Numpy

#01. Numpy 개요

> Numeric Python

- 다차원 배열을 쉽게 처리하고 효율적으로 사용할 수 있도록 지원하는 파이썬 패키지
- 데이터 구조 외에도 수치 계산을 위해 효율적으로 구현된 기능 제공
- 데이터 분석에서 Pandas와 함께 자주 사용하는 도구
- 실제로 데이터 분석을 수행하기 위한 전제 조건은 컴퓨터가 이해할 수 있도록 데이터를 숫자 형식으로 변환하는 것
- 파이썬의 list 자료형의 경우 데이터의 크기가 커질수록 저장 및 가공에 효율성을 보장하지 못함
- 이러한 단점을 보완하기 위한 패키지이기 때문에 Data Science에서 핵심적인 도구로 인식되고 있음

#02. 준비과정

[1] 패키지 참조

```
import numpy as np
```

#03. Numpy가 제공하는 데이터 타입

리스트의 업그레이드 버전인 **배열(array)** 라는 데이터 타입을 제공한다.

- list 타입: 서로 다른 타입의 원소를 허용한다. '[1, "hello", True]'
- array 타입: 반드시 모든 원소의 데이터 타입이 동일해야 한다.

[1] numpy 배열 생성과 기본 활용

리스트를 통한 1차원(=1행으로 구성) 배열 만들기

```
arr = np.array([1,3,5,7,9])
print(type(arr))
arr
```

```
<class 'numpy.ndarray'>
```

```
array([1, 3, 5, 7, 9])
```

(2) 배열의 크기와 각 원소에 접근하기

기본 특성은 list와 동일

- 인덱스 번호와 길이의 존재
- 반복문을 통한 원소의 탐색
- 인덱싱, 슬라이싱

```
size = len(arr)
print("배열의 원소는 %d개 입니다." % size)
```

```
배열의 원소는 5개 입니다.
```

(3) 배열의 원소

리스트와 마찬가지로 인덱스 번호가 존재

```
print(arr[0])
print(arr[1])
print(arr[2])
```

```
1
3
5
```

(4) 반복문을 통한 활용

리스트와 동일하게 사용할 수 있다.

```
for i,v in enumerate(arr):
    print("%d번째 원소 >> %d"% (i,v))
```

```
0번째 원소 >> 1
1번째 원소 >> 3
2번째 원소 >> 5
3번째 원소 >> 7
4번째 원소 >> 9
```

[2] Numpy 배열의 특성

(1) list 타입의 경우

서로 다른 타입의 원소를 허용

```
arr2 = [100, 3.14, True]
arr2
```

```
[100, 3.14, True]
```

(2) Numpy 배열의 경우

'arr2'를 배열로 변환할 경우 원소의 타입이 서로 다른것을 허용하지 않기 때문에 가장 포괄적인 형태의 자료형으로 통일함

> 여기서는 실수의 범위가 더 크므로 모든 원소가 실수형으로 변환 됨.

```
arr3 = np.array(arr2)
arr3        # '100.' '1.' 은 소수점 뒤의 0을 생략한 것임
```

```
array([100.  ,   3.14,   1.  ])
```

강제 형 변환

'np.array()'메서드에 'dype'파라미터로 특정 타입을 강제 지정할 수 있다.

실수형 값을 정수로 변환하는 과정에서 소수점 아래는 모두 버림.

```
arr4 = np.array(arr2, dtype='int')
arr4
```

```
array([100,   3,   1])
```

문자열이 포함된 경우

모든 프로그래밍 언어에서 가장 표현범위가 넓은 타입은 문자열이다.

그러므로 문자열이 포함된 리스트를 배열로 변환할 경우 모든 원소가 문자열이 된다.

```
arr5 = np.array([1.2, 3, True, '4'])
print(arr5)
arr5        # 이것은 print형태와 다른 형식.
# 참고. list는 사이사이에 쉼표가 찍히나, np는 그 사이에 쉼표가 없다. (print 형
```

```
['1.2' '3' 'True' '4']
```

```
array(['1.2', '3', 'True', '4'], dtype='<U32')
```

(3) list 타입으로 역 변환

서로 호환 가능

```
mylist = list(arr5)
print(type(mylist))
mylist
```

```
<class 'list'>
```

```
['1.2', '3', 'True', '4']
```