list.append(i)  
4 print(list)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,  
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,  
37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]
```

```
1 arr = np.array(list)  
2 arr
```

```
array([ 1,  2,  3, ..., 48, 49, 50])
```

1,2,3,...,50 이 담긴 배열 생성하기

```
1 arr = np.arange(1,51)
2 arr
```

```
array([ 1,  2,  3, ..., 48, 49, 50])
```

```
1 arr = np.arange(1,51,10)
2 arr
```

```
array([ 1, 11, 21, 31, 41])
```

랜덤 값 배열 생성하기

```
1 arr = np.random.rand(2,3)
2 arr
```

```
array([[0.49813944, 0.13250847, 0.90819973],
       [0.43032648, 0.228296  , 0.12117948]])
```

랜덤 값 배열 생성하기

```
1 arr = np.random.randint(2,10)
2 arr
```

2

```
1 arr = np.random.randint(2,10, size=(2,3))
2 arr
```

```
array([[3, 5, 9],
       [8, 2, 2]])
```

타입 지정하여 배열 생성하기

```
1 arr_type = np.array([1.2,2.3,3.4], dtype=np.int64)  
2 arr_type
```

```
array([1, 2, 3], dtype=int64)
```

타입 변경하기

```
1 arr_type = arr_type.astype("float64")  
2 arr_type
```

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

```
1 arr_type.dtype
```

```
dtype('float64')
```

Array Opertaion

array연산(요소별 연산)

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

```
1 arr+arr  
  
array([2, 4, 6])
```

```
1 arr*arr  
  
array([1, 4, 9])
```


Indexing & Slicing

array 인덱싱

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

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

```
1 print(arr[0])
```

```
[1 2 3]
```

```
1 print(arr[0][0])
```

```
1
```

```
1 print(arr[0,0])
```

```
1
```

1차원 array 슬라이싱

3번 인덱스 ~ 7번 인덱스 슬라이싱

```
arr1 = np.arange(10)  
arr1
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
arr1[3:8]
```

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

슬라이싱한 곳에 한번에 데이터 넣기

```
arr1[3:8] = 12
```

```
arr1
```

```
array([ 0,  1,  2, 12, 12, 12, 12, 12,  8,  9])
```

2차원 array 생성

```
arr2 = np.arange(50).reshape(5,10)  
arr2
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],  
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],  
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],  
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],  
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49]])
```

2차원 array 슬라이싱

0행(가로)부터 1행까지 전체 열(세로) 출력

```
arr2[:2, :]
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],  
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])
```

전체행의 1열만 출력

```
arr2[:, 0]
```

```
array([ 0, 10, 20, 30, 40])
```

2차원 array 슬라이싱

0행(가로)부터 3행까지 0열(세로)부터 4열까지 출력

```
arr2[:4, :5]
```

```
array([[ 0,  1,  2,  3,  4],  
       [10, 11, 12, 13, 14],  
       [20, 21, 22, 23, 24],  
       [30, 31, 32, 33, 34]])
```

```
arr2[2,1]
```



21

```
arr2[[2,3],[2,3]]
```

행 열



```
array([22, 33])
```

10명에 대한 키와 몸무게가 들어있는 파일
'height_weight.txt'을 읽어 각 사람별 BMI 지수를 구하시오.

```
1 data = np.loadtxt("height_weight.txt", delimiter=",")  
2 data
```

```
array([[175.2, 180.3, 175. , ..., 178.2, 177. , 179. ],  
       [ 65.6,  88. ,  79.2, ...,  68.9,  74. ,  82. ]])
```

$$\text{BMI 지수} = \frac{\text{몸무게(kg)}}{\text{키(m)} \times \text{키(m)}}$$

결과값

```
array([21.37153104, 27.07018468, 25.86122449, 24.20652885, 16.03543423,  
       20.14486193, 23.14392095, 21.69720651, 23.62028791, 25.59220998])
```


Boolean 색인

Boolean 색인

```
1 name = np.array(['운비', '세욱', '승준', '동원'])  
2 name
```

```
array(['운비', '세욱', '승준', '동원'], dtype='<U2')
```

```
1 bol = np.array([True, False, True, False])  
2 bol
```

```
array([ True, False,  True, False])
```

```
1 name[bol]
```

```
array(['운비', '승준'], dtype='<U2')
```

Boolean 색인

```
1 name = np.array(['운비', '세욱', '승준', '동원'])  
2 name_score = np.array([[60, 60], [100, 90], [80, 80], [90, 90]])  
3 name_score
```

```
array([[ 60,  60],  
       [100,  90],  
       [ 80,  80],  
       [ 90,  90]])
```

Boolean 색인

```
1 name == '운비 '
```

```
array([ True, False, False, False])
```

```
1 name_score[name=='운비 ']
```

```
array([[60, 60]])
```

Universally Function

sum함수(합계)

```
arr = np.random.randint(1, 10 ,size=(2,5))  
arr
```

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

```
print(arr.sum())  
print(np.sum(arr))
```

57

57

mean함수(평균)

```
arr = np.random.randint(1, 10 ,size=(2,5))
```

```
arr
```

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

```
print(arr.mean())  
print(np.mean(arr))
```

```
5.7
```

```
5.7
```

sqrt함수(제곱근)

