30분 요약강좌 2

Python 활용편

이호준 김유진 김혜원 김대현 이승신 이현창 장하림 차경림

이호준

(現) 바울랩아이씨티기술연구원 대표이사
(現) 바울랩아이씨티컴퓨터학원 대표이사
(現) 사도출판 대표이사
(現) 바울랩미디어 대표이사
(現) 주식회사 위니브 대표이사
(現) GDG(Google Developers Group) JEJU Organizer
(現) 제주스타트업협회 부회장
(現) 제주대학교 풀스택 수업 강사
(現) 제주코딩베이스캠프 운영진

직책과 감투가 많지만, 사회에 공헌하며 미래를 꿈꾸는 아이들 가르치며 소박하게 살고 싶은 제주 도민입니다.

김대현

(前) 제주대학교 전산통계학과 전공
(現) 빅데이터 전략 마에스트로 교육생
(現) 바울랩아이씨티기술연구원 연구원

김유진

(前) 제주대학교 컴퓨터공학 전공
(現) 바울랩아이씨티기술연구원 연구원
제주코딩베이스캠프 Code Festival: Python 100제 집필

김혜원

(前) 제주대학교 컴퓨터공학 전공
(現) 바울랩아이씨티기술연구원 연구원
카카오와 함께하는 제주 코딩 베이스캠프 11기 스탭
코딩도장 튜토리얼로 배우는 Python 문제풀이 외 4권 집필

이승신

(前) 제주대학교 언론홍보학 전공
(現) 더큰내일센터 탐나는 인재 1기

이현창

(前) 제주대학교 전산통계학과 전공
(現) 빅데이터 전략 마에스트로 교육생
(現) 바울랩아이씨티기술연구원 연구원

장하림

(前) Liberty University mathematics major
(現) 빅데이터 전략 마에스트로 교육생
(現) 바울랩아이씨티기술연구원 연구원

차경림

(前) 제주대학교 산업디자인학부 문화조형디자인 전공
(現) 바울랩아이씨티기술연구원 연구원

이 책은 인쇄용으로 사용하시고,
아래 링크에서 노션(Notion)으로 된
Code Page를 보실 수 있으십니다.
실습을 하실 때에는 노션에서 Code를 복사해 사용하세요.

30분 요약강좌 2 – python 활용편

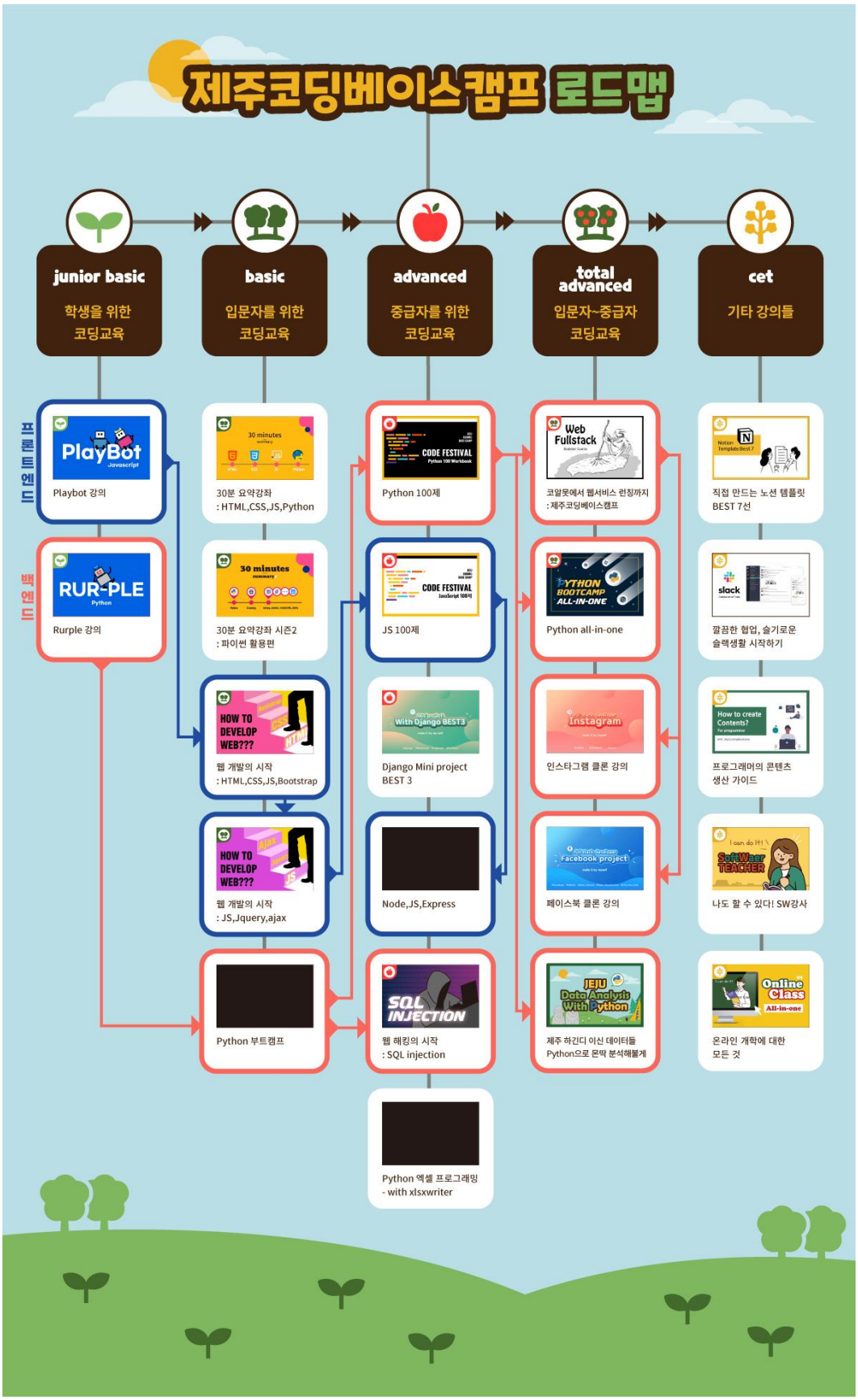
https://www.notion.so/30-f20d59b2401c4404b6cd1ee3a31556d1

이 책에 있는 30분 요약 강좌는
인프런에서 무료로 만나보실수 있습니다.

30분 요약강좌 2 - python 활용편

https://www.inflearn.com/course/제주코딩-웹개발-30분요약-시즌2

Road Map


제주코딩베이스캠프 로드맵
junior basic
학생을 위한 코딩교육
basic
입문자를 위한 코딩교육
advanced
중급자를 위한 코딩교육
total advanced
입문자~중급자 코딩교육
cet
기타 강의들
프론트엔드
백엔드
Playbot 강의
30분 요약강좌 : HTML,CSS,JS,Python
Python 100제
코알못에서 웹서비스 런칭까지 : 제주코딩베이스캠프
직접 만드는 노션 템플릿 BEST 7선
Rurple 강의
30분 요약강좌 시즌2 : 파이썬 활용편
JS 100제
Python all-in-one
깔끔한 협업, 슬기로운 슬랙생활 시작하기
웹 개발의 시작 : HTML,CSS,JS,Bootstrap
Django Mini project BEST 3
인스타그램 클론 강의
프로그래머의 콘텐츠 생산 가이드
웹 개발의 시작 : JS,Jquery,ajax
Node,JS,Express
페이스북 클론 강의
나도 할 수 있다! SW강사
Python 부트캠프
웹 해킹의 시작 : SQL injection
제주 하긴디 이신 데이터들 Python으로 몬딱 분석해볼게
온라인 개학에 대한 모든 것
Python 엑셀 프로그래밍 - with xlsxwriter

대상 독자

이 책은 Python을 배우고 싶은 코딩 초보자 분들에게 추천합니다. 함께 Python 기초부터 실제 데이터 분석을 해보면서 데이터 분석을 처음부터 끝까지 경험해 보세요. 코딩으로 데이터 분석 시각화를 하고자 하는 분, 실제 데이터를 수집하여 분석하는 기존의 마케팅 및 영업 관련 업무를 담당하고 계신 분들에게 추천합니다.

코딩을 많이 해본 적 없는 초보자를 위해 기초 이론부터 실전 프로젝트까지 하나하나 탄탄하게 배울 수 있습니다.

이 책의 내용

튜토리얼 별로 Jupyter Notebook 코드와 Python, 데이터 분석을 위한 이론적인 학습 내용이 있습니다. 이론을 통해 배우고 실전 코드를 치면서 쉽고 재미있게 Python을 배울 수 있습니다. Python의 기본 문법부터 자료 구조 및 데이터 시각화를 배울 수 있고, 기본 분석부터 실무자를 위한 실전 프로젝트까지 진행 할 수 있습니다. 데이터 분석을 위한 수학적 이론을 쉽게 배울 수 있습니다. 기초 이론부터 실전 프로젝트까지 빠르고 깊게 배울 수 있는 단기 집중 코스로 구성된 책입니다.

Contents

1. Numpy

2. Pandas

Contents

3. Visualization

4. Crawling

1. Numpy

📥 이 장에서 다루는 내용

Numpy란
연산
차원
Matrix 유형
집계함수

Numpy란

- C언어를 기반으로 내부 반복문을 사용하여 속도가 매우 빠르다.
- 다차원 배열(ndarray)을 편리하게 처리할 수 있도록 지원해주는 파이썬 라이브러리다.

※ array vs list

- array : 적은 메모리로 데이터를 빠르게 처리하고, 모든 원소는 같은 자료형을 가진다.
- list : 속도가 매우 느리고, 원소가 각각 다른 자료형을 가질 수 있다.

ndarray

- N-dimensional Array
- 다차원 배열을 지원한다.
- 서로 다른 타입의 데이터를 담을 수 없다.
- 내부 반복문을 사용해서 속도가 빠르다.
- 배열 간에 산술 연산이 가능하다.

데이터 타입 종류

1. int(8bit, 16bit, 32bit, 64bit) i1, i2, i4, i8

- 부호가 있다.
- 비트수 만큼 크기를 가지는 정수형이다.
- 저장할 수 있는 값의 범위 : -2^{n-1} ~ $2^{n-1}-1$

2. uint(8bit, 16bit, 32bit, 64bit) u1, u2, u4, u8

- 부호가 없다.
- 비트수 만큼 크기를 가지는 정수형이다.
- 저장할 수 있는 값의 범위 : 0 ~ $2^{n}-1$

3. float(16bit, 32bit, 64bit, 128bit) f2, f4, f8, f16

- 부호가 있다.
- 비트수 만큼 크기를 가지는 실수형이다.

4. 복소수형

- complex64 : 두개의 32비트 부동 소수점으로 표시되는 복소수 c8
- complex128 : 두개의 64비트 부동소수점으로 표시되는 복소수 c16

5. unicode

- 고정 길이 문자열 unicode

6. bool

- True, False

- dtype : 자료형 확인
- astype : 자료형 변경

```python
import numpy as np
```

Python

```python
data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

a = np.array(data)
a
```

Python

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

Python

```python
a.dtype
```

Python

```
Out[-]
dtype('int32')
```

Python

```python
a = a.astype(np.float32)
a
```

Python

```
Out[-]
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]], dtype=float32)
```

Python

```python
a[0][1]
```

Python

```
Out[-]
2.0
```

Python

```python
a[0]
```

Python

```
Out[-]
array([1., 2., 3.], dtype=float32)
```

Python

- np.arange(start, stop, step) : step에 따라 증가하는 수열을 생성한다.
- reshape : 만들어진 배열의 데이터를 그대로 가지고 형태를 바꿔준다.

```python
np.arange(1, 10, 2) # (start, stop, step)
```

Python

Copy to clipboard

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

Python

```python
np.arange(1, 10).reshape(3, 3)
```

Python

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

Python

```python
np.arange(1, 13).reshape(3, 2, 2)
```

Python

```
Out[-]
array([[[ 1,  2],
        [ 3,  4]],

       [[ 5,  6],
        [ 7,  8]],

       [[ 9, 10],
        [11, 12]]])
```

Python

- nan
  - 자료형이 Float형이므로 Integer에서는 오류가 발생한다.
  - 배열에서 연산할 경우 오류가 발생하지 않지만 결과값이 NaN이 된다.

```python
np.nan * 10
```

Python

```
Out[-]
nan
```

Python

```python
a[0][1] = np.nan
a
```

Python

```
Out[-]
array([[ 1., nan,  3.],
       [ 4.,  5.,  6.],
       [ 7.,  8.,  9.]], dtype=float32)
```

Python

```python
a = np.arange(1, 10).reshape(3, 3)
a
```

Python

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

Python

```python
a[0][1] = np.nan # integer에서는 nan을 사용 할 수 없다.
```

Python

```
Out[-]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-10bef5bd968c> in <module>
----> 1 a[0][1] = np.nan

ValueError: cannot convert float NaN to integer
```

Python

```python
a = np.linspace(1, 10, 20) # (start, stop, num)
a
```

Python

```
Out[-]
array([ 1.        ,  1.47368421,  1.94736842,  2.42105263,  2.89473684,
        3.36842105,  3.84210526,  4.31578947,  4.78947368,  5.26315789,
        5.73684211,  6.21052632,  6.68421053,  7.15789474,  7.63157895,
        8.10526316,  8.57894737,  9.05263158,  9.52631579, 10.        ])
```

