

빅데이터 테크 아카데미

(파이썬+머신러닝)



과정 목표

- 빅데이터를 수집하고 통계적, 시각적으로 분석할 수 있으며 데이터를 정제해서 그것으로부터 스스로 학습하는 프로그램을 작성 합니다. 파이썬의 Numpy, Pandas, Scikit-learn 패키지를 이용하여 데이터를 가공, 분석, 예측 및 분류를 하는 모델을 활용할 수 있습니다.

Contents

1. 머신러닝

1. 머신러닝 소개

3. 머신러닝 & 딥러닝

1. 머신러닝 알고리즘

2. 프로세스

2. Tensorflow 기본문법

2. 파이썬

1. 기본 문법

3. Linear/logistic regression,

CNN, RNN

2. Numpy

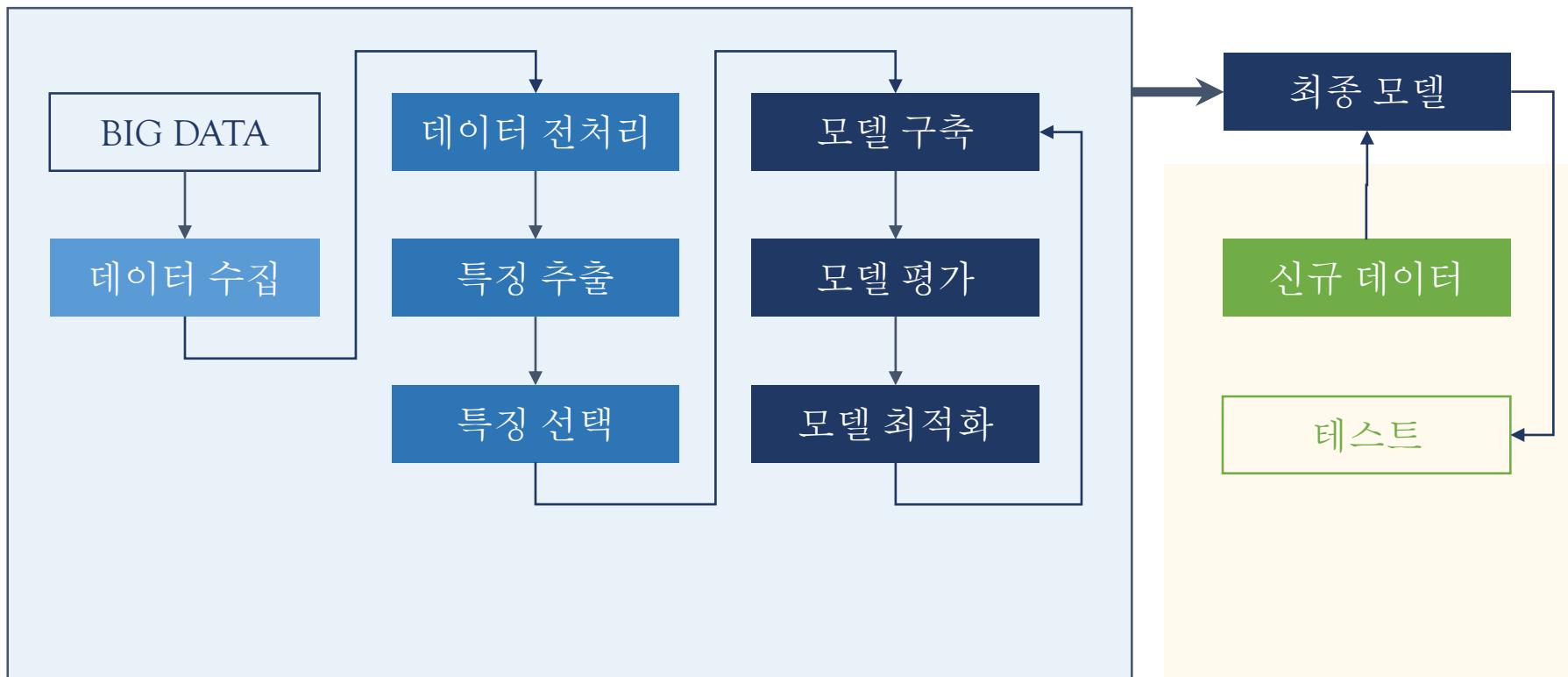
4. 개인별 challange

3. Jupyter Notebook

4. Pandas

5. 데이터 전처리

DataWork Flow



과정 소개 - 1일차

- 머신 러닝 소개, 파이썬 기본 문법



- Data Work Flow○] 해
- Python 기본 문법
- Numpy 패키지

Chap.1.1 머신러닝 소개

머신러닝이란

데이터에서 지식을 추출하는 작업입니다.(vs 데이터 마이닝)

통계학, 인공지능, 컴퓨터 과학이 어우러진 연구분야입니다.

예측 분석predictive analytics이나 통계적 머신러닝statistical learning으로도 불립니다.

위키피디아의 소개: “인공지능의 한 분야로 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야”

Arthur Samuel “Field of study that gives **computers** the ability to **learn** without being **explicitly programmed**”

명확한 해결 방법을 모르거나 해결 방법을 찾기 어려운 문제를 다룹니다.

어플리케이션

영화, 쇼핑, 음악 추천

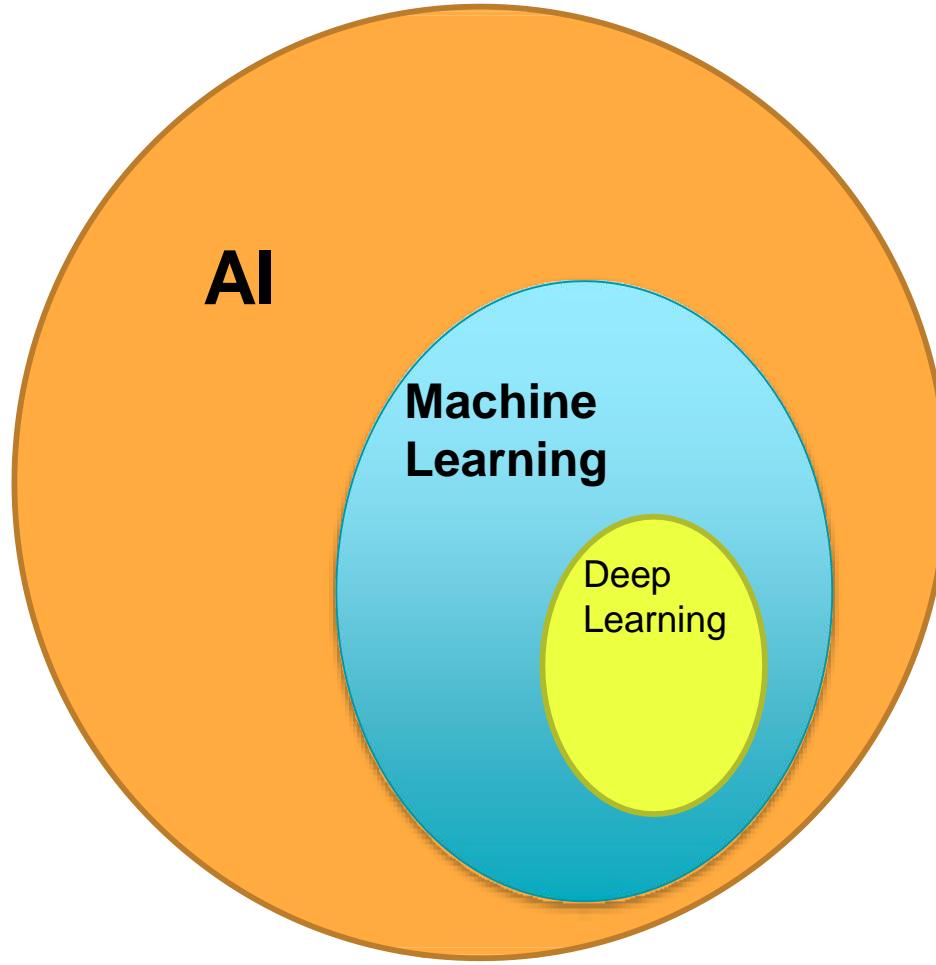
사진 분류, 검색, 친구 추천, 광고, 번역, 음성 인식

의학, 자율주행 자동차, 천문, 양자역학

유럽입자물리연구소
James Beacham



AI → 머신러닝 → 딥러닝



왜 머신러닝인가?

규칙기반 전문가 시스템 rule based expert system의 문제점(고전적인 스팸 필터)

결정에 필요한 로직이 한 분야나 작업에 국한, 작업 변경에 따라 시스템 개발을
다시 수행할 수도 있습니다.

규칙을 설계하려면 분야의 전문가들의 결정 방식에 대해 잘 알아야 합니다.

2001년 이전까지 이미지에서 얼굴을 인식하는 문제를 풀지 못했음

컴퓨터의 픽셀 단위는 사람이 인식하는 방식과 다름, 얼굴이 무엇인지 일련의
규칙을 만들기 어렵습니다.

머신러닝을 사용해 많은 얼굴 이미지를 제공하면 얼굴을 특정하는 요소를 찾을 수
있습니다.

지도 학습 Supervised Learning

알고리즘에 입력과 기대하는 출력을 제공합니다.

알고리즘은 입력으로부터 기대하는 출력을 만드는 방법을 찾습니다. 혹은 기대하는 출력이 나오도록 알고리즘을 가르칩니다.

스팸 문제의 경우 이메일(입력)과 스팸 여부(기대 출력)을 제공해야 합니다.

지도 학습의 예

손글씨 숫자 판별: 손글씨 스캔 이미지(입력), 우편번호(기대 출력)

데이터 수집에 수작업이 많음. 비교적 쉽고 적은 비용 소모.

의료 영상에 기반한 암진단: 영상(입력), 종양 여부(기대 출력)

도덕적/개인정보 문제 고가의 장비, 전문가 의견 필요

신용카드 부정거래 감지: 신용카드 거래내역(입력), 부정거래 여부(기대 출력)

고객에게 신고가 올 때까지 기다리면 됨.

비지도 학습 Unsupervised Learning

알고리즘에 입력은 주어지지만 출력은 제공되지 않습니다.

따라서 비지도 학습의 이해하거나 평가하기 어렵습니다.

비지도 학습의 예

블로그 글의 주제 구분: 사전에 어떤 주제가 있는지 얼마나 많은 주제가 있는지 모릅니다.

고객들을 취향에 따라 그룹으로 묶기: 부모, 독서광, 게이머 등 어떤 그룹이 있는지 미리 알 수 없고 얼마나 많이 있는지 모릅니다.(평가하기 어렵습니다)

비정상적 웹사이트 접근 탐지: 비정상적인 패턴은 각기 다를 수 있고 가지고 있는 비정상 데이터가 없을 수 있습니다. 현재 트래픽만 관찰할 수 있습니다.

데이터, 특성



좋은 입력 데이터를 만들어 내는 일을 특성 추출 feature extraction, 특성 공학 feature engineering이라고 합니다.

문제와 데이터 이해

머신러닝 프로세스에서 데이터를 이해하고 해결할 문제와 어떤 관련이 있는지 이해하는 것이 가장 중요합니다.

알고리즘마다 잘 들어맞는 데이터나 문제의 종류가 다릅니다.

알고리즘이나 방법론은 문제를 푸는 전체 과정 중 일부일 뿐입니다.

머신러닝 솔루션을 만들 동안 마음에 새겨야 할 질문들

어떤 질문에 대답을 원하는가? 원하는 답을 만들 수 있는 데이터를 가지고 있는가?
머신러닝의 문제로 가장 잘 기술할 수 있는 방법은 무엇인가?

문제를 풀기에 충분한 데이터가 있는가?

추출한 데이터의 특성은 무엇이고 좋은 예측을 위한 특성을 가지고 있는가?

애플리케이션의 성과를 어떻게 측정할 것인가?

다른 연구나 제품과 어떻게 협력할 수 있는가?

파이썬(Python)

과학 분야를 위한 표준 프로그래밍 언어가 되어 가고 있습니다.(vs Julia)

MATLAB, R 같이 도메인에 특화된 언어와 Java, C 같은 범용 언어의 장점을 모두 가지고 있습니다.

통계, 머신러닝, 자연어, 이미지, 시각화 등을 포함한 풍부한 라이브러리가 있습니다: Scikit-Learn, pandas, Numpy, Scipy, matplotlib, ...

브라우저 기반 인터랙티브 프로그래밍 환경인 주피터 노트북 Jupyter Notebook 0| 있습니다.

파이썬 주도 딥러닝 라이브러리: TensorFlow, Keras, PyTorch, Caffe2, CNTK, MXNet, Theano, ...

Scikit-Learn

오픈소스: <https://github.com/scikit-learn/scikit-learn>

소스 코드를 보고 어떻게 알고리즘이 구현되어 있는지 확인할 수 있습니다.

회귀, 분류, 군집, 차원축소, 특성공학, 전처리, 교차검증, 파이프라인 등 머신러닝에 필요한 도구를 두루 갖추고 있습니다.

풍부한 문서 (영문): <http://scikit-learn.org/stable/documentation>

학교, 산업 현장에서 널리 사용됩니다.

폭넓은 커뮤니티와 많은 튜토리얼, 예제 코드가 있습니다.

Apple's Core ML Support

Model type	Supported models	Supported tools
Neural networks	Feedforward, convolutional, recurrent	Caffe
		Keras 1.2.2+
Tree ensembles	Random forests, boosted trees, decision trees	scikit-learn 0.18
		XGBoost 0.6
Support vector machines	Scalar regression, multiclass classification	scikit-learn 0.18
		LIBSVM 3.22
Generalized linear models	Linear regression, logistic regression	scikit-learn 0.18
Feature engineering	Sparse vectorization, dense vectorization, categorical processing	scikit-learn 0.18
Pipeline models	Sequentially chained models	scikit-learn 0.18

Scikit-Learn 설치

NumPy, SciPy를 기반으로 합니다.

대화식 환경을 위해서는 IPython 커널과 Jupyter Notebook 설치가 필요합니다.

Anaconda(<https://www.continuum.io/anaconda-overview>)

무료, 과학전문 파이썬 배포판, 수백개의 패키지, 맥/윈도우/리눅스 지원, 인텔 MKL 라이브러리 포함

Enthought Canopy(<https://www.enthought.com>)

과학전문 파이썬 배포판, 학생, 교육 기관에 무료(scikit-learn 미포함)

Python(x, y)

윈도우즈 전용 과학 파이썬 배포판

필수 라이브러리

Jupyter Notebook

프로그램 코드 + 문서 + 결과 (텍스트, 그래프)를 위한 대화식 개발 환경입니다.

탐색적 데이터 분석에 유리하여 많은 과학자, 엔지니어들이 사용합니다.

<https://jupyter.org>

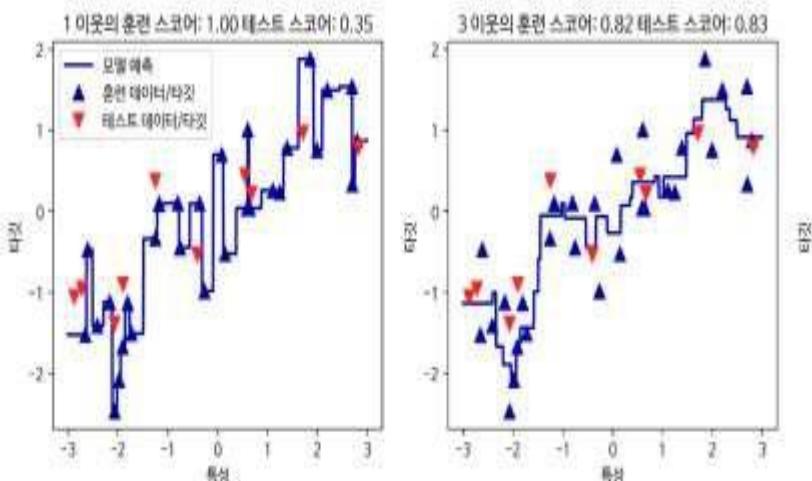


KNeighborsRegressor 분석

```
In [25]: fig, axes = plt.subplots(1, 3, figsize=(15, 4))
# -3과 3 사이에 1,000 개의 데이터 포인트를 만듭니다
line = np.linspace(-3, 3, 1000).reshape(-1, 1)
for n_neighbors, ax in zip([1, 3, 9], axes):
    # 1, 3, 9 이웃을 사용한 예측을 합니다
    reg = KNeighborsRegressor(n_neighbors=n_neighbors)
    reg.fit(X_train, y_train)
    ax.plot(line, reg.predict(line))
    ax.plot(X_train, y_train, '^', c=mlearn.cm2(0), markersize=8)
    ax.plot(X_test, y_test, 'v', c=mlearn.cm2(1), markersize=8)

    ax.set_title(
        '{} 이웃의 훈련 스코어: {:.2f} 테스트 스코어: {:.2f}".format(
            n_neighbors, reg.score(X_train, y_train), reg.score(X_test, y_test)))
    ax.set_xlabel('특성')
    ax.set_ylabel('타깃')
    axes[0].legend(["모델 예측", "훈련 데이터/타깃", "테스트 데이터/타깃"], loc="best")
```

Out[25]: <matplotlib.legend.Legend at 0x114152b00>



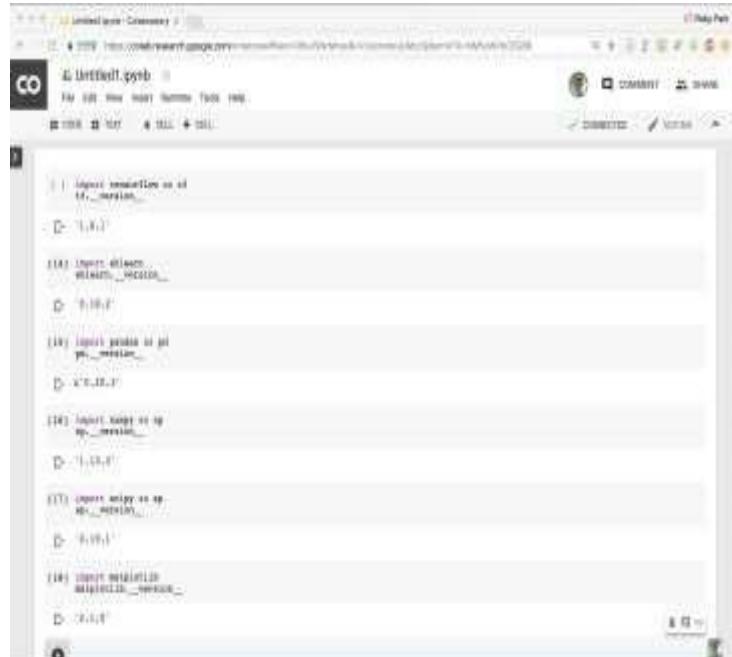
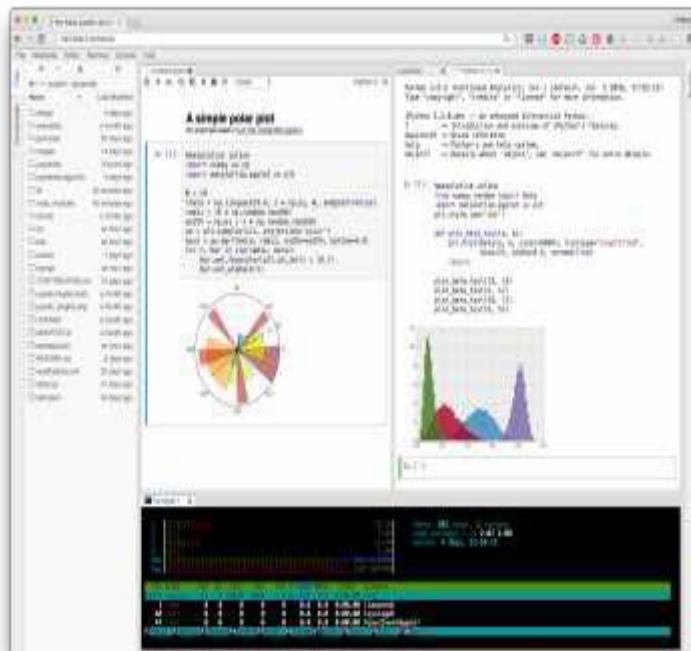
IDE for Jupyter Notebook

PyCharm: 파이썬 IDE로 노트북 파일을 지원합니다.

