데사 B0002

데이터마이닝이해와실습

김 태 완

kimtwan21@dongduk.ac.kr

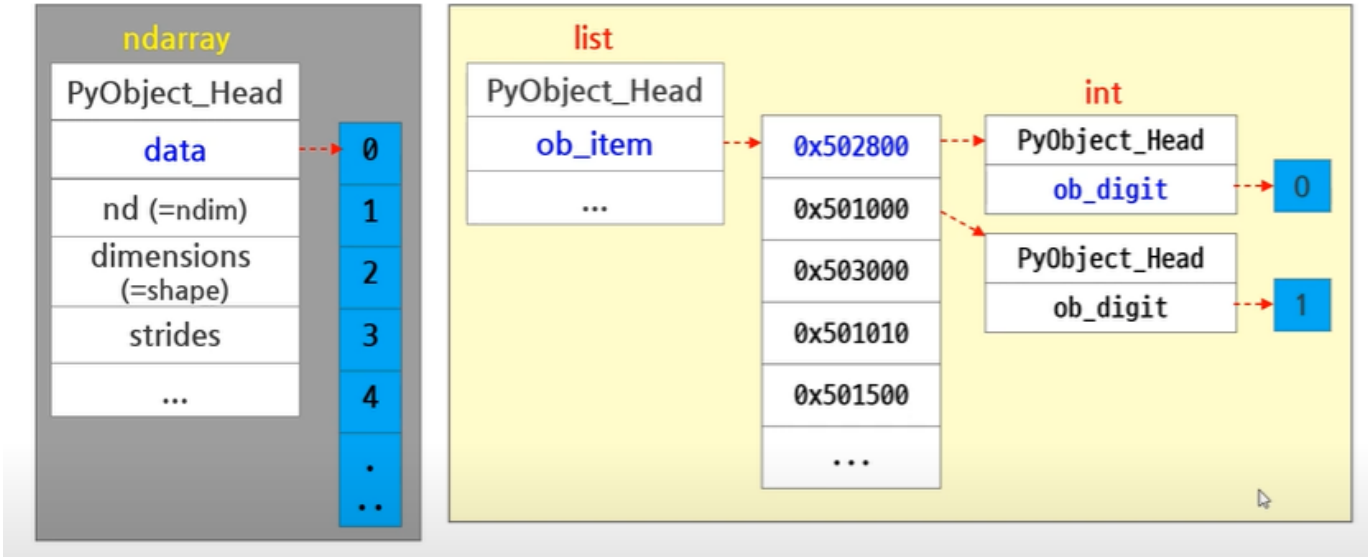
Numpy

- Numpy는 C언어로 구현된 파이썬 라이브러리로써, 고성능의 수치계산을 위해 제작
- Numerical Python의 줄임말 Numpy는 벡터 및 행렬 연산에 있어서 매우 편리한 기능을 제공
- 기본적으로 array (행렬)라는 단위로 데이터를 관리하며 이에 대해 연산을 수행
- Python의 list는 여러 자료형의 값들을 하나의 변수에 저장할 수 있는 자료구조로서 강력하고 활용도가 높음
- 하지만 데이터를 처리할 때는 리스트와 리스트 간의 다양한 연산이 필요
  - 이러한 점에서 기능이 다소 부족 / 연산 속도도 느림
  - 따라서 Data Science 영역에서는 Numpy array를 선호
- 설치

```
pip install numpy
```

