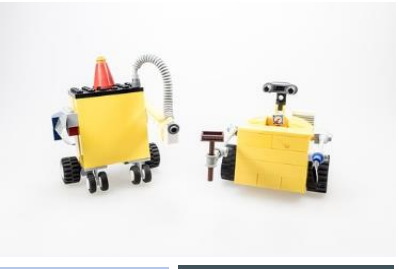
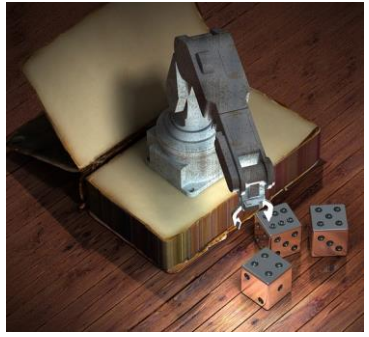
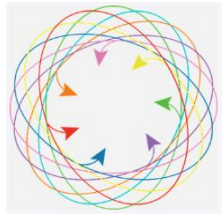
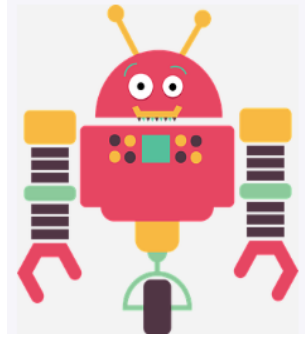
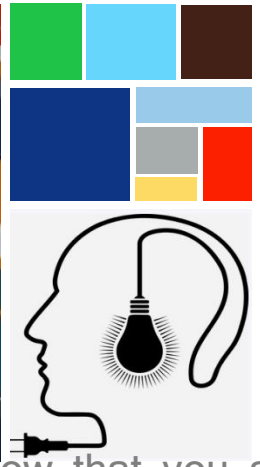










In the future, an AI agent will know that you are at work and have ten minutes free, and then help you accomplish something that is high on your to-do list.