Rodeo(<https://www.yhat.com/products/rodeo>) : 데이터 분석용 전문 IDE입니다.

JupyterLab(<https://github.com/jupyter/jupyterlab>) : 차세대 주피터 노트북 환경

Colaboratory(<https://colab.research.google.com>) : 구글 드라이브와 연동되는 연구와 교육
목적의 노트북 환경



NumPy

다차원 배열을 위한 기능과 선형 대수를 비롯해 고수준 수학 함수와 유사 난수 생성기를 포함하고 있습니다.

scikit-learn의 기본 데이터 구조입니다.(TensorFlow 도)

<http://www.numpy.org>

모듈의 별칭을 만듦

<http://www.scipy-lectures.org/> 의 1장

In [2]: `import numpy as np`

```
x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))
```



NumPy

ndarray

x:
[[1 2 3]
 [4 5 6]]

동일한 데이터 타입

SciPy

선형 대수, 최적화, 통계 등 많은 과학 계산 함수를 모아놓은 파이썬 패키지

scikit-learn은 알고리즘 구현에 SciPy에 많이 의존하고 있습니다.

0이 많이 포함된 행렬을 효율적으로 표현하기 위한 희소 행렬 sparse matrix
scipy.sparse 패키지를 사용합니다.

<https://www.scipy.org/scipylib>

<http://www.scipy-lectures.org/>
의 2.5절

```
In [3]: from scipy import sparse  
  
# 대각선 원소는 1이고 나머지는 0인 2차원 NumPy 배열을 만듭니다.  
eye = np.eye(4)  
print("NumPy 배열:\n{}".format(eye))
```

NumPy 배열:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

단위 행렬

SciPy

밀집 배열로 부터 희소 행렬을 생성
(메모리 부족 가능성)

In [4]: # NumPy 배열을 CSR 포맷의 SciPy 희박 행렬로 변환합니다.

0이 아닌 원소만 저장됩니다.

sparse_matrix = sparse.csr_matrix(eye)

print("\nSciPy의 CSR 행렬:\n{}\n".format(sparse_matrix))

SciPy의 CSR 행렬:

(0, 0)	1.0
(1, 1)	1.0
(2, 2)	1.0
(3, 3)	1.0

Compressed Sparse Row Format

Coordinate Format

대각 행렬의 위치

In [5]: data = np.ones(4)

row_indices = np.arange(4)

col_indices = np.arange(4)

eye_coo = sparse.coo_matrix((data, (row_indices, col_indices)))

print("COO 표현:\n{}\n".format(eye_coo))

COO 표현:

(0, 0)	1.0
(1, 1)	1.0
(2, 2)	1.0
(3, 3)	1.0

인덱스를 지정하여 희소 행렬을 생성

matplotlib

과학 계산용 그래프 라이브러리입니다.

선, 히스토그램, 산점도 등 다양한
그래프를 그릴 수 있으며 출판 수준의
고품질을 제공합니다.

노트북에 그래프 포함: %matplotlibinline

<https://matplotlib.org/>



그외 대표적인 그래프 라이브러리: Bokeh, Seaborn,
Plotly

별칭 이름

```
In [6]: %matplotlib inline
import matplotlib.pyplot as plt

# -10에서 10까지 100개의 간격으로 나뉘어진 배열을 생성합니다.
x = np.linspace(-10, 10, 100)
# 사인 함수를 사용하여 y 배열을 생성합니다.
y = np.sin(x)
# plot 함수는 한 배열의 값을 다른 배열에 대응해서 선 그래프를 그립니다.
plt.plot(x, y, marker="x")
```

Out[6]: [`<matplotlib.lines.Line2D at 0x110403b00>`]

A line graph showing a sine wave plotted against x from -10 to 10. The y-axis ranges from -1.00 to 1.00. The plot shows three full cycles of the sine function, with peaks at approximately x = -9.4, -1.5, and 7.9, and troughs at approximately x = -7.9, -1.5, and 9.4.

pandas

데이터 처리와 분석을 위한 라이브러리

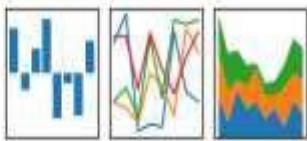
DataFrame inspired by R's data.frame

엑셀과 비슷하게 이종 데이터 포함 가능합니다.(numpy와 크게 다른 점)

웨스 매킨니 Wes Makinney의 “파이썬 라이브러리를 활용한 데이터 분석”

<http://pandas.pydata.org>

pandas
 $y_t = \beta' x_t + \mu_t + \epsilon_t$



별칭 이름

In [7]:

```
import pandas as pd
```

회원 정보가 들어간 간단한 데이터셋을 생성합니다.

```
data = {'Name': ["John", "Anna", "Peter", "Linda"],  
        'Location' : ["New York", "Paris", "Berlin", "London"],  
        'Age' : [24, 13, 53, 33]  
       }
```

```
data_pandas = pd.DataFrame(data)
```

IPython.display는 주피터 노트북에서 Dataframe을 미려하게 출력해줍니다.
display(data_pandas)

	Age	Location	Name
0	24	New York	John
1	13	Paris	Anna
2	53	Berlin	Peter
3	33	London	Linda

CSV, SQL, 엑셀 등에서 데이터 읽을 수 있음

In [8]:

```
# Age 열의 값이 30 이상인 모든 행을 선택합니다.  
display(data_pandas[data_pandas.Age > 30])
```

	Age	Location	Name
2	53	Berlin	Peter
3	33	London	Linda

Python 2 vs Python 3

어떤 것을 써야할 지 잘 모르겠다면 Python 3이 적절합니다.

이미 파이썬 2로 작성한 코드가 많이 있거나 사용하고 있는 라이브러리가 파이썬 2 밖에 지원하지 않는 경우에는 파이썬 2를 사용합니다.

six 패키지를 사용해 호환된 코드를 만들 수 있습니다. (<https://pythonhosted.org/six/>)

Python 2.7은 2020년까지만 지원합니다.

사용하는 Version

Python 3.6

scikit-learn 0.19.x

matplotlib 2.0.x

NumPy 1.13.x

SciPy 0.19.x

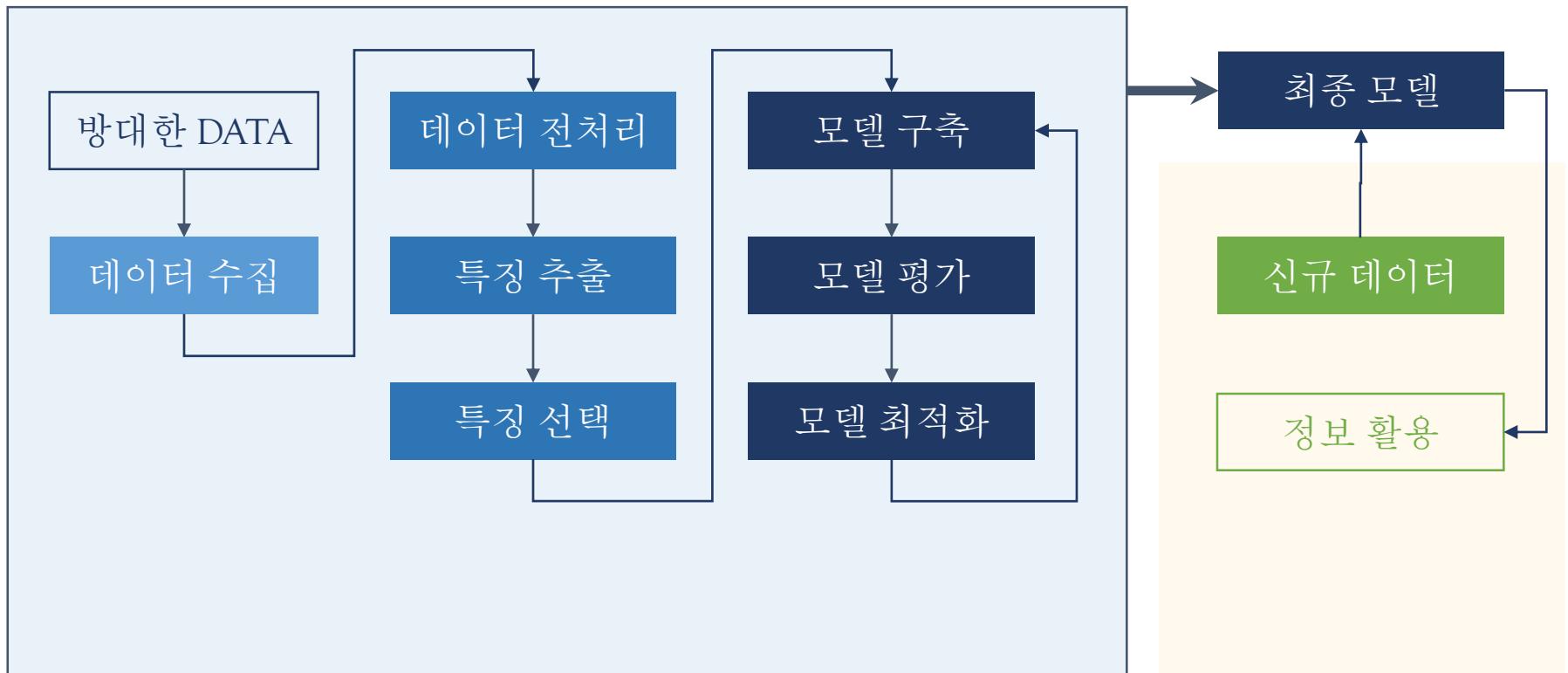
pandas 0.20.x

머신러닝 언어

	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

Chap.1.2 머신러닝 프로 세스

Data Work Flow



데이터 수집

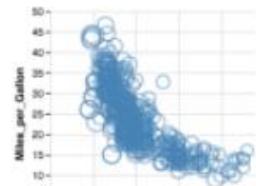
- 이미 수집된 데이터가 있는 경우 vs 완전히 새로 수집해야 하는 경우
- 데이터가 그냥 많다고 다 쓸 수 있는 것은 아님 ← 분석가가 판단
- 새로 수집해야 하는 경우
 - 데이터의 수집 주기는 어떻게 되는가?
 - 얼마나 많은 데이터가 필요한가?
 - 어떤 특성을 포함해야 하는가?
 - 훈련 데이터가 충분히 대표성을 띠는가?
 - 데이터는 어떤 형식으로 가져올 수 있는가?

데이터 전처리

- 데이터를 그대로 특징으로 사용할 수 있는 경우
 - 형식화 - 분석 목적에 맞는 적절한 데이터 형식으로 변환(범주형 / 수치형)
 - 정제 - 문제 해결에 도움이 되지 않는 데이터는 제거, 누락된(결측) 데이터 처리
 - 정규화 - 개별 특징이 동일한 숫자 범위 안에 들어가도록 변환
 - 분해 - 하나의 특징이 복잡한 여러 개념을 포함하는 경우 → 분리 or 선택
 - 결합 - 두 가지를 모두 고려했을 때 더 큰 의미를 가지는 특징은 결합
- 이 모든 과정은 '데이터 시각화(visualization)' 와 함께 수행

데이터 전처리

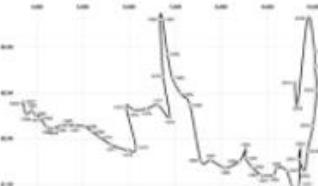
Dot & Scatter Plots



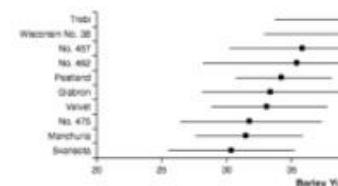
Scatter Plot



Scatter Plot Null Values



Connected Scatter Plot



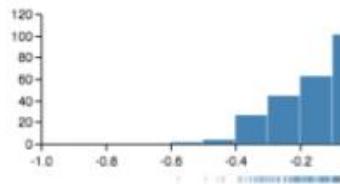
Error Bars



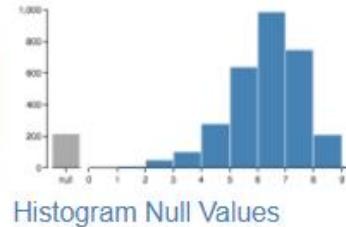
Barley Trellis Plot

★★★ 그래프마다 역할과 장단점이 있음!

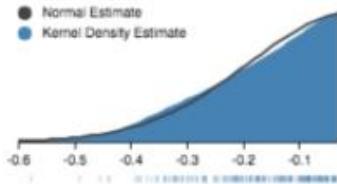
Distributions



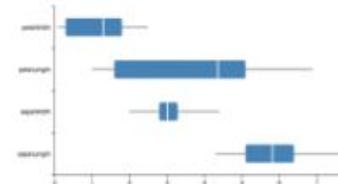
Histogram



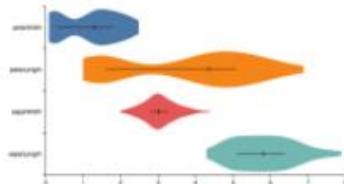
Histogram Null Values



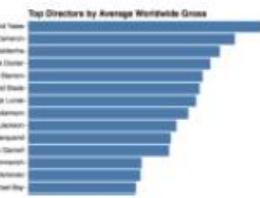
Probability Density



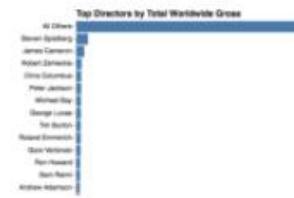
Box Plot



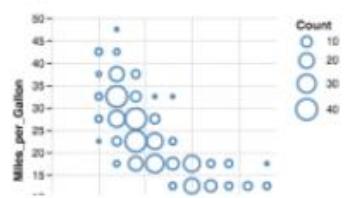
Violin Plot



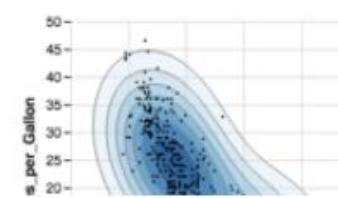
Top K Plot



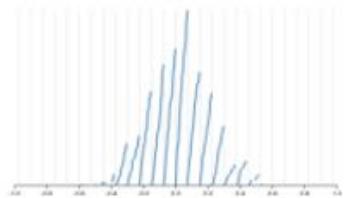
Top K Plot With Others



Binned Scatter Plot



Contour Plot



Wheat Plot

알고리즘 선택과 평가

- 머신러닝의 주요 알고리즘은 이미 다 증명이 되어 있다
- 머신러닝의 모든 알고리즘을 다 증명하고 구현할 수 있는 사람이 과연 몇 명이나 될까?
- 머신러닝의 사용 목적에 따라 알고리즘 이해도 수준이 다르다
- 직접 구현 vs 라이브러리 사용
- 알고리즘을 왜 직접 구현하려고 하는가? > 성능 튜닝
- 성능 검증할 수 있는 능력이 필요하다 ← 여러 개 비교해 보면 된다 :)

Chap.2.1 파일 쓰기

Python

- 등장시기: 1990년
 - 설계자: 귀도 반 로섬
 - 동기: 1989년 크리스마스때 연구실이 달혀있어서 심심
 - 목표: 강력한 동적 프로그램
-
- 1.x: 1994年
 - 2.x: 2000年。Major 언어화
 - 3.x: 2008年。2x와 호환 안됨

귀도 반 로섬



본명	Guido van Rossum
출생	1956년 1월 31일 (62세)
	 네덜란드 노르트홀란트 주 하클럼
학력	암스테르담 대학교 출신
직업	소프트웨어 엔지니어



파이썬과 머신러닝

- 데이터 분석 필수 라이브러리

라이브러리	설명
numpy	고성능의 과학계산 컴퓨팅과 데이터 분석에 필요한 기본 패키지
scipy	과학적인 파이썬 라이브러리 NumPy와 SciPy는 역사적으로 그 코드 베이스를 공유하지만 나중에는 분리
matplotlib	Numpy를 기반으로 도면을 그리는 라이브러리
pandas	고수준의 자료 구조와 파이썬을 통한 빠르고 쉬운 데이터 분석 도구 포함

- 이 모든 라이브러리를 다 익히고 머신러닝 모델에 들어간다? ← 기본 조작 법만 알고 그때 그때 찾으면 됨!

Pandas

- 고수준의 자료 구조와 빠른 데이터 분석을 지원하는 도구를 포함하는 라이브러리
- Pandas의 특징
 - 자동적으로 또는 명시적으로 축의 이름을 따라 데이터를 정렬할 수 있는 자료 구조
 - 잘못 정렬된 데이터에 의한 일반적인 오류를 예방
 - 다양한 소스에서 가져온 다양한 방식으로 색인돼 있는 데이터를 다루는 기능
 - 통합된 시계열 기능
 - 시계열/비시계열 데이터를 함께 다룰 수 있는 통합 자료 구조
 - 데이터 축약 연산을 축의 이름 같은 메타 데이터로 전달 가능
 - 누락된 데이터를 유연하게 처리하는 기능
 - SQL 같은 일반 데이터베이스처럼 데이터를 합치고 관계연산 수행 기능
- **데이터프레임!**

Numpy

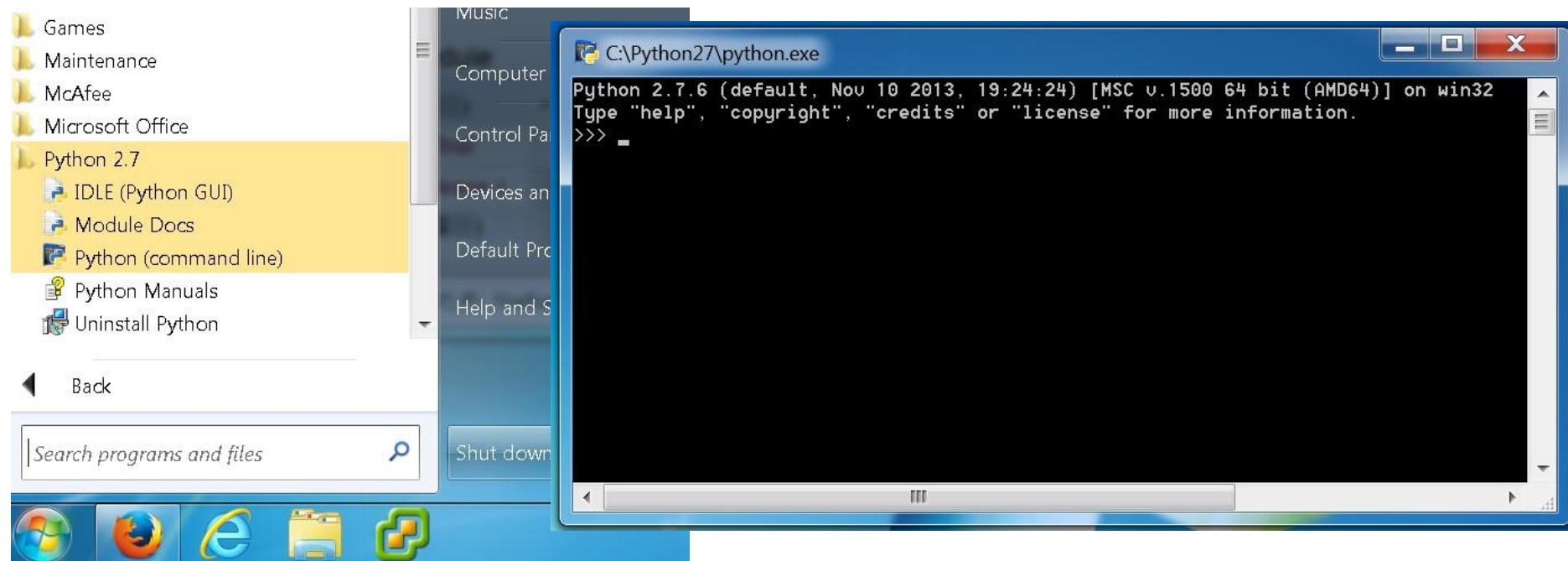
- 고성능 과학 계산 컴퓨팅과 데이터 분석에 필요한 기본 패키지
- Numpy의 특징
 - 빠르고 메모리를 효율적으로 사용
 - 다차원 배열인 ndarray(벡터 산술연산과 브로드캐스팅 기능)
 - 반복문을 작성 불필요
 - 전체 데이터 배열에 대한 빠른 연산을 제공하는 표준 수학 함수
 - 배열 데이터를 디스크에 쓰기/읽기
 - 선형대수, 난수 발생기, 푸리에 변환 가능
 - C, C++, 포트란으로 쓰여진 코드 통합(글루 언어)
- 배열을 만들고, 자르고, 붙이고, 뽑아내고, 정리하는 라이브러리