Numpy

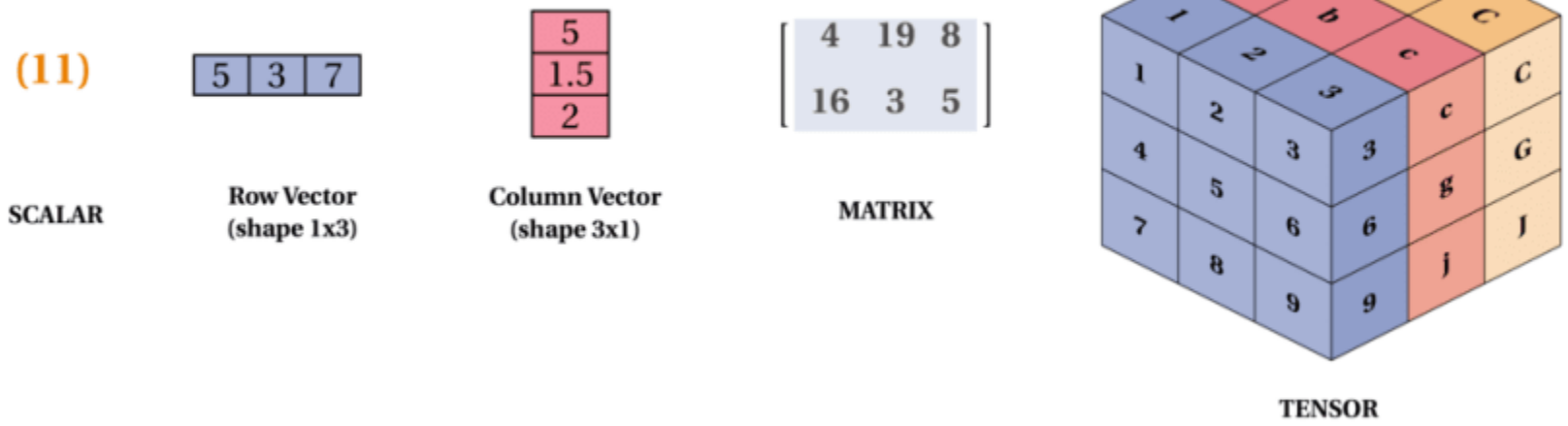
- Numpy 특징
  - 속도가 빠르고 메모리 사용이 효율적
  - 데이터를 메모리에 할당하는 방식이 기존과 다름
  - 반복문을 사용하지 않음
  - 연산할 때 병렬로 처리
  - 함수를 한 번에 많은 요소에 적용
  - 다양한 선형대수 관련 함수 제공
  - C, C++, 포트란 등 다른 언어와 통합 사용 가능



Numpy 기본 사용법

- Numpy의 차원
  - 1차원 축 (행) : axis 0 → vector
  - 2차원 축 (열) : axis 1 → matrix (행렬)
  - 3차원 축 (채널) : axis 2 → tensor (3차원 이상)

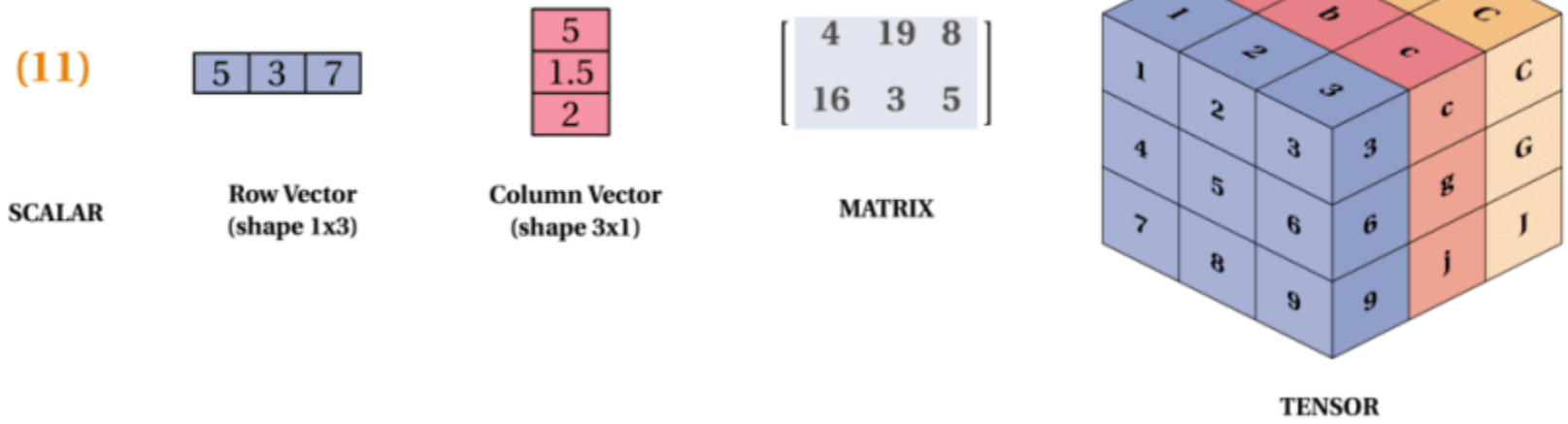
```python
import numpy as np

a= np.array(1)
#a= np.array([1])
#a= np.array([1,2,1])
print(a, a.shape, a.ndim)
```