Python

연산

- 반복문을 사용하지 않아도, 배열의 모든 원소는 반복연산이 사용된다.
- 선형 대수 식을 사용하여 연산 가능하다.
  - np.dot, @ : 행렬 곱 구하는 연산

```python
data = np.arange(1, 10).reshape(3, 3)
data
```

Python

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

Python

```python
data + data
```

Python

```
Out[-]
array([[ 2,  4,  6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

Python

```python
[1, 2, 3] + [4, 5, 6] # 리스트끼리 더하면 연결된다.
```

Python

```
Out[-]
[1, 2, 3, 4, 5, 6]
```

Python

```python
data - data
```

Python

```
Out[-]
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

Python

```python
data * data
```

Python

```
Out[-]
array([[ 1,  4,  9],
       [16, 25, 36],
       [49, 64, 81]])
```

Python

```python
data / data
```

Python

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

Python

```python
np.dot(data, data)
```

Python

```
Out[-]
array([[ 30,  36,  42],
       [ 66,  81,  96],
       [102, 126, 150]])
```

Python

```python
data@data
```

Python

```
Out[-]
array([[ 30,  36,  42],
       [ 66,  81,  96],
       [102, 126, 150]])
```

Python

차원

- 0차원 : Scalar (하나의 데이터 값으로만 존재하는 것)
- 1차원 : Vector (숫자들의 배열 (1D array))
- 2차원 : Matrix (숫자들의 2D array (rows: 행, columns: 열))
- 3차원 이상 : Tensor (숫자들의 다차원 배열)

```python
# 0차원
a = np.array(1)
print(a)
print(a.shape)
print(a.ndim)
```

Python

```
Out[-]
1
()
0
```

Python

```python
# 1차원
a = np.array([1])
print(a)
print(a.shape)
print(a.ndim)
```

Python

```
Out[-]
[1]
(1,)
1
```

Python

```python
# 1차원
a = np.array([1, 2, 3, 4, 5])
print(a)
print(a.shape)
print(a.ndim)
```

Python

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

Python

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

Python

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

Python

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

Python

```
Out[-]
[[1]]
(1, 1)
2
```

Python

```python
# 3차원
a = np.array([[[1, 2], [3, 4], [5, 6]], [[7, 8], [9, 10], [11, 12]]])
print(a)
print(a.shape)
print(a.ndim)
```

Python

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

 [[ 7  8]
  [ 9 10]
  [11 12]]]

(2, 3, 2)

3
```

Python

matrix 유형

- zeros : 0으로 초기화된 배열 생성
- ones : 1로 초기화 된 배열 생성
- eye : 주대각선의 원소가 모두 1이고 나머지 원소는 0인 정사각행렬 (단위행렬)
- empty : 초기화 하지 않고 배열만 생성, 기존에 메모리에 저장되어 있는 값으로 나타남
- full : 지정한 숫자로 초기화
- linespace(start, end(포함), number, endpoint =) : 선형 구간을 지정한 개수만큼 분할한다.
  - endpoint = True or False : 마지막 값을 포함시킬지 시키지 않을지 선택

```python
a = np.arange(12).reshape(2,3,2)
a
```

Python

```
Out[-]
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5]],

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

Python

```python
b = np.ones(12)
b
```

Python

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

Python

```python
b = np.ones(12).reshape(2, 3, 2)
b
```

Python

```
Out[-]
array([[[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]]])
```

Python

```python
c = np.zeros(12).reshape(2, 3, 2)
c
```

Python

```
Out[-]
array([[[0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.]]])
```

Python

```python
d = np.eye(3)
d
```

Python

```
Out[-]
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Python

```python
e = np.zeros([3, 4])
e
```

Python

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

Python

```
e = np.zeros([3, 4, 2])
e
```

Python

```
Out[-]
array([[[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]]])
```

Python

```
f = np.empty([2, 3])
f
```

Python

```
Out[-]
array([[2.12199579e-314, 6.36598737e-314, 1.06099790e-313],
       [1.48539705e-313, 1.90979621e-313, 2.33419537e-313]])
```

Python

```python
g = np.full((3, 4), 1000)
g
```

Python

```
Out[-]
array([[1000, 1000, 1000, 1000],
       [1000, 1000, 1000, 1000],
       [1000, 1000, 1000, 1000]])
```

Python

```python
h = np.linspace(2, 10, 6)
h
```

Python

```
Out[-]
array([ 2. ,  3.6,  5.2,  6.8,  8.4, 10. ])
```

Python

집계함수

- mean : 평균을 구하는 함수
- median : 데이터를 크기로 정렬하고 그 중 가운데 수 출력하는 함수

  ※ 만약 데이터 개수가 짝수일 경우 가장 가운데의 두 수의 평균을 구한다.
- var : 분산 구하는 함수
- std : 표준편차 구하는 함수
- sum : 합계 구하는 함수
- max : 가장 큰 수를 구하는 함수
- min : 가장 작은 수를 구하는 함수

```python
a = np.arange(10).reshape(2,5)
a
```

Python

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

Python

```python
a[0][0] = 100
a
```

Python

```
Out[-]
array([[100,   1,   2,   3,   4],
       [  5,   6,   7,   8,   9]])
```

Python

```
a[0, 1] = 1000
a
```

Python

```
Out[-]
array([[ 100, 1000,    2,    3,    4],
       [   5,    6,    7,    8,    9]])
```

Python

```
np.mean(a)
```

Python

```
Out[-]
114.4
```

Python

```
np.median(a)
```

Python

```
Out[-]
6.5
```

Python

```
np.std(a)
```

Python

```
Out[-]
296.5485457728633
```

Python

```python
np.var(a)
```

Python

```
Out[-]
87941.04
```

Python

```python
np.sum(a)
```

Python

```
Out[-]
1144
```

Python

```python
sum(a)
```

Python

```
Out[-]
array([ 105, 1006,    9,   11,   13])
```

Python

```python
a
```

Python

```
Out[-]
array([[ 100, 1000,    2,    3,    4],
       [   5,    6,    7,    8,    9]])
```

Python

```python
np.sum(a, axis=0)
```

Python

```
Out[-]
array([ 105, 1006,    9,   11,   13])
```

Python

```python
np.sum(a, axis=1)
```

Python

```
Out[-]
array([1109,   35])
```

Python

```python
np.max(a)
```

Python

```
Out[-]
1000
```

Python

```python
np.min(a)
```

Python

```
Out[-]
2
```

Python

2. Pandas

이 장에서 다루는 내용

Pandas

Series

DataFrame

Pandas

- 데이터를 분석할 때 가장 많이 쓰이는 라이브러리다.
- 행과 열을 쉽게 처리할 수 있는 함수를 제공하는 도구이다.

  ※ 각 열은 단일 데이터 형식만 저장
- numpy보다 유연하게 수치 연산 가능하다.

```python
import pandas as pd

pd.__version__

pd? # tab 누르면 사용가능한 여러가지 method를 볼 수 있다.
```

Python

```
Out[-]
1.0.1
```

Python

Series

- index와 values로 이루어진 1차원 배열이다.
- 모든 유형의 데이터를 보유할 수 있다.
- 인덱스를 지정해 줄 수 있다.
- 인덱스 길이는 데이터의 길이와 같아야 한다.
- 명시적 인덱스와 암묵적 인덱스를 가진다.
  - loc : 인덱스값을 기반으로 행 데이터를 읽는다.
  - iloc : 정수 인덱스을 기반으로 행 데이터를 읽는다.

```python
import pandas as pd
import numpy as np
```

Python

```python
data = np.arange(0, 50, 10)
data
```

Python

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

Python

```python
a = pd.Series(data, index=['a', 'b', 'c', 'd', 'e'])
print(a)
```

Python

```
Out[-]
a     0
b    10
c    20
d    30
e    40
dtype: int32
```

Python

```python
b = pd.Series(data)
b
```

Python

Copy to clipboard

```
Out[-]
0     0
1    10
2    20
3    30
4    40
dtype: int32
```

Python

```python
a['b']
```

Python

```
Out[-]
10
```

Python

```python
a.loc['b']
```

Python

```
Out[-]
10
```

Python

```python
a.iloc[1]
```

Python

```
Out[-]
10
```

Python

산술연산

```
a
```

Python

```
Out[-]
a     0
b    10
c    20
d    30
e    40
dtype: int32
```

Python

```
# 더하기
a + 10
```

Python

```
Out[-]
a    10
b    20
c    30
d    40
e    50
dtype: int32
```

Python

```python
# 빼기
a - 10
```

Python

```
Out[-]
a   -10
b     0
c    10
d    20
e    30
dtype: int32
```

Python

```python
# 곱하기
a * 10
```

Python

Copy to clipboard

```
Out[-]
a      0
b    100
c    200
d    300
e    400
dtype: int32
```

Python

```python
# 거듭제곱
a ** 2
```

Python

```
Out[-]
a       0
b     100
c     400
d     900
e    1600
dtype: int32
```

Python

```python
# 나누기
a / 5
```

Python

```
Out[-]
a    0.0
b    2.0
c    4.0
d    6.0
e    8.0
dtype: float64
```

Python

```python
# 몫
a // 5
```

Python

```
Out[-]
a    0
b    2
c    4
d    6
e    8
dtype: int32
```

Python

```python
# 나머지
a % 3
```

Python

```
Out[-]
a    0
b    1
c    2
d    0
e    1
dtype: int32
```

Python

비교 연산자

```python
a
```

Python

```
Out[-]
a     0
b    10
c    20
d    30
e    40
dtype: int32
```

Python

```python
a > 15
```

Python

```
Out[-]
a    False
b    False
c     True
d     True
e     True
dtype: bool
```

Python

```python
a[a>15]
```

Python

```
Out[-]
c    20
d    30
e    40
dtype: int32
```

Python

집계함수

- add : 더하기 함수
- sub : 빼기 함수
- mul : 곱하기 함수
- div : 나누기 함수
- mod : 나머지 구하는 함수
- min : 최소값 구하는 함수
- max : 최대값 구하는 함수
- mean : 평균 구하는 함수
- median : 중앙값 구하는 함수
- std : 표준편차 구하는 함수
- var : 분산 구하는 함수

```python
a.add(100)
```

Python

```
Out[-]
a    100
b    101
c    102
d    103
e    104
dtype: int32
```

Python

```python
a.sub(100)
```

Python

```
Out[-]
a   -100
b    -90
c    -80
d    -70
e    -60
dtype: int32
```

Python

```python
a.mul(100)
```

Python

```
Out[-]
a       0
b    1000
c    2000
d    3000
e    4000
dtype: int32
```

Python

```python
a.div(100)
```

Python

```
Out[-]
a    0.0
b    0.1
c    0.2
d    0.3
e    0.4
dtype: float64
```

Python

```python
a.mod(3)
```

Python

```
Out[-]
a    0
b    1
c    2
d    0
e    1
dtype: int32
```

Python

```python
a.min()
```

Python

```
Out[-]
0
```

Python

```python
a.max()
```

Python

```
Out[-]
40
```

Python

```python
a.sum()
```

Python

```
Out[-]
100
```

Python

```python
a.mean()
```

Python

```
Out[-]
20.0
```

Python

```python
a.median()
```

Python

```
Out[-]
20.0
```

Python

```python
a.std()
```

Python

```
Out[-]
15.811388300841896
```

Python

```python
a.var()
```

Python

```
Out[-]
250.0
```

Python

DataFrame

- 2차원 배열에 행과 열에 인덱스를 붙인 것이다.
- 가장 기본적인 데이터 구조이다.

```python
# 50부터 100사이, 4x3형태로 랜덤 생성
rawData = np.random.randint(50, 100, size=(4, 3))
rawData
```

Python

```
Out[-]
array([[72, 61, 98],
       [83, 90, 91],
       [55, 71, 89],
       [71, 71, 76]])
```

Python

```python
df = pd.DataFrame(rawData,
                  index=['1반', '2반', '1반', '2반'],
                  columns=['국', '영', '수'])
df
```

Python

```
Out[-]
    국  영  수
1반 72  61  98
2반 83  90  91
1반 55  71  89
2반 71  71  76
```

Python

```python
#df[0] -> Error
df['국']
```

Python

```
Out[-]
1반    72
2반    83
1반    55
2반    71
Name: 국, dtype: int32
```

Python

```python
df
```

Python

```
Out[-]
    국  영  수
1반 72  61  98
2반 83  90  91
1반 55  71  89
2반 71  71  76
```

Python

```python
# 열을 추가하여 평균 구하기
df['평균'] = round((df['국']+df['영']+df['수'])/3, 2)
df
```

Python

```
Out[-]
    국  영  수  평균
1반 72  61  98  77.00
2반 83  90  91  88.00
1반 55  71  89  71.67
2반 71  71  76  72.67
```

Python

```python
# 열('na') 추가하고 NaN값을 넣기
df["na"] = np.nan
df
```

Python

```
Out[-]
    국  영  수  평균  na
1반 72  61  98  77.00 NaN
2반 83  90  91  88.00 NaN
1반 55  71  89  71.67 NaN
2반 71  71  76  72.67 NaN
```

Python

```python
# del 삭제
del df['na']
df
```

Python

```
Out[-]
    국  영  수  평균
1반 72  61  98  77.00
2반 83  90  91  88.00
1반 55  71  89  71.67
2반 71  71  76  72.67
```

Python

```python
df[df.평균 > 75]
```

Python

```
Out[-]
    국  영  수  평균
1반 72  61  98  77.0
2반 83  90  91  88.0
```

Python

```python
df = df.drop(["평균"], axis = 'columns')
df
```

Python

```
Out[-]
    국  영  수
1반 72  61  98
2반 83  90  91
1반 55  71  89
2반 71  71  76
```

Python

결측값 처리

1. NaN
   - 자료형이 Float형이다.
   - 배열에서 연산할 경우 오류가 발생하지 않지만 결과값이 NaN이 된다.
2. None
   - 자료형이 None이다.
   - 배열 연산을 할 경우 오류가 발생한다.
3. 처리방법
   - isnull() : 결측값 확인 (결측 이면 True , 결측이 아니면 False)
   - notnull() : 결측값 확인 (결측 이면 False , 결측이 아니면 True)
   - dropna() : 결측값 삭제
     - inplace = True : drop후 원본에 반영
   - fillna(Num) : 결측값을 Num으로 채워 넣는다.