Scipy / Scikit

- Scipy - 과학 기술 계산용 함수 및 알고리즘을 제공하는 패키지
- Scipy의 특징
 - Numpy 배열 프레임워크를 기반으로 만들어짐
 - 통계, 클러스터링 분석, 신호 및 이미지 처리 등 다양한 고급 분석 기능을 제공
- Scikit-learn - 머신러닝 알고리즘을 제공해 주는 패키지
- Scikit-learn의 특징
 - 머신러닝에 필요한 거의 모든 기능들을 구현
 - 라이브러리 설명뿐만 아니라 알고리즘도 함께 볼 수 있음
 - 파이썬 머신러닝 = sklearn + tensorflow

Prompt에서 python을 호출

- Windows: Start --> python (command line)
- Mac: terminal에서 python 타이핑



Prompt 사용

- Prompt에서 아래 코드를 실행시켜 본다

```
print('hello world')
```

```
6+1*3
```

```
(6+1)*3
```

```
5/2
```

```
5/2.0
```

```
5/0
```

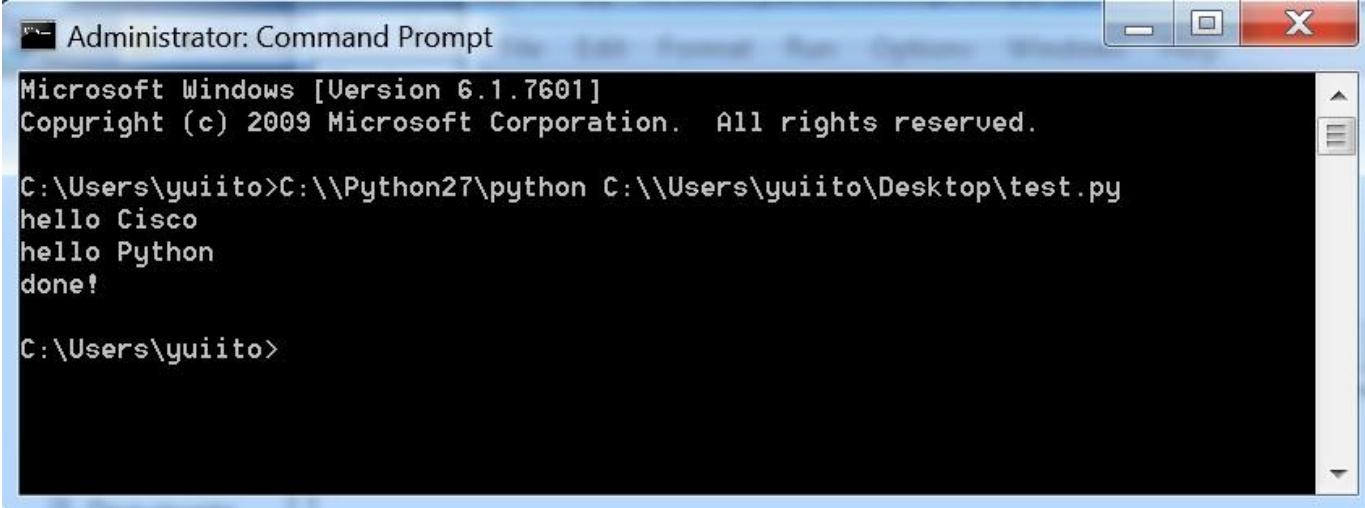
```
print("{} + {} = {}".format(2,3,5))
```

```
print """hello I love  
myself  
I love Python too """
```

```
import sys  
sys.exit()
```

command로 script 파일 실행

- 파일에 프로그램 작성후, 커맨드로 실행
- Python 의 스크립트 파일의 확장자는 .py
- 필요한 경우 python 의 패스를 지정



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The window displays the following output:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

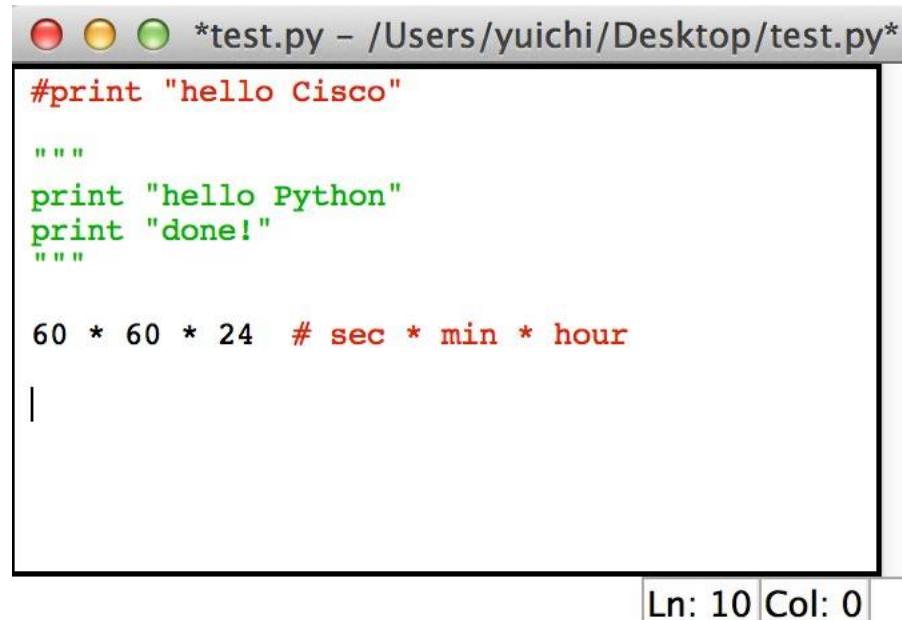
C:\Users\yuiito>C:\\\\Python27\\\\python C:\\\\Users\\\\yuiito\\\\Desktop\\\\test.py
hello Cisco
hello Python
done!

C:\Users\yuiito>
```

Windows에서 실행한 화면

comment

- comment : 프로그램의 주석。 실행시 무시됨
- 1행의 경우: #
- 여러행의 경우:
 - """ 복수행 지정가능"""
 - ''' '''



A screenshot of a Python code editor window titled "test.py - /Users/yuichi/Desktop/test.py*". The code contains the following:

```
#print "hello Cisco"
"""
print "hello Python"
print "done!"
"""

60 * 60 * 24 # sec * min * hour
```

The code editor shows syntax highlighting where comments are in green and code in red. A cursor is visible at the end of the last line. The status bar at the bottom right indicates "Ln: 10 Col: 0".

변수와 타입

- 타입선언: 이 변수는 어떤 타입이라고 명시
- Python 의 변수는 타입선언이 없음
- 변수에 어떤 타입의 데이터도 모두 저장 가능

```
int x = 5;  
x = "Java" //Error
```

Java

```
x = 5  
print(x)  
x = "Python" # OK  
print(x, type(x))
```

Python

Python 데이터 타입

- 기본 타입
 - 숫자: byte, character, int, float32, float64 등
 - 문자형: 영어, 한글 등
 - 리스트: 배열과 비슷
 - Bool: True or False
 - 함수: Python에서는 함수도 타입의 일종
 - 그 외
- 유저가 정의하는 타입(Class)

숫자형

- 정수(int)와 실수(float)、복소수 등
- 매우 큰 수도 걱정 없음>> $1234567 * 3456789 = 4267637625363$ (42 조)

사용가능한 연산자	설명
$M + N$	더하기
$M - N$	빼기
$M * N$	곱하기
M / N	나누기
$M \% N$	나머지
$M^{**}N$	승수($M * M * M \dots N$ 회)

함수

- 증가는 사용할 수 없음 > 대입 연산자를 사용

Java

```
int i = 0;  
i++;
```

Python

```
i = 0  
i = i+1  
i += 1
```

연산자	설명
$M += N$	$M = M + N$
$M -= N$	$M = M - N$
$M *= N$	$M = M * N$
$M /= N$	$M = M / N$
$M %= N$	$M = M \% N$

문자타입

- 3종류의 할당방식

홑따옴표

```
string = 'Hello Python'
```

쌍따옴표

```
string = "Hello Python"
```

셋따옴표

```
string = """Hello World  
Hello Python # this isn't comment  
Hello Cisco"""
```

줄바꿈과 주석을 모두
포함해 문자열



문자타입

- 문자타입도 연산기호 사용 가능

```
print "I love " + "python"  
=> "I love Python"
```

```
print "hello " * 3  
=> "hello hello hello "
```

```
print "I say " + ("hello " * 3)  
=> "I say hello hello hello"
```

```
x = "I love "  
y = "Python"  
print(x+y)  
=> "I love Python"
```

Bool 타입

- True, False 을 갖는 타입
- If 등의 제어문에 자주 사용됨

연산자	의미
A and B	A와 B 가 True 면 True
A or B	A또는 B 가 True 면 True
A == B	A와 B가 같으면 True
A != B	A와 B가 다르면 True
A <> B	위와 같음
not A	A가 False 면 True

List 탐색

- 배열길이 구하기: `len(list)`
- 오브젝트의 `index` 구하기: `.index(x)`
- 배열에 오브젝트 추가하기: `.append(x)`
- 배열에서 오브젝트 제거하기: `del`

```
list1 = [1,2,3,4,5]
print(list1.index(3))
=> 2

print(len(list1))
=> 5
```

```
list1.append(6)
print(list1)
=> [1,2,3,4,5,6]

del list2[1]
print(list2)
=> [1,3,4,5,6]
```

그외의 타입들

- Tuple
- Dictionary
- File
- Dates, Times
- 사용자 정의 타입(class)
- 함수

코드 블록과 들여쓰기

- C, Java 등: {} 기호로 코드 정리
- Python: indent (들여쓰기)로 코드 정리

```
for(int i=0; i<10;  
    i++){ if(i%2  
    == 0){  
        System.out.println(i + " is  
        even");  
    }  
    else{  
        System.out.println(i + " is  
        odd");  
    }  
}
```

```
for i in range(10):  
    if(i%2 == 0):  
        print("{} is even".format(i))  
    else:  
        print("{} is odd".format(i))  
print("done")
```

Python

조건분기 SWITCH-CASE

- 없음! If 문으로 작성

```
switch(x){  
    case A: x>100:  
        break;  
    case B: x>10  
        break  
    default: ...  
}
```

Java

```
if(x>100):  
    ...  
elif(x>10):  
    ...  
else:  
    ...
```

Python

for 문법

for 변수 in List:
 실행할 코드

List의 0번째 값을 변수에 대입해서 코드 실행,
List의 1번째 값을 변수에 대입해서 코드 실행,
List의 2번째.....

```
string = ""  
for char in ["H","e","l","l","o"]:  
    string += char  
print(string)
```

```
>>>  
Hello
```

열거자(Iteration)

- python 의 for 문이 iterator 방식
- 튜플、문자열、dictionary 등도 key 를 이용해 열거

```
for char in "hello":  
    print(char)  
=> h e l l o
```

for 기준형태

- 일정 횟수 반복 시 `range` 함수를 사용
- `range(x)`: 0 부터 $x-1$ 까지의 정수 리스트를 리턴
- `range(x,y)`: x 부터 $y-1$ 까지의 정수 리스트를 리턴

```
for(int i=0; i<10; i++){  
    System.out.println(i)  
}
```

Java

```
for i in range(10):  
    print(i)
```

Python

while 으로 반복 처리

- 몇번 반복될지 모르는 경우에 사용
- 무한 반복 주의

```
x = 6789329  
i = 1  
while(2**i < x):  
    i += 1  
  
print("2 ^ {} > {}".format(i,x))
```

$$2^{23} > 6789329$$

연습(반복, 조건분기)

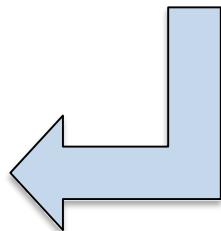
- 0 ~ 100 까지 짹수를 출력하세요
 - 0 2 4 6 … 98 100
- 100 ~ 1 까지 출력하세요
 - 100 99 98 … 2 1

함수 호출

- 특정 처리를 수행하기 위한 호출
- 함수명과 인자를 명시해서 호출
- 함수는 반환값을 돌려줌(리턴)

함수명(인자)

반환값



```
>>> length = len([1,2,3,4,5])
>>> print(length)
5
```

함수 : 복잡한 처리과정을 은닉

- 함수를 사용함으로써 코드가 읽기 쉬워짐

```
i = -5
```

```
if(i<0): i  
    = -i
```

```
print i  
=> 5
```

얼핏 봄서는
의도를 알기
힘듦

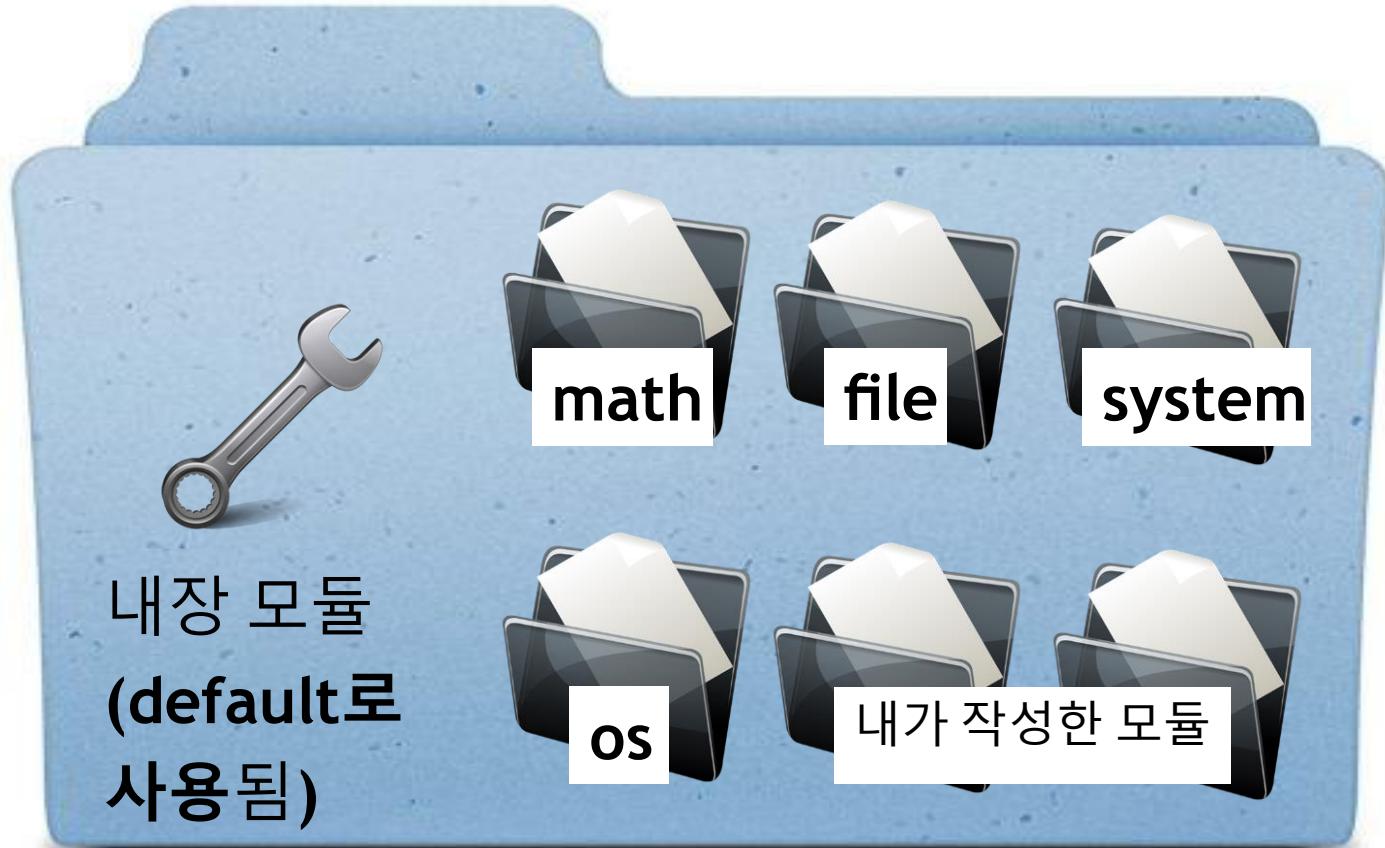
```
i = -5
```

```
i = abs(i)  
print i  
=> 5
```

함수명만 봄도 「절
대값」을 구하려는
의도를 알수 있다

module 을 이용한 계층화

- 기능별로 프로그램 정리



module access

- import 로 모듈을 이용

```
>>> random()  
Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
NameError: name 'random' is not defined
```

import 안하면 이용
불가/에러

```
>>> import random  
>>> random.random()  
0.6019003331149728
```

「모듈명.함수」로 호출

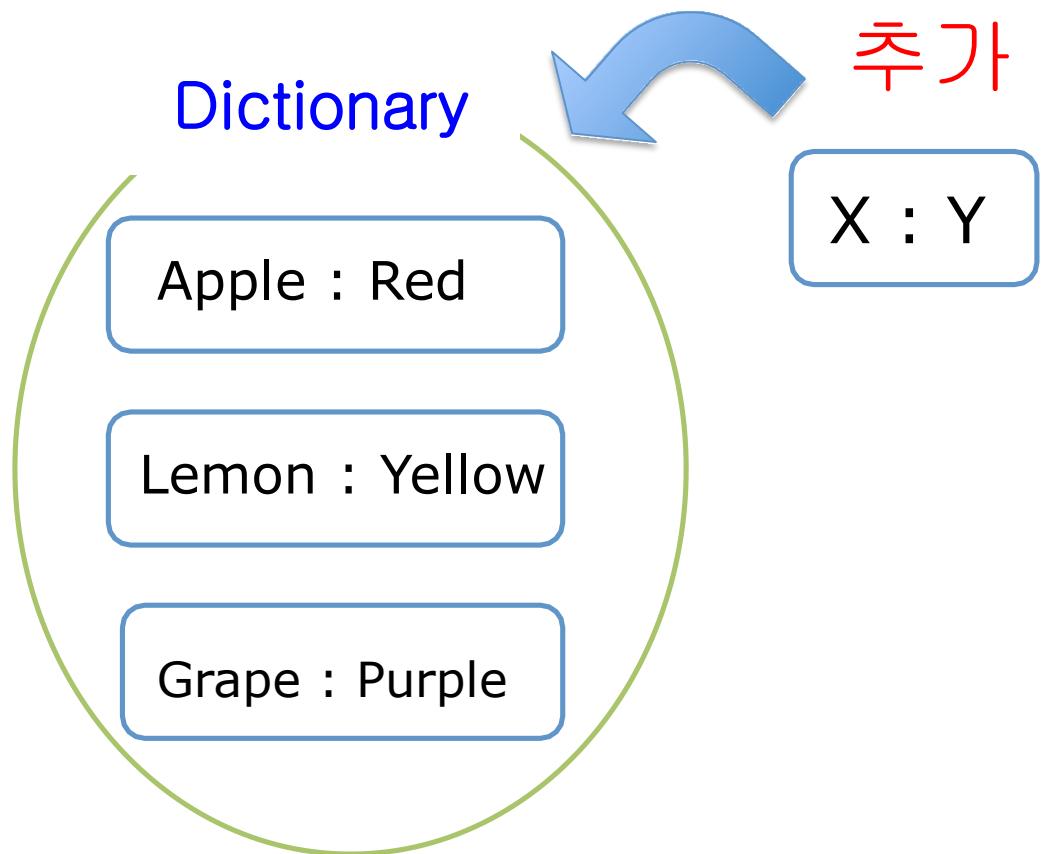
Dictionary type

- Key 와 Value 의 쌍으로 저장하는 데이터 타입
- Key로 Value 를 얻음

가져오기

Apple의 색은?

Red !!



Dictionary 만들기

- dict 형식 : {key1:value1, key2:value2}
- 값 가져오기: dic_object[key]
- 값 추가(수정): dic_object[key] = value
- 값 제거: del dic_object[key]
- key 전부 가져오기: dic_object.keys()
- key 존재 유무 확인: dic_object.has_key(key)

Dict 조작

dict object 생성

```
>>> d = {"Apple": "Red", "Lemon": "Yellow"}
```

key 를 이용해서 Value 가져오기

```
>>> d["Apple"] 'Red'
```

dict object 에 새 요소 추가

```
>>> d["Grape"] = "Purple"
```

```
>>> d
```

```
{'Grape': 'Purple', 'Lemon': 'Yellow', 'Apple': 'Red'}
```

dict object 에서 요소 제거