Numpy 기본 사용법

- Numpy의 차원
  - 1차원 축 (행) : axis 0 → vector
  - 2차원 축 (열) : axis 1 → matrix (행렬)
  - 3차원 축 (채널) : axis 2 → tensor (3차원 이상)

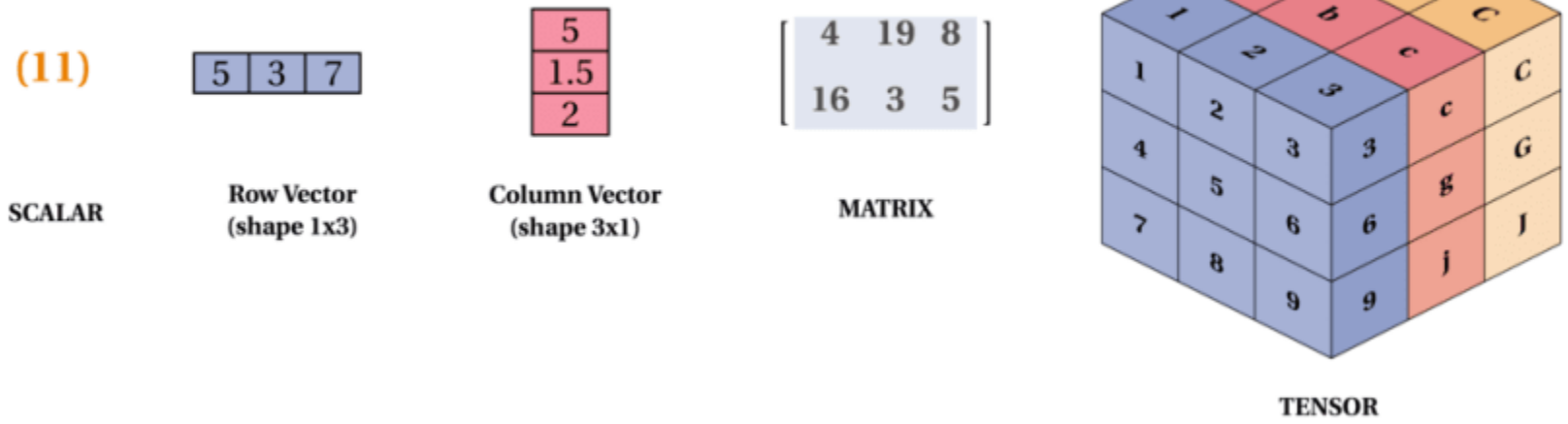
```
import numpy as np

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

print(a, a.shape, a.ndim)
```



Numpy 기본 사용법

- Numpy의 차원
  - 1차원 축 (행) : axis 0 → vector
  - 2차원 축 (열) : axis 1 → matrix (행렬)
  - 3차원 축 (채널) : axis 2 → tensor (3차원 이상)

```
import numpy as np

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

print(a, a.shape, a.ndim)
```