```python
df = df.astype('float64') # 타입 변경
df
```

Python

```
Out[-]
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
1반 55.0  71.0  89.0
2반 71.0  71.0  76.0
```

Python

```python
df['수'][2] = np.nan
df
```

Python

```
Out[-]
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
1반 55.0  71.0   NaN
2반 71.0  71.0  76.0
```

Python

```python
df.dropna(axis=0) # inplace=True # axis=0 : 행 삭제
```

Python

```
Out[-]
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
2반 71.0  71.0  76.0
```

Python

```python
df
```

Python

```
Out[-]
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
1반 55.0  71.0   NaN
2반 71.0  71.0  76.0
```

Python

```python
df.dropna(axis=1) # axis=1 : 열 삭제
```

Python

```
Out[-]
      국    영
1반 72.0  61.0
2반 83.0  90.0
1반 55.0  71.0
2반 71.0  71.0
```

Python

```python
df.fillna('hello')
```

Python

```
Out[-]
      국    영      수
1반 72.0  61.0     98
2반 83.0  90.0     91
1반 55.0  71.0  hello
2반 71.0  71.0     76
```

Python

```python
df.fillna(0) # 0으로 대체
```

Python

```
Out[-]
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
1반 55.0  71.0   0.0
2반 71.0  71.0  76.0
```

Python

```python
df.fillna(df.mean()) # 평균으로 대체
```

Python

```
Out[-]
      국    영         수
1반 72.0  61.0  98.000000
2반 83.0  90.0  91.000000
1반 55.0  71.0  88.333333
2반 71.0  71.0  76.000000
```

Python

MultiIndex

- Index를 설정할 때 리스트의 리스트 형태로 넣어주면 다중 인덱스가 설정이 된다.

```
df
```

Python

```
Out[-]
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
1반 55.0  71.0   NaN
2반 71.0  71.0  76.0
```

Python

```
# index 와 columns을 바꿈
df.T
```

Python

```
Out[-]
     1반   2반   1반   2반
국  72.0  83.0  55.0  71.0
영  61.0  90.0  71.0  71.0
수  98.0  91.0   NaN  76.0
```

Python

```
df
```

Python

```
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
1반 55.0  71.0   NaN
2반 71.0  71.0  76.0
```

Python

```python
df.index = [['1학년', '1학년', '2학년', '2학년'], ['1반', '2반', '1반', '2반']]
df
```

Python

```
Out[-]
            국    영    수
1학년 1반 72.0  61.0  98.0
      2반 83.0  90.0  91.0
2학년 1반 55.0  71.0   NaN
      2반 71.0  71.0  76.0
```

Python

```python
df.columns = [['언어', '언어', '수리'],['국', '영', '수']]
df
```

Python

```
Out[-]
               언어   수리
            국    영    수
1학년 1반 72.0  61.0  98.0
      2반 83.0  90.0  91.0
2학년 1반 55.0  71.0   NaN
      2반 71.0  71.0  76.0
```

Python

```python
df['언어']['국']
```

Python

```
Out[-]
1학년  1반    72.0
     2반    83.0
2학년  1반    55.0
     2반    71.0
Name: 국, dtype: float64
```

Python

```python
df.iloc[0]
```

Python

```
Out[-]
언어  국    72.0
    영    61.0
수리  수    98.0
Name: (1학년, 1반), dtype: float64
```

Python

```python
df.loc['1학년']
```

Python

```
Out[-]
          언어   수리
      국    영    수
1반 72.0  61.0  98.0
2반 83.0  90.0  91.0
```

Python

```python
df.loc['1학년'].loc['1반']
```

Python

```
Out[-]
언어  국    72.0
    영    61.0
수리  수    98.0
Name: 1반, dtype: float64
```

Python

데이터 사전 분석

- info() : DataFrame을 구성하는 행과 열에 대한 정보를 나타내 주는 함수
- head(n) : DataFrame의 처음부터 n줄의 행을 출력
- tail(n) : DataFrame의 마지막 n줄의 행을 출력
- describe() : Series, DataFrame의 각 열에 대한 요약 통계
- dtypes : 데이터 자료형 확인

```
df
```

Python

```
Out[-]
                 언어   수리
              국   영    수
1학년 1반 72.0  61.0  98.0
      2반 83.0  90.0  91.0
2학년 1반 55.0  71.0   NaN
      2반 71.0  71.0  76.0
```

Python

```
df.info()
```

Python

```
Out[-]
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 4 entries, (1학년, 1반) to (2학년, 2반)
Data columns (total 3 columns):
(언어, 국)    4 non-null float64
(언어, 영)    4 non-null float64
(수리, 수)    3 non-null float64
dtypes: float64(3)
memory usage: 248.0+ bytes
```

Python

```python
df.head()
```

Python

```
Out[-]
                언어   수리
             국    영    수
1학년 1반  72.0  61.0  98.0
      2반  83.0  90.0  91.0
2학년 1반  55.0  71.0   NaN
      2반  71.0  71.0  76.0
```

Python

```python
df.tail()
```

Python

```
Out[-]
                언어   수리
             국    영    수
1학년 1반  72.0  61.0  98.0
      2반  83.0  90.0  91.0
2학년 1반  55.0  71.0  NaN
      2반  71.0  71.0  76.0
```

Python

```python
df.dtypes
```

Python

```
Out[-]
언어  국    float64
    영    float64
수리  수    float64
dtype: object
```

Python

```python
df.describe()
```

Python

```
Out[-]
                    언어       수리
             국        영         수
count  4.000000  4.000000  3.000000
 mean 70.250000 73.250000 88.333333
  std 11.528949 12.120919 11.239810
  min 55.000000 61.000000 76.000000
  25% 67.000000 68.500000 83.500000
  50% 71.500000 71.000000 91.000000
  75% 74.750000 75.750000 94.500000
  max 83.000000 90.000000 98.000000
```

Python

```python
df.isnull() # 결측값 확인, 결과는 bool 형태로 출력
```

Python

```
Out[-]
                 언어   수리
             국    영    수
1학년 1반 False False False
      2반 False False False
2학년 1반 False False  True
      2반 False False False
```

Python

```python
df.isnull().sum() # 결측값 개수 확인
```

Python

```
Out[-]
언어  국    0
    영    0
수리  수    1
dtype: int64
```

Python

값의 연결

- concat : DataFrame끼리 결합
  - axis=0 or 1 : 아래로 데이터 연결 / 옆으로 데이터 연결
- append : 마지막 행에 데이터를 추가

※ concatenate : 배열끼리 결합

```python
a = pd.DataFrame(np.arange(1, 10).reshape(3, 3))
a
```

Python

```
Out[-]
  0 1 2
0 1 2 3
1 4 5 6
2 7 8 9
```

Python

```python
b = pd.Series(np.arange(10, 40, 10))
b
```

Python

```
Out[-]
0    10
1    20
2    30
dtype: int32
```

Python

```python
pd.concat([a, b], axis=1)
```

Python

```
Out[-]
  0 1 2  0
0 1 2 3 10
1 4 5 6 20
2 7 8 9 30
```

Python

```python
pd.concat([a, b], axis=1, ignore_index=True)
```

Python

```
Out[-]
  0 1 2  3
0 1 2 3 10
1 4 5 6 20
2 7 8 9 30
```

Python

```python
a.append(b, ignore_index=True)
```

Python

```
Out[-]
   0   1   2
0  1   2   3
1  4   5   6
2  7   8   9
3 10  20  30
```

Python

3. Visualization

이 장에서 다루는 내용

Matplotlib

Plotly

이미지 분석

Matplotlib

- 데이터를 Chart, Plot으로 시각화해주는 라이브러리이다.
- 2003년 0.1 출시, 17년이된 패키지
- 가장 많이 사용되는 시각화 패키지이다.
- 기본 구성 : 그림(figure), 축(axes)
- % matplotlib inline : jupyter notebook 내에서 output을 보여준다.

Basic Attributes

- alpha : 투명도
- logy : Y축 Log scaling
- kind : line, bar, barh, kde
- subplots : 여러개의 plot 그리기
- legend : subplot의 범례 지정
- xlim, ylim : X축과 Y축의 경계(axis로 한번에 그릴 수 있음)
- grid : 그리드 표현(True, False)
- title : 그래프의 제목 지정
- linewidth : 라인 넓이
- color : 색
- linestyle : 실선, 점선, 1점 쇄선 등
- marker : 마커 지정
- markerfacecolor : 마커 색
- markersize : 마커 크기
- 그 외 Attribute : https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html

Line Plot

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Python

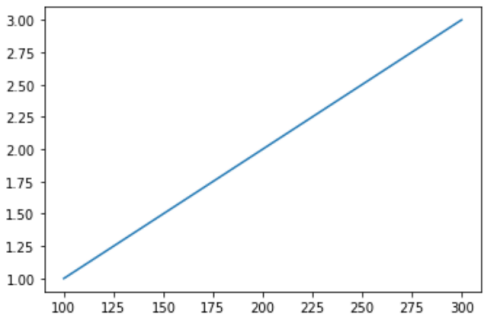
```python
%matplotlib inline
```

Python

```python
x = [100, 200, 300]
y = [1, 2, 3]
plt.plot(x, y)
```

Python

Out[-]

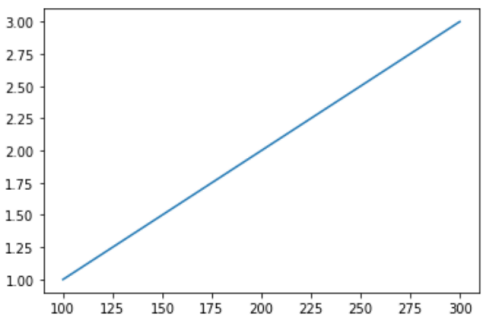


```python
x = [100, 200, 300]
y = [1, 2, 3]

# y 먼저 입력
value = pd.Series([1, 2, 3], [100, 200, 300])
plt.plot(value)
```

Python

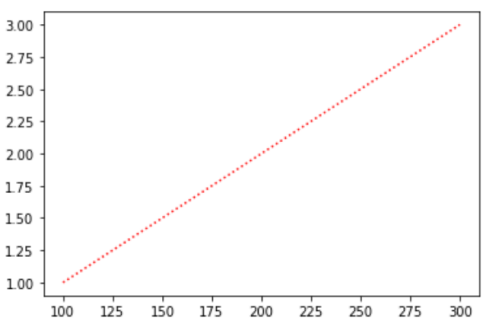
Out[-]




```python
x = [100, 200, 300]
y = [1, 2, 3]
plt.plot(x, y, ':r') # red색 점선
```

Python

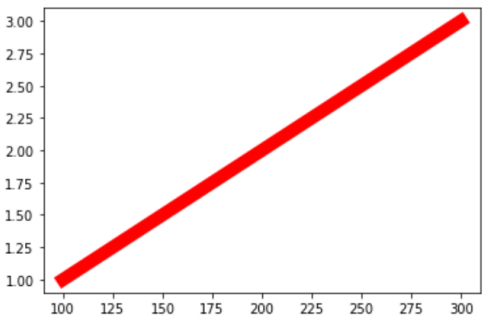
Out[-]



```python
x = [100, 200, 300]
y = [1, 2, 3]
plt.plot(x, y, color='#ff0000', linewidth=10) # 16진수 색상 값 (RGB) , 선두께
```

Python

Out[-]

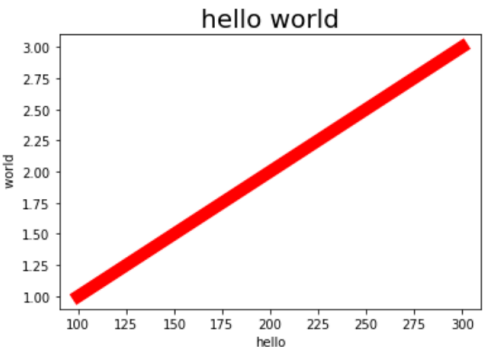



```python
x = [100, 200, 300]
y = [1, 2, 3]
plt.plot(x, y, color='#ff0000', linewidth=10)
plt.title('hello world', fontsize=20)  # 제목
plt.xlabel('hello', fontsize=10)  # x축 이름
plt.ylabel(' world', fontsize=10) # y축 이름
```

Python

Out[-]

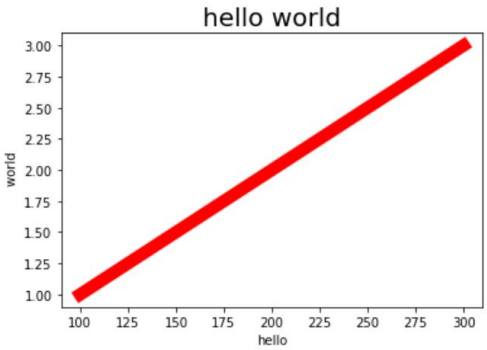
Text(0, 0.5, ' world')