```
>>> del d["Apple"]
```

```
>>> d
```

```
{'Grape': 'Purple', 'Lemon': 'Yellow'}
```

Tuple type

- 수정 불가능
- 요소의 수가 결정된 「복수개의 데이터」

show logging

Link down
OSPF Neighbor XXX down
Admin up
Link up
OSPF Neighbor XXX up
....

행수(요소수)가 늘어남
=> List

사원정보

Park
Korea
2011-04-01
Seoul
developer

행수(요소수)늘지 않고 고정
=> Tuple

Tuple 조작

- Tuple 생성: tupleObject = (elem1, elem2,...)
- 요소 가져오기: item = tupleObject[index]
- 요소 변경: 불가능

Tuple 조작

Tuple 생성

```
>>> me = ("Park", "Chuncheon", 2011)
```

```
>>> me
```

```
(Park ', ' Chuncheon ', 2011)
```

요소 참조

```
>>> me[1] 'Park'
```

```
>>> me = ("Park", "Chuncheon", 2011)
```

```
>>> (family, name, year) = me
```

```
>>> year 2011
```

요소 변경

```
>>> me[2] = 2009
```

```
Traceback (most recent call last): File
```

```
 "<stdin>", line 1, in <module>
```

TypeError: 'tuple' object does not support item assignment

Set

- 중복, 순서없이 저장하는 type
- Value 없는 Map 과 비슷

```
>>> a = set([1,2,3,4,5,3,2])
>>> print(a)
set([1, 2, 3, 4,5])
```

```
>>> a = set("hello")
>>> b = set("world")
>>> print(a)
set(['h', 'e', 'l', 'o'])

>>> print(b)
set(['d', 'r', 'o', 'w', 'l'])

>>> print(a & b)
set(['l', 'o'])
```

Set 조작

- 초기화: `set(sequence type의 object)`
- 추가: `set_object.add(value)`
- 제거: `set_object.discard(value)`

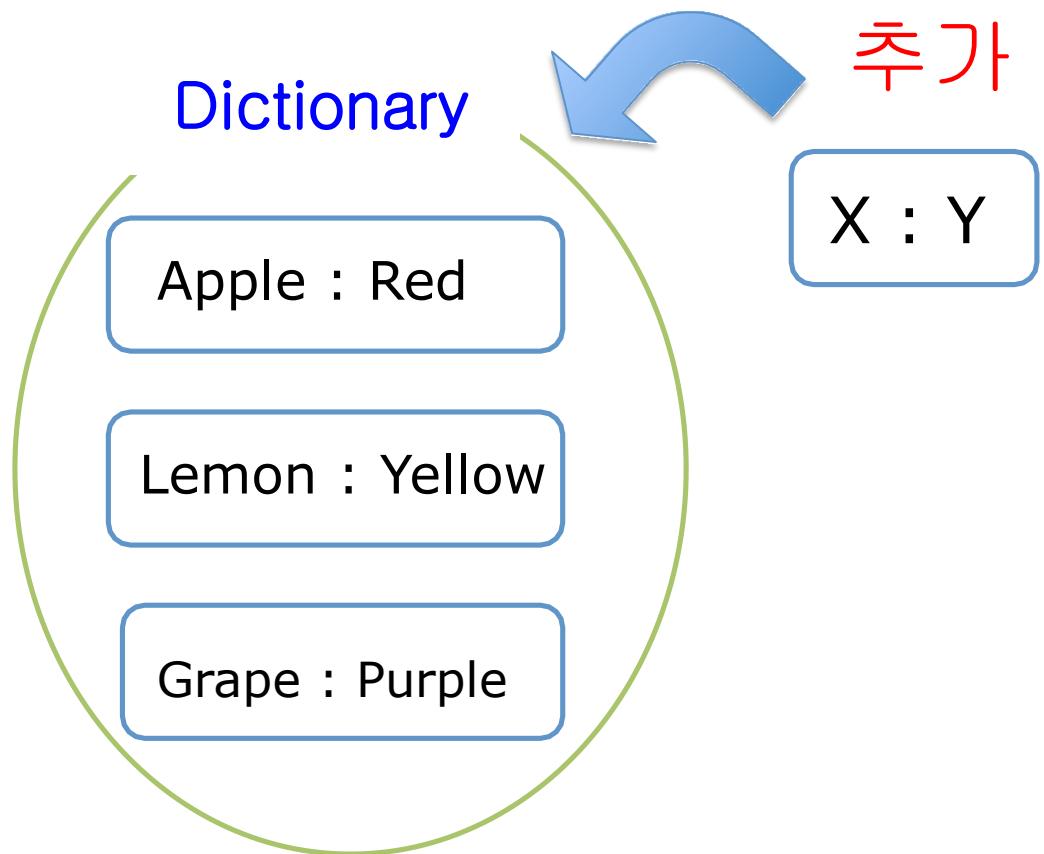
Dictionary type

- Key 와 Value 의 쌍으로 저장하는 데이터 타입
- Key로 Value 를 얻음

가져오기

Apple의 색은?

Red !!



Dictionary 만들기

- dict 형식 : {key1:value1, key2:value2}
- 값 가져오기: dic_object[key]
- 값 추가(수정): dic_object[key] = value
- 값 제거: del dic_object[key]
- key 전부 가져오기: dic_object.keys()
- key 존재 유무 확인: dic_object.has_key(key)

Dict 조작

dict object 생성

```
>>> d = {"Apple": "Red", "Lemon": "Yellow"}
```

key 를 이용해서 Value 가져오기

```
>>> d["Apple"] 'Red'
```

dict object 에 새 요소 추가

```
>>> d["Grape"] = "Purple"
```

```
>>> d
```

```
{'Grape': 'Purple', 'Lemon': 'Yellow', 'Apple': 'Red'}
```

dict object 에서 요소 제거

```
>>> del d["Apple"]
```

```
>>> d
```

```
{'Grape': 'Purple', 'Lemon': 'Yellow'}
```

None

- 아무것이 아님을 명시하는 특수한 type
- C, Java 의 NULL 과 비슷하지만 다름
 - 함수의 return을 명시하지 않으면 None을 리턴

함수 작성 1

- 인자, 반환 없는 형태
- 반환을 명시하지 않으면 **None** 을 반환

```
def 함수명():  
    코드
```

```
def print_hello():  
    print("hello")
```

```
print_hello()  
=> hello
```

함수 작성2

- 인수, 반환 있는 경우
- 인수는 함수명 뒤()에 필요한 만큼 지정
- 반환값은 return에 작성

```
def 함수명(인수):  
    코드  
    return 반환값
```

```
def adder(a, b):  
    return a + b  
  
print(add(3,4))  
=> 7
```

인수의 default 식

- 지정하지 않는 경우 설정되는 인수

```
def func(a, b=1):  
    print(a)  
    print(b)
```

```
func(5,6)
```

```
=> 5
```

```
    6
```

```
func(5)
```

```
=> 5
```

```
    1
```

import

- import 해서 모듈을 사용

```
def add(a,b):  
    return a + b  
  
def minus(a, b):  
    return a - b  
  
def multiply(a, b):  
    return a * b  
  
def divide(a, b):  
    return a / b
```

```
import mymath  
  
print(mymath.add(2,3)) # -> 5  
print(mymath.minus(2,3)) # -> -1  
print(mymath.multiply(2,3)) # -> 6  
print(divide(2,3)) #ERROR HAPPEN  
  
Traceback (most recent call last):  
  File "</main.py>", line 6, in <module>  
    print(divide(2,3))  
NameError: name 'divide' is not defined
```

from

- **from 파일명(패키지명) import ***
파일명 지정 없이도 가능

```
def add(a,b):  
    return a + b  
  
def minus(a, b):  
    return a -- b  
  
def multiply(a, b):  
    return a * b  
  
def divide(a, b):  
    return a / b
```

```
from mymath import *  
  
print(add(2,3)) # --> 5  
print(minus(2,3)) # --> --1  
print(multiply(2,3)) # --> 6  
print(mymath.divide(2,3)) #ERROR HAPPEN  
  
Traceback (most recent call last):  
  File "<~/main.py>", line 6, in <module>  
    print(divide(2,3))  
NameError: name 'mymath' is not defined
```

class와 instance 관계

- class: instance를 작성하는 템플릿
- 인스턴스: 클래스로 만든 object



type (Class) String

value (Instance) "Hello Python", "Hello Cisco"

클래스 선언

- 클래스 선언 : **class** 클래스이름:
- 메서드 선언 : 제 1인자로 **self** 를 정의
- 속성에 접근할때는 **self.속성**

```
class MyClass:  
    string = ""  
  
    def set_string(self, string):  
        self.string = string  
  
    def get_string(self):  
        return self.string  
  
    def double(self):  
        self.string *= 2
```

class 선언
class 가 저장할 데이터 선언
메서드 선언
첫번째 인수를 self 로 하는게 python 규칙

속성에 access
self.속성 으로 접근
(그렇지 않으면 로컬 변수를 사용하게 됨)

Chap.2.2 파이썬 Numpy

Arrays – Numerical Python (Numpy)

1차원 데이터를 List로 저장해서 사용해도 좋지만...

```
>>> a = [1, 3, 5, 7, 9]
>>> print(a[2:4])
[5, 7]
>>> b = [[1, 3, 5, 7, 9], [2, 4, 6, 8, 10]]
>>> print(b[0])
[1, 3, 5, 7, 9]
>>> print(b[1][2:4])
[6, 8]
```

```
>>> a = [1, 3, 5, 7, 9]
>>> b = [3, 5, 6, 7, 9]
>>> c = a + b
>>> print c
[1, 3, 5, 7, 9, 3, 5, 6, 7, 9]
```

- But, 직접 수학 연산을 할 수 없다 (+, -, *, /, ...)
- 다차원 배열을 효율적으로 다룰 도구가 필요하다
- **Numpy**

```
>>> import numpy
```
- 리스트와 비슷하지만 성능과 호환성이 좋다

Numpy – Creating vectors

From lists numpy.array

```
# as vectors from lists
>>> a = numpy.array([1,3,5,7,9])
>>> b = numpy.array([3,5,6,7,9])
>>> c = a + b
>>> print c
[4, 8, 11, 14, 18]

>>> type(c)
<type 'numpy.ndarray'>

>>> c.shape
(5,)
```

List로 생성

list, list comprehension으로 ndarray 객체 인스턴스 생성

```
import numpy as np

# integer array:
i = np.array([1, 4, 2, 5, 3])
print(i)
print(i.dtype)

# float array:
f = np.array([3.14, 4, 2, 3])
print(f)
print(f.dtype)

# nested lists result in multi-dimensional arrays
mi = np.array([range(i, i + 3) for i in [2, 4, 6]])
print(mi)

[1 4 2 5 3]
int32
[ 3.14  4.     2.     3.   ]
float64
[[2 3 4]
 [4 5 6]
 [6 7 8]]
```

zeros/ones/full함수로 생성

zeros/ones/full함수로 ndarray 객체 인스턴스
생성

```
# Create a length-10 integer array filled with zeros
ze = np.zeros(10, dtype=int)
print(ze)
# Create a 3x5 floating-point array filled with ones
one = np.ones((3, 5), dtype=float)
print(one)
# Create a 3x5 array filled with 3.14
f1 = np.full((3, 5), 3.14)
print(f1)
```

```
[0 0 0 0 0 0 0 0 0 0]
[[ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]]
[[ 3.14  3.14  3.14  3.14  3.14]
 [ 3.14  3.14  3.14  3.14  3.14]
 [ 3.14  3.14  3.14  3.14  3.14]]
```

eye/empty 함수로 생성

eye/empty 함수로 ndarray 객체 인스턴스 생성

```
import numpy as np

# Create a 3x3 identity matrix
ey = np.eye(3)
print(ey)

# Create an uninitialized array of three integers
# The values will be whatever happens to already exist at that memory location
em = np.empty(3)
print(em)
```

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[1. 1. 1.]

arange/linspace함수로 생성

arange/linspace함수로 ndarray 객체 인스턴스 생성

```
import numpy as np  
|  
# Create an array filled with a Linear sequence  
# Starting at 0, ending at 20, stepping by 2  
# (this is similar to the built-in range() function)  
ar = np.arange(0, 20, 2)  
print(ar)  
  
# Create an array of five values evenly spaced between 0 and 1  
ls = np.linspace(0, 1, 5)  
print(ls)  
  
[ 0  2  4  6  8 10 12 14 16 18]  
[ 0.    0.25  0.5   0.75  1.   ]
```

random 모듈의 함수로 생성

random 모듈의 함수로 ndarray 객체 인스턴스 생성

```
import numpy as np

# Create a 3x3 array of uniformly distributed
# random values between 0 and 1
rr = np.random.random((3, 3))
print(rr)

# Create a 3x3 array of normally distributed random values
# with mean 0 and standard deviation 1
rn = np.random.normal(0, 1, (3, 3))
print(rn)

# Create a 3x3 array of random integers in the interval [0, 10)
ri = np.random.randint(0, 10, (3, 3))
print(ri)

[[ 0.88050071  0.79042114  0.81195593]
 [ 0.10219646  0.477742   0.71939974]
 [ 0.29968495  0.31610957  0.42161167]]
[[ 0.27567609 -0.00974696  0.35981263]
 [ 1.55050238  0.26103931 -0.06907681]
 [-0.24800385 -1.48016813 -0.45919365]]
[[3 2 4]
 [8 1 1]
 [7 2 6]]
```

ndarray 생성시 data type 지정

ndarray 객체 인스턴스 생성시 데이터 타입을
 문자열이나 np.int16 처럼 지정 가능

```
import numpy as np

z1 =np.zeros(10, dtype='int16')
print(z1)
print(z1.dtype)
```

```
z2 =np.zeros(10, dtype=np.int16)
print(z2)
print(z2.dtype)
```

```
[0 0 0 0 0 0 0 0 0 0]
int16
[0 0 0 0 0 0 0 0 0 0]
int16
```

ndarray 생성시 data type 1

ndarray 객체 인스턴스 생성시 데이터 타입

Data type	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C long; normally either int64 or int32)
intc	Identical to C int (normally int32 or int64)
intp	Integer used for indexing (same as C ssize_t; normally either int32 or int64)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)

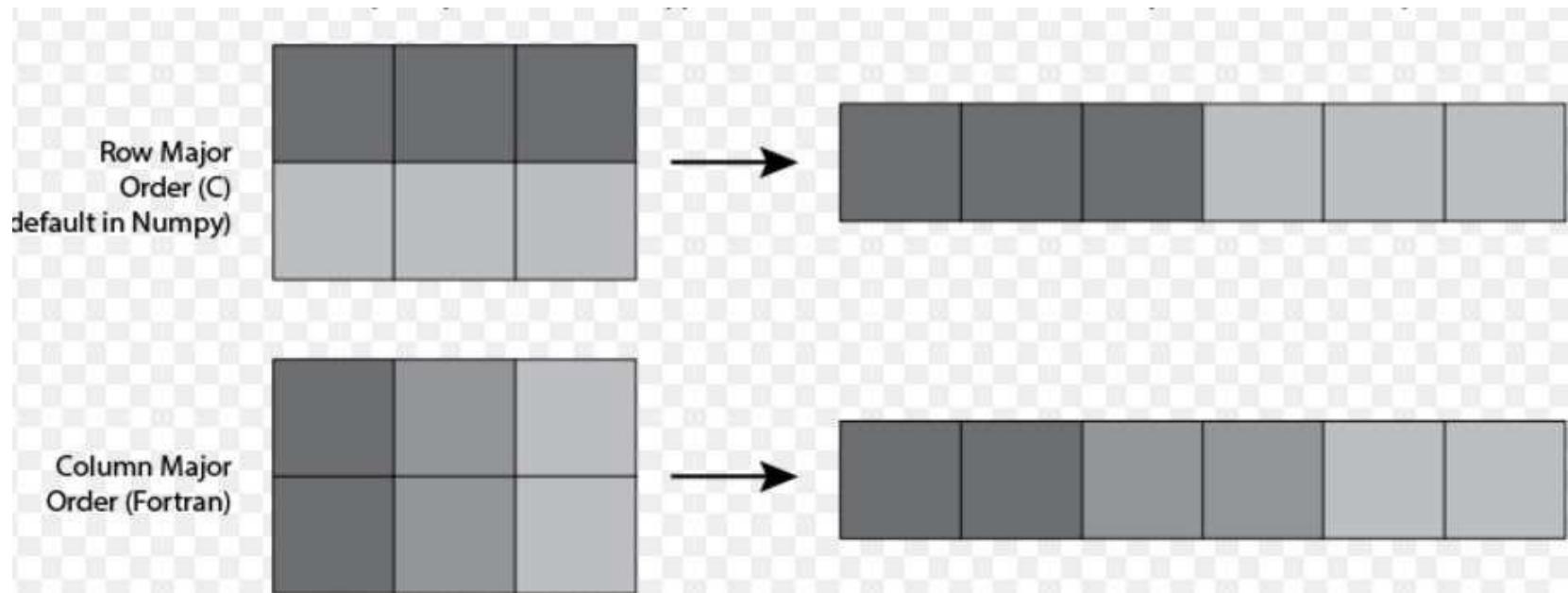
ndarray 생성시 data type 2

ndarray 객체 인스턴스 생성시 데이터 타입

Data type	Description
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for float64.
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for complex128.
complex64	Complex number, represented by two 32-bit floats
complex128	Complex number, represented by two 64-bit floats

ndarray 배열의 구성

ndarray 객체의 배열의 구성은 평면 배열로 구성하면 기본이 행단위로 분리되고 Fortran 언어 스타일일 경우 열단위로 구성



ndarray : 1 차원

ndarray 객체 인스턴스 생성된 1차원 속성

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x1 = np.random.randint(10, size=6) # One-dimensional array

print(x1.flatten())
print("x1 차원 : ", x1.ndim)
print("x1 형태 : ", x1.shape)
print("x1 size: : ", x1.size)
print("x1 dtype : ", x1.dtype)
print("itemsize:", x1.itemsize, "bytes")
print("nbytes:", x1.nbytes, "bytes")
```

```
[5 0 3 3 7 9]
x1 차원 : 1
x1 형태 : (6,)
x1 size: : 6
x1 dtype : int32
itemsize: 4 bytes
nbytes: 24 bytes
```

ndarray : 2차원

ndarray 객체 인스턴스 생성된 2차원 속성

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array

print(x2.flatten())
print("x2 차원 : ", x2.ndim)
print("x2 형태 : ", x2.shape)
print("x2 size: : ", x2.size)
print("x2 dtype : ", x2.dtype)
print("itemsize:", x2.itemsize, "bytes")
print(" nbytes:", x2.nbytes, "bytes")
```

```
[5 0 3 3 7 9 3 5 2 4 7 6]
x2 차원 : 2
x2 형태 : (3, 4)
x2 size: : 12
x2 dtype : int32
itemsize: 4 bytes
nbytes: 48 bytes
```

ndarray : 3차원

ndarray 객체 인스턴스 생성된 3차원 속성

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print(x3.flatten())
print("x3 차원 : ", x3.ndim)
print("x3 형태 : ", x3.shape)
print("x3 size: : ", x3.size)
print("x3 dtype : ", x3.dtype)
print("itemsize:", x3.itemsize, "bytes")
print(" nbytes:", x3.nbytes, "bytes")
```

[5 0 3 3 7 9 3 5 2 4 7 6 8 8 1 6 7 7 8 1 5 9 8 9 4 3 0 3 5 0 2 3 8 1 3 3 3
7 0 1 9 9 0 4 7 3 2 7 2 0 0 4 5 5 6 8 4 1 4 9]
x3 차원 : 3
x3 형태 : (3, 4, 5)
x3 size: : 60
x3 dtype : int32
itemsize: 4 bytes
nbytes: 240 bytes

