

PYTHON for 데이터 분석

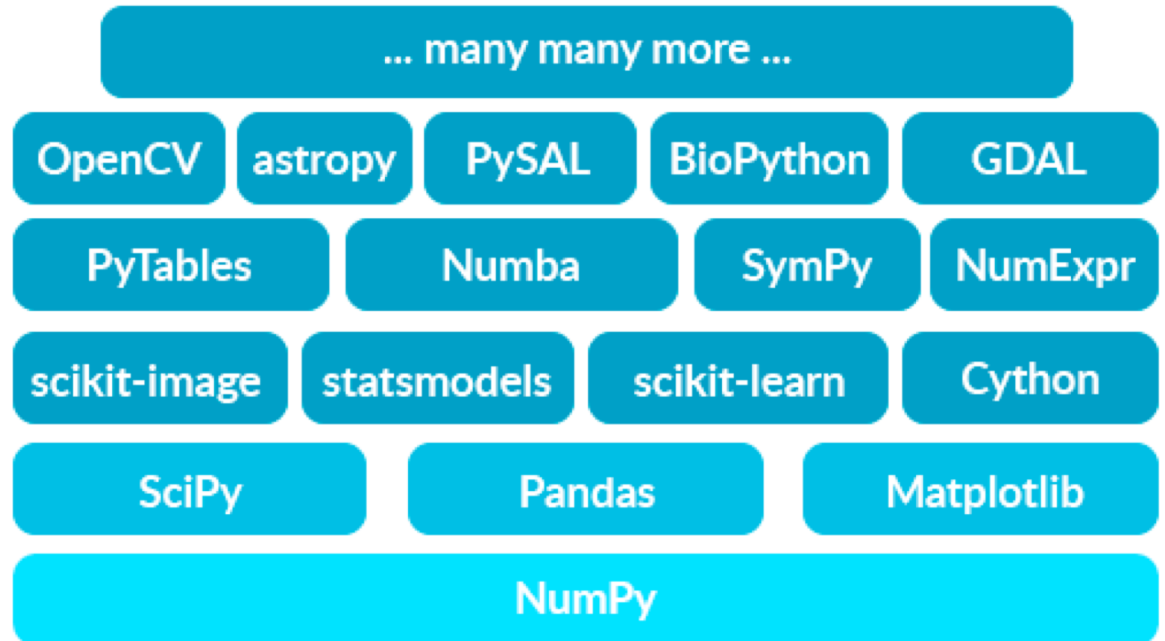
2019. 03



1. **Python Coding Basics**
2. **Python Data Type**
 - number, string
 - list, tuple, dictionary, set
3. **Python Operators**
 - Arithmetic, Relational, Logical, ...
4. **Python Control Statement**
 - if, while, for
5. **Major Packages**
6. **Python Class, Function**

주요 패키지

- 데이터 처리
 - NumPy
 - Pandas
- 시각화
 - Matplotlib
 - Seaborn
- 머신 러닝
 - Scikit-learn



□ numpy 개요

- numpy는 과학 계산을 위한 라이브러리로서 다차원 배열을 처리하는데 필요한 여러 유용한 기능을 제공
- numpy 기본 데이터 타입은 ndarray
- array() 함수
 - 리스트와 같은 다양한 인자를 입력받아서 ndarray로 변환하는 기능
- 한 개의 ndarray 내의 데이터 타입은 그 연산의 특성상 같은 데이터 타입만 가능

□ Indexing and slicing

- **indexing**
- **slicing**
 - 각 차원별로 슬라이스 범위를 지정
- **fancy indexing**
 - 정수 배열을 indexer로 사용해서 다차원 배열로 부터 Indexing하는 방법
- **boolean indexing**
 - 배열 각 요소의 선택여부를 True, False로 표현하는 방식

```
import numpy as np

# 1차원 배열 생성
a = np.array([1, 2, 3, 4, 5, 6])
# 2차원 배열 생성
b = np.array([[1, 2, 3], [4, 5, 6]])

# 1차원 indexing & slicing
a[0], a[0:3], a[1:], a[:-2]

# 2차원 indexing & slicing
b[1,1], b[0:2,0:2]

# fancy indexing
b[0,1]
b[[0,1], 1]

# boolean indexing
c = a[a > 3]
```

NumPy

□ Indexing & slicing 참고(<https://rfriend.tistory.com/290>)



[Python NumPy] Indexing and Slicing of an ndarray

Indexing a subset of 1D array

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

`a[0:5]`

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

Not a copy, but a VIEW!!!

<http://rfriend.tistory.com>

Indexing a subset of 2D array

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

`d[0:3, 1:3]`

```
array([[1, 2],  
       [6, 7],  
       [11, 12]])
```



[Python NumPy] Slicing and Indexing with Boolean values

Expression

arr

axis_ABC

arr[axis_ABC == 'A']

Shape

(5, 4)

(5,)

(2, 4)

**Array
Respresen-
-tation**

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

(['A', 'A', 'B', 'C', 'C'])

```
[[0, 1, 2, 3],  
 [4, 5, 6, 7]]
```

<http://rfriend.tistory.com>



[Python NumPy] Fancy Indexing by using integer arrays

indexer

From the first
0
1
2
3
4

From the end
-5
-4
-3
-2
-1

ndarray a

0	1	2
3	4	5
6	7	8
9	10	11
12	13	14

selecting a subset of the rows

from the first

`a[[1, 2]]`

3	4	5
6	7	8

from the end

`a[[-1, -2]]`

12	13	14
9	10	11

<http://rfriend.tistory.com>



[Python NumPy] Fancy Indexing by using integer arrays

axis 1	0	1	2
axis 0	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8
3	9	10	11
4	12	13	14

selecting a subset of the rows and columns

`a[[0, 2, 4]][:, [0, 2]]`

or

`a[np.ix_([0, 2, 4], [0, 2])]`

0	2
6	8
12	14

<http://rfriend.tistory.com>

□ array

▪ 주요 변수

- ndim, shape, dtype

▪ 주요 함수

• 생성

- ✓ array()
- ✓ arange(), zeros(), ones(), linspace()