```python
# 이미지 저장하기
x = [100, 200, 300]
y = [1, 2, 3]
plt.plot(x, y, color='#ff0000', linewidth=10)
plt.title('hello world', fontsize=20)
plt.xlabel('hello', fontsize=10)
plt.ylabel(' world', fontsize=10)
plt.savefig('sample.png')
```

Python

Out[-]

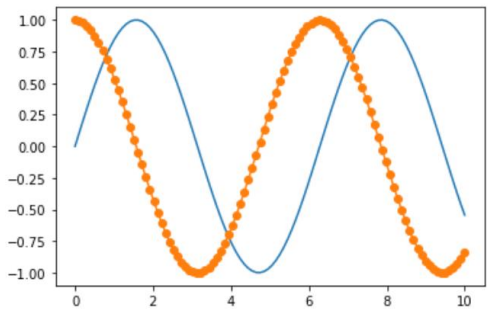



```python
# 두개 그래프 동시에 그리기
x = np.linspace(0, 10, 100)
y = np.sin(x)
y_ = np.cos(x)

plt.plot(x, y)
plt.plot(x, y_, '-o') # -o 모양으로 그리기
```

Python

Out[-]



label

- 무슨 데이터인지 표시해주는 방법

legend

- 좌표축에 범례를 추가해주는 방법
- loc : legend 위치
  - 'best' 0
  - 'upper right' 1
  - 'upper left' 2
  - 'lower left' 3
  - 'lower right' 4
  - 'right' 5
  - 'center left' 6
  - 'center right' 7
  - 'lower center' 8
  - 'upper center' 9
  - 'center' 10

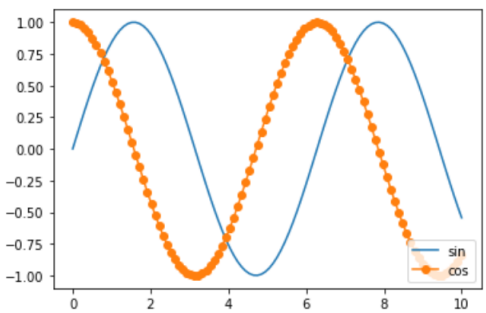
```python
x = np.linspace(0, 10, 100)
y = np.sin(x)
y_ = np.cos(x)

# 범례 추가하기
plt.plot(x, y, label='sin')
plt.plot(x, y_, '-o', label='cos')

# 4번 위치에 범례 넣기
plt.legend(loc=4)
```

Python

Out[-]



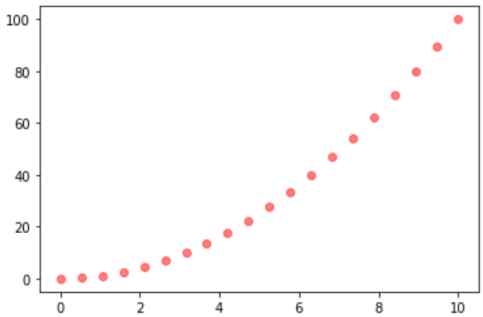
Scatter Plot

- 변수들의 상관관계, 밀집 위치를 나타낸다.

```python
x = np.linspace(0, 10, 20)
y = x ** 2

plt.scatter(x, y, color='r', alpha=0.5)
plt.show()
# 우상향 그래프로 x,y 두 변수는 양의 상관관계임을 나타낸다.
```

Out[-]



histogram

- 도수분포표를 그래프로 나타낸 방법이다.
- 막대그래프는 계급 즉 가로를 생각하지 않고 세로의 높이로만 나타낸다. (출처 : 위키백과)
- 연속형 자료를 계급으로 나누어 계급별 도수를 막대로 나타낸다.

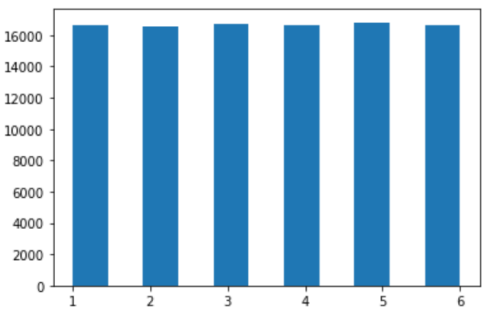
```python
x = [np.random.randint(1, 7) for i in range(100000)]
```

Python

```python
plt.hist(x, bins=11)
plt.show()
```

Python

Out[-]



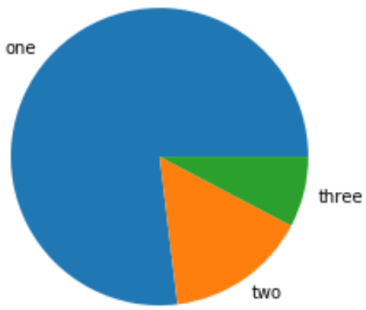
Pie Chart

```python
labels = ['one', 'two', 'three']
size = [100, 20, 10]

plt.pie(size, labels=labels)
plt.show()
```

Python

Out[-]

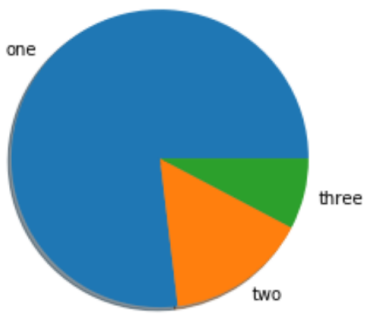



```python
labels = ['one', 'two', 'three']
size = [100, 20, 10]

plt.pie(size, labels=labels, shadow=True)  # 그림자 넣기
plt.show()
```

Python

Out[-]



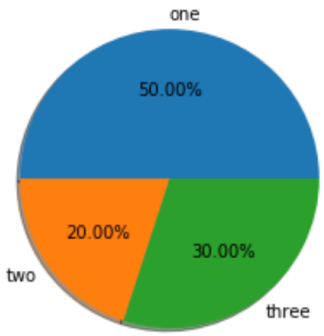
```python
labels = ['one', 'two', 'three']

size = [50, 20, 30]

plt.pie(size, labels = labels, shadow = True, autopct = '%1.2f%%') # 자동 비율 추가하기
plt.show()
```

Python

Out[-]



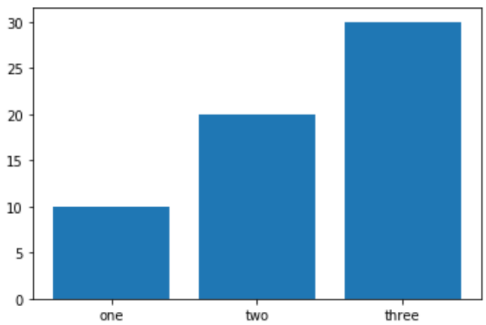
Bar Chart

- 범주형 자료의 분포를 파악한다.

```python
plt.bar(['one', 'two', 'three'], [10, 20, 30])
plt.show()
```

Python

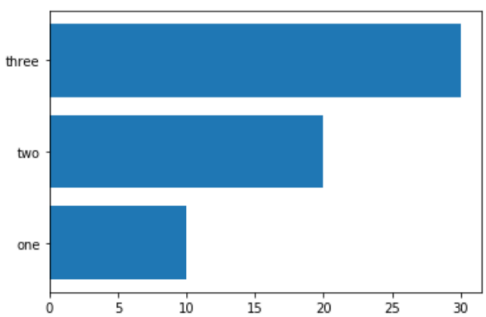
Out[-]




```python
plt.barh(['one', 'two', 'three'], [10, 20, 30]) # 가로방향 그래프
plt.show()
```

Python

Out[-]



선 색상과 스타일

- 공식문서 : https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.plot.html
- 공식문서 : https://matplotlib.org/3.2.1/api/pyplot_summary.html

---

- plt.plot(x, y, color='red') -> red, green, blue 등 색상 이름
- plt.plot(x, y, color='r') -> r, g, b, y, m(자홍), k(검정) 등 색상 이름
- plt.plot(x, y, color='0.2') -> 회색조(0-1사이 값)
- plt.plot(x, y, color='#ff0000') -> 16진수 색상 값

- plt.plot(x, y, linestyle='solid') -> 실선('-')
- plt.plot(x, y, linestyle='dashed') -> 파선('--')
- plt.plot(x, y, linestyle='dashdot') -> 1점 쇄선('-.')
- plt.plot(x, y, linestyle='dotted') -> 점선(':')

- plt.plot(x, y, '--r') -> 빨간색 파선

- '.' point marker
- 'o' circle marker(plt.plot(x, y_, '-o'))
- '^' triangle_up marker
- 's' square marker
- 'p' pentagon marker
- '*' star marker
- '+' plus marker
- 'x' x marker

Plotly

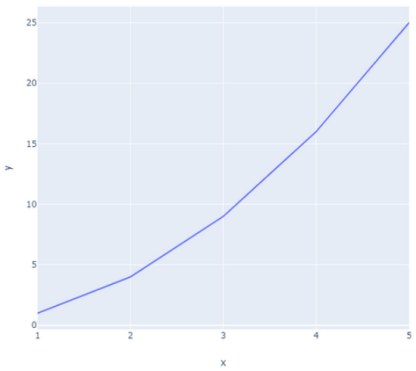
- 인터랙티브한 그래프를 그리기에 적합한 패키지
- 웹 시각화인 자바스크립트의 라이브러리 D3를 이용해 그래프가 웹에서 빠르게 그려진다.
- 공식홈페이지(https://plotly.com/python/) 튜토리얼

Line Plot

```python
import plotly.express as px

x_ = np.array([1, 2, 3, 4, 5])
y_ = x_ ** 2
fig = px.line(x=x_, y=y_)
fig.show()
```

Out[-]

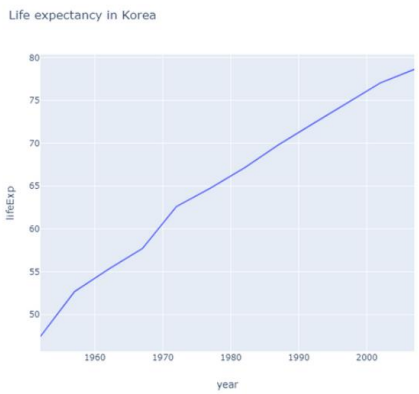


```python
import plotly.express as px
import plotly.graph_objects as go

# gapminder 데이터 셋 사용하기
korea_life = px.data.gapminder().query("country == 'Korea, Rep.'") # 한국인 기대 수명
fig = px.line(korea_life, x="year", y="lifeExp", title='Life expectancy in Korea')
fig.show()
```

Python

Out[-]



```python
korea_life["year"]
korea_life["lifeExp"]
```

Python

```
Out[-]
840    47.453
841    52.681
842    55.292
843    57.716
844    62.612
845    64.766
846    67.123
847    69.810
848    72.244
849    74.647
850    77.045
851    78.623
Name: lifeExp, dtype: float64
```

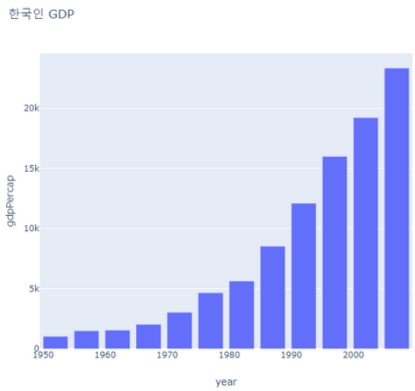
Python

Bar Chart

```python
# 한국 GDP 데이터 셋
korea_gdp = px.data.gapminder().query("country == 'Korea, Rep.'")
fig = px.bar(korea_gdp, x='year', y='gdpPercap', title = "한국인 GDP")
fig.show()
```

Python

Out[-]

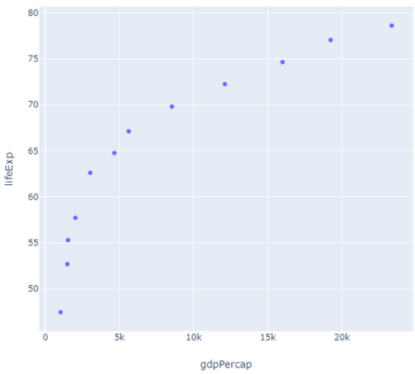


Scatter Plot

```python
# 한국 기대 수명과 GDP의 상관관계
korea_data = px.data.gapminder().query("country == 'Korea, Rep.'")
fig = px.scatter(korea_data, x = 'gdpPercap', y = 'lifeExp')
fig.show()
```

Python

Out[-]

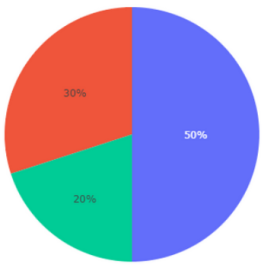


Pie Chart

```python
fig = px.pie(values = [20, 30, 50])
fig.show()
```

Python

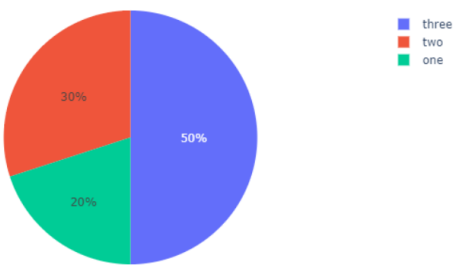
Out[-]




```python
labels = ['one', 'two', 'three']
values = [20, 30, 50]
fig =go.Figure(data = [go.Pie(values = values, labels = labels)]) # 범례 추가하기
fig.show()
```

Python

Out[-]



이미지 분석

- 이미지는 작은 사각형 모양의 픽셀을 모아서 구성되어있다.
- 이미지 크기는 (세로픽셀수X가로픽셀수)로 표현한다.
- 이미지를 저장할 때는 색을 표현하는 스칼라 값이나 2차원 vector로 표현한다.
- RGB는 색공간을 말한다. Red, Green, Blue로 세개가 합쳐진 벡터이다.

- skimage(scikit-image) : 이미지 처리를 위한 파이썬 라이브러리이다. numpy array로 동작한다.

- PIL(Python Image Library)
  - Pillow : 이미지 처리와 그래픽 기능을 제공하는 파이썬 라이브러리이다.

    (더이상 PIL은 지원하지 않고 그 후속 프로젝트인 Pillow를 사용한다)

- cv2 : OpenCV는 Open Source Computer Vision Library의 약자로 오픈소스 컴퓨터 비전 및 머신러닝 라이브러리이다. 이 라이브러리를 불러올 때는 cv2를 사용한다.