원소 읽기(indexing)

ndarray 객체 인스턴스 생성된 차원에 따라 원소 읽기

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x1 = np.random.randint(6, size=6) # One-dimensional array
x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
x3 = np.random.randint(12, size=(3, 2, 2)) # Three-dimensional array

print(x1[0])
print(x2[0])
print(x3[0])

print(x1[-1])
print(x2[-1])
print(x3[-1])
```

4
[7 1]
[[7 6]
 [8 8]]
3
[2 4]
[[7 8]
 [1 5]]

subarrays 읽기(slicing)

ndarray 객체 인스턴스 생성된 차원에 따라 동일 탑입의 subarray를 읽기

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x1 = np.random.randint(6, size=6) # One-dimensional array
x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
x3 = np.random.randint(12, size=(3, 2, 2)) # Three-dimensional array

print(x1[:2])
print(x2[:2])
print(x3[:2])

print(x1[-1:])
print(x2[-1:])
print(x3[-1:])

[[4 5]
 [[7 1]
 [3 5]]
 [[[ 7   6]
 [ 8   8]]]

 [[[10  1]
 [ 6  7]]]
 [3]
 [[[2 4]]
 [[[7 8]
 [1 5]]]]
```

slices separated by commas

ndarray 객체 인스턴스 생성된 차원에 따라
commas로 분리해서 읽기

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array

print(x2)
print(x2[1, 1]) # 두번째 행, 두번째 열 원소 검색
print(x2[1, :]) # 두번째 행
print(x2[::-1, ::-1]) # 역으로 출력
```

```
[[4 7]
 [5 0]
 [3 3]]
0
[5 0]
[[3 3]
 [0 5]
 [7 4]]
```

ndarray는 기본 view 제공

ndarray는 기본 view만 제공하므로 조회 후 변경시 원본 배열도 갱신됨

```
import numpy as np

x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
print(x2)
print(" slice 배열 조회")
x2_sub = x2[:2, :2]
print(x2_sub)
print(" slice 배열 변경")
x2_sub[0, 0] = 99
print(x2_sub)
print("원본 배열 ")
print(x2)
```

```
[[4 7]
 [6 0]
 [0 4]]
slice 배열 조회
[[4 7]
 [6 0]]
slice 배열 변경
[[99  7]
 [ 6  0]]
원본 배열
[[99  7]
 [ 6  0]
 [ 0  4]]
```

ndarray 조회 후 copy 사용

ndarray는 기본 view만 제공하므로 원본을 유지
하려면 copy해서 사용해야 함

```
import numpy as np

x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
print(x2)
print(" slice 배열 조회 및 복사 ")
x2_sub_copy = x2[:2, :2].copy()
print(x2_sub_copy)
print(" slice 배열 변경")
x2_sub_copy[0, 0] = 42
print(x2_sub_copy)
print("원본 배열 ")
print(x2)
```

```
[[0 1]
 [5 1]
 [5 0]]
slice 배열 조회 및 복사
[[0 1]
 [5 1]]
slice 배열 변경
[[42  1]
 [ 5  1]]
원본 배열
[[0 1]
 [5 1]
 [5 0]]
```

reshape

배열을 바꾸면 새로운 배열이 만들어짐

```
import numpy as np

#reshape 메소드 처리
grid = np.arange(1, 10).reshape((3, 3))
print(grid)

#reshape 함수 처리
```

```
x = np.array([1, 2, 3, 4, 5, 6])
xx = np.reshape(x, (2, 3))
print(xx)
```

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

newaxis

기존 배열에 축을 추가해서 새로운 배열이 만들기

```
import numpy as np

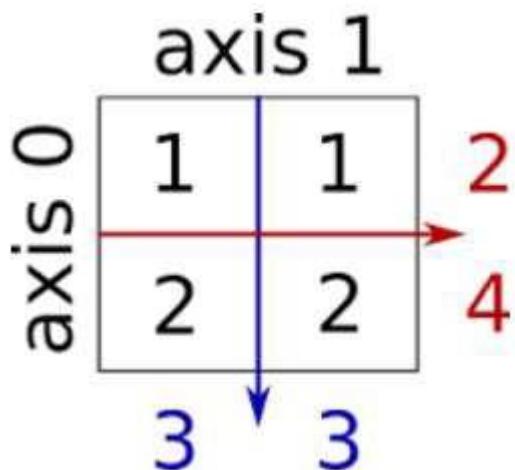
# reshape 메소드 처리
x = np.arange(0, 3)
# row vector via newaxis
rx = x[np.newaxis, :]
print(rx)

# column vector via reshape
xs = x.reshape((3, 1))
print(xs)
xc = x[:, np.newaxis]
print(xc)

[[0 1 2]]
[[0]
 [1]
 [2]]
[[0]
 [1]
 [2]]
```

concatenate

배열을 행 또는 열 축으로 통합 axis=0이면 행
이 추가, axis= 1이면 열로 추가



```
import numpy as np

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

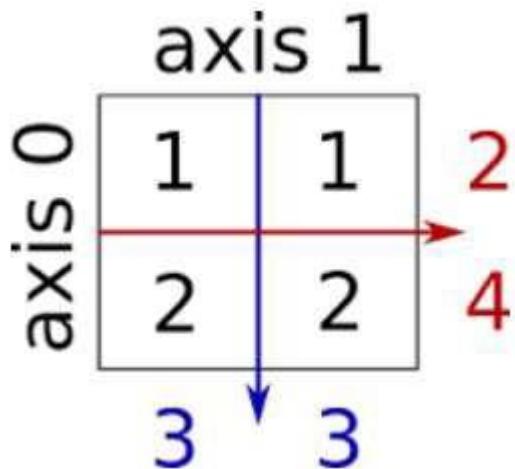
# concatenate along the first axis
ax0 = np.concatenate([grid, grid], axis=0)
print(ax0)

# concatenate along the second axis (zero-indexed)
ax1 = np.concatenate([grid, grid], axis=1)
print(ax1)
```

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

vstack/hstack

배열을 행 또는 열 축으로 통합 vstack이면 행
이 추가, hstack이면 열로 추가



```
import numpy as np
grid = np.array([[1, 2, 3], [4, 5, 6]])
```

```
# vertically stack the arrays
av = np.vstack([grid, grid])
print(av)
```

```
# horizontally stack the arrays
```

```
ah = np.hstack([grid, grid])
print(ah)
```

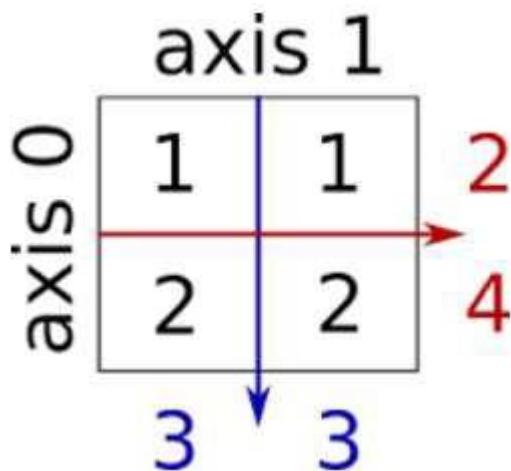
```
[[1 2 3]
 [4 5 6]
 [1 2 3]
 [4 5 6]]
 [[1 2 3 1 2 3]
 [4 5 6 4 5 6]]
```



배열 행/열 축으로 분리

split

배열을 행 또는 열 축으로 통합 axis=0이면 행
분리, axis= 1이면 열 분리



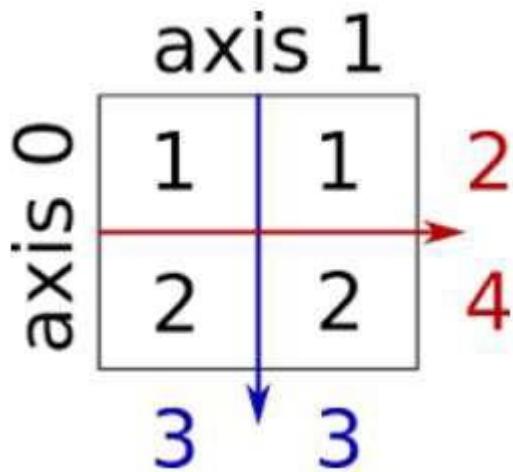
```
import numpy as np

x = [1, 2, 3, 99, 99, 3, 2, 1]
x1, x2, x3 = np.split(x, [3,5])
print(x1, x2, x3)
x = [[1, 2, 3, 99,100, 88,99], [99, 3, 2, 1,100,88,99]]
x1, x2, x3, x4 = np.split(x, [2,5,7], axis=1)
print(x1)
print(x2)
print(x3)
print(x4)
```

```
[1 2 3] [99 99] [3 2 1]
[[ 1  2]
 [99  3]]
[[ 3  99 100]
 [ 2   1 100]]
[[88 99]
 [88 99]]
[]
```

vsplit/hsplit

배열을 행 또는 열 축으로 통합 vsplit이면 행 분리, hsplit이면 열로 분리



```
import numpy as np

x = [1, 2, 3, 99, 99, 3, 2, 1]

xa = np.array(x).reshape(4,2)
upper, lower = np.vsplit(xa, [2])
print(upper)
print(lower)

left, right = np.hsplit(xa , [1])
print(left)
print(right)
```

```
[[ 1  2]
 [ 3 99]]
[[99  3]
 [ 2  1]]
[[ 1]
 [ 3]
 [99]
 [ 2]]
[[ 2]
 [99]
 [ 3]
 [ 1]]
```

연습(reduce)

- 2개의 학급이 시험을 보았습니다.
 - 1번째 반의 성적 [10,30,50,70]
 - 2번째 반의 성적 [30,40,50,60]
- 반별 성적의 평균을 구하세요 (mean 사용)
- 어느 반의 평균성적이 더 높은지 구하세요 (argmax 사용)

Array Slicing

파이썬의 일반적인 슬라이싱

```
>>> a[0,3:5]  
array([3, 4])
```

```
>>> a[4:,4:]  
array([[44, 45],  
      [54, 55]])
```

```
>>> a[:,2]  
array([2,12,22,32,42,52])
```

간격을 지정하기

```
>>> a[2::2,::2]  
array([[20, 22, 24],  
      [40, 42, 44]])
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

Array Broadcasting

4x3

0	1	2
0	1	2
0	1	2
0	1	2

+

4x3

0	0	0
10	10	10
20	20	20
30	30	30

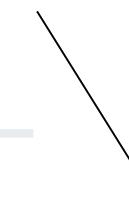
=

0	1	2
0	1	2
0	1	2
0	1	2

+

0	0	0
10	10	10
20	20	20
30	30	30

=



4x3

0	0	0
10	10	10
20	20	20
30	30	30

+

3

0	1	2

=

0	0	0
10	10	10
20	20	20
30	30	30

+

0	1	2
0	1	2
0	1	2
0	1	2

=

0	1	2
10	11	12
20	21	22
30	31	32

stretch

4x1

0
10
20
30

+

3

0	1	2

=

0	0	0
10	10	10
20	20	20
30	30	30

+

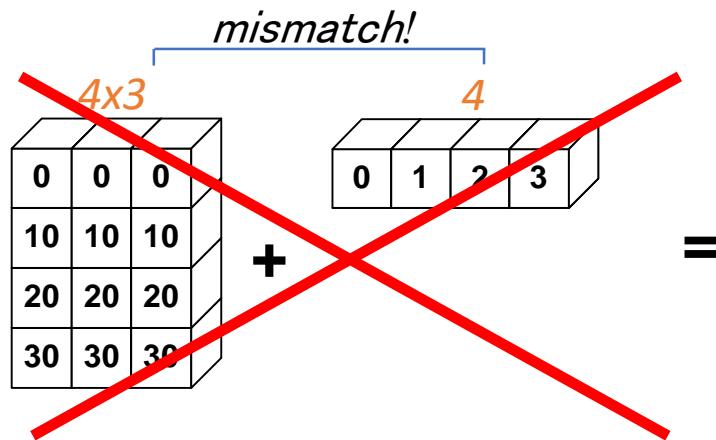
0	1	2
0	1	2
0	1	2
0	1	2

=

*stretch**stretch*

Broadcasting Rules

반드시 모든 축의 길이가 서로 같거나 1이어야 합니다. 그렇지 않으면
“`ValueError: frames are not aligned`” exception