Numpy

- 배열

```
import numpy as np

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

print(array.size)     # 배열의 크기
print(array.dtype)    # 배열 원소의 타입
print(array[2])       # 인덱스 2의 원소
```

Numpy

- 배열

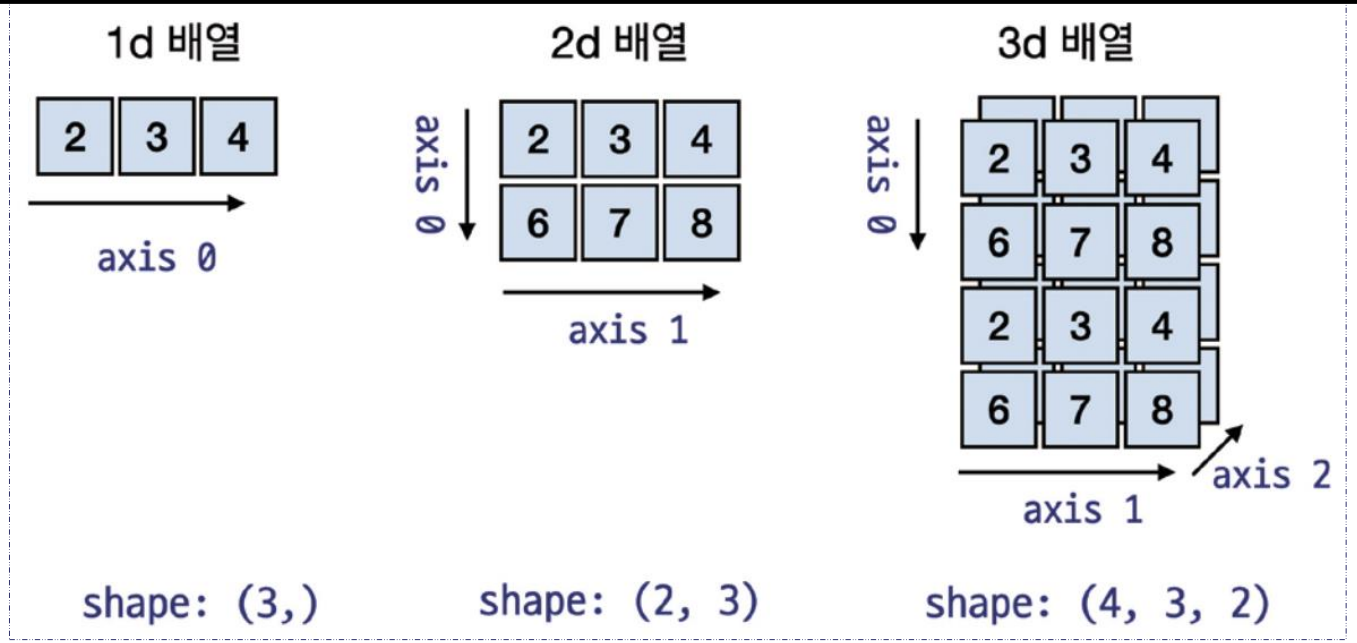
```
import numpy as np

arr = np.array([0, 2, 3, 4, 5, 6, 7, 8, 9])

print(type(arr))  # <class 'numpy.ndarray'>
```

```
import numpy as np

a = [0, 2, 3, 4, 5, 6, 7, 8, 9]
arr = np.array(a)

print(type(a))
print(type(arr))
```

Numpy

- Shape
  - 여러 가지 속성 중에서 다차원 배열의 행과 열을 비롯한 차원의 형상을 출력하는 shape 속성은 넘파이 이용자들이 가장 자주 사용하는 속성

```
import numpy as np

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



Numpy

- dtype
  - 매개변수 dtype으로 넘파이 배열의 데이터 타입 지정
    - 넘파이의 ndarray는 모든 원소가 동일한 타입(보통 숫자)을 가지기 때문에 효율적
    - dtype을 실수형인 float으로 지정한다면 모든 데이터가 실수형으로 저장되는 것을 확인 가능