```python
import numpy as np
from skimage import io
# from PIL import Image
# from cv2
import matplotlib.pyplot as plt
```

Python

```python
jeju = io.imread('jeju.jpg') # 이미지 불러오기
```

Python

```python
type(jeju) # 이미지 타입 확인하기
```

Python

```
Out[-]
imageio.core.util.Array
```

Python

```python
jeju.shape
```

Python

```
Out[-]
(960, 1280, 3)  # 세로, 가로, RGB(색)
```

Python

```python
jeju
```

Python

```
Out[-]
Array([[[171, 222, 251],
        [172, 223, 252],
        [172, 223, 252],
        ...,
        [124, 189, 255],
        [121, 189, 254],
        [120, 188, 253]],

       [[173, 224, 253],
        [173, 224, 253],
        [173, 224, 253],
        ...,
        [124, 189, 255],
        [122, 190, 255],
        [121, 189, 254]],

       ...,

       [[ 44,  71,   4],
        [ 23,  50,   0],
        [ 29,  52,   0],
        ...,
        [ 40,  67,  22],
        [  0,  19,   0],
        [ 16,  41,   0]],

       [[ 29,  58,   0],
        [ 44,  71,   2],
        [ 84, 110,  37],
        ...,
        [ 17,  44,   1],
        [ 33,  60,  17],
        [ 18,  43,   1]]], dtype=uint8)
```

Python

```python
np.min(jeju), np.max(jeju)
```

Python

```
Out[-]
(0, 255)
```

Python

```python
plt.imshow(jeju)
```

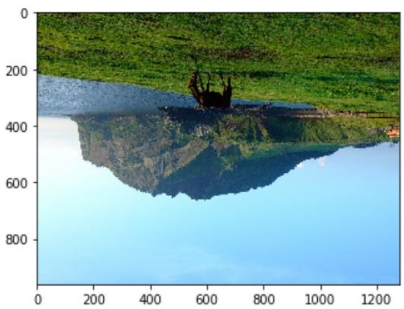
Python

Out[-]


```python
plt.imshow(jeju[::-1]) # 상하로 뒤집기
```

Python

Out[-]



```python
plt.imshow(jeju[:, ::-1]) # 좌우로 뒤집기
```

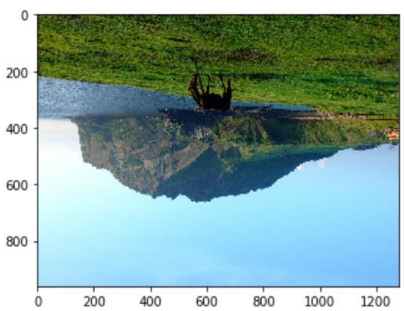
Python

Out[-]


```python
plt.imshow(jeju[::-1, :])
```

Python

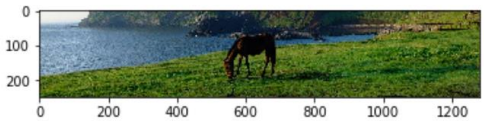
Out[-]



```python
plt.imshow(jeju[550:800, :]) # 부분만 잘라내기
```

Python

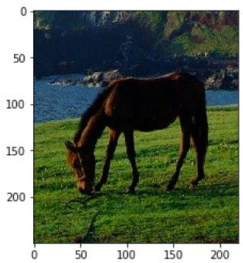
Out[-]




```python
plt.imshow(jeju[550:800, 500:720])
```

Python

Out[-]




```python
plt.imshow(jeju[::3, ::3]) # 3칸씩 건너 뛰기
```

Python

Out[-]

```python
plt.imshow(jeju[::6, ::6]) # 6칸씩 건너 뛰기 : 화질이 깨진 것을 볼 수 있다
```

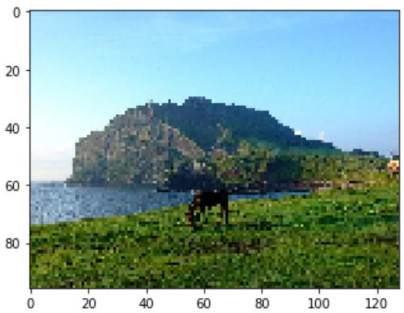
Python

Out[-]


```python
plt.imshow(jeju[::10, ::10]) # 10칸씩 건너 뛰기 : 화질이 많이 깨짐
```

Python

Out[-]



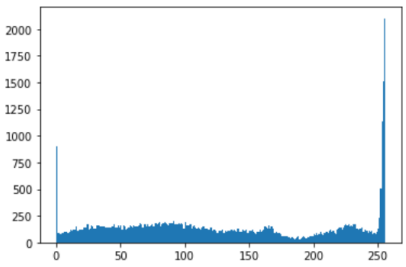
```python
miniJeju = jeju[::10, ::10]
```

Python

```python
# 색 정보를 일렬로 나열하고 어디에 많이 분포 되었는 지를 확인한다.
plt.hist(miniJeju.ravel(), 256, [0, 256])
plt.show()
```

Python

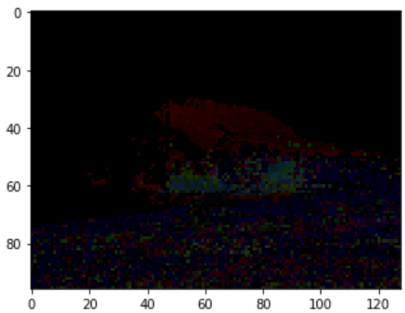
Out[-]




```python
# 특정한 값(50)을 지정하고 그 이하는 원래의 색으로 나오고 다른 값은 검정색(0)으로 지정하여
# 이미지를 출력한다.
miniJeju_ = np.where(miniJeju < 50, miniJeju, 0)
plt.imshow(miniJeju_)
```

Python

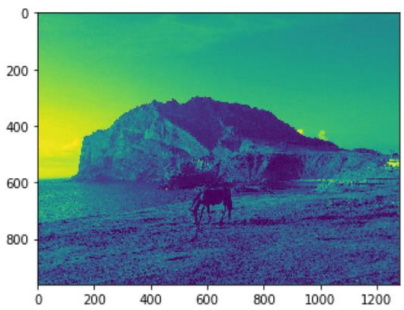
Out[-]



```python
plt.imshow(jeju[:, :, 0]) # 0번째 색만 출력하기
```

Python

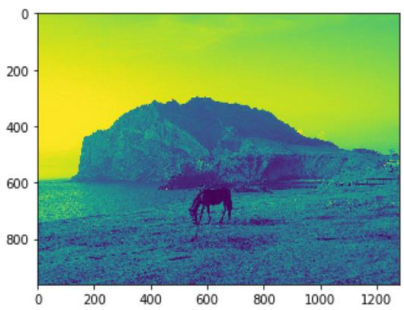
Out[-]




```python
plt.imshow(jeju[:, :, 1]) # 1번째 색만 출력하기
```

Python

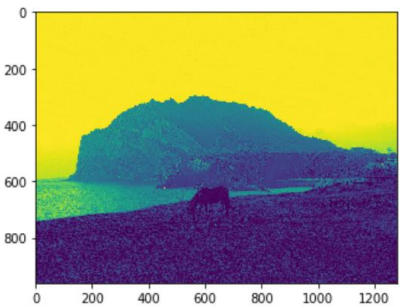
Out[-]



```python
plt.imshow(jeju[:, :, 2]) # 2번째 색만 출력하기
```

Python

Out[-]




```python
jeju[:, :, 0] # 0번째 색 정보 출력
```

Python

```
Out[-]
Array([[171, 172, 172, ..., 124, 121, 120],
       [173, 173, 173, ..., 124, 122, 121],
       [174, 174, 175, ..., 125, 122, 122],
       ...,
       [ 66,  89,  49, ...,   2,  34,  40],
       [ 44,  23,  29, ...,  40,   0,  16],
       [ 29,  44,  84, ...,  17,  33,  18]], dtype=uint8)
```

Python

```python
jeju[:, :]
```

Python

```
Out[-]
Array([[[171, 222, 251],
        [172, 223, 252],
        [172, 223, 252],
        ...,
        [124, 189, 255],
        [121, 189, 254],
        [120, 188, 253]],

       [[173, 224, 253],
        [173, 224, 253],
        [173, 224, 253],
        ...,
        [124, 189, 255],
        [122, 190, 255],
        [121, 189, 254]],

       ...,

       [[ 44,  71,   4],
        [ 23,  50,   0],
        [ 29,  52,   0],
        ...,
        [ 40,  67,  22],
        [  0,  19,   0],
        [ 16,  41,   0]],

       [[ 29,  58,   0],
        [ 44,  71,   2],
        [ 84, 110,  37],
        ...,
        [ 17,  44,   1],
        [ 33,  60,  17],
        [ 18,  43,   1]]], dtype=uint8)
```

Python

```python
# 회색으로 이미지 출력하기
from skimage import color

plt.imshow(color.rgb2gray(jeju), cmap=plt.cm.gray)
```

Python

Out[-]

4. Crawling

이 장에서 다루는 내용

Crawling

URL

HTTP

requests

BeautifulSoup

연습문제

Crawling

웹 페이지에 접속해서 정보를 찾는 과정을 프로그램을 통해 찾아 수집하고 원하는 형태에 맞게 가공하는 모든 과정.

- 사이트의 운영자의 의사에 반하지 않으면 합법이고 그렇지 않으면 불법
- 사이트 디렉토리의 robots.txt파일을 보면 크롤링을 금지하는지 안하는지표시되어있음 (Disallow라는 표시 있으면 크롤링하면 안 됨)
- 웹페이지 소스 중 웹 프로그래밍 요소는 저작물로 인정될 수 있으므로 불법 복제는 저작권 침해에 해당.

필요패키지

- (필수) pip3 install BeautifulSoup4 or pip3 install bs4
- (필수) pip3 install requests
- (필수) pip3 install pandas
- (필수) pip3 install plotly
- (선택) pip3 install lxml

라이브러리 설치 (대부분 설치되어있다는 가정 하)

- !pip3 install requests
- !pip3 install beautifulsoup4

```
!pip3 install requests
```

Python

```
!pip3 install beautifulsoup4
```

Python

```
# mac , Linux
!ls
# window
!dir
```

Python

```
Out[-]
C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: CC5E-6766

 C:\Users\leehojun\Google 드라이브\11_1. 콘텐츠 동영상 결과물\007. 크롤링 강의 디렉터리

2020-04-03  13:16    <DIR>          .
2020-04-03  13:16    <DIR>          ..
2020-04-03  13:10    <DIR>          .ipynb_checkpoints
2020-04-03  12:18           391,939 001.ipynb
2020-04-03  01:58    <DIR>          참고자료
2020-04-03  13:16               999 최종강의자료_크롤링.ipynb
               2개 파일             392,938 바이트
               4개 디렉터리  13,425,782,784 바이트 남음
```

Python

URL(Uniform Resource Locator)

- 자원이 어디 있는지를 알려주기 위한 규약
- 흔히 웹 사이트 주소로 알고 있지만, URL은 웹 사이트 주소뿐만 아니라 컴퓨터 네트워크상의 자원을 모두 나타낼 수 있음
- 그 주소에 접속하려면 해당 URL에 맞는 프로토콜을 알아야 하고, 그와 동일한 프로토콜로 접속 (FTP 프로토콜인 경우에는 FTP 클라이언트를 이용해야 하고, HTTP인 경우에는 웹 브라우저를 이용해야 한다. 텔넷의 경우에는 텔넷 프로그램을 이용해서 접속)

출처 : Wiki

HTTP(Hypertext Transger Protocol)

- HTML, XML, Javascript, 오디오, 비디오, 이미지, PDF, Etc
- 요청 또는 상태 라인 / 해더(생략가능) / 빈줄(해더의 끝) / 바디(생략가능)

HTTP request ex 1

```
GET /stock.html HTTP/1.1
Host www.paullab.co.kr
```

Plain Text

HTTP request ex 2