```
1 arr = np.arange(1,6)
2 arr
```

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

```
1 np.sqrt(arr)
```

```
array([1.          , 1.41421356, 1.73205081, 2.          , 2.23606798])
```


abs함수(절댓값)

```
1 arr = np.array([-1,2,-3,4,-5])  
2 arr
```

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

```
1 np.abs(arr)
```

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

Universally 함수

- 단일 배열에 사용하는 함수

함수	설명
abs, fabs	각 원소의 절대값을 구한다. 복소수가 아닌 경우에는 fabs로 빠르게 연산가능
sqrt	제곱근을 계산 arr ** 0.5와 동일
square	제곱을 계산 arr ** 2와 동일
Exp	각 원소에 지수 ex를 계산
Log, log10, log2, logp	각각 자연로그, 로그10, 로그2, 로그(1+x)
sign	각 원소의 부호를 계산
ceil	각 원소의 소수자리 올림
floor	각 원소의 소수자리 버림
rint	각 원소의 소수자리 반올림. dtype 유지
modf	원소의 몫과 나머지를 각각 배열로 반환
isnan	각 원소가 숫자인지 아닌지 NaN 나타내는 불리언 배열
isfinite, isinf	배열의 각 원소가 유한한지 무한한지 나타내는 불리언 배열
cos, cosh, sin, sinh, tan, tanh	일반 삼각함수와 쌍곡삼각 함수
logical_not	각 원소의 논리 부정(not) 값 계산. -arr와 동일

Universally 함수

-서로 다른 배열 간에 사용하는 함수

함수	설명
add	두 배열에서 같은 위치의 원소끼리 덧셈
subtract	첫번째 배열 원소 - 두번째 배열 원소
multiply	배열의 원소끼리 곱셈
divide	첫번째 배열의 원소에서 두번째 배열의 원소를 나눴셈
power	첫번째 배열의 원소에 두번째 배열의 원소만큼 제곱
maximum, fmax	두 원소 중 큰 값을 반환. fmax는 NaN 무시
minimum, fmin	두 원소 중 작은 값 반환. fmin는 NaN 무시
mod	첫번째 배열의 원소에 두번째 배열의 원소를 나눈 나머지
greater, greater_equal, less, less_equal, equal, not_equal	두 원소 간의 >, >=, <, <=, ==, != 비교연산 결과를 불리언 배열로 반환
logical_and, logical_or, logical_xor	각각 두 원소 간의 논리연산, &, , ^ 결과를 반환

최종 실습

-영화평점 데이터 분석-

ratings.txt

User_id :: item_id :: rating :: timestamp

```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
```

데이터 분석하기

```
data = np.loadtxt('ratings.txt', delimiter="::", dtype="int")  
data
```

```
array([[      1,      1193,      5, 978300760],  
       [      1,       661,      3, 978302109],  
       [      1,       914,      3, 978301968],  
       ...,  
       [    6040,       562,      5, 956704746],  
       [    6040,      1096,      4, 956715648],  
       [    6040,      1097,      4, 956715569]])
```

`np.loadtxt("파일경로", 파일에서 사용한 구분자, 데이터 타입)`

데이터 분석하기

```
1 print(data.ndim)
2 print(data.shape)
3 print(data.size)
```

```
2
(1000209, 4)
4000836
```

전체 평점 평균 구하기

```
1 rating_all_mean = data[:,2].mean()  
2 rating_all_mean
```

3.581564453029317

각 사용자별 평점 평균 구하기

```
1 user_id = np.unique(data[:,0])  
2 print(user_id)  
3 print(user_id.shape)
```

```
[ 1  2  3 ... 6038 6039 6040]  
(6040,)
```

사용자 아이디만 Boolean 색인

```
1 data[:,0]
```

```
array([ 1, 1, 1, ..., 6040, 6040, 6040], dtype=int64)
```

```
1 data[:,0]==1
```

```
array([ True,  True,  True, ..., False, False, False])
```

사용자 아이디가 1인 데이터만 Boolean 색인

```
1 data[data[:,0]==1]
array([[ 1, 1193, 5, 978300760],
       [ 1,  661, 3, 978302109],
       [ 1,  914, 3, 978301968],
       ...,
       [ 1, 3114, 4, 978302174],
       [ 1,  608, 4, 978301398],
       [ 1, 1246, 4, 978302091]], dtype=int64)
```

각 사용자 별 평균 평점 구하기

```
[[1, 4.188679245283019],  
 [2, 3.7131782945736433],  
 [3, 3.9019607843137254],  
 [4, 4.190476190476191],  
 [5, 3.146464646464666]]
```

각 사용자 별 평균 평점이 4점 이상인 사용자 구하기

```
array([ 1, 4, 7, ..., 6027, 6032, 6034], dtype=int64)
```

csv 파일로 저장

```
1 np.savetxt('user_id_mean.csv', list, delimiter=',', fmt="%.3f")
```

Python script가 있는 위치 폴더가 루트

	A	B
1	1	4.189
2	2	3.713
3	3	3.902
4	4	4.19
5	5	3.146



다음시간에는?

Pandas