데이터처리&분석하기

강희숙



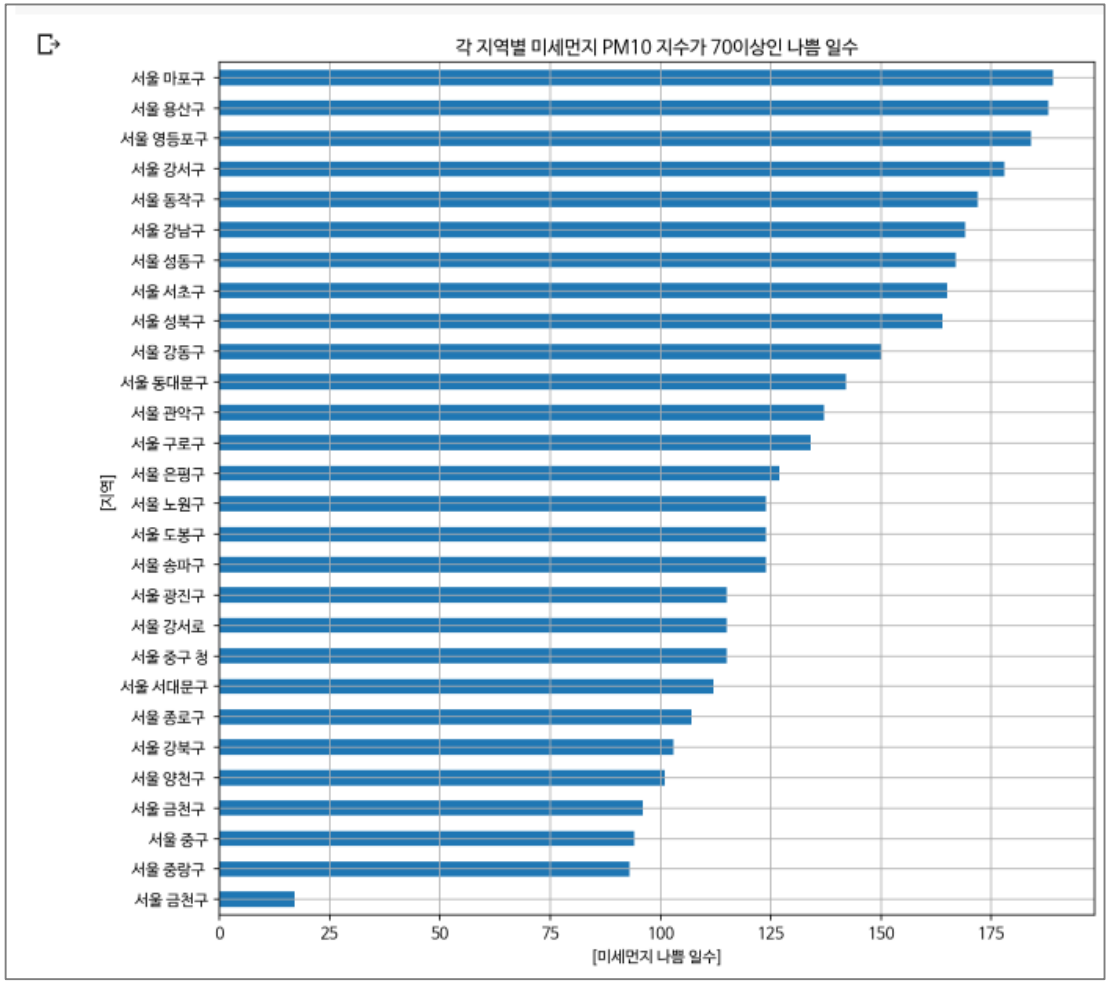

PROJECT. 공공 데이터 분석하기

Enjoy your possibility

서울지역 미세먼지 측정 데이터 분석

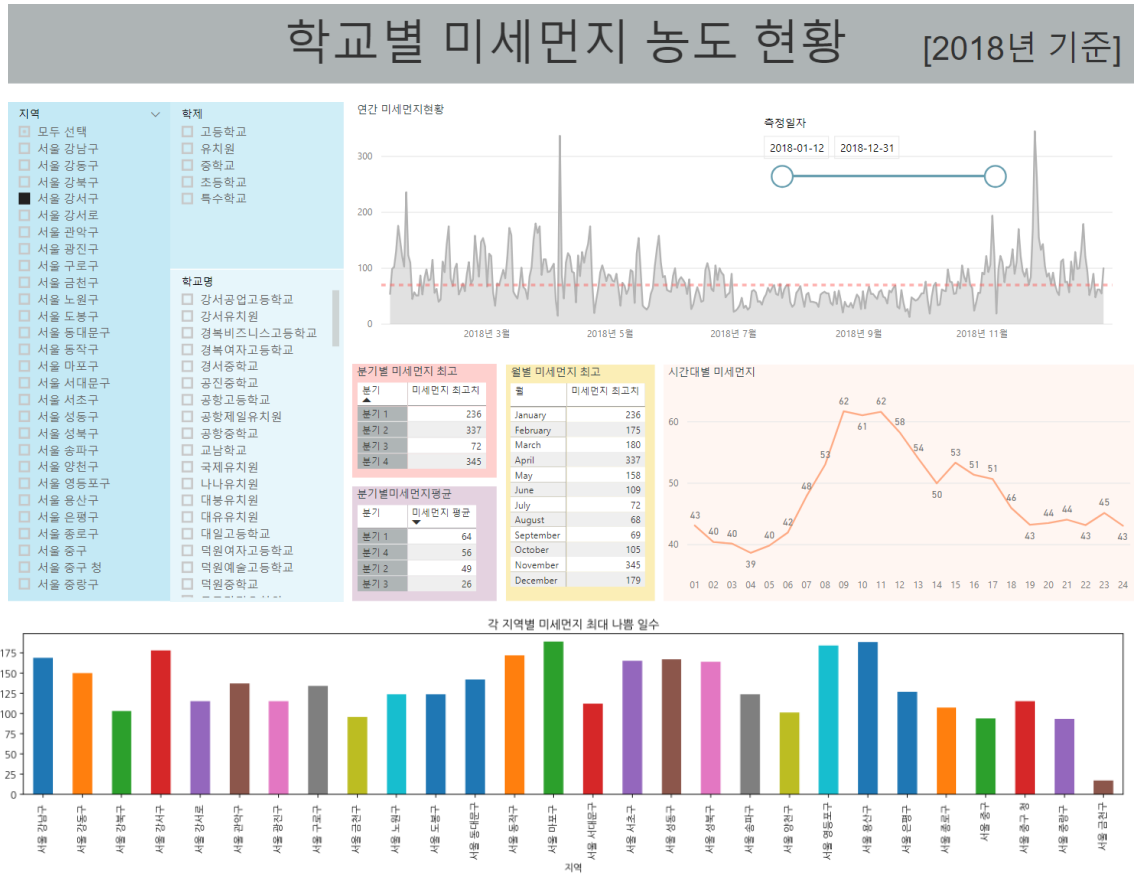
	지역	측정소코드	측정소명	측정일시	SO2	CO	O3	NO2	PM10	PM25	주소
2	서울 중구	111121	중구	2018010101	0.004	0.5	0.02	0.02	34	19	서울 중구 덕수궁길 15
3	서울 중구	111121	중구	2018010102	0.004	0.4	0.024	0.016	27	14	서울 중구 덕수궁길 15
4	서울 중구	111121	중구	2018010103	0.004	0.4	0.018	0.022	26	14	서울 중구 덕수궁길 15
5	서울 중구	111121	중구	2018010104	0.004	0.5	0.01	0.03	26	15	서울 중구 덕수궁길 15
6	서울 중구	111121	중구	2018010105	0.004	0.6	0.011	0.029	28	16	서울 중구 덕수궁길 15
7	서울 중구	111121	중구	2018010106	0.004	0.5	0.012	0.027	29	17	서울 중구 덕수궁길 15
8	서울 중구	111121	중구	2018010107	0.004	0.5	0.009	0.03	28	16	서울 중구 덕수궁길 15
9	서울 중구	111121	중구	2018010108	0.004	0.5	0.009	0.032			
		111121	중구	2018010109	0.004						
				2018123116	0.005	0.5	0.008	0.031	52	31	서울 노원구 화랑로 429
343098	서울 노원구	111312	화랑로	2018123117	0.004	0.3	0.01	0.028	36	20	서울 노원구 화랑로 429
343099	서울 노원구	111312	화랑로	2018123118	0.004	0.4	0.006	0.034	29	17	서울 노원구 화랑로 429
343100	서울 노원구	111312	화랑로	2018123119	0.004	0.5	0.004	0.036	30	18	서울 노원구 화랑로 429
343101	서울 노원구	111312	화랑로	2018123120	0.004	0.5	0.005	0.036	35	22	서울 노원구 화랑로 429
343102	서울 노원구	111312	화랑로	2018123121	0.005	0.9	0.004	0.041	40	22	서울 노원구 화랑로 429
343103	서울 노원구	111312	화랑로	2018123122	0.005	1.1	0.004	0.044	45	23	서울 노원구 화랑로 429
343104	서울 노원구	111312	화랑로	2018123123	0.005	1	0.004	0.04	51	23	서울 노원구 화랑로 429
343105	서울 노원구	111312	화랑로	2018123124	0.005	1	0.004	0.037	43	26	서울 노원구 화랑로 429

거대한 양의 데이터를 가공하여 정보를 효율적으로 가시화



Step0_분석 목적 및 범위 설정

- 서울특별시 광진구에 위치한 00중학교에서 1학기 야외활동 일정과 시간을 2018년 미세먼지 빅데이터에 근거하여 정하고 싶다면?