```
GET /index.html HTTP/1.1
user-agent: MSIE 6.0; Windows NT 5.0
accept: text/html; */*
cookie: name = value
referer: http://www.naver.com
host: www.paullab.co.kr
```

Plain Text

1. 데이터 처리 방식, 기본 페이지, 프로토콜 버전.
2. User-Agent: 사용자 웹 브라우저 종류 및 버전 정보.
3. Accept: 웹 서버로부터 수신되는 데이터 중 웹 브라우저가 처리할 수 있는 데이터 타입을 의미.

   여기서 text/html은 text, html 형태의 문서를 처리할 수 있고, /는 모든 문서를 처리할 수 있다는 의미. (이를 MIME 타입이라 부르기도 한다.)
4. Cookie: HTTP 프로토콜 자체가 세션을 유지하지 않는 State-less(접속상태를 유지하지 않는) 방식이기 때문에 로그인 인증을 위한 사용자 정보를 기억하려고 만든 인위적인 값. 즉 사용자가 정상적인 로그인 인증 정보를 가지고 있다는 것을 판단하고자 사용.
5. Referer: 현재 페이지 접속 전에 어느 사이트를 경유했는지 알려주는 도메인 혹은 URL 정보.
6. Host: 사용자가 요청한 도메인정보.

HTTP Response ex 1

```
HTTP/1.1 200 OK                                          ## 상태라인
Content-Type: application/xhtml+xml; charset=utf-8 ## 해더
                                                         ## 빈줄
<html>                                                   ## 바디
...
</html>
```

Plain Text

HTTP Response ex 2

```
HTTP/1.1 OK 200
Server: NCSA/1.4.2
Content-type: text/html
Content-length: 107

<html>
...
</html>
```

Plain Text

1. 웹 프로토콜 버전 및 응답 코드 정보가 포함.
2. 웹 애플리케이션 종류 및 버전 정보가 포함.
3. MIME 타입 정보가 포함.
4. 수신 Body 사이즈 정보가 포함.
5. 사용자가 요청한 웹 페이지 정보가 포함.

HTTP 처리방식

- GET : 리소스 취득 (? 뒤에 이어붙이는 방식 - 작은 값들)
- POST : 리소스 생성 (Body에 붙이는 방식 - 상대적으로 큰 용량)
- PUT : 리소스 수정 요청
- DELETE : 리소스 삭제 요청
- HEAD : HTTP 헤더 정보만 요청, 해당 자원 존재 여부 확인 목적
- OPTIONS : 웹서버가 지원하는 메소드 종류 반환 요청
- TRACE : 요청 리소스가 수신되는 경로 확인
- CONNECT : 요청 리소스에 대해 양방향 연결 시작

상태 코드

- 200 : 서버가 요청을 제대로 처리.
- 201 : 성공적으로 요청되었으며 서버가 새 리소스를 작성.
- 202 : 서버가 요청을 접수했지만 아직 처리하지 않음.
- 301 : 요청한 페이지를 새 위치로 영구적으로 이동.
- 403 : 서버가 요청을 거부.
- 404 : 서버가 요청한 페이지를 찾을 수 없음.
- 500 : 서버에 오류가 발생하여 요청을 수행할 수 없음.
- 503 : 서버가 오버로드되었거나 유지관리를 위해 다운되었기 때문에 현재서버 사용 불가.

출처 : WIKI

```python
import requests
import bs4
```

Python

```python
requests.__version__ # requests 버전 확인
```

Python

```
Out[-]
'2.22.0'
```

Python

```python
bs4.__version__ # bs4 버전 확인
```

Python

```
Out[-]
'4.7.1'
```

Python

```python
from datetime import datetime

datetime.now() # 현재 시간 출력
```

Python

```
Out[-]
datetime.datetime(2020, 4, 3, 13, 36, 59, 789815)
```

Python

requets

- HTTP 요청을 보내는데 사용하는 라이브러리
- .text : str타입의 데이터를 return
- .headers : header(key/value 형식으로 데이터 저장)의 내용 확인
- .encoding : 인코딩 방식 확인
- .status_code : HTTP 요청에 대해서 요청이 성공했는지 실패했는지 혹은 어떤 상태인지 말해줌
- .ok : 데이터를 잘 불러오고 있는지 확인

```python
import requests

html = requests.get('http://www.paullab.co.kr/stock.html')
html
```

Python

```
Out[-]
<Response [200]> # 서버가 요청을 제대로 처리
```

Python

```
html.text # 한글 깨지는 현상이 발생
```

Python

```
Out[-]
'<!DOCTYPE html>\n<html lang="en">\n\n<head>\n
<meta charset="UTF-8">\n
<meta name="viewport" content="width=device-width, initial-scale=1.0">\n
<meta http-equiv="X-UA-Compatible" content="ie=edge">\n
<title>Document</title>\n  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">\n
<style>\n    h1{\n      margin: 2rem;\n    }\n
h1>span{\n      font-size: 1rem;\n    }\n
.main {\n      width: 70%;\n      margin: 3rem auto auto auto;\n
text-align: center\n    }\n\n    table {\n      width: 100%;\n    }\n
</style>\n</head>\n\n<body>\n
<h1>í\x81¬ë¡¤ë§\x81 ì\x97°ì\x8aµì\x9a© í\x8e\x98ì\x9d´ì§\x80

                ...

<td class="num"><span>139,085</span></td>\n
</tr>\n
</tbody>\n
</table>\n  </div>\n</body>\n\n</html>\n'
```

Python

```
html.headers
```

Python

```
Out[-]
{'Server': 'nginx', 'Date': 'Sat, 04 Apr 2020 11:13:06 GMT',
'Content-Type': 'text/html', 'Transfer-Encoding': 'chunked',
'Connection': 'keep-alive', 'Vary': 'Accept-Encoding',
'P3P': "CP='NOI CURa ADMa DEVa TAIa OUR DELa BUS IND PHY ONL UNI COM NAV INT DEM PRE'",
'X-Powered-By': 'PHP/5.5.17p1', 'Content-Encoding': 'gzip'}
```

Python

```python
dir(html)
```

Python

```
Out[-]
['__attrs__',
 '__bool__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',

  ...

 'ok',
 'raise_for_status',
 'raw',
 'reason',
 'request',
 'status_code',
 'text',
 'url']
```

Python

```python
#ASCII 기반의 확장 인코딩 방식
html.encoding
```

Python

```
Out[-]
'ISO-8859-1'
```

Python

```python
html.encoding = 'utf-8' # 한글 출력
```

Python

```python
html.text
```

Python

```
Out[-]
'<!DOCTYPE html>\n<html lang="en">\n\n<head>\n
<meta charset="UTF-8">\n  <meta name="viewport"
content="width=device-width, initial-scale=1.0">\n
<meta http-equiv="X-UA-Compatible" content="ie=edge">\n
<title>Document</title>\n  <link rel="stylesheet" href=
"https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
\n  <style>\n    h1{\n      margin: 2rem;\n    }\n
h1>span{\n      font-size: 1rem;\n    }\n    .main {\n      width: 70%;\n
margin: 3rem auto auto auto;\n      text-align: center\n    }\n\n
table {\n      width: 100%;\n    }\n  </style>\n</head>\n\n<body>\n
<h1>크롤링 연습용 페이지

         ...

<td class="num"><span>139,085</span></td>\n
</tr>\n
</tbody>\n
</table>\n  </div>\n</body>\n\n</html>\n'
```

Python

```python
html.status_code
```

Python

```
Out[-]
200 # 200 : 성공했다는 의미
```

Python

```python
html.ok
```

Python

```
Out[-]
True
```

Python

GET 방식으로 parameter 전달하는 방법

```
<html>
<head>
</head>
<body>
  <form action="test.html" method="GET">
    <input type="text" name="user_id">
    <input type="password" name="user_pw">
    <input type="submit" name="submit">
  </form>
</body>
</html>
```

Plain Text

```
params = {'pa1': 'val1', 'pa2': 'value2'}
response = requests.get('http://www.paullab.co.kr', params=params)
```

Python

```
response.url
```

Python

```
Out[-]
'http://www.paullab.co.kr/?pa1=val1&pa2=value2'
```

Python

POST 요청할 때 data 전달법

```python
import requests, json

data = {'pa1': 'val1', 'pa2': 'value2'}
response = requests.post('http://www.paullab.co.kr', data=json.dumps(data))
```

Python

헤더 추가, 쿠키 추가

```python
headers = {'Content-Type': 'application/json; charset=utf-8'}
cookies = {'session_id': 'sorryidontcare'}
response = requests.get('http://www.paullab.co.kr', headers=headers, cookies=cookies)
```

Python

인증추가

```python
response = requests.post('http://www.paullab.co.kr', auth=("id","pass"))
```

Python

requests를 이용한 크롤링

```python
import requests
from bs4 import BeautifulSoup

response = requests.get('http://www.paullab.co.kr/stock.html')
response.encoding = 'utf-8'
html = response.text

soup = BeautifulSoup(html, 'html.parser') # 원하는 문자열로 잘라줌
```

Python

```python
print(soup.prettify()) # html 문서형식으로 출력
```

Python

```
Out[-]
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8"/>
  <meta content="width=device-width, initial-scale=1.0" name="viewport"/>
  <meta content="ie=edge" http-equiv="X-UA-Compatible"/>
  <title>
   Document
  </title>

      ...

<td class="num">
       <span>
        139,085
       </span>
      </td>
     </tr>
    </tbody>
   </table>
  </div>
 </body>
</html>
```

Python

특정 페이지의 소스코드를 파일로 저장

```python
import requests
from bs4 import BeautifulSoup

response = requests.get('http://www.paullab.co.kr/stock.html')
response.encoding = 'utf-8'
html = response.text
# url 코드를 파일로 저장
f = open('test.html', 'w', encoding='utf-8')
f.write(html)
f.close()
```

Python

```python
!dir
```

Python

```
Out[-]
C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: CC5E-6766

 C:\Users\leehojun\Google 드라이브\11_1. 콘텐츠 동영상 결과물\007. 크롤링 강의 디렉터리

2020-04-03  13:55    <DIR>          .
2020-04-03  13:55    <DIR>          ..
2020-04-03  13:10    <DIR>          .ipynb_checkpoints
2020-04-03  13:50           392,338 001.ipynb
2020-04-03  13:55            48,527 test.html # test.html이 생성되는것을 확인
2020-04-03  01:58    <DIR>          참고자료
2020-04-03  13:54           221,527 최종강의자료_크롤링.ipynb
               3개 파일             662,392 바이트
               4개 디렉터리  13,301,800,960 바이트 남음
```

Python

```python
# url 파일에서 특정단어 찾기
s = html.split(' ') # 띄어쓰기 단위로 분할 # 앞뒤로 띄어쓰기 안되어 있으면 검색이 안됨
word = input('페이지에서 검색할 단어를 입력하세요 : ')
s.count(word)
```

Python

```
Out[-]
페이지에서 검색할 단어를 입력하세요 : 제주
0
```

Python

BeautifulSoup

- str타입의 html 데이터를 html 구조를 가진 데이터로 가공해주는 라이브러리
- BeautifulSoup(markup, "html.parser")
- BeautifulSoup(markup, "lxml")
- BeautifulSoup(markup, "lxml-xml") BeautifulSoup(markup, "xml")
- BeautifulSoup(markup, "html5lib")

```python
import requests
from bs4 import BeautifulSoup

response = requests.get('http://www.paullab.co.kr/stock.html')

response.encoding = 'utf-8'
html = response.text

soup = BeautifulSoup(html, 'html.parser')
```

Python

```python
soup.title # title 태그 출력
```

Python

```
Out[-]
<title>Document</title>
```

Python

```python
soup.title.string # title태그에서 문자열만 출력
```

Python

```
Out[-]
'Document'
```

Python

```python
soup.title.text # String 같은 기능
```

Python

```
Out[-]
'Document'
```

Python

```python
soup.title.parent.name # 부모 태그
```

Python

```
Out[-]
'head'
```

Python

```python
soup.tr # table row
```

Python

```
Out[-]
<tr>
<th scope="col">날짜</th>
<th scope="col">종가</th>
<th scope="col">전일비</th>
<th scope="col">시가</th>
<th scope="col">고가</th>
<th scope="col">저가</th>
<th scope="col">거래량</th>
</tr>
```

Python

```python
soup.td # table data
```

Python

```
Out[-]
<td align="center "><span class="date">2019.10.23</span></td>
```

Python

```python
soup.th # table header cell
```

Python

```
Out[-]
<th scope="col">날짜</th>
```

Python

```
soup.table
```

Python

```
Out[-]
<table class="table table-hover">
<tbody>
<tr>
<th scope="col">날짜</th>
<th scope="col">종가</th>
<th scope="col">전일비</th>
<th scope="col">시가</th>
<th scope="col">고가</th>
<th scope="col">저가</th>
<th scope="col">거래량</th>
</tr>
<tr>
<td align="center "><span class="date">2019.10.23</span></td>
<td class="num"><span>6,650</span></td>

                ...