```
import numpy as np

a = np.array( [ [1, 2, 3.5], [4, 5, 6.5]], dtype=int)
b = np.array( [ [1, 2, 3.5], [4, 5, 6.5]], dtype=float)

print(a)
print(b)
```

Numpy

- 파이썬 list와 ndarray 비교 (1/3)

```
import numpy as np

a = [1, 2, 3, 4]
b = [1, 2, 3, 4]

arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([1, 2, 3, 4])

print(a + b)
print(arr1 + arr2)
print(arr1 - arr2)
print(arr1 * arr2)
print(arr1 / arr2)
```

- 파이썬 list와 ndarray 비교 (2/3)

```
import numpy as np

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

print(a * 5)
print(arr1 + 5)
print(arr1 - 5)
print(arr1 * 5)
print(arr1 / 5)
```

Numpy

- 파이썬 list와 ndarray 비교 (3/3)

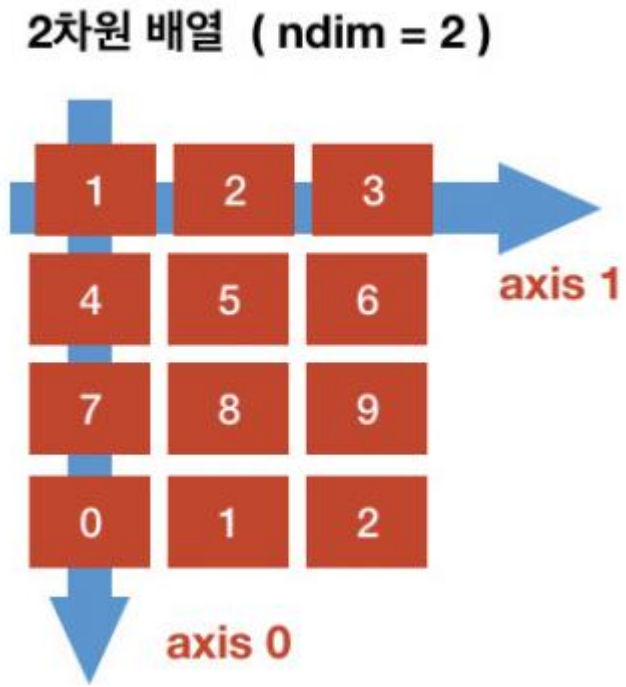
```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([10, 20, 30])

print(2*a + b)
print(a == 2)
print(b > 10)
print((a == 2) & (b > 10))
```

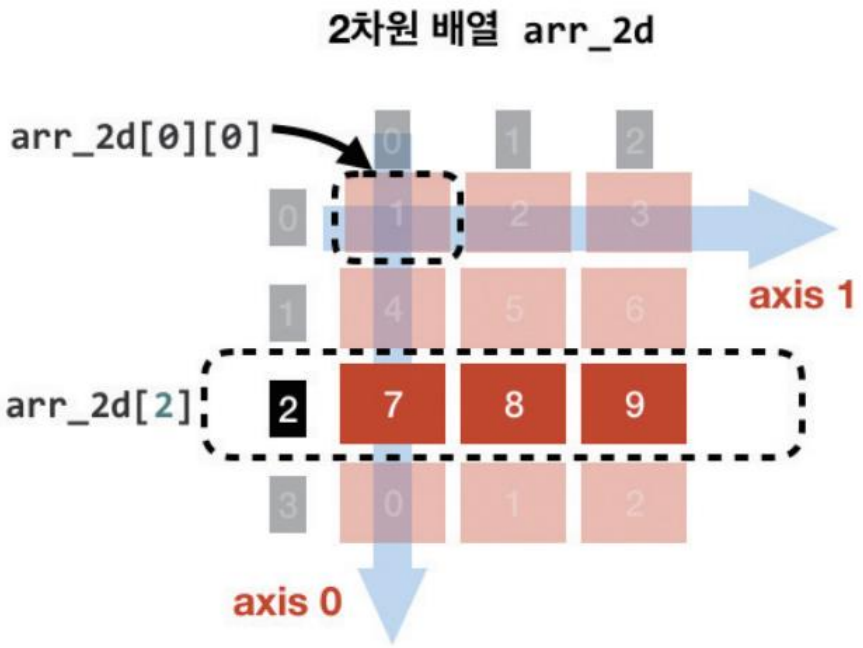
Numpy

- 2차원 배열