Step1: 데이터 수집

미세먼지 데이터, 에어코리아

Step2: 데이터 전처리

지역별, 기간별

Step3: 분석

지역별 PM10 지수 70이상 일수 파악

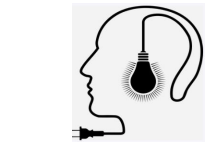
가시화

지역별 일수 히스토그램으로 나타내기

빅데이터 분석 처리 흐름

분석주제 설정

Collecting < 데이터 수집 >	Processing <데이터 전처리>	Analysis <분석>	Visualization <정보 표현>
- 실시간 대용량 데이터 자동 수집	- 데이터 분석에 맞도록 처리 - 대용량 데이터를 처리하는 분산처리 시스템 구축	- 데이터 간의 관계를 파악 - 파악된 관계를 사용하여 우리가 원하는 새로운 데이터를 만들어내는 과정	- 결과 출력 및 시각화 (Visualization)
- Data Warehousing, 실시간(주가, 메신저 내용) 혹은 저장된 데이터(통계청 마이크로 데이터)가 제공되고 있음	- Python, Hadoop	- 빅데이터 통계 소프트웨어 활용 : R, SAS	- Unsupervised Learning - Social Media analysis - Predictive modeling - Visualization - Simulation

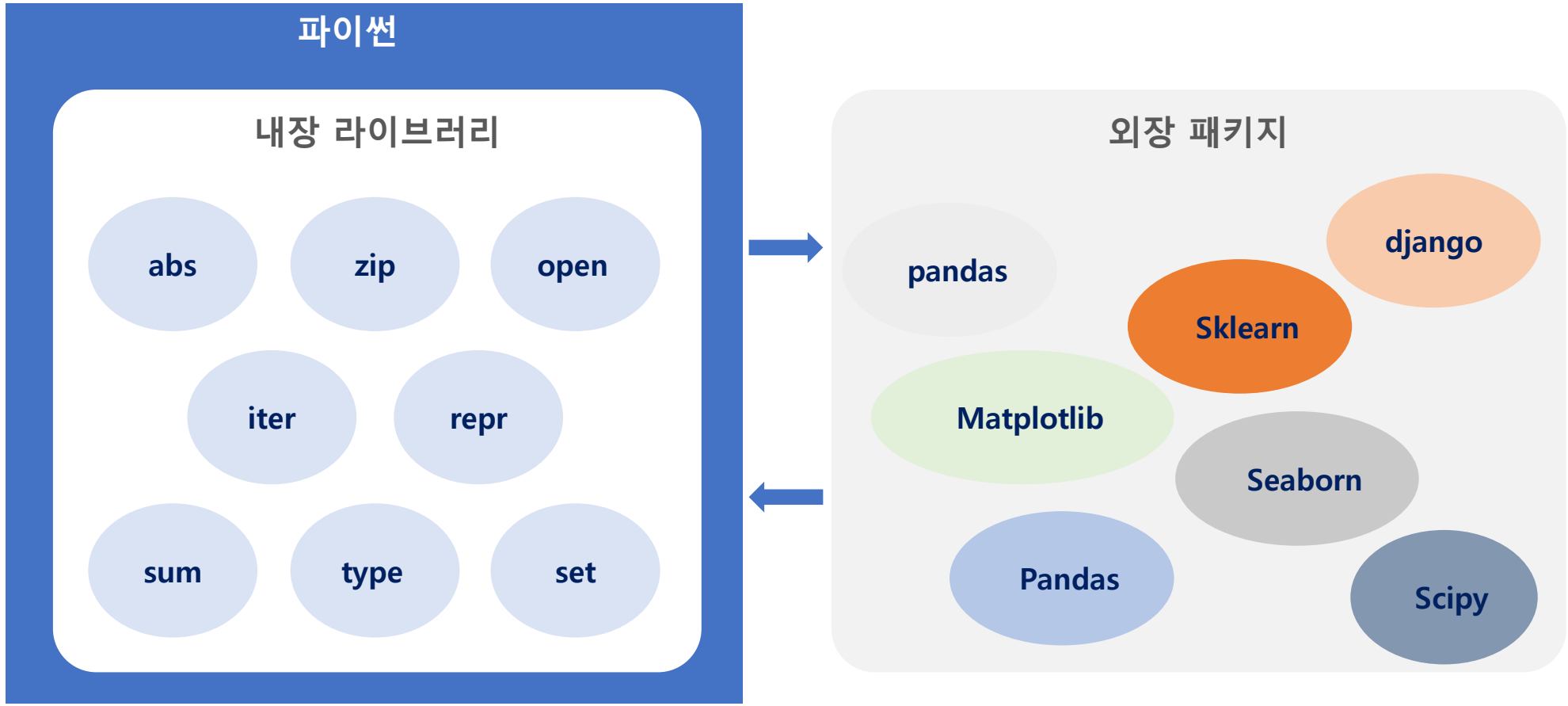



PART1. 데이터 분석 라이브러리(1)