</td>
<td class="num"><span>5,300</span></td>
<td class="num"><span>5,370</span></td>
<td class="num"><span>5,280</span></td>
<td class="num"><span>211,019</span></td>
</tr>
</tbody>
</table>
```

Python

```python
soup.find('title') # find() : 조건에 맞는 하나의 태그를 출력
```

Python

```
Out[-]
<title>Document</title>
```

Python

```python
soup.find('tr')
```

Python

```
Out[-]
<tr>
<th scope="col">날짜</th>
<th scope="col">종가</th>
<th scope="col">전일비</th>
<th scope="col">시가</th>
<th scope="col">고가</th>
<th scope="col">저가</th>
<th scope="col">거래량</th>
</tr>
```

Python

```python
soup.find('th')
```

Python

```
Out[-]
<th scope="col">날짜</th>
```

Python

```python
soup.find(id=('update')).text # 특정 id의 text 출력
```

Python

```
Out[-]
'update : 20.12.30'
```

Python

```python
soup.find('head').find('title') # head 안에 title 출력
```

Python

```
Out[-]
<title>Document</title>
```

Python

```python
soup.find('h2', id='제주코딩베이스캠프연구원')
# h2의 'id가 제주코딩베이스캠프연구원'를 출력
```

Python

```
Out[-]
<h2 id="제주코딩베이스캠프연구원">제주코딩베이스캠프 연구원</h2>
```

Python

```python
soup.find_all('h2') # find_all() : 조건에 맞는 모든 태그들을 출력
```

Python

```
Out[-]
[<h2 id="제주코딩베이스캠프연구원">제주코딩베이스캠프 연구원</h2>,
 <h2 id="제주코딩베이스캠프공업">제주코딩베이스캠프 공업</h2>,
 <h2 id="제주코딩베이스캠프출판사">제주코딩베이스캠프 출판사</h2>,
 <h2 id="제주코딩베이스캠프학원">제주코딩베이스캠프 학원</h2>]
```

Python

```python
soup.find_all('h2')[0]
```

Python

```
Out[-]
<h2 id="제주코딩베이스캠프연구원">제주코딩베이스캠프 연구원</h2>
```

Python

```python
soup.find_all('table', class_='table') # class_ : 예약어
# 예약어 : 특정한 기능을 수행하도록 미리 예약되어 있는것
```

Python

```
Out[-]
[<table class="table table-hover">
 <tbody>
 <tr>
 <th scope="col">날짜</th>
 <th scope="col">종가</th>
 <th scope="col">전일비</th>
 <th scope="col">시가</th>
 <th scope="col">고가</th>
 <th scope="col">저가</th>
 <th scope="col">거래량</th>
 </tr>
 <tr>
 <td align="center "><span class="date">2019.10.23</span></td>
 <td class="num"><span>6,650</span></td>

             ...

</td>
 <td class="num"><span>2,020</span></td>
 <td class="num"><span>2,090</span></td>
 <td class="num"><span>2,020</span></td>
 <td class="num"><span>139,085</span></td>
 </tr>
 </tbody>
 </table>]
```

Python

```
soup = BeautifulSoup('''
<hojun id='jeju' class='codingBaseCamp codingLevelUp'>
   hello world
</hojun>
''')
# tag = hojun , id = 'jeju' , class = 'codingBaseCamp codingLevelUp'
tag = soup.hojun
tag
```

Python

```
Out[-]
<hojun class="codingBaseCamp codingLevelUp" id="jeju">
   hello world
</hojun>
```

Python

```
type(tag)
```

Python

```
Out[-]
bs4.element.Tag
```

Python

```python
dir(tag) # tag의 method
```

Python

```
Out[-]
['HTML_FORMATTERS',
 'XML_FORMATTERS',
 '__bool__',
 '__call__',
 '__class__',
 '__contains__',
 '__copy__',
 '__delattr__',

   ...

 'setup',
 'string',
 'strings',
 'stripped_strings',
 'text',
 'unwrap',
 'wrap']
```

Python

```python
tag.name
```

Python

```
Out[-]
'hojun'
```

Python

```python
tag['class']
```

Python

```
Out[-]
['codingBaseCamp', 'codingLevelUp']
```

Python

```python
tag['id']
```

Python

```
Out[-]
'jeju'
```

Python

```python
tag.attrs # 정보를 한번에 보고싶을때 사용
```

Python

```
Out[-]
{'id': 'jeju', 'class': ['codingBaseCamp', 'codingLevelUp']}
```

Python

```python
tag.string # 문자열 출력
```

Python

```
Out[-]
'\n   hello world\n'
```

Python

```python
tag.text # 문자열 출력
```

Python

```
Out[-]
'\n   hello world\n'
```

Python

```python
tag.contents # list로 출력
```

Python

```
Out[-]
['\n   hello world\n']
```

Python

```python
for i in tag.children: # children : 종속 태그
    print(i)
```

Python

```
Out[-]
hello world
```

Python

```python
tag.children
```

Python

```
Out[-]
<list_iterator at 0x1cd52f37550>
```

Python

```
soup = BeautifulSoup('''
<ul>
    <li id='jeju' class='codingBaseCamp codingLevelUp'>hello world</li>
    <li id='jeju' class='codingBaseCamp codingLevelUp'>hello world</li>
    <li id='jeju' class='codingBaseCamp codingLevelUp'>hello world</li>
</ul>
''')
tag = soup.ul
tag
```

Python

```
Out[-]
<ul>
<li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>
<li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>
<li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>
</ul>
```

Python

```
tag.contents # list
```

Python

```
Out[-]
['\n',
 <li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>,
 '\n',
 <li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>,
 '\n',
 <li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>,
 '\n']
```

Python

```python
tag.li # li 태그 출력
```

Python

```
Out[-]
<li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>
```

Python

```python
tag.li.parent # li 태그의 부모 태그 출력
```

Python

```
Out[-]
<ul>
<li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>
<li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>
<li class="codingBaseCamp codingLevelUp" id="jeju">hello world</li>
</ul>
```

Python

Selector

- 태그에 좀 더 세밀한 접근이 가능
- class를 지칭할 때는 '.'을 사용하고, id를 지칭할 때는 '#'를 사용
- 탐색하고자 하는 태그가 특정태그 하위에 있을 때 '>'를 사용

```python
import requests
from bs4 import BeautifulSoup

response = requests.get('http://www.paullab.co.kr/stock.html')

response.encoding = 'utf-8'
html = response.text

soup = BeautifulSoup(html, 'html.parser')
```

Python

```python
soup.select('#update')
```

Python

```
Out[-]
[<span id="update">update : 20.12.30</span>]
```

Python

```
soup.select('.table > tr') # 'table' class 안에 모든 tr 태그 출력
# 순서 : table > tbody > tr (바로 아래 아니면 실행안됨)
```

Python

```
Out[-]
[]
```

Python

```
soup.select('.table > tbody > tr')[2] # 'table' class 안에 tbody 안에 모든 tr 태그 출력
```

Python

```
Out[-]
<tr>
<td align="center"><span class="date">2019.10.22</span></td>
<td class="num"><span>6,630</span></td>
<td class="num">
<img alt="하락" height="6" src="ico_down.gif" style="margin-right:4px;" width="7"/>
<span class="tah p11 nv01">
                            190
                        </span>
</td>
<td class="num"><span>6,830</span></td>
<td class="num"><span>6,930</span></td>
<td class="num"><span>6,530</span></td>
<td class="num"><span>919,571</span></td>
</tr>
```

Python

```python
# 요소 선택 방법
soup.select("p > a:nth-of-type(2)") # p > a tag 인데 2번째 요소
soup.select("p > a:nth-child(even)") # p > a tag 인데 짝수나 홀수번째 요소
soup.select('a[href]') # 특정 attribute 요소
soup.select("#link1 + .sister") # id와 class를 동시에 가진 요소
```

Python

```python
oneStep = soup.select('.main')[0] # 제주코딩베이스캠프 연구원
oneStep
```

Python

```
Out[-]
<div class="main">
<h2 id="제주코딩베이스캠프연구원">제주코딩베이스캠프 연구원</h2>
<h3><span style="color: salmon">일별</span> 시세</h3>
<table class="table table-hover">
<tbody>
<tr>
<th scope="col">날짜</th>
<th scope="col">종가</th>
<th scope="col">전일비</th>
<th scope="col">시가</th>
<th scope="col">고가</th>
<th scope="col">저가</th>
<th scope="col">거래량</th>
</tr>

        ...

<td class="num"><span>5,300</span></td>
<td class="num"><span>5,370</span></td>
<td class="num"><span>5,280</span></td>
<td class="num"><span>211,019</span></td>
</tr>
</tbody>
</table>
</div>
```

Python

```python
twoStep = oneStep.select('tbody > tr')[1:]
twoStep
```

Python

```
Out[-]
<tr>
 <td align="center "><span class="date">2019.10.23</span></td>
 <td class="num"><span>6,650</span></td>
 <td class="num">
 <img alt="상승 " height="6 " src="ico_up.gif " style="margin-right:4px; " width="7 "/>
 <span>
                            20
                        </span>

                            ...

                            10
                        </span>
 </td>
 <td class="num"><span>5,300</span></td>
 <td class="num"><span>5,370</span></td>
 <td class="num"><span>5,280</span></td>
 <td class="num"><span>211,019</span></td>
 </tr>]
```

Python

```python
twoStep[0].select('td')[0].text # 날짜
```

Python

```
Out[-]
'2019.10.23'
```

Python

```python
twoStep[0].select('td')[1].text  # 종가
```

Python

```
Out[-]
'6650'   # 문자형이기때문에 만약 계산에 이용하게 된다면 숫자형으로 바꿔줘야함.
# twoStep[0].select('td')[1].text.replace(',', '')
```

Python

```python
날짜 = []
종가 = []

for i in twoStep:
    날짜.append(i.select('td')[0].text)
    종가.append(int(i.select('td')[1].text.replace(',', '')))
```

Python

```
날짜
```

Python

```
Out[-]
['2019.10.23',
 '2019.10.22',
 '2019.10.21',
 '2019.10.18',
 '2019.10.17',
 '2019.10.16',
 '2019.10.15',
 '2019.10.14',
 '2019.10.11',
 '2019.10.10',
 '2019.10.08',
 '2019.10.07',
 '2019.10.04',
 '2019.10.02',
 '2019.10.01',
 '2019.09.30',
 '2019.09.27',
 '2019.09.26',
 '2019.09.25',
 '2019.09.24']
```

Python

```
종가
```

Python

```
Out[-]
[6650,
 6630,
 6820,
 6430,
 5950,
 5930,
 5640,
 5380,
 5040,
 5100,
 5050,
 4940,
 5010,
 4920,
 5010,
 5000,
 5010,
 5060,
 5060,
 5330]
```

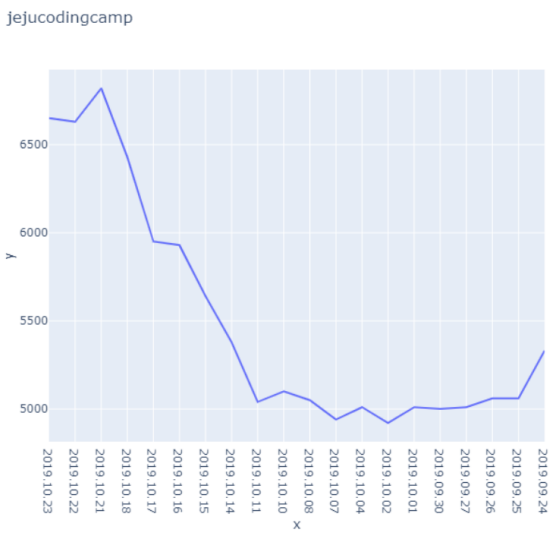
Python

```python
# 시각화
# 날짜별로 가격 변동 추이
import plotly.express as px

fig = px.line(x=날짜, y=종가, title='jejucodingcamp')
fig.show()
```

Python

Out[-]



연습문제

라이브러리 설치 (대부분 설치되어있다는 가정 하)

- !pip3 install requests
- !pip3 install beautifulsoup4
- 크롤링 URL : http://www.paullab.co.kr/stock.html

문제 1번

각 회사별 1만주씩 있다고 가정했을 때, 전그룹사 시가총액을 구해주세요.

- 그룹사 : [제주코딩베이스캠프 연구원, 제주코딩베이스캠프 공업, 제주코딩베이스캠프 출판사, 제주코딩베이스캠프 학원]

```python
import requests
from bs4 import BeautifulSoup

response = requests.get("http://www.paullab.co.kr/stock.html")

response.encoding = 'utf-8'
html = response.text

soup = BeautifulSoup(html, 'html.parser')
```

Python

```python
soup.select('.main')[0] # 제주코딩베이스캠프 연구원
soup.select('.main')[1] # 제주코딩베이스캠프 공업
soup.select('.main')[2] # 제주코딩베이스캠프 출판사
soup.select('.main')[3] # 제주코딩베이스캠프 학원
```

Python

```python
soup.select('.main')[0] # 제주코딩베이스캠프 연구원
soup.select('.main')[1] # 제주코딩베이스캠프 공업
soup.select('.main')[2] # 제주코딩베이스캠프 출판사
soup.select('.main')[3] # 제주코딩베이스캠프 학원
```

Python

```
Out[-]
<div class="main">
<h2 id="제주코딩베이스캠프학원">제주코딩베이스캠프 학원</h2>
<h3><span style="color: salmon">일별</span> 시세</h3>
<table class="table table-hover">
<tbody>
<tr>
<th scope="col">날짜</th>
<th scope="col">종가</th>
<th scope="col">전일비</th>
<th scope="col">시가</th>
<th scope="col">고가</th>
<th scope="col">저가</th>
<th scope="col">거래량</th>
</tr>

    ...