과정 소개 - 2일차

- Interactive tool “Jupyter Notebook”
- 데이터 획득, 분석, 시각화, 전처리

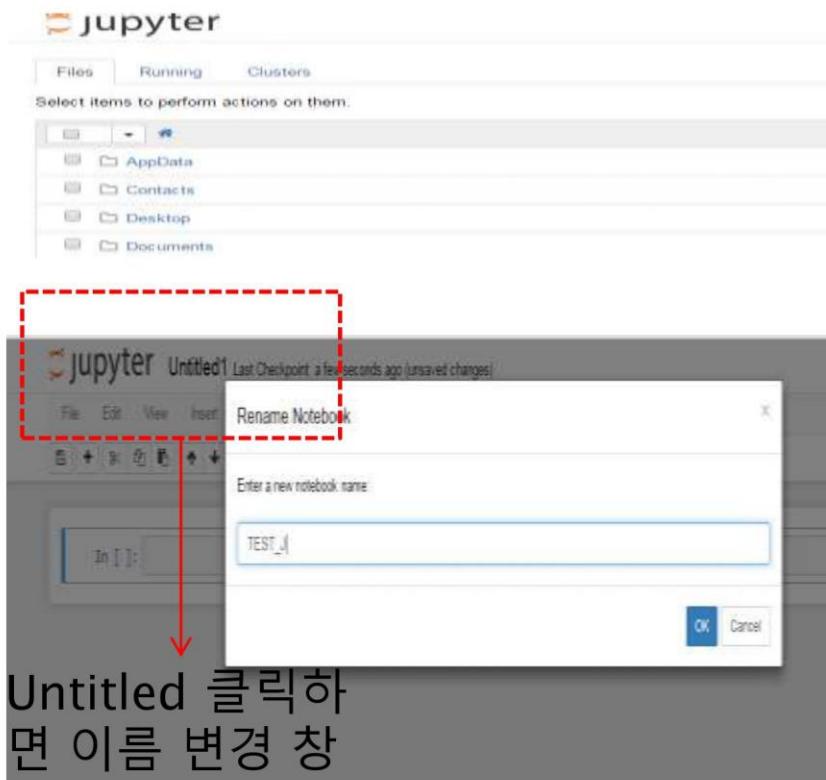


- Jupyter Notebook
- CSV, Excel Data Load
- Pandas 데이터 분석, 시각화, 전처리

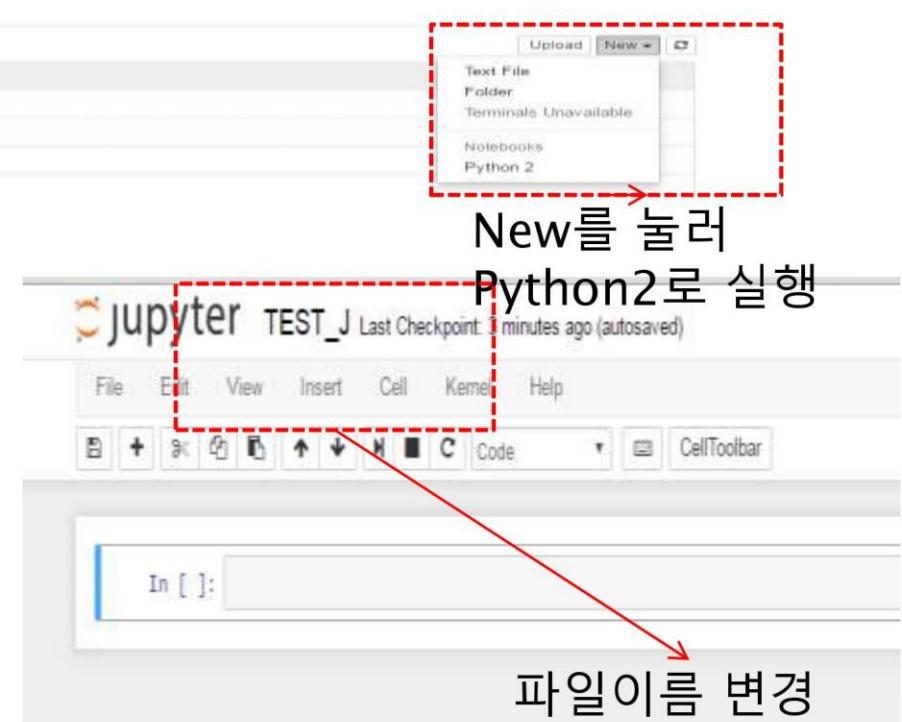
Chap.2.3 Jupyter Notebook

Jupyter notebook 커널 실행

jupyter notebook 커널 선택하여 실행



Untitled 클릭하면
이름 변경 창
이 나오고 이름
변경

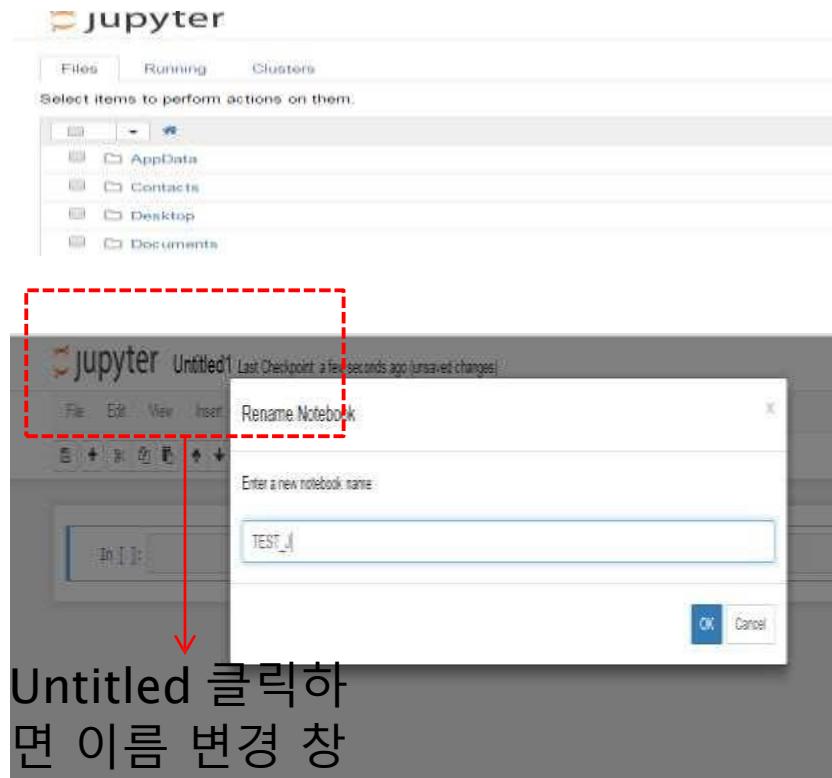


New를 눌러
Python2로 실행

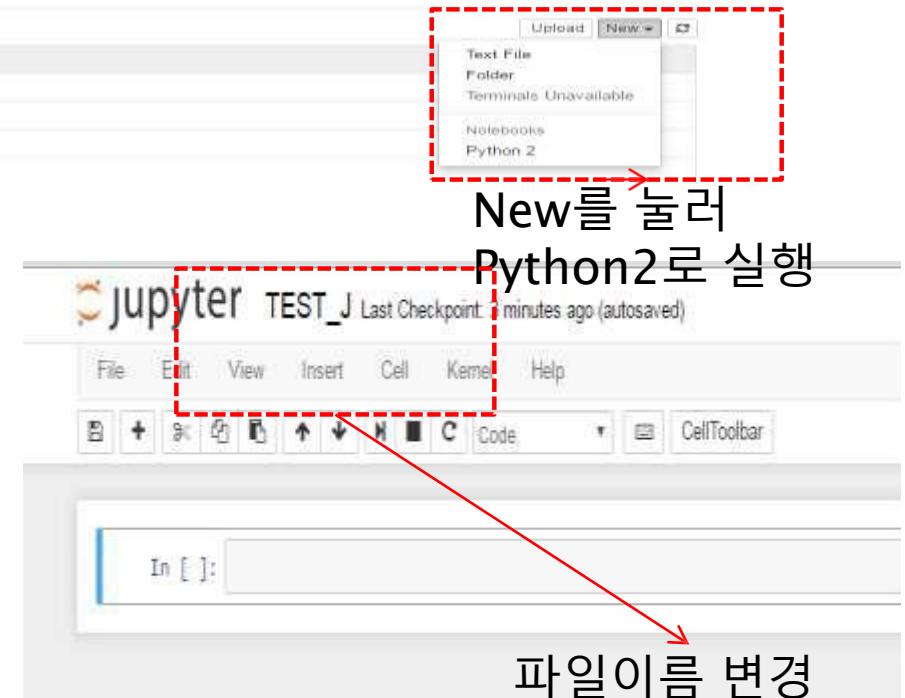
파일이름 변경

Jupyter notebook 커널 실행

jupyter notebook 커널 선택하여 실행



Untitled 클릭하면
이름 변경 창
이 나오고 이름
변경

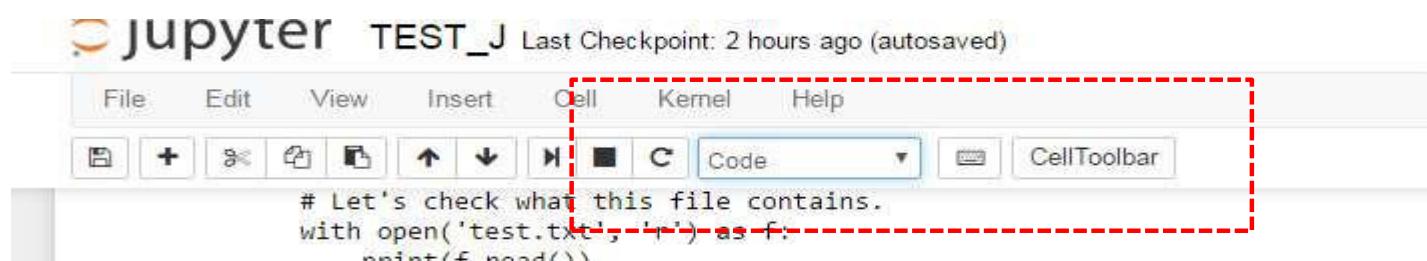


New를 눌러
Python2로 실행

파일이름 변경

Cell type : code

Cell에 Python 코드가 입력되어 실행

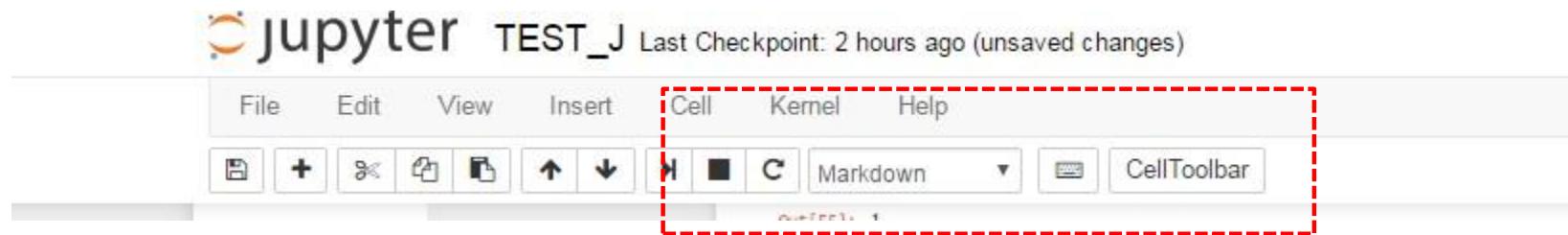


The screenshot shows a Jupyter Notebook interface with the title "jupyter TEST_J Last Checkpoint: 2 hours ago (autosaved)". The menu bar includes File, Edit, View, Insert, Cell (which is highlighted), Kernel, and Help. Below the menu is a toolbar with various icons. A red dashed box highlights the "Cell" menu item and the "Cell" icon in the toolbar. The main area contains a code cell with the following Python code:

```
# Let's check what this file contains.  
with open('test.txt', 'r') as f:  
    print(f.read())
```

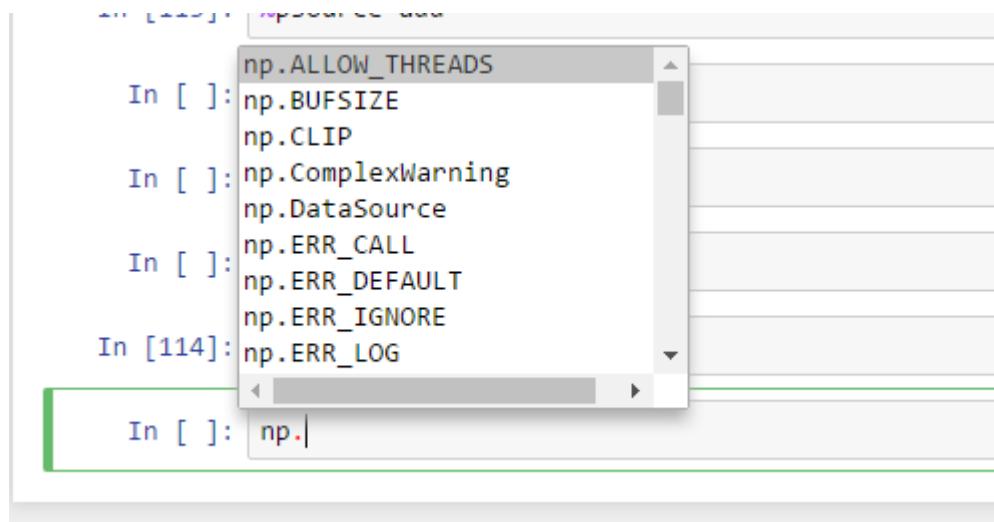
Cell type : markdown

Cell에 markdown에 대한 표기법으로 수학식이나 문서 등을 작성



자동완성: tab

ipython 처럼 입력하고 tab 키를 누르면 내부에 있는 요소들을 보여주므로 선택해서 사용 가능



객체 정보 조회 : shift+TAB

변수를 키인하고 shift+TAB을 누르면 내부 특성이 조회 됨

```
import numpy as np

x = [1, 2, 3, 99, 99, 3, 2, 1]
x
```

Type: list
String form: [1, 2, 3, 99, 99, 3, 2, 1]
Length: 8
Docstring:

함수 정보 조회 : shift+TAB

함수를 키인하고 shift+TAB을 누르면 내부 특성이 조회 됨

```
def add(x:int, y:int) ->int :  
    return x+y  
  
add  
  
Signature: add(x:int, y:int) -> int  
Docstring: <no docstring>  
File:      c:\users\06411\documents\<ipython-input-820-5bfdc7793909>  
Type:      function
```

객체 정보 조회 : ?

ipython 처럼 입력한 후 ?를 붙이고 실행시키면
내부 정보가 보임

The screenshot shows a Jupyter Notebook interface. In the top cell (In [115]), the user has typed `list?`. In the bottom cell (In []), the user has started typing `In []:`. Below the cells, the docstring for the `list` constructor is displayed:

```
Docstring:  
list() -> new empty list  
list(iterable) -> new list initialized from iterable's items  
Type: type
```

객체 정보 조회 : ??

정의된 객체에 다음에 ??를 이용하면 help정보
(소스)가 나옴

```
In [117]: add??  
In [ ]:  
  
Signature: add(x, y)  
Source:  
def add(x,y) :  
    return x+y  
  
File:      c:\users\06411\test\<ipython-input-111-b603eada704c>  
Type:      function
```

단축키:ctrl+enter

cell에 표현식을 입력하고 ctrl+ enter를 치면 현재 cell이 실행되고 다음 cell이 활성화 되지 않음



```
In [160]: c = 10
In [161]: print c
           10
```

단축키:shift+enter

cell에 표현식을 입력하고 shift+enter(Alt-Enter)를치면 다음 셀이 활성화

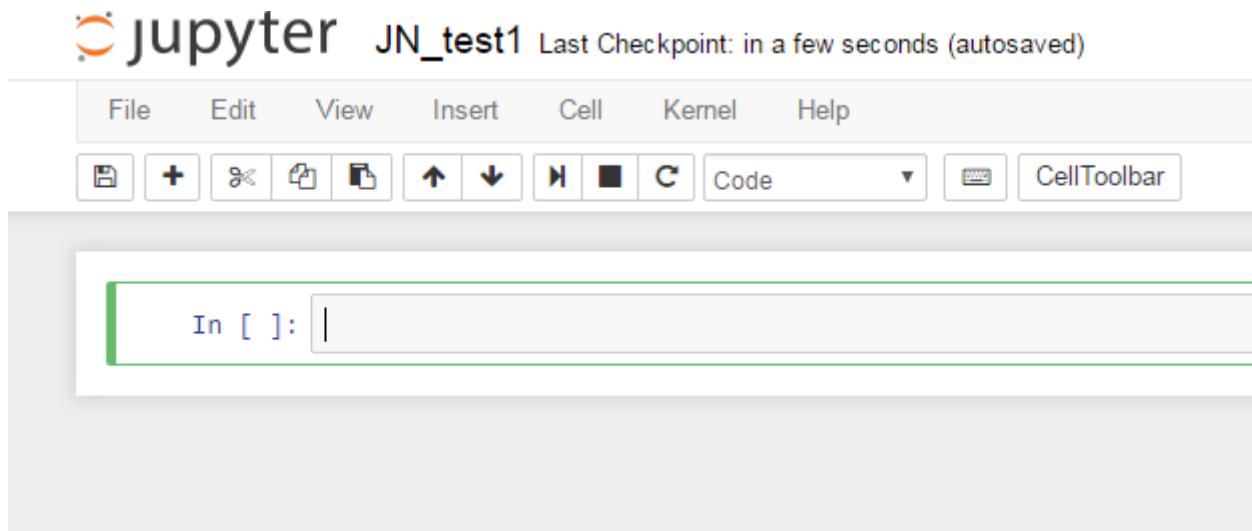
The image shows a Jupyter Notebook interface with three cells:

- In [155]:** `a=10`
- In [156]:** `print a`
10
- In []:** (empty)

The third cell is highlighted with a blue border, indicating it is the active cell.

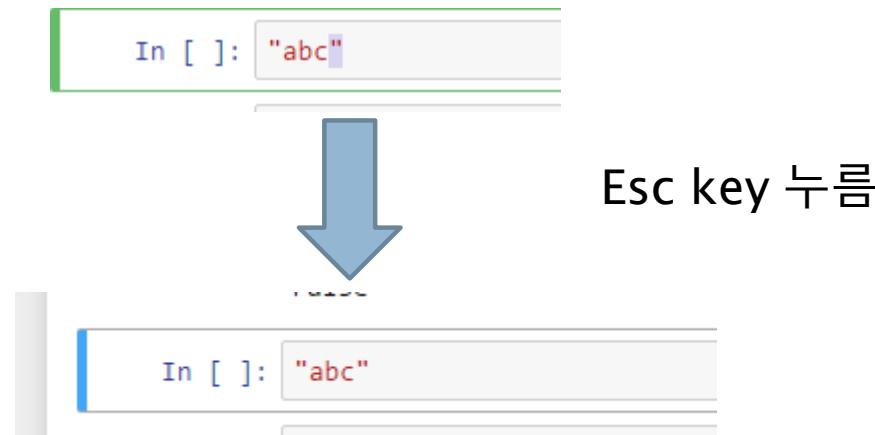
단축키:ctrl+S

현재 jupyter notebook에 저장되었다는 표시가
남음



단축키: Esc

편집모드에서 cell을 명령모드로 변경
초록색에서 파란색으로 변경





```
%matplotlib notebook
```

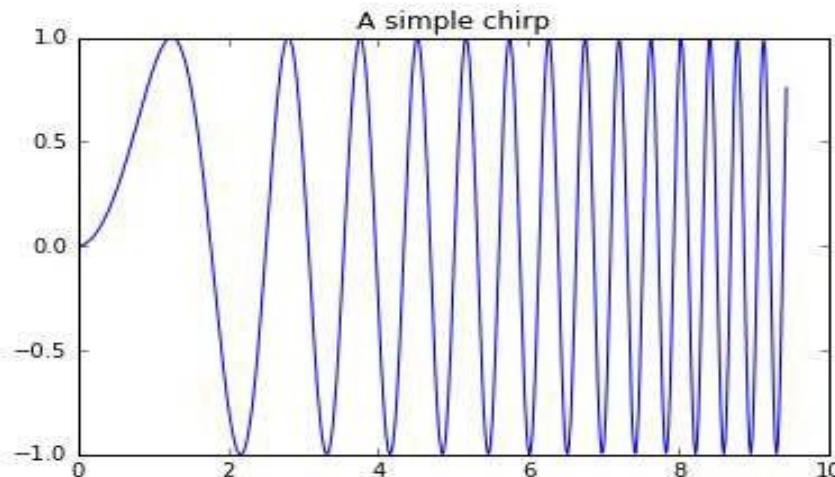
Matplotlib 사용

%matplotlib inline 을 실행한 후에 코딩해서 실행하면 matplotlib이 처리된 결과가 출력됨

```
In [18]: %matplotlib inline
```

```
In [19]: import matplotlib.pyplot as plt
import numpy as np

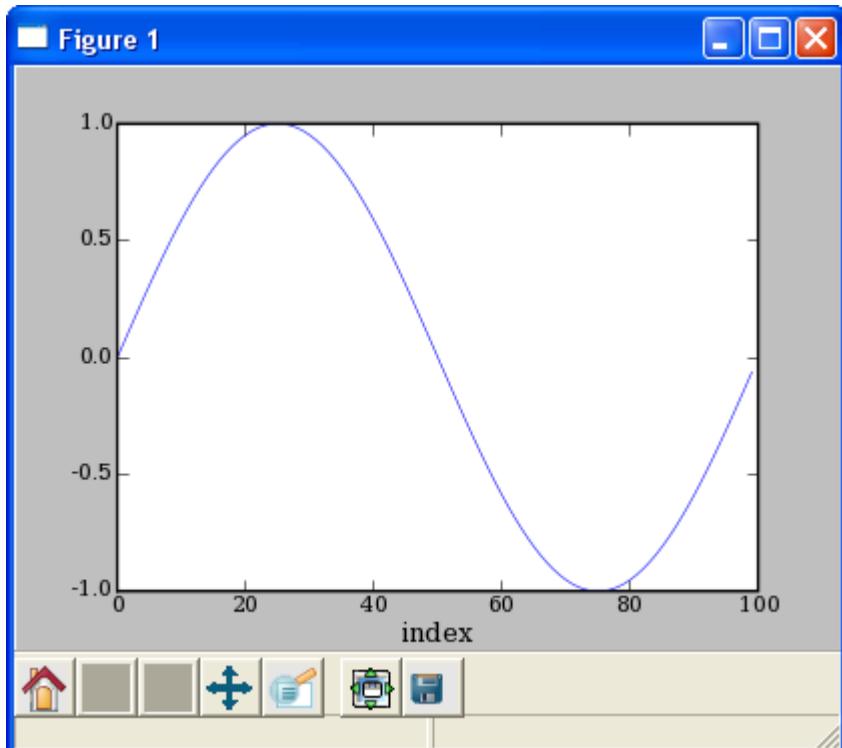
x = np.linspace(0, 3*np.pi, 500)
plt.plot(x, np.sin(x**2))
plt.title('A simple chirp');
```



Line Plots

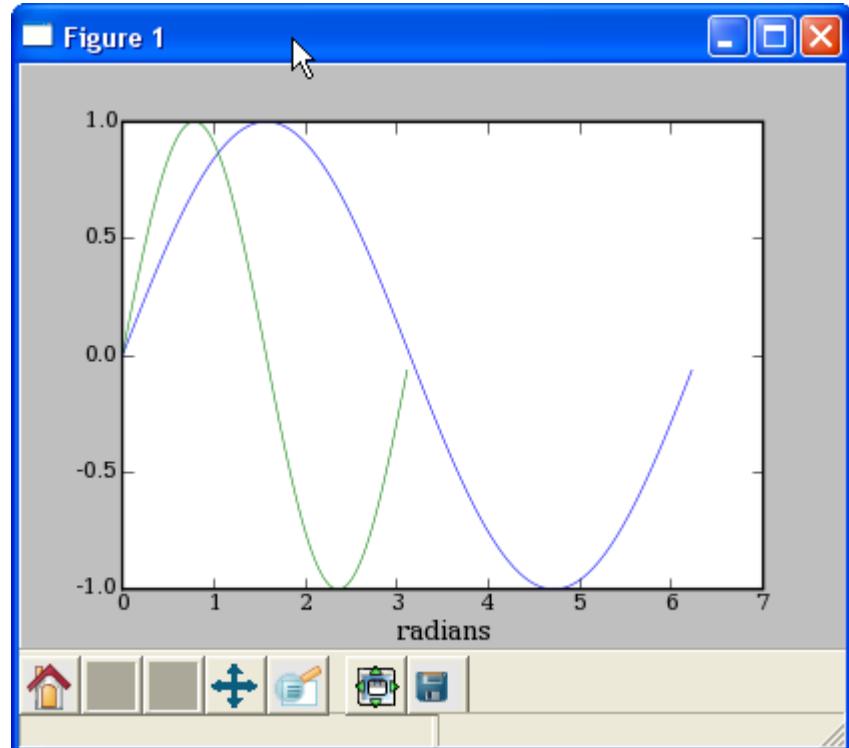
PLOT AGAINST INDICES

```
>>> x = arange(50)*2*pi/50.  
>>> y = sin(x)  
>>> plot(y)  
>>> xlabel('index')
```



MULTIPLE DATA SETS

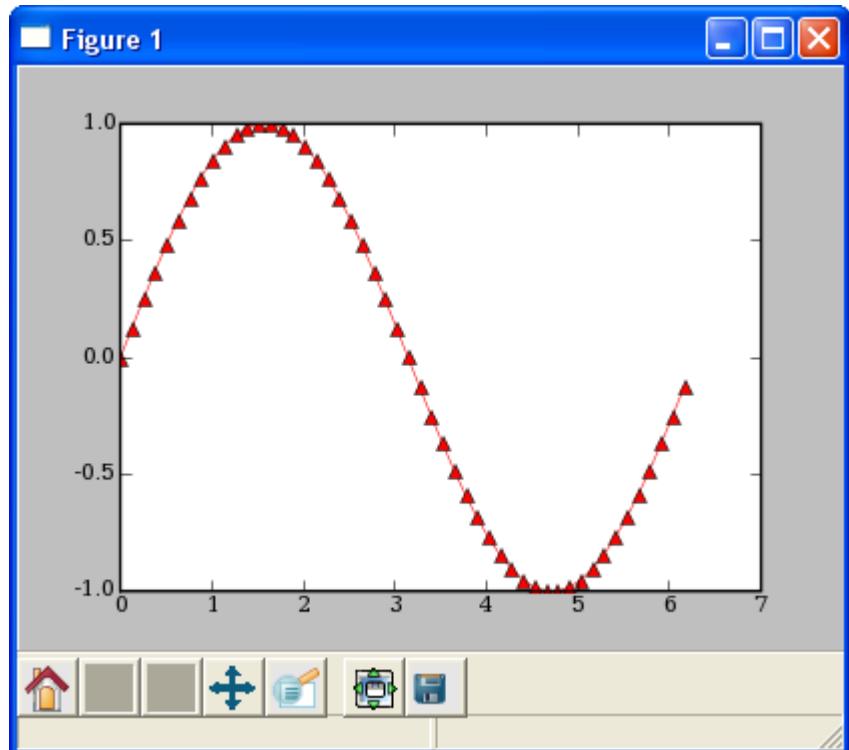
```
>>> plot(x,y,x2,y2)  
>>> xlabel('radians')
```



Line Plots

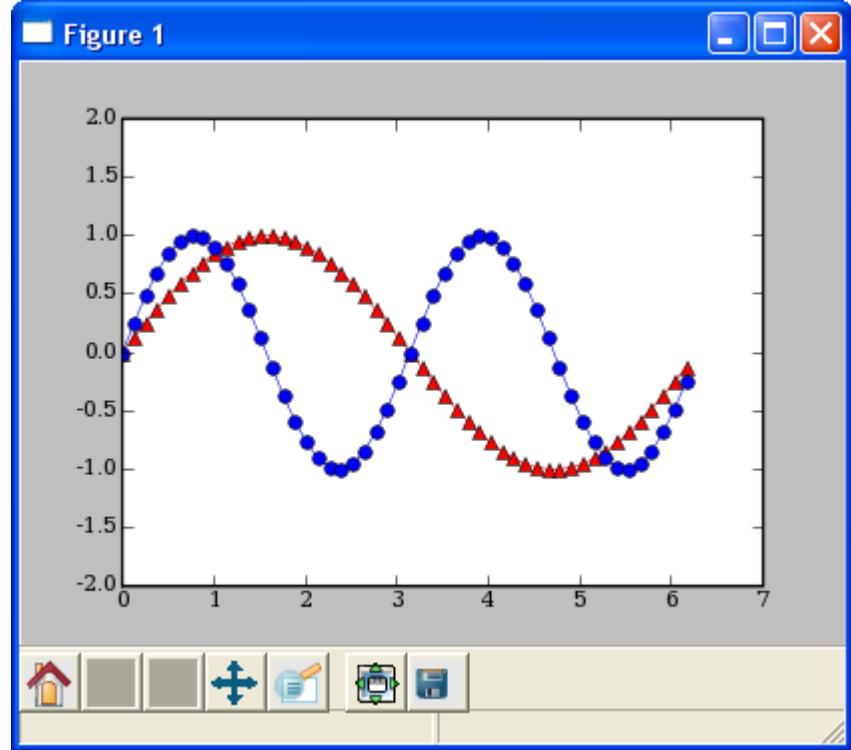
LINE FORMATTING

```
# red, dot-dash, triangles  
>>> plot(x,sin(x),'r-^')
```