```
import numpy as np

a = [[1, 2, 3, 4], [5, 6, 7, 8]]

arr2 = np.array(a)
print(arr2)

arr3 = np.array([[0, 1, 2], [3, 4, 5]]) #2x3 array

print(len(arr3))     # row
print(len(arr3[0]))  # column
print(arr3.shape)    # shape
```

- Index / slicing





Numpy

- index

```
arr = np.array([0, 1, 2, 3, 4])
print(arr[2])
print(arr[-1])
print(arr[1] * arr[3])
```

- slicing

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

Numpy

- How to create a basic array
  - np.array(), np.zeros(), np.ones(), np.full(), np.empty(), np.arange(), np.linspace()

  - np.array()

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

https://numpy.org/doc/stable/user/absolute_beginners.html

Numpy

- How to create a basic array
  - np.zeros()
  - np.ones()

```
a = np.zeros(5)

b = np.zeros((2, 3), dtype=float)

c = np.ones((2, 5), dtype=int)
```

  - np.full()

```
a = np.full((2,3),1)
b = np.full((3,2,3),10)
```

Numpy

- How to create a basic array
  - np.empty()

```
a = np.empty(2)
b = np.empty([2,2],dtype=int)
```

  - np.arrange()

```
a = np.arange(4)
b = np.arange(1,5)
c = np.arange(1,11,2)
```

  - np.linspance

```
a = np.linspace(0,1)
b = np.linspace(0,10,num=5)
```

Numpy

- 배열 생성
  - random한 초기 값 설정

```
import numpy as np
array = np.random.randint(0, 10, (3,3))
print(array)
```

  - 평균이 0이고, 표준편차가 1인 표준 정규를 띄는 배열 생성

```
import numpy as np
array = np.random.normal(0, 1, (3,3))
print(array)
```

Numpy

- 배열 크기 변형
  - reshape : 배열의 구조를 변경

```
a = np.zeros(12)
b = a.reshape(3, 4)

print(a)
print(b)
```

  - flatten : convolutional neural network (CNN) 마지막 layer에서 많이 사용

```
c = b.flatten()

print(b.reshape(-1,))
print(c)
```

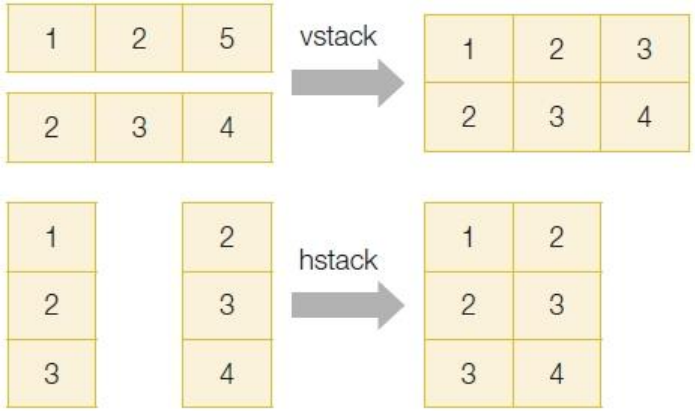
Numpy

- 연결 함수(concatenation functions) : 두 객체 간의 결합을 지원하는 함수
  - concatenate 함수