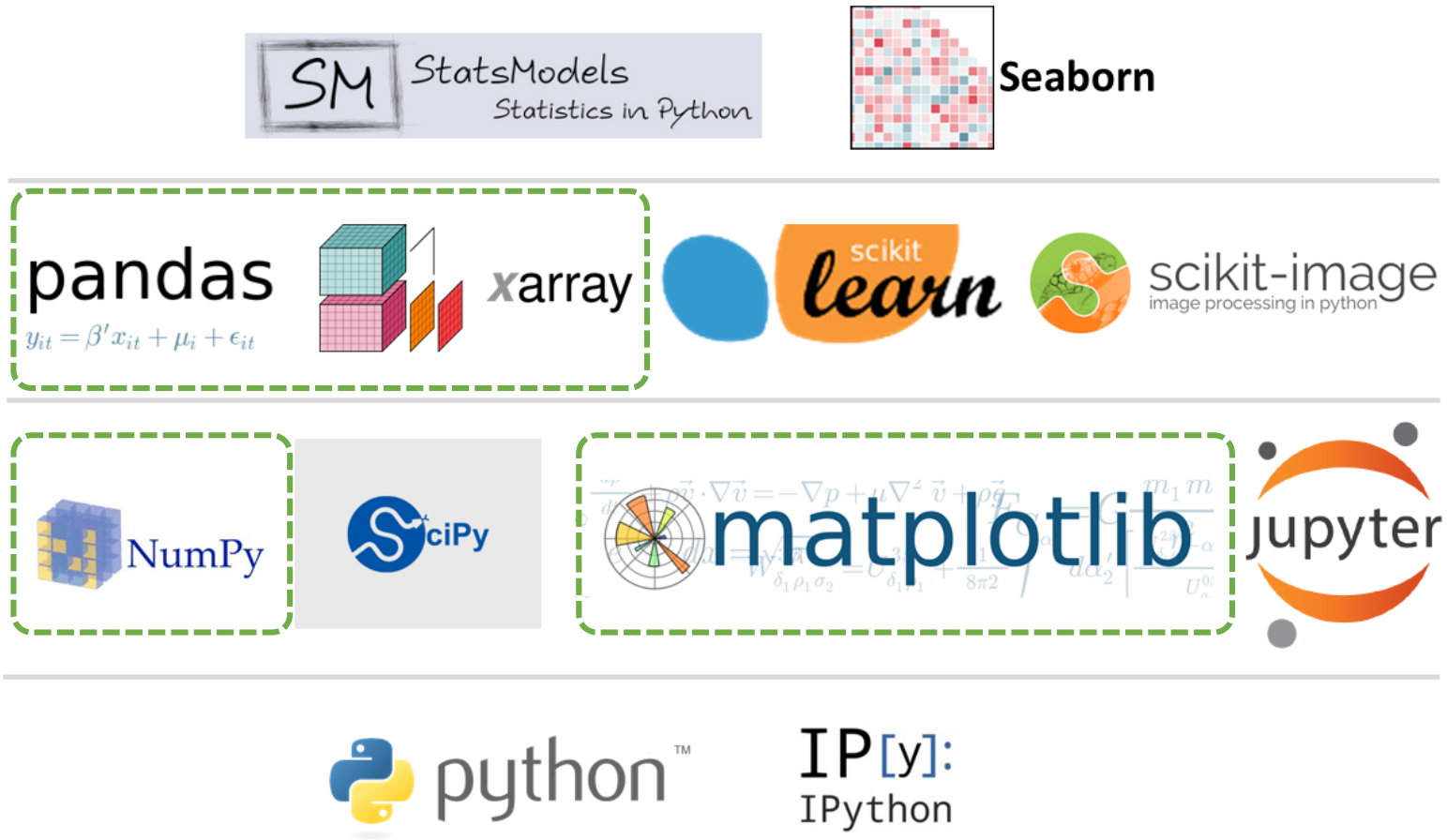
Enjoy your possibility

파이썬 데이터 처리/분석 라이브러리 - Numpy

파이썬 분석 라이브러리


파이썬
내장 라이브러리
abs
zip
open
iter
repr
sum
type
set
외장 패키지
pandas
django
Sklearn
Matplotlib
Seaborn
Pandas
Scipy

파이썬 데이터 분석 라이브러리



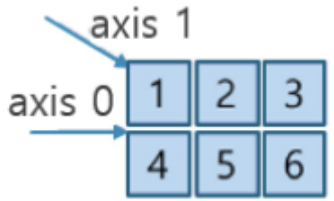
파이썬 라이브러리 - numpy

- 수치 계산 라이브러리 (Numpy Documentation- https://numpy.org/doc/1.17/contents.html)

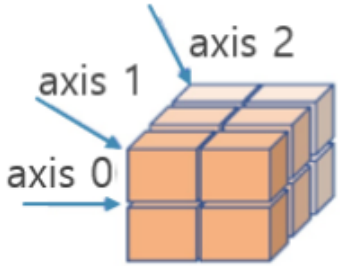
1) Numpy 개요

- numpy는 수치해석용 파이썬 패키지로 numerical python의 줄임말
- 다차원의 배열 자료구조 클래스인 ndarray 클래스를 지원
- 벡터와 행렬을 사용하는 선형대수 계산 사용
- 2005, Travis Oliphant 개발