MULTIPLE PLOT GROUPS

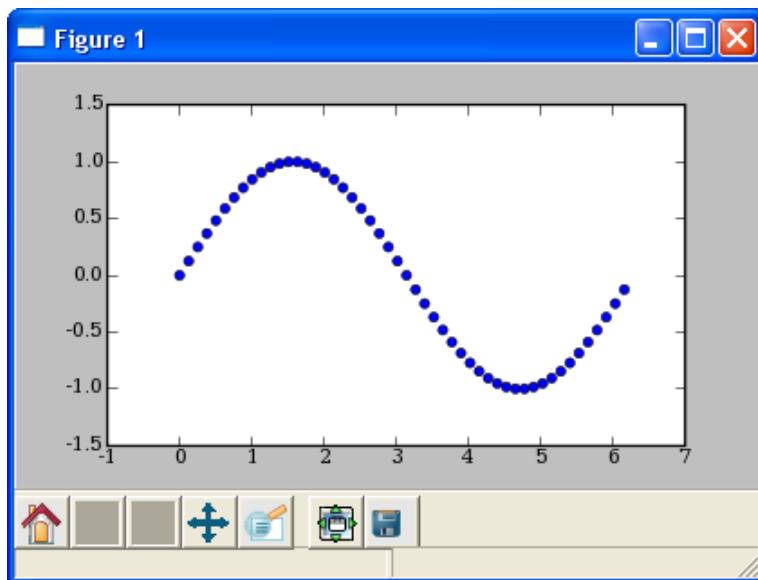
```
>>> plot(x,y1,'b-o', x,y2), r-^')  
>>> axis([0,7,-2,2])
```



Scatter Plots

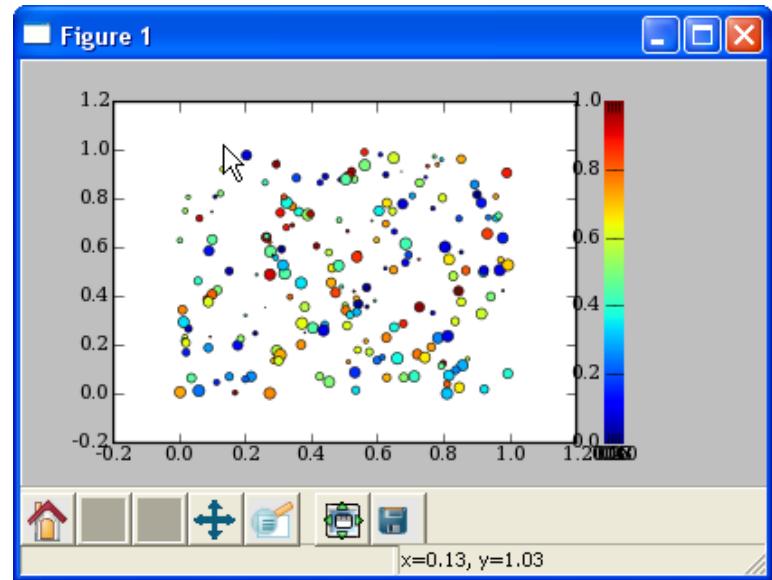
SIMPLE SCATTER PLOT

```
>>> x = arange(50)*2*pi/50.  
>>> y = sin(x)  
>>> scatter(x,y)
```



COLORMAPPED SCATTER

```
# marker size/color set with data  
>>> x = rand(200)  
>>> y = rand(200)  
>>> size = rand(200)*30  
>>> color = rand(200)  
>>> scatter(x, y, size, color)  
>>> colorbar()
```



Bar Plots

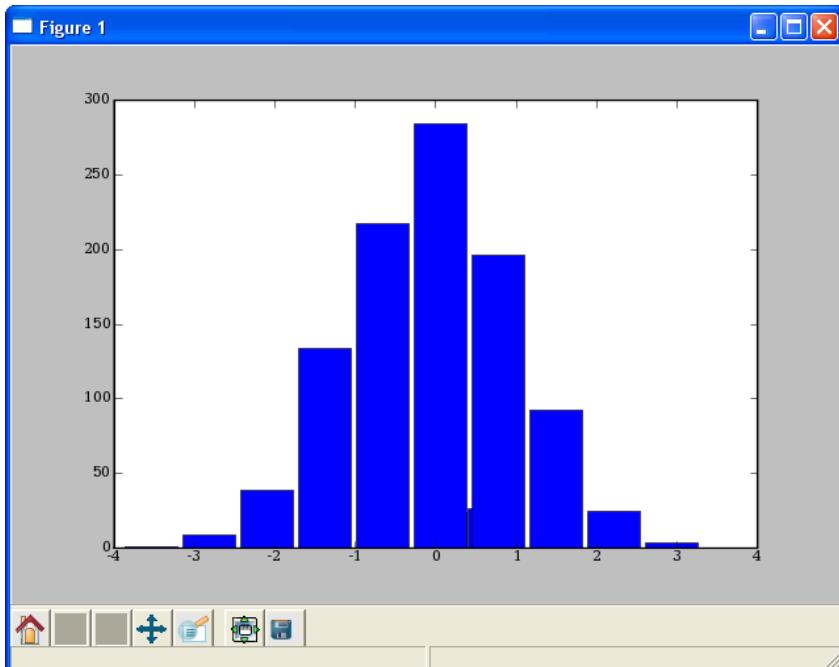
DEMO/MATPLOTLIB_PLOTTING/BARCHART_DEMO.PY



HISTOGRAMS

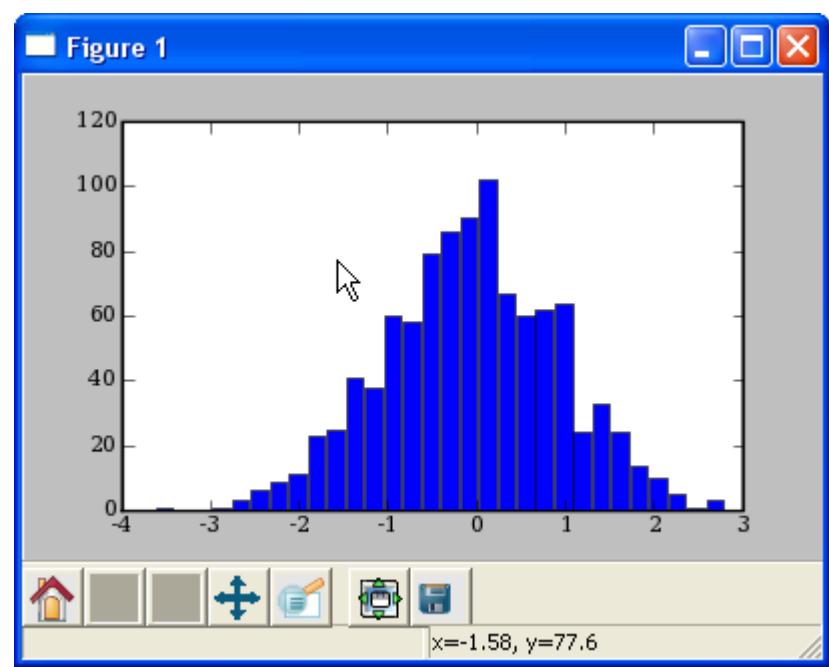
HISTOGRAM

```
# plot histogram  
# default to 10 bins  
>>> hist(randn(1000))
```



HISTOGRAM 2

```
# change the number of bins  
>>> hist(randn(1000), 30)
```



Chap.2.4 Pandas

pandas.read_csv 파일 읽기

pandas.read_csv("xxx.csv")로 읽고 확인하기

```
data_filename = 'nyc_data_01.csv'
print data_filename
fare_filename = 'nyc_fare_01.csv'
print fare_filename

nyc_data_01.csv
nyc_fare_01.csv

import pandas as pd

data = pd.read_csv(data_filename)
fare = pd.read_csv(fare_filename)

data.head(5)
```

	medallion	hack_license	vendor
0	76942C3205E17D7E7FE5A9F709D16434	25BA06A87905667AA1FE5990E33F0E2E	VTS
1	517C6B330DBB3F055D007B07512628B3	2C19FBEE1A6E05612EFE4C958C14BC7F	VTS
2	ED15611F168E41B33619C83D900FE266	784AEBD7C80DA17BA1D81D89FB6F4D1D	CMT
3	B33E704CC189E80C9671230C16627BBC	6789C77E1CBDC850C450D72204702976	VTS
4	BD5CC6A22D05EB2D5C8235526A2A4276	5E8F2C93B5220A922699FEBAFC2F7A54	VTS

```
<fare.tail(5)
```

	medallion	hack_license	vendor
72326	6C8EA4103E23404EA0E0AD0808FB5FEC0	390D55D0BCD554802CD22FC3D34FEC1B	VT
72327	2DBC1E95106344420C9818DA00FEA9AC	E467C8BCF6F93D3CFB184BC124BFA43C	VT
72328	5BAA40F73352E3A1C8648ACDC5445D05	9D21F436A113D106FC8169F6AC67F126	VT
72329	4150704BFA77421C4FC9C41D3A9E2A52	D019D1DF479317FE0918D35D6D9169EC	CN
72330	7B42640E2575D1C97848B1D1CCFA7307	74809FB2CFCFCCE1D5A0FE52AAFD84F46	CN

DataFrame.describe()

data.describe()를 이용해서 통계 기본 가져오기

```
data.describe()
```

	rate_code	passenger_count	trip_time_in_secs	trip_distance	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
count	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000
mean	1.026987	1.702023	681.408151	2.742523	-73.975341	40.750951	-73.974562	40.751526
std	0.233373	1.364674	489.732353	3.236069	0.034110	0.026638	0.031892	0.030094
min	0.000000	0.000000	0.000000	0.000000	-74.089478	40.524044	-74.099770	40.523472
25%	1.000000	1.000000	360.000000	1.000000	-73.992176	40.736893	-73.991318	40.736305
50%	1.000000	1.000000	553.000000	1.700000	-73.982010	40.753441	-73.980499	40.754227
75%	1.000000	2.000000	880.000000	3.030000	-73.968018	40.767635	-73.965538	40.768353
max	6.000000	6.000000	8401.000000	96.300000	-73.370308	40.899021	-73.370308	40.941383

DataFrame 속성

data.column(열정보), data.index(행정보) 가
져오기

```
data.columns
```

```
Index([u'medallion', u'hack_license', u'vendor_id', u'rate_code',
       u'store_and_fwd_flag', u'pickup_datetime', u'dropoff_datetime',
       u'passenger_count', u'trip_time_in_secs', u'trip_distance',
       u'pickup_longitude', u'pickup_latitude', u'dropoff_longitude',
       u'dropoff_latitude'],
      dtype='object')
```

```
data.index
```

```
RangeIndex(start=0, stop=72331, step=1)
```

DataFrame 특정 칼럼 추출

data[[칼럼명, 칼럼명]]을 넣어서 인덱싱 처리

```
: data.head(5)
```

	medallion	hack_license	vendor_id	rate_code
0	76942C3205E17D7E7FE5A9F709D16434	25BA06A87905667AA1FE5990E33F0E2E	VTS	1
1	517C6B330DBB3F055D007B07512628B3	2C19FBEE1A6E05612EFE4C958C14BC7F	VTS	1
2	ED15611F168E41B33619C83D900FE266	754AEBD7C80DA17BA1D81D89FB6F4D1D	CMT	1
3	B33E704CC189E80C9671230C16527BBC	6789C77E1CBDC850C450D72204702976	VTS	1
4	BD5CC6A22D05EB2D5C8235526A2A4276	5E8F2C93B5220A922699FEBAFC2F7A54	VTS	1


```
: data[['pickup_datetime', 'dropoff_datetime']].head(5)
```

	pickup_datetime	dropoff_datetime
0	2013-01-01 00:00:00	2013-01-01 00:05:00
1	2013-01-01 00:05:00	2013-01-01 00:21:00
2	2013-01-01 00:05:00	2013-01-01 00:12:00
3	2013-01-01 00:06:00	2013-01-01 00:06:00
4	2013-01-01 00:06:00	2013-01-01 00:12:00

DataFrame 특정 행 추출

data.loc[행번호]을 넣어서 인덱싱 처리

```
fare.tail(2)
```

	medallion	hack_license	vendor_id	pickup_datetime
72329	4150704BFA77421C4FC9C41D3A9E2A52	D019D1DF479317FE0918D35D6D9169EC	CMT	2013-01-31 23:59:00
72330	7B42640E2575D1C97848B1D1CCFA7307	74809FB2CFCFCE1D5A0FE52AAFD84F46	CMT	2013-01-31 23:59:00

```
fare.loc[0].head(3)
```

```
medallion      76942C3205E17D7E7FE5A9F709D16434
hack_license   25BA06A87905667AA1FE5990E33F0E2E
vendor_id      VTS
Name: 0, dtype: object
```

```
fare.loc[0].tail(5)
```

```
surcharge      0.5
mta_tax        0.5
tip_amount     0
tolls_amount   0
total_amount   6
Name: 0, dtype: object
```

Data Frame 데이터 타입

Pandas Type	Native Python Type	Description
object	string	문자 데이터 문자이거나 숫자와 문자 가 혼합되었을 때
int64	int	정수 데이터
float64	float	실수 데이터
datetime64, timedelta[ns]	datetime	날짜나 시간 데이터

Data Frame 데이터 타입

```
In [4]: #특정 컬럼 타입 체크하기  
df['salary'].dtype
```

```
Out[4]: dtype('int64')
```

```
In [5]: #모든 컬럼의 타입 체크하기  
df.dtypes
```

```
Out[4]: rank          object  
discipline      object  
phd            int64  
service         int64  
sex            object  
salary          int64  
dtype: object
```

Data Frames 속성

Python 오브젝트는 속성과 메서드를 갖습니다

df.attribute	description
dtypes	컬럼의 타입 리스트
columns	컬럼의 이름 리스트
axes	행 라벨과 컬럼 이름 리스트
ndim	차원 갯수
size	요소 갯수
shape	차원 형태
values	numpy로 표현한 데이터 값

Data Frames methods

속성은 인자가 없지만 메서드는 인자를 받습니다
모든 속성과 인자를 보려면 with a *dir()* function: **dir (df)**

df.method()	description
head([n]), tail([n])	앞/뒤의 n 개 행 선택
describe()	통계 정보 출력(수치형 데이터에 대해서만)
max(), min()	모든 수치형 데이터의 최대/최소값
mean(), median()	모든 수치형 데이터의 평균/중간값
std()	표준 편차
sample([n])	임의의 n 개 샘플링(n 개 행을 랜덤하게 뽑습니다)
dropna()	누락값 있는 행 제거

컬럼 선택하기

2가지 방식이 있습니다

- Dictionary 의 Key 접근 방식입니다
 - df['age']
- 점(.)을 이용해서 접근하는 방식입니다
 - df.age
 - 주의 : 컬럼 이름에 특수문자나 공백이 있는 경우에는 사용할 수 없습니다
- 복수 개 컬럼 선택시 컬럼 이름의 리스트를 사용
 - df[['age', 'city']]

Data Frames: 행 선택

특정 범위의 행을 선택하고 싶을 때는 ":" 를 사용합니다

```
In [ ]: # 위치인덱스로 행 선택하기:  
df[10:20]
```

Numpy와 마찬가지로 행은 0부터 시작하고 마지막 인덱스는 포함되지 않습니다:
그러므로 0:10 범위를 지정하면 처음 10개의 행이 리턴됩니다. 위치 번호는 0~9

Data Frames: loc 함수

특정행과 특정 컬럼을 선택하고 싶을때:

```
In [ ]: df_sub.loc[10:20, ['rank', 'sex', 'salary']]
```

```
Out[ ]:
```

	rank	sex	salary
10	Prof	Male	128250
11	Prof	Male	134778
13	Prof	Male	162200
14	Prof	Male	153750
15	Prof	Male	150480
19	Prof	Male	150500

Data Frames: iloc 함수 요약

```
df.iloc[0]    # 데이터 프레임의 첫번째 행  
df.iloc[i]    # (i+1) 번째 행  
df.iloc[-1]   # 마지막 행
```

```
df.iloc[:, 0]  # 첫번째 열  
df.iloc[:, -1] # 마지막 열
```

```
df.iloc[0:7]      # 위 7개 행  
df.iloc[:, 0:2]    # 왼쪽 2개 열  
df.iloc[1:3, 0:2]  # 1~2행의 왼쪽 2개 열  
df.iloc[[0,5], [1,3]] # 0st 과 5th 행의 1nd 과 4th 의 열
```

Data Frames: 정렬

컬럼의 값을 정렬할 수 있습니다.