<td class="num"><span>2,020</span></td>
<td class="num"><span>2,090</span></td>
<td class="num"><span>2,020</span></td>
<td class="num"><span>139,085</span></td>
</tr>
</tbody>
</table>
</div>
```

Python

```
그룹사별일일시가 = soup.select('.main')
오늘종가 = []
오늘시가총액 = []

for i in 그룹사별일일시가:
    print(i.select('.table > tbody > tr')[1].select('td')[1])
    print(i.select('.table > tbody > tr')[1].select('td')[1].text)
    print(i.select('.table > tbody > tr')[1].select('td')[1].text.replace(',', ''))
```

Python

```
Out[-]
<td class="num"><span>6,650</span></td> # 10.23일, 제주코딩베이스캠프 연구원, 종가
6,650
6650
<td class="num"><span>31,300</span></td> # 10.23일, 제주코딩베이스캠프 공업, 종가
31,300
31300
<td class="num"><span>13,250</span></td> # 10.23일, 제주코딩베이스캠프 출판사, 종가
13,250
13250
<td class="num"><span>2,600</span></td> # 10.23일, 제주코딩베이스캠프 학원, 종가
2,600
2600
```

Python

```python
그룹사별일일시가 = soup.select('.main')
오늘종가 = []
오늘시가총액 = []

for i in 그룹사별일일시가:
    오늘종가.append(int(i.select('.table > tbody > tr')[1].select('td')[1].
                    select('td > span')[0].text.replace(',', '')))
print(오늘종가)
```

Python

```
Out[-]
[6650, 31300, 13250, 2600]
```

Python

```python
오늘시가총액 = [i*10000 for i in 오늘종가]
전그룹사시가총액 = format(sum(오늘시가총액), ',')
전그룹사시가총액
```

Python

```
Out[-]
'538,000,000'
```

Python

문제 2번

전그룹사 시가총액 추이를 그래프로 그려주세요. x축은 날짜, y축은 가격입니다.

```python
# 각그룹사의 일일 시가총액을 구함
그룹사별일일시가 = soup.select('.main')
오늘종가 = []
오늘시가총액 = []

for i in 그룹사별일일시가:
    오늘종가.append(int(i.select('.table > tbody > tr')[1].select('td')[1].
                    select('td > span')[0].text.replace(',', '')))
```

Python

```python
그룹사별일일시가 = soup.select('.main')
오늘종가 = []
오늘시가총액 = []
for j in range(1, len(soup.select('.main')[0].select('table > tbody > tr'))):
    오늘종가 = []
    for i in 그룹사별일일시가:
        오늘종가.append(int(i.select('.table > tbody > tr')[j].select('td')[1].
                        select('td > span')[0].text.replace(',', '')))
    오늘시가총액.append(sum(오늘종가))
```

Python

```
오늘시가총액
```

JavaScript

```
Out[-]
[53800,
 53180,
 53615,
 52305,
 49035,
 48755,
 46970,
 46140,
 45900,
 45765,
 44000,
 43210,
 43830,
 44310,
 44850,
 44370,
 43935,
 44180,
 44410,
 46245]
```

Python

```python
# 날짜 table 크롤링
날짜전체 = soup.select('.main')[0].select('.table > tbody > tr > td > .date')
date = []
for i in 날짜전체:
    date.append(i.text)
date
```

Python

```
Out[-]
['2019.10.23',
 '2019.10.22',
 '2019.10.21',
 '2019.10.18',
 '2019.10.17',
 '2019.10.16',
 '2019.10.15',
 '2019.10.14',
 '2019.10.11',
 '2019.10.10',
 '2019.10.08',
 '2019.10.07',
 '2019.10.04',
 '2019.10.02',
 '2019.10.01',
 '2019.09.30',
 '2019.09.27',
 '2019.09.26',
 '2019.09.25',
 '2019.09.24']
```

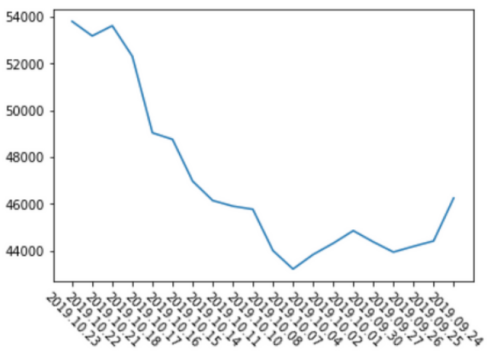
Python

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(date, 오늘시가총액)
plt.xticks(rotation = -45 ) # y 축 변수 기울기 설정
plt.show()
```

Python

Out[-]

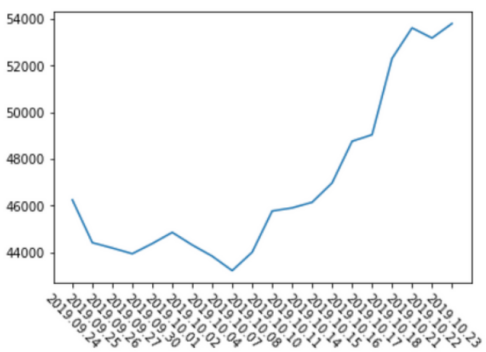



```
%matplotlib inline
# 날짜순을 정렬하여 재출력
import matplotlib.pyplot as plt

plt.plot(date[::-1], 오늘시가총액[::-1])
plt.xticks(rotation = -45 )
plt.show()
```

Python

Out[-]



```python
import requests
from bs4 import BeautifulSoup

response = requests.get("http://www.paullab.co.kr/stock.html")

response.encoding = 'utf-8'
html = response.text

soup = BeautifulSoup(html, 'html.parser')
```

Python

```
Out[-]
[538000000,
 531800000,
 536150000,
 523050000,
 490350000,
 487550000,
 469700000,
 461400000,
 459000000,
 457650000,
 440000000,
 432100000,
 438300000,
 443100000,
 448500000,
 443700000,
 439350000,
 441800000,
 444100000,
 462450000]
```

Python

```python
# 날짜 table 크롤링
날짜 = soup.select('.main')[0].select('.table > tbody > tr > td > .date')
date = []
for i in 날짜:
    date.append(i.text)
date
```

Python

```
Out[-]
['2019.10.23',
 '2019.10.22',
 '2019.10.21',
 '2019.10.18',
 '2019.10.17',
 '2019.10.16',
 '2019.10.15',
 '2019.10.14',
 '2019.10.11',
 '2019.10.10',
 '2019.10.08',
 '2019.10.07',
 '2019.10.04',
 '2019.10.02',
 '2019.10.01',
 '2019.09.30',
 '2019.09.27',
 '2019.09.26',
 '2019.09.25',
 '2019.09.24']
```

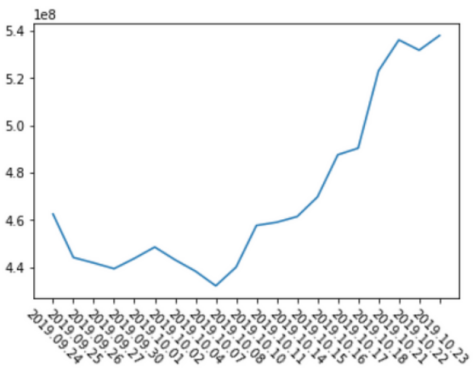
Python

```python
# 시각화
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(date[::-1], 오늘시가총액[::-1])
plt.xticks(rotation = -45)
plt.show()
```

Python

Out[-]



문제 3번

각 회사별 거래 총량과 전그룹사 거래 총량을 subplot으로 그려주세요.

```python
그룹사별일일데이터 = soup.select('.main')
그룹사별일일거래량 = [[],[],[],[]]
그룹사전체일일거래량 = []
# 데이터 구조 :
# 그룹사별일일거래량 = [[출판사], [연구원], [공업사], [학원]]
```

Python

```python
그룹사별일일데이터[0].select('.table > tbody > tr')[0]
그룹사별일일데이터[0].select('.table > tbody > tr')[1].select('td')[-1].text.replace(',','')
```

Python

```
Out[-]
'398421'
```

Python

```python
for j in range(1, len(soup.select('.main')[0].select('table > tbody > tr'))):
    그룹사별일일거래량[0].append(int(그룹사별일일데이터[0].select('.table > tbody > tr')[j].
                            select('td')[-1].text.replace(',', '')))
    그룹사별일일거래량[1].append(int(그룹사별일일데이터[1].select('.table > tbody > tr')[j].
                            select('td')[-1].text.replace(',', '')))
    그룹사별일일거래량[2].append(int(그룹사별일일데이터[2].select('.table > tbody > tr')[j].
                            select('td')[-1].text.replace(',', '')))
    그룹사별일일거래량[3].append(int(그룹사별일일데이터[3].select('.table > tbody > tr')[j].
                            select('td')[-1].text.replace(',', '')))
```

Python

```python
그룹사별일일거래량[0] # 출판사
그룹사별일일거래량[1] # 연구원
그룹사별일일거래량[2] # 공업사
그룹사별일일거래량[3] # 학원
len(그룹사별일일거래량[0])
그룹사별일일거래량[0]
```

Python

```
Out[-]
[398421,
 919571,
 1678055,
 2168857,
 1982922,
 839434,
 702104,
 764800,
 134558,
 288563,
 223839,
 199580,
 188467,
 160510,
 246145,
 705046,
 408859,
 404633,
 441923,
 211019]
```

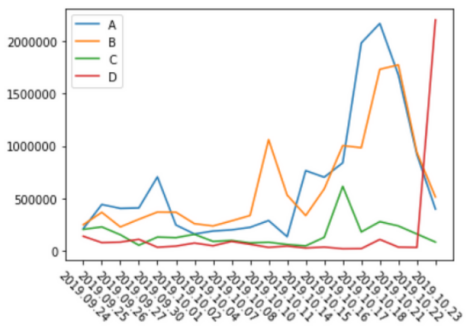
Python

```python
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(date[::-1], 그룹사별일일거래량[0][::-1], label='A')
plt.plot(date[::-1], 그룹사별일일거래량[1][::-1], label='B')
plt.plot(date[::-1], 그룹사별일일거래량[2][::-1], label='C')
plt.plot(date[::-1], 그룹사별일일거래량[3][::-1], label='D')
plt.xticks(rotation = -45 )
plt.legend(loc=2)
plt.show()
```

Python

Out[-]



```python
for i in range(len(그룹사별일일거래량[0])):
    s = 0
    for j in range(4):
        s += 그룹사별일일거래량[j][i]
    그룹사전체일일거래량.append(s)
그룹사전체일일거래량
```

Python

```
Out[-]
[3198301,
 2051067,
 3724291,
 4286651,
 3167249,
 2477184,
 1456343,
 1174487,
 771938,
 1463947,
 698527,
 673095,
 562816,
 650582,
 784490,
 1239662,
 872050,
 868624,
 1115164,
 803201]
```

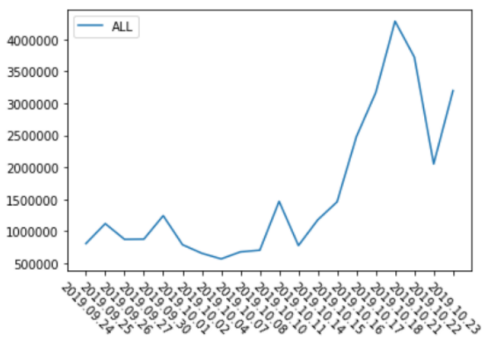
Python

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(date[::-1], 그룹사전체일일거래량[::-1], label='ALL')
plt.xticks(rotation = -45 )
plt.legend(loc=2)
plt.show()
```

Python

Out[-]

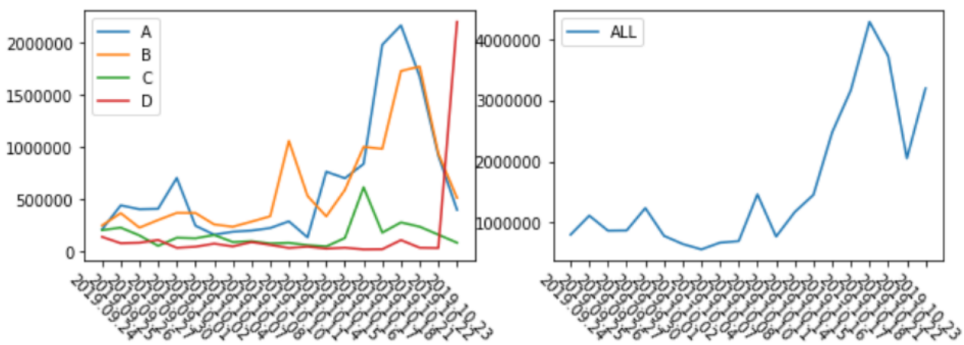


```python
f = plt.figure(figsize=(10,3))
# 1번 그림 (그룹사별)
ax = f.add_subplot(121)
ax.plot(date[::-1], 그룹사별일일거래량[0][::-1], label='A')
ax.plot(date[::-1], 그룹사별일일거래량[1][::-1], label='B')
ax.plot(date[::-1], 그룹사별일일거래량[2][::-1], label='C')
ax.plot(date[::-1], 그룹사별일일거래량[3][::-1], label='D')
plt.xticks(rotation = -45)
ax.legend(loc=2)
# 2번 그림 (전체)
ax2 = f.add_subplot(122)
ax2.figsize=(15,15)
ax2.plot(date[::-1], 그룹사전체일일거래량[::-1], label='ALL')
plt.xticks(rotation = -45)
ax2.legend(loc=2)
```

Python

Out[-]

```
<matplotlib.legend.Legend at 0x2987b073cf8>
```



초판 1쇄 발행 | 2020년 5월 4일
지은이 | 이호준 김유진 김혜원 이현창 장하림 차경림 이숭신 김대현
편　집 | 이호준
총　괄 | 이호준
펴낸곳 | 사도출판
주　소 | 제주특별자치도 제주시 동광로 137 대동빌딩 2층
표지디자인 | 차경림
홈페이지 | http://www.paullab.co.kr
E - mail | paul-lab@naver.com
ISBN | 979-11-88786-31-2

Python 활용편

비매품/무료
95560
9 791188 786329
ISBN 979-11-88786-32-9 (PDF)