1D array



2D array



3D array

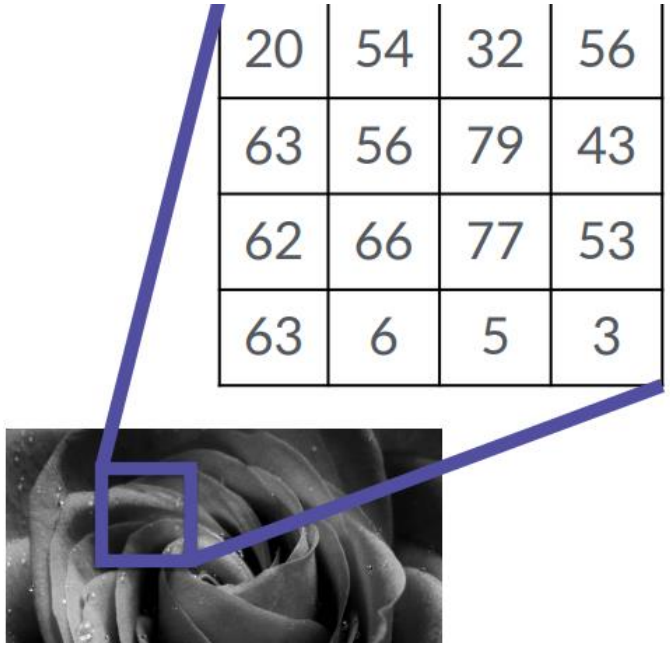
2) Numpy 특징

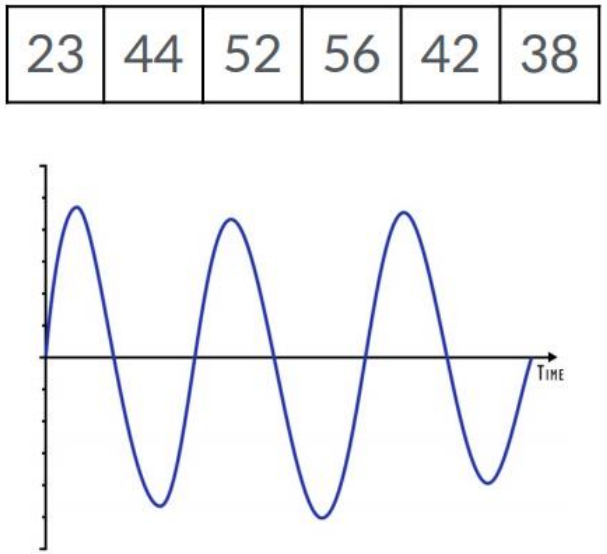
- numpy의 배열 연산은 c로 구현된 내부 반복문을 사용
- 파이썬 반복문에 비해 빠른 속도
- 벡터화 연산(vectorized operation)을 이용
- 간단한 코드로도 복잡한 선형 대수 연산을 수행
- 배열 인덱싱(array indexing)을 사용한 질의(query) 기능
- 하지만 고수준의 데이터 분석 기능을 제공하지 않아 Pandas 를 이용해 분석 수행함

- axis : 각 차원의 축
- rank : 축의 개수
- shape : 배열의 축 길이 (예: shape 가 (1000,10) 이면?)
- size : 배열의 사이즈

파이썬 라이브러리 - Numpy

- 넘파이를 왜 사용하는가?





반복문 없이
배열 처리 가능!

파이썬 리스트에 비해,
빠른 연산을 지원하고
메모리를 효율적으로
사용

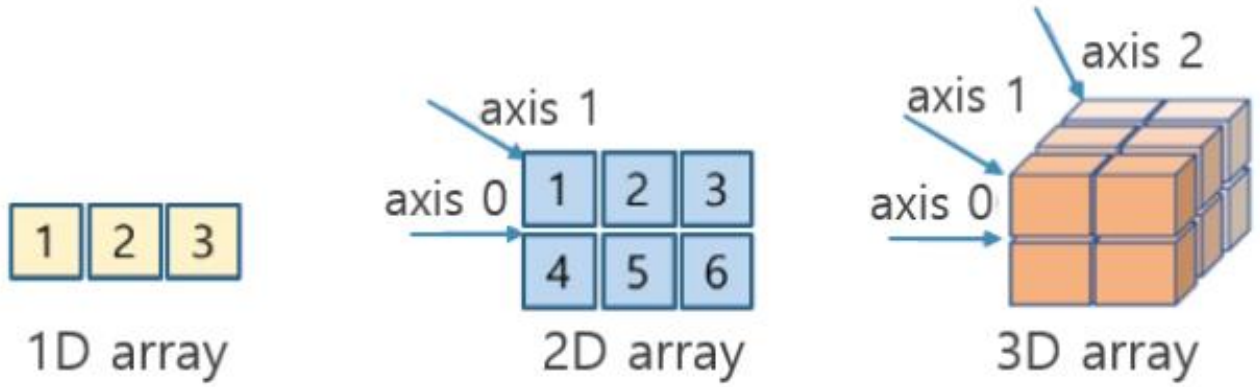
데이터의 대부분은 숫자 배열 형태로 이루어짐

numpy 설치 및 사용하기

설치하기	pip install numpy
가져오기	import numpy as np

- 모듈(라이브러리)을 호출하여 속성과 메서드를 사용한다
- numpy.sum( )을 간단히 별칭(alias)을 사용해 np.sum( )으로 사용

배열(Array)이란

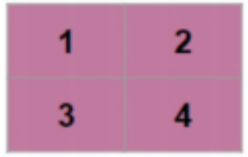


- 각 차원을 축(axis)라고 함
- 축의 개수는 차원의 개수인데 rank 라고도 합니다. ndim
- 배열의 차원을 shape라고 하고 tuple 로 표시합니다. (3,), (2, 3)
- shape 안의 숫자는 각 차원에 있는 원소의 개수입니다.
- 전체 원소의 개수는 size라고 합니다.