```
import numpy as np


array1 = np.array([1,2,3])
array2 = np.array([4,5,6])
array3 = np.concatenate([array1, array2])


print(array3)
```

Numpy

- 연결 함수(concatenation functions) : 두 객체 간의 결합을 지원하는 함수
  - vstack 함수 : 배열을 수직으로 붙여 하나의 행렬을 생성
  - hstack 함수 : 배열을 수평으로 붙여 하나의 행렬을 생성




```
import numpy as np

v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])

print(np.hstack((v1, v2)))
print(np.vstack((v1, v2)))
```

Numpy

- 최대, 최소 및 기술 통계 값

```
x = np.array([1, 2, 3, 4])
print(np.sum(x))
print(x.sum())
print(x.min())
print(x.max())
print(x.argmin())
print(x.argmax())
print(x.mean())
print(np.var(x))
print(np.std(x))
```

Numpy

- 최대, 최소 및 기술 통계 값

```
import numpy as np

a = np.array([[-2.5, 3.1, 7], [10, 11, 12]])

for func in (a.min, a.max, a.sum, a.prod, a.std, a.var):
    print(func.__name__, "=", func())
```

Examples

- **예제 1 : 배열에 저장한 모든 값에 1씩 더해서 배열로 출력**

```
a = [2,5,1,17,21]

## coding here

print(result)
```

```
import numpy as np

## coding here

print(result)
```

Examples

- **예제 2 : 넘파이를 사용해 1차원 배열을 2차원 배열로 변경**

```
import numpy as np

a = np.array([1,3,5,7,8,11])

## coding here

print(b)
```

[1, 3, 5, 7, 8, 11]

[[1, 3, 5]
[7, 8, 11]]

Examples

- 예제 4 : 넘파이를 사용해 아래의 사칙 연산 수행

10 + 30 = ?
40 – 93 = ?
55 * 71 = ?
250 / 25 = ?

```
import numpy as np

a = np.array([1,3,5,7,8,11])

## coding here
import numpy as np

res1 = np.[ ](10,30)
res2 = np.[ ](40,93)
res3 = np.[ ](55,71)
res4 = np.[ ](250,25)

print(res1, res2, res3, res4)
```

Examples

- **예제 5 : 랜덤 값이 있는 10x10 배열을 만들고 최소값과 최대값을 찾기**

```
import numpy as np

a = np.random.random(          )

min_a = a.    ()
max_a = a.    ()

print(a)
print(min_a, max_a)
```

Examples

- **예제 6 :** linspace() 함수를 이용하여 sin 0 부터 sin π사이의 값 11개를 다음과 같이 출력해 보자.

```
values = np.linspace(                    )
x = (values)
print(x)
```

```
[0.00000000e+00 3.09016994e-01 5.87785252e-01 8.09016994e-01
 9.51056516e-01 1.00000000e+00 9.51056516e-01 8.09016994e-01
 5.87785252e-01 3.09016994e-01 1.22464680e-16]
```

Examples

- **예제 7 :** 아래 점수를 보고 다음을 구해보자.
  - 각 교과목의 총점
  - 각 교과목의 평균
  - 가장 잘 본 교과목의 평균

	midexam	finalexam	homework
Python	90	92	37
R	85	88	91
Math	94	91	84

Examples

- **예제 8 :** 총 30일의 매출이 적혀 있는 “sales” list 중에서 40000원 이상의 매출을 달성하지 못한 날과 그 매출을 구해보자.

```
import numpy as np
import random

sales = [86623, 33030, …, 38117]

n_sales = np.array(sales)
filter = [          ] # 매출을 달성하지 못한 날

bad_sales = [          ] # 매출을 달성하지 못한 날의 매출액
```

Examples

- **예제 9 :** numpy에서 txt 파일을 읽는 함수는 loadtxt 이다. 평균을 구해보자.