• 차원과 크기 변경

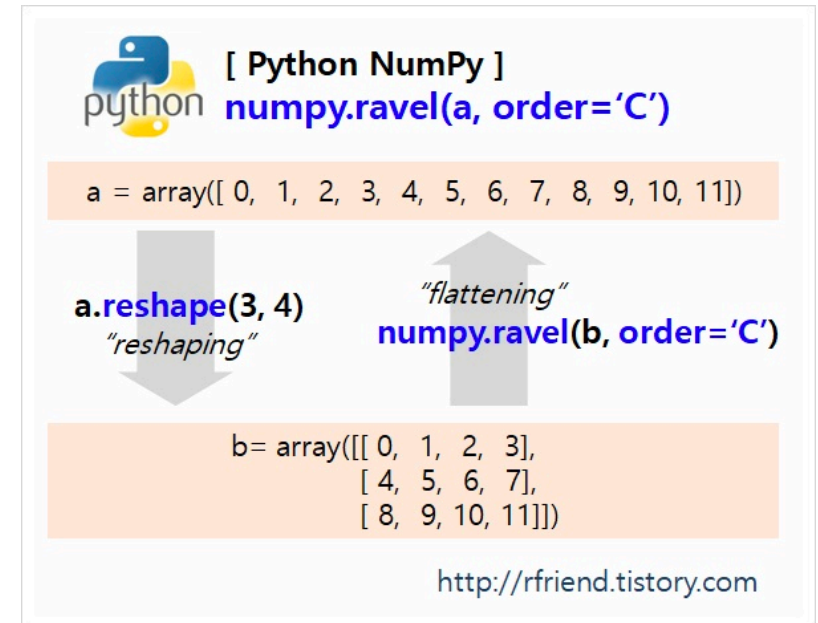
- ✓ reshape(),
- ✓ ravel(),
- ✓ flatten()
- ✓ astype()

• 정렬

- ✓ sort()
- ✓ argsort()

▪ array 사용 시 주의할 점

- 길이가 5인 1차원 배열과 행(5,), 열의 갯수가 (5,1)인 2차원 배열 또는 행, 열의 갯수가 (1, 5)인 2차원 배열은 데이터가 같아도 엄연히 다른 객체라는 점이다.



NumPy

- **numpy array 장점**
 - **Size** - Numpy data structures take up **less space**
 - **Performance** - they have a need for speed and are faster than lists
 - **Functionality** - SciPy and NumPy have optimized functions such as linear algebra operations built in.
- **numpy array vs. list**
 - 하나의 array는 동일한 데이터 타입만 지원
 - ✓ [1, 2, 3]
 - ✓ [[1, 2, 3]
 - [4, 5, 6]]
 - 하나의 list는 여러가지 데이터 타입으로 가능
 - ✓ [1, 2, 3, 'a', 'b', 'c']
- **numpy array vs. matrix**
 - matrix는 배열 중 2차원과 동일
- Numpy array의 indexing and slicing의 결과는 original array에 대한 **view**를 제공
 - 즉, slicing 결과에 대한 변경이 original array에 영향을 미침
 - Indexing한 배열을 copy(복사)하려면 copy() 사용
 - 단 fancy indexing은 copy를 제공하므로 영향이 없음

□ pandas 개요

- pandas는 데이터 분석(Data Analysis)을 위해 널리 사용되는 파이썬 라이브러리 패키지이다
- **Series**
 - 1차원 자료구조
 - 배열/리스트 같은 일련의 시퀀스 데이터를 받아들인다
 - 별도의 인덱스 레이블을 지정하지 않으면 자동적으로 0부터 시작되는 정수 인덱스 사용
- **DataFrame**
 - 2차원 자료구조
 - 행과 열이 있는 테이블 데이터(Tabular Data)를 받아들인다.
- **Panel**
 - 3차원 자료구조
 - Axis0(items), Axis1(major_axis), Axis2(minor_axis)등 3개 축을 가지고 있다.
 - Axis 0은 하나의 요소가 2차원의 DataFrame에 해당하며
 - Axis 1은 DataFrame의 행(row)에 해당하고,
 - Axis 2는 DataFrame의 열(column)에 해당된다

□ Data selecting and filtering

- **'[]' operator**
 - **column name**
 - ✓ `df['col1']`
 - **column name list**
 - ✓ `df[['col1', 'col2']]`
 - **인덱스로 변환 가능한 표현식**
 - ✓ 슬라이싱
 - `df[0:2], df[0:], df[:-1]`
 - ✓ 불린 인덱싱
 - `df[df['col1'] == 0]`
- ~~**ix(deprecated 예정)**~~
- **iloc**
 - 위치(position) 기반
- **loc**
 - 명칭(label) 기반

□ 주요 함수

- Aggregation
- groupby
- sort_values
- isnull
- fillna
- apply
 - 행이나 열 단위로 더 복잡한 처리를 하고 싶을 때 사용
 - 인수로 행 또는 열을 받는 함수를 apply 메서드의 인수로 넣으면 각 열(또는 행)을 반복하여 그 함수에 적용시킨다