배열(Array) 생성하기

- np.array( )함수를 써서 만들기

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




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

- arange( )와 reshape( )를 써서 만들기

```
a = np.arange(8)
a.shape
a.reshape(2, 4)
```

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

배열의 연산(Operation)

- 리스트의 연산과 비슷하지만, 약간 다르다.: 리스트는 – 연산 지원 안함
- Numpy 함수를 쓰는 것보다 객체.함수( )가 편리하다
- Numpy의 연산은 원소들끼리 이루어진다 - Element wise
- Array의 형태(shape)가 안 맞으면, 자동으로 맞추어 주기도 한다.
  - broadcasting

배열의 연산(Operation)

- numpy 함수를 쓰는 것보다 객체.함수( )가 편리함

```
arr1 = np.array([[1,2], [3,4]])
arr2 = np.array([[5,6], [7,8]])
```

1	2
3	4

5	6
7	8

```
np.multiply(arr1, arr2)    # arr1*arr2
```

$$AB = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 1(5) + 2(7) & 1(6) + 2(8) \\ 3(5) + 4(7) & 3(6) + 4(8) \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

```
np.matmul(arr1, arr2) # arr1@arr2
```

```
array([[19, 22],
       [43, 50]])
```

배열의 연산(Operation)

- 리스트의 연산과 비슷하지만, 약간 다르다.: 리스트는 – 연산 지원 안함

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

a + b → [1, 2, 3, 4, 5, 6, 7, 8]

a - b

→ TypeError: unsupported operand type(s) for -: 'list' and 'list'

```
a = np.array( [1,2,3,4] )
b = np.array( [5,6,7,8] )
```

a + b np.add(a,b)

→ array([6, 8, 10, 12])

a – b np.subtract(a,b)

→ array([-4, -4, -4, -4])




PART2. 데이터 분석 라이브러리(2)

Enjoy your possibility

파이썬 데이터 처리/분석 라이브러리 - Pandas

파이썬 라이브러리 - pandas

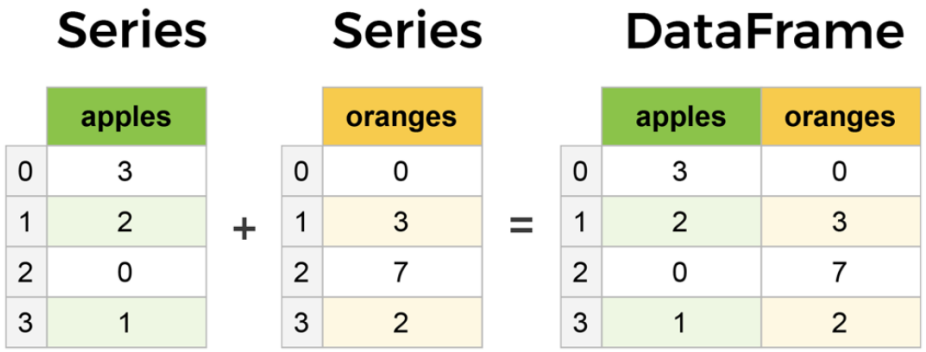
- 수치 지원 라이브러리 (Pandas Documentation - https://pandas.pydata.org/pandas-docs/stable/)

1) pandas 개요

- Pandas: 계량 경제학 용어인 panel data와 analysis의 합성어
- 구조화된 데이터를 빠르고 쉬우면서 다양한 형식으로 가공할 수 있는 풍부한 자료 구조와 함수를 제공
- R의 data.frame 자료구조 구현
- 금융회사에 다니고 있던 Wes Mckinney가 처음에 금융 데이터 분석을 위해 2008년 설계

2) pandas 특징

- 빅데이터 분석에 최적화 된 필수 패키지
- 데이터는 시계열(series)이나 표(table)의 형태
- 표 데이터를 다루기 위한 시리즈(series) 클래스 변환
- 데이터프레임(dataframe) 클래스 변환 :
  엑셀같은 스프레드시트 형태의 데이터를 관리할 수 있는 구조임



pandas 설치 및 사용하기

설치하기	pip install pandas
가져오기	import pandas as pd

- 모듈(라이브러리)을 호출하여 속성과 메서드를 사용
- pandas를 간단히 별칭(alias)을 사용해 pd로 사용

시리즈 (series) 주요 사항

- 시리즈 정의 및 특성
- 시리즈 생성
- 시리즈 확인
- 시리즈의 인덱스
- 시리즈 연산
- 시리즈 인덱싱
- 시리즈 슬라이싱(slicing)
- 시리즈의 데이터 갱신, 추가, 삭제
- 시리즈와 딕셔너리 자료형

시리즈 생성 및 확인

- 데이터를 리스트나 1차원 배열 형식으로 series 클래스 생성자에 넣어줌

```
# Series 정의하여 생성하기
obj = pd.Series([4, 5, -2, 8])

obj
```

- 시리즈 확인

```
# Series의 값만 확인하기
obj.values
```

```
# series의 인덱스 확인하기
obj.index
```

```
# series의 데이터타입 확인하기
obj.dtypes
```

시리즈 인덱싱

- 시리즈는 numpy 배열의 인덱스 방법처럼 사용 외에 인덱스 라벨을 이용한 인덱싱

```
1  a = pd.Series([1024, 2048, 3096, 6192],
2                index=["서울", "부산", "인천", "대구"])
3  a
```

```
서울    1024
부산    2048
인천    3096
대구    6192
dtype: int64
```

```
1  a[1], a["부산"]
```

```
(2048, 2048)
```

시리즈 슬라이싱

- 배열 인덱싱이나 인덱스 라벨을 이용한 슬라이싱(slicing)도 가능

```
1  a[1:3]
```

```
부산    2048
인천    3096
dtype: int64
```

```
1  a["부산":"대구"]
```

```
부산    2048
인천    3096
대구    6192
dtype: int64
```