```
import numpy as np
data = np.loadtxt('data.txt', delimiter=',')

print(                )
```

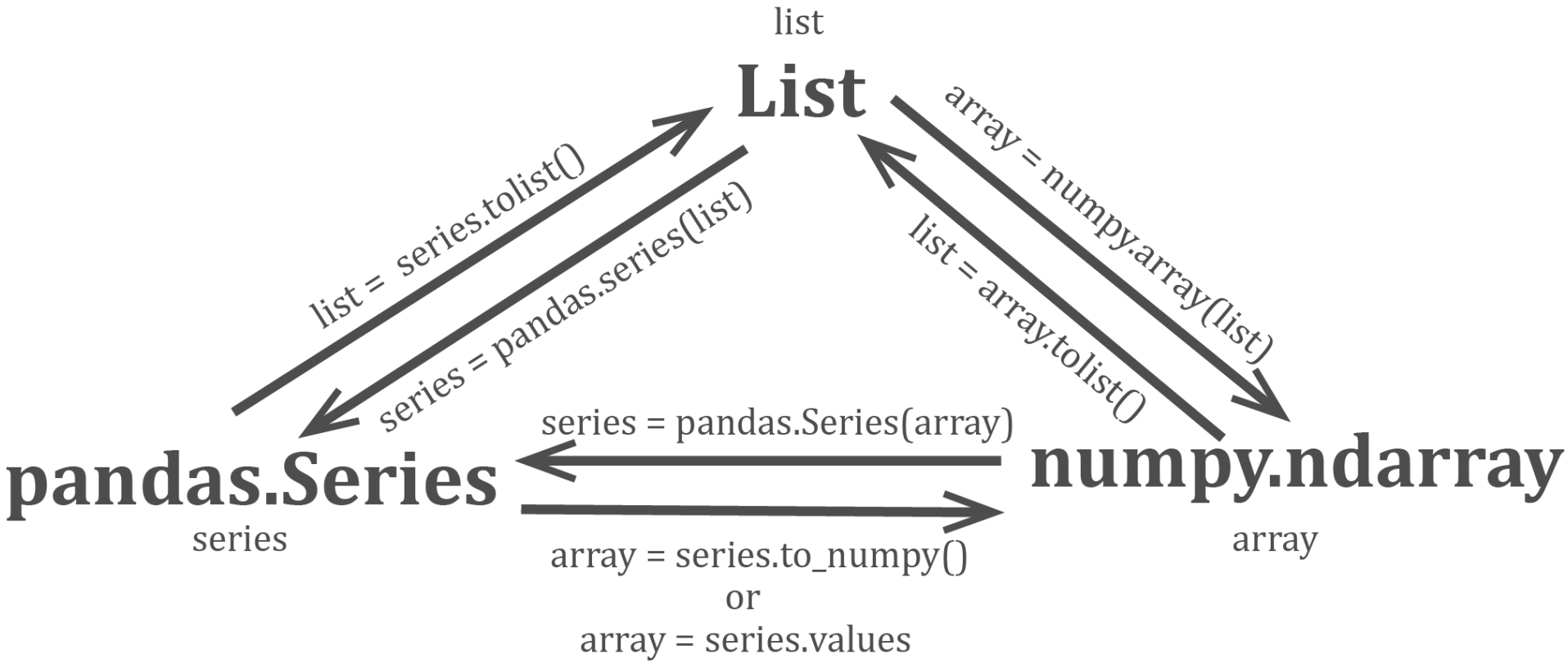
Examples

- **예제 10 :** numpy에서 txt 파일을 읽는 함수는 loadtxt 이다. 평균을 구해보자.

```
data = np.loadtxt('data2.txt')
average = np.mean(data)
print(average)
```

Numpy

- Transformation
  - 1차원 배열



감사합니다

kimtwan21@dongduk.ac.kr

김 태 완