데이터 프레임(DataFrame) 주요 사항

- 데이터프레임(DataFrame) 정의 및 특성
- 데이터프레임 생성
- 데이터프레임 열 갱신 추가
- 데이터프레임 인덱싱
- 데이터프레임 고급 인덱싱
- Boolean 인덱싱
- 데이터프레임 다루기
- 데이터 입출력
- 데이터 처리하기
- pandas 시계열 분석

데이터프레임 열 갱신

- 데이터를 리스트나 1차원 배열 형식으로 series 클래스 생성자에 넣어줌

```
# DataFrame을 만들면서 columns와 index를 설정

df=pd.DataFrame(data, columns=["year","name","points","penalty"],
                      index=(["one", "two", "three", "four", "five"]))
df
```

	year	name	points	penalty
one	2013	Choi	1.5	NaN
two	2014	Choi	1.7	NaN
three	2015	Choi	3.6	NaN
four	2016	Kim	2.4	NaN
five	2017	Park	2.9	NaN

```
# 특정 열을 선택하고, 값(0.5)을 대입
df["penalty"] = 0.5
df
```

	year	name	points	penalty
one	2013	Choi	1.5	0.5
two	2014	Choi	1.7	0.5
three	2015	Choi	3.6	0.5
four	2016	Kim	2.4	0.5
five	2017	Park	2.9	0.5

판다스 데이터 파일 읽고 쓰기

- csv 데이터 파일 읽어오기

```
# csv 파일로부터 데이터 읽어 데이터프레임 만들기: pandas.read_csv()
# sample1.csv 파일은 열 인덱스(c1, c2, c3)는 있으나 행 인덱스 없음.
pd.read_csv('data/sample1.csv')
```

	c1	c2	c3
0	1	1.11	one
1	2	2.22	two
2	3	3.33	three

- csv 데이터 파일 쓰기

```
# 데이터프레임 값을 csv 파일로 출력: pandas.to_csv()
df.to_csv("data/sample2.csv")
```




PART3. 데이터 분석 라이브러리(3)

Enjoy your possibility

파이썬 데이터 처리/분석 라이브러리 - Matplotlib

파이썬 라이브러리 - Matplotlib

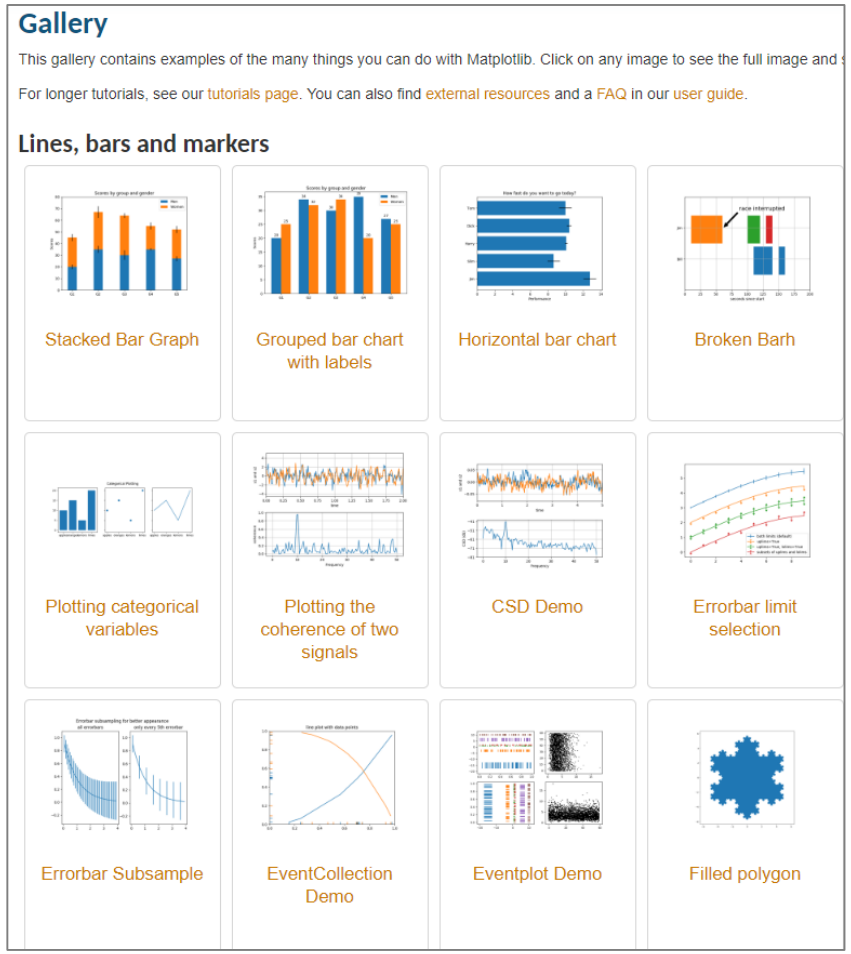
- 시각화 지원 라이브러리(Matplotlib Documentation - https://matplotlib.org/3.1.1/contents.html#)

1) Matplotlib 개요

- 시각화 라이브러리
- 각종 그래프나 챠트 등을 그리는 시각화 기능을 제공
- Tkinter, wxPython, Qt, GTK+ 등의 다양한 그래픽 엔진을 사용할 수 있음
- 2002, John D. Hunter 개발

2) Matplotlib 특징

- 빅데이터 분석에 최적화 된 필수 패키지
- 데이터는 시계열(series)이나 표(table)의 형태
- 표 데이터를 다루기 위한 시리즈(series) 클래스 변환
- 데이터프레임(dataframe) 클래스 변환



matplotlib 설치 및 사용하기

설치하기

```
pip install matplotlib
```

가져오기

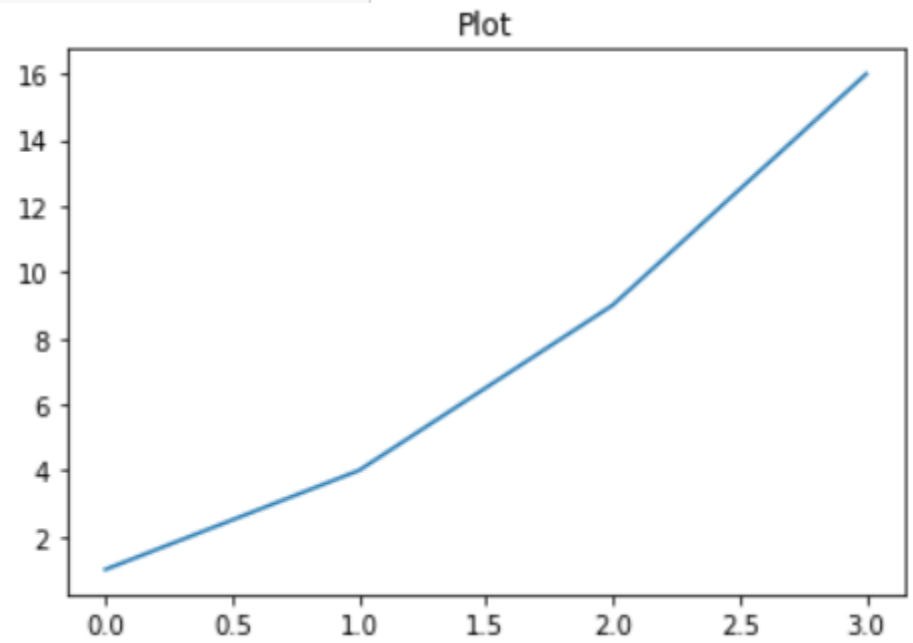
```
import matplotlib as mpl
import matplotlib.pylab as plt
```

- 모듈(라이브러리)을 호출하여 속성과 메서드를 사용한다
- **matplotlib** 을 간단히 별칭(alias)을 사용해 mpl 또는 하위 패키지 plt로 사용

라인 플롯

- 선을 그림
- 데이터가 시간, 순서등에 따라 어떻게 변화하는지 보여주기위해 사용
- 명령은 pylab 서브패키지의 plot 명령을 사용

```
1 import matplotlib.pylab as plt
2 %matplotlib inline
3
4 plt.title('Plot')
5 plt.plot([1, 4, 9, 16])    # y축 값
6 plt.show
```



라인 플롯 – 스타일 지정

- 색상, 마커, 선 순서로 지정하고 지정하지 않은 경우 디폴트 값이 적용됨

색상(color)

색 이름 혹은 약자를 사용하거나 # 문자로 시작하는 RGB 코드 사용

- blue : b
- green : g
- red : r
- black : k
- white : w

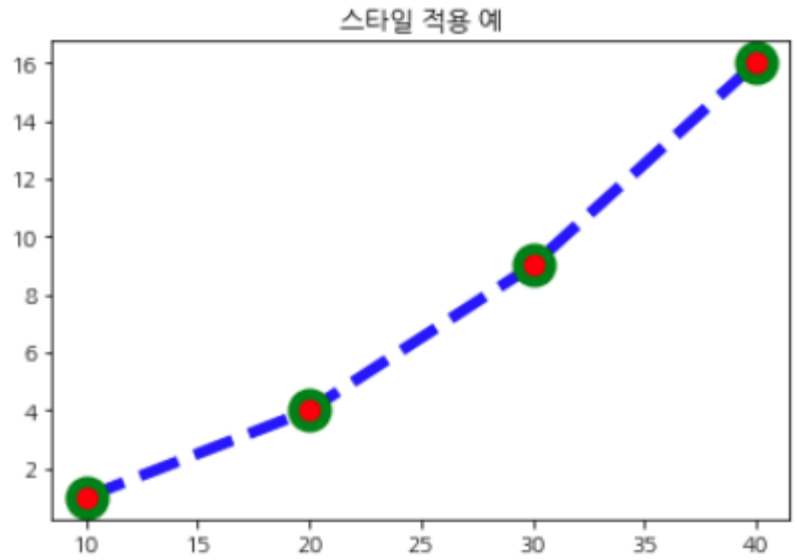
마커(maker)

- 스타(*) 마커
- 플러스(+) 마커
- 원(o) 마커

선(line style)

- 실선(-)
- 대시선(--)
- 대시-점선(-.)
- 점선(:)

```
1 plt.plot([10, 20, 30, 40], [1, 4, 9, 16], c="b",
2        lw=5, ls="--", marker="o", ms=15, mec="g", mew=5, mfc="r")
3 plt.title("스타일 적용 예")
4 plt.show()
```



라인 플롯 – 여러 개의 선 그리기

- 선을 여러개를 그리려면 x 데이터, y 데이터, 스타일 문자열을 반복하여 인수로 넘긴다.
- 이 경우에는 하나의 선을 그릴 때 처럼 x 데이터나 스타일 문자열을 생략할 수 없다.

```
1  import numpy as np
2
3  t = np.arange(0., 5., 0.5)
4  plt.title('라인 플롯에서 여러개 선 그리기')
5  plt.plot(t, t, 'r--', t, 0.5* t**2, 'bs:', t, 0.2* t**3, 'g^-')
6  plt.show()
```