기본적으로 오름차순 정렬이 되고 정렬된 데이터프레임이 반환됩니다.

```
In [ ]: df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

```
Out[ ]:
```

		rank	discipline	phd	service	sex	salary
55	AsstProf	A	2	0	Female	72500	
23	AsstProf	A	2	0	Male	85000	
43	AsstProf	B	5	0	Female	77000	
17	AsstProf	B	4	0	Male	92000	
12	AsstProf	B	1	0	Male	88000	

Data Frames: Sorting

2개 이상의 컬럼을 값을 정렬할 수 있습니다.

In []:

```
df_sorted = df.sort_values( by =['service', 'salary'],
                           ascending = [True, False])
df_sorted.head(10)
```

Out[]:

		rank	discipline	phd	service	sex	salary
52	Prof	A	12	0	Female	105000	
17	AsstProf	B	4	0	Male	92000	
12	AsstProf	B	1	0	Male	88000	
23	AsstProf	A	2	0	Male	85000	
43	AsstProf	B	5	0	Female	77000	
55	AsstProf	A	2	0	Female	72500	

Missing Values

누락값은 NaN 으로 표시됩니다

```
In [ ]: # 누락값이 하나라도 존재하는 행 선택  
flights[flights.isnull().any(axis=1)].head()
```

Out[]:

	year	month	day	dep_time	dep_delay	arr_time	arr_
330	2013		1	1	1807.0	29.0	2251.0
403	2013		1	1	NaN	NaN	NaN
404	2013		1	1	NaN	NaN	NaN
855	2013		1	2	2145.0	16.0	NaN
858	2013		1	2	NaN	NaN	NaN

Missing Values

누락값을 처리하는 함수입니다

df.method()	description
dropna()	누락이 있는 행을 제거
dropna(how='all')	모든 셀이 누락인 경우에만 행 제거
dropna(axis=1, how='all')	모든 값이 누락인 경우 컬럼 제거
dropna(thresh = 5)	5개 이상의 값을 가지고 있는 않은 행 제거
fillna(0)	누락값을 0으로 채우기
isnull()	누락인 경우 True 반환
notnull()	모두 누락인 경우 True 반환

그래프

```
In [ ]: %matplotlib inline
```

description	
distplot	histogram
barplot	estimate of central tendency for a numeric variable
violinplot	similar to boxplot, also shows the probability density of the data
jointplot	Scatterplot
regplot	Regression plot
pairplot	Pairplot
boxplot	boxplot
swarmplot	categorical scatterplot
factorplot	General categorical plot

Chap.2.5 데이터전처리

연속형 vs 범주형

출력값이 연속형 à 회귀 , 출력값이 범주형 à 분류

연속형 특성 continuous feature : 실수 데이터, ex) 0.13493, 100.0

범주형(이산형) 특성 categorical feature : 연속적이지 않은 숫자나 속성,
ex) 픽셀 강도, 브랜드, 쇼핑카테고리

범주형 특성의 사이에는 중간값이 없고 순서가 없습니다. ex) 책과 옷

특성 공학 feature engineering : 애플리케이션에 가장 적합한 데이터 표현을 찾는 것

알맞은 데이터 표현 >>> 매개변수 탐색 ex) inch와 cm의 스케일

범주형 변수

$$\hat{y} = w[0] \times x[0] + w[1] \times x[1] + \cdots + w[p] \times x[p] + b > 0$$

이진 출력: 분류문제
 $\leq 50, > 50k$

age	workclass	education	gender	hours-per-week	occupation	income
0 39	State-gov	Bachelors	Male	40	Adm-clerical	$\leq 50K$
1 50	Self-emp-not-inc	Bachelors	Male	13	Exec-managerial	$\leq 50K$
2 38	Private	HS-grad	Male	40	Handlers-cleaners	$\leq 50K$
3 53	Private	11th	Male	40	Handlers-cleaners	$\leq 50K$
4 28	Private	Bachelors	Female	40	Prof-specialty	$\leq 50K$

범주형 특성

연속형 특성

1994년 인구조사 데이터베이스: 미국 성인의 소득 데이터셋(Adult Data Set)

원-핫-인코딩 one-hot-encoding(가변수 dummy variable)

범주형 변수를 0 또는 1의 값을 가진 여러개의 새로운 특성으로 바꿈(이진 특성)

workclass	Government Employee	Private Employee	Self Employed	Self Employed Incorporated
Government Employee	1	0	0	0
Private Employee	0	1	0	0
Self Employed	0	0	1	0
Self Employed Incorporated	0	0	0	1

* 통계에서의 더미 인코딩은 마지막 값을 모든 변수가 0인 것으로 대신함(열 랭크 부족 현상 때문)

pandas.read_csv

```
53, Private, 234721, 11th, 7, Married-civ-spouse, Handlers-cleaners, Husband, Black, Male, 0, 0, 40, United-States, <=50K  
28, Private, 338409, Bachelors, 13, Married-civ-spouse, Prof-specialty, Wife, Black, Female, 0, 0, 40, Cuba, <=50K  
37, Private, 284582, Masters, 14, Married-civ-spouse, Exec-managerial, Wife, White, Female, 0, 0, 40, United-States, <=50K  
49, Private, 160187, 9th, 5, Married-spouse-absent, Other-service, Not-in-family, Black, Female, 0, 0, 16, Jamaica, <=50K  
52, Self-emp-not-inc, 209642, HS-grad, 9, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 45, United-States, >50K  
31, Private, 45781, Masters, 14, Never-married, Prof-specialty, Not-in-family, White, Female, 14084, 0, 50, United-States, >50K
```

```
data = pd.read_csv(  
    os.path.join(mglearn.datasets.DATA_PATH, "adult.data"), header=None, index_col=False,  
    names=['age', 'workclass', 'fnlwgt', 'education', 'education-num',  
           'marital-status', 'occupation', 'relationship', 'race', 'gender',  
           'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',  
           'income'])  
# 예제를 위해 몇개의 열만 선택합니다  
data = data[['age', 'workclass', 'education', 'gender', 'hours-per-week',  
            'occupation', 'income']]
```

	age	workclass	education	gender	hours-per-week	occupation	income
0	39	State-gov	Bachelors	Male	40	Adm-clerical	<=50K
1	50	Self-emp-not-inc	Bachelors	Male	13	Exec-managerial	<=50K
2	38	Private	HS-grad	Male	40	Handlers-cleaners	<=50K

pandas.get_dummies

```
In [4]: print(data.gender.value_counts())
```

```
Male      21790  
Female    10771  
Name: gender, dtype: int64
```

```
data_dummies = pd.get_dummies(data)
```

	age	workclass	education
0	39	State-gov	Bachelors
1	50	Self-emp-not-inc	Bachelors
2	38	Private	HS-grad

범주형 데이터 자동 변환

age	hours-per-week	workclass_?	workclass_Federal-gov	workclass_Local-gov	...	occupation_Tech-support	occupation_Transport-moving	income_<=50K	income_>50K
0	39	40	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	50	13	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	38	40	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	53	40	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	28	40	0.0	0.0	0.0	0.0	0.0	1.0	0.0

로지스틱 회귀 적용

occupation_ Transport-moving까지 포함됨

```
In [7]: features = data_dummies.loc[:, 'age':'occupation_Transport-moving']  
# NumPy 배열 추출  
X = features.values  
y = data_dummies['income_ >50K'].values  
print("X.shape: {} y.shape: {}".format(X.shape, y.shape))
```

X.shape: (32561, 44) y.shape: (32561,)

```
In [8]: from sklearn.linear_model import LogisticRegression  
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)  
logreg = LogisticRegression()  
logreg.fit(X_train, y_train)  
print("테스트 점수: {:.2f}".format(logreg.score(X_test, y_test)))
```

테스트 점수: 0.81

훈련세트 나누기 전에 먼저 get_dummies 적용

숫자로 된 범주형 변수

숫자로 되어 있다고 무조건 연속형 특성은 아닙니다.

ex) workclass를 객관식으로 골랐다면 1, 2, 3 와 같은 숫자로 저장될 수 있습니다.

연속형인지 범주형인지는 특성의 의미를 알아야 판단할 수 있는 경우가 많습니다.

ex) 별 다섯개의 평점 데이터, 영화 관람 등급(범주형이지만 순서가 있음)

pandas의 get_dummies 또는 scikit-learn의 OneHotEncoder를 사용할 수 있음

OneHotEncoder는 숫자로된 범주형 변수에만 사용 가능

숫자 범주형 변환 예제

```
In [10]: display(pd.get_dummies(demo_df))
```

	범주형 특성	숫자 특성		숫자 특성	범주형 특성_상자	범주형 특성_양말	범주형 특성_여우
0	양말	0		0	0	1	0
1	여우	1		1	1	0	1
2	양말	2		2	2	0	1
3	상자	1		3	1	1	0

```
In [11]: demo_df['숫자 특성'] = demo_df['숫자 특성'].astype(str)  
display(pd.get_dummies(demo_df, columns=['숫자 특성', '범주형 특성']))
```

	숫자 특성_0	숫자 특성_1	숫자 특성_2	범주형 특성_상자	범주형 특성_양말	범주형 특성_여우
0	1	0	0	0	1	0
1	0	1	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	1	0	0

구간 분할 binning

연속형 특성 하나를 구간을 나누어 여러개의 범주형 특성으로 만듦

```
In [13]: bins = np.linspace(-3, 3, 11)
print("bins: {}".format(bins))
```

```
bins: [-3. -2.4 -1.8 -1.2 -0.6 0. 0.6 1.2 1.8 2.4 3.]
```

[11] : $3 \leq x$

```
In [14]: which_bin = np.digitize(X, bins=bins)
print("\n데이터 포인트:\n", X[:5])
print("\n데이터 포인트의 소속 구간:\n", which_bin[:5])
```

데이터 포인트:

```
[[ -0.753]
 [ 2.704]
 [ 1.392]
 [ 0.592]
 [-2.064]]
```

[10] : $2.4 \leq x < 3$

연속형

데이터 포인트의 소속 구간:

```
[[ 4]
 [10]
 [ 8]
 [ 6]
 [ 2]]
```

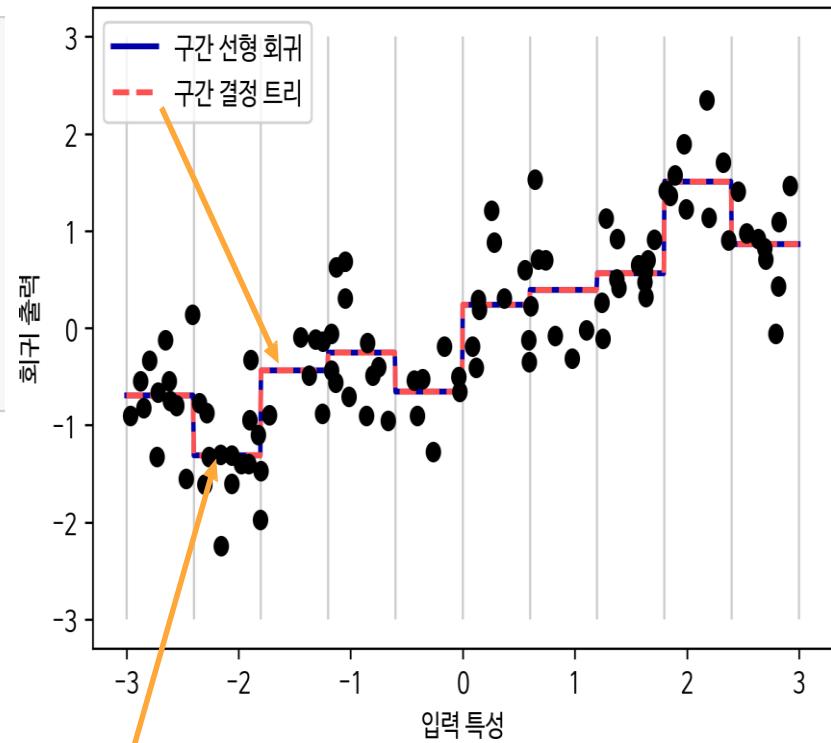
범주형

OneHotEncoder

일반 numpy 배열을 받기 위해

```
In [15]: from sklearn.preprocessing import OneHotEncoder  
# 변환을 위해 OneHotEncoder를 사용합니다  
encoder = OneHotEncoder(sparse=False)  
# encoder.fit은 which_bin에 나타난 유일한 값을 찾습니다  
encoder.fit(which_bin)  
# 원-핫-인코딩으로 변환합니다  
X_binned = encoder.transform(which_bin)  
print(X_binned[:5])
```

```
[[ 4]  [[ 0.  0.  0.  1.  0.  0.  0.  0.  0.]  
[10]  [ 0.  0.  0.  0.  0.  0.  0.  0.  1.]  
[ 8]  [ 0.  0.  0.  0.  0.  0.  1.  0.  0.]  
[ 6]  [ 0.  0.  0.  0.  0.  1.  0.  0.  0.]  
[ 2]]  [ 0.  1.  0.  0.  0.  0.  0.  0.  0.]]
```



선형 모델은 상수 특성으로 인해 유연해졌으나
결정트리는 오히려 더 나빠졌음

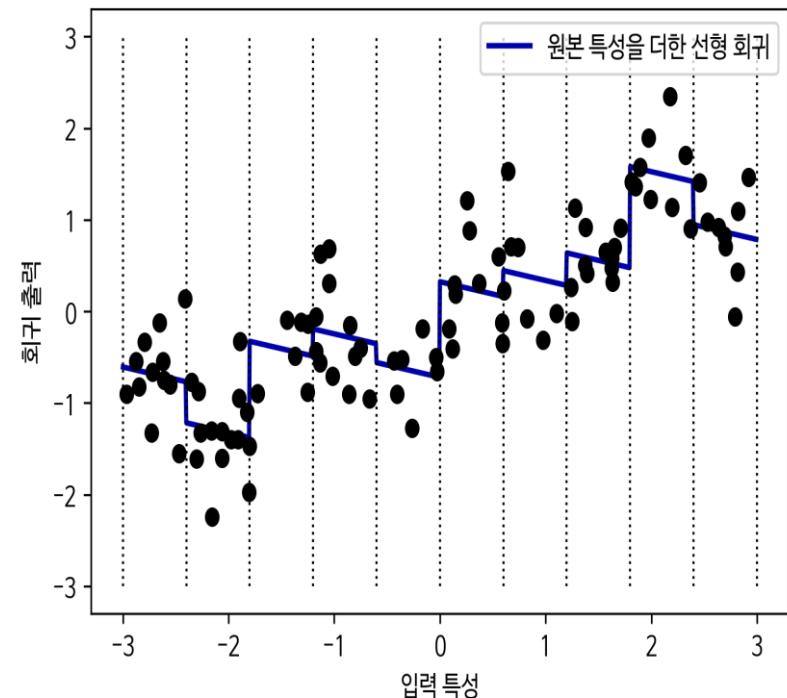
구간 분할 + 원본데이터

```
In [18]: X_combined = np.hstack([X, X_binned])
print(X_combined.shape)
```

```
(100, 11)
```

```
[-0.753,  0. ,  0. ,  ...,  0. ,  0. ,  0. ]
[ 2.704,  0. ,  0. ,  ...,  0. ,  0. ,  1. ]
[ 1.392,  0. ,  0. ,  ...,  1. ,  0. ,  0. ]
...
[-0.435,  0. ,  0. ,  ...,  0. ,  0. ,  0. ]
[-2.847,  1. ,  0. ,  ...,  0. ,  0. ,  0. ]
[-2.353,  0. ,  1. ,  ...,  0. ,  0. ,  0. ]
```

```
In [37]: reg = LinearRegression().fit(X_combined, y)
```



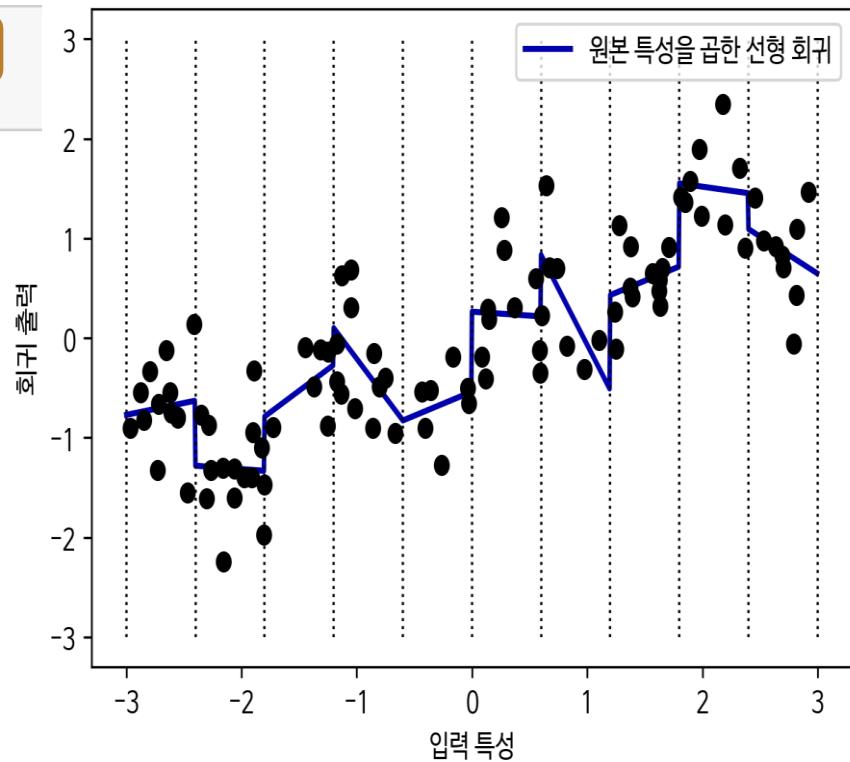
구간 분할 * 원본데이터

```
In [39]: X_product = np.hstack([X_binned, X * X_binned])
print(X_product.shape)
```

```
(100, 20)
```

```
[ 0. ,  0. ,  0. , ..., -0. , -0. , -0. ]
[ 0. ,  0. ,  0. , ...,  0. ,  0. ,  2.704]
[ 0. ,  0. ,  0. , ...,  1.392,  0. ,  0. ]
...
[ 0. ,  0. ,  0. , ..., -0. , -0. , -0. ]
[ 1. ,  0. ,  0. , ..., -0. , -0. , -0. ]
[ 0. ,  1. ,  0. , ..., -0. , -0. , -0. ]
```

```
reg = LinearRegression().fit(X_product, y)
```



PolynomialFeatures

구간 분할의 예제처럼 원본 특성에 상호작용이나 제곱항을 추가함

```
In [22]: from sklearn.preprocessing import PolynomialFeatures  
  
# x ** 10까지 고차항을 추가합니다  
# 기본값인 "include_bias=True"는 절편에 해당하는 1인 특성을 추가합니다  
poly = PolynomialFeatures(degree=10, include_bias=False)  
poly.fit(X)  
X_poly = poly.transform(X)
```

```
In [23]: print("X_poly.shape: {}".format(X_poly.shape))
```

X_poly.shape: (100, 10)

		-0.753	0.567	-0.427	0.321	-0.242	0.182
		-0.137	0.103	-0.078	0.058]		
[-0.753]		2.704	7.313	19.777	53.482	144.632	391.125
[2.704]		1057.714	2860.36	7735.232	20918.278]		
[1.392]	→	1.392	1.938	2.697	3.754	5.226	7.274
[0.592]		10.125	14.094	19.618	27.307]		
[-2.064]		0.592	0.35	0.207	0.123	0.073	0.043
		0.025	0.015	0.009	0.005]		
		-2.064	4.26	-8.791	18.144	-37.448	77.289
		-159.516	329.222	-679.478	1402.367]]		

boston + Ridge vs RandomForestClassifier

```
In [31]: from sklearn.linear_model import Ridge  
ridge = Ridge().fit(X_train_scaled, y_train)  
print("상호작용 특성이 없을 때 점수: {:.3f}".format(ridge.score(X_test_scaled, y_test)))  
ridge = Ridge().fit(X_train_poly, y_train)  
print("상호작용 특성이 있을 때 점수: {:.3f}".format(ridge.score(X_test_poly, y_test)))
```

상호작용 특성이 없을 때 점수: 0.621

상호작용 특성이 있을 때 점수: 0.753

```
In [32]: from sklearn.ensemble import RandomForestRegressor  
rf = RandomForestRegressor(n_estimators=100, random_state=0).fit(X_train_scaled, y_train)  
print("상호작용 특성이 없을 때 점수: {:.3f}".format(rf.score(X_test_scaled, y_test)))  
rf = RandomForestRegressor(n_estimators=100, random_state=0).fit(X_train_poly, y_train)  
print("상호작용 특성이 있을 때 점수: {:.3f}".format(rf.score(X_test_poly, y_test)))
```

상호작용 특성이 없을 때 점수: 0.795

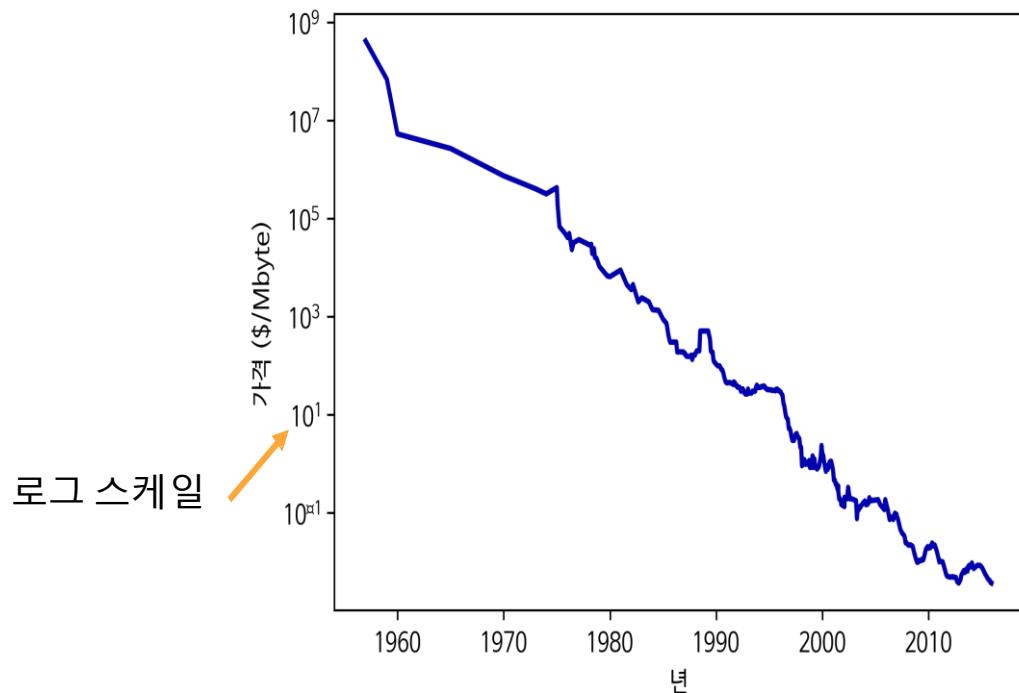
상호작용 특성이 있을 때 점수: 0.773

일변량 변환

log, exp, sin 같은 수학 함수를 적용하여 특성 값을 변환함

선형이나 신경망 모델 같은 경우 데이터 스케일에 민감함

ex) 2장의 컴퓨터 메모리 가격 데이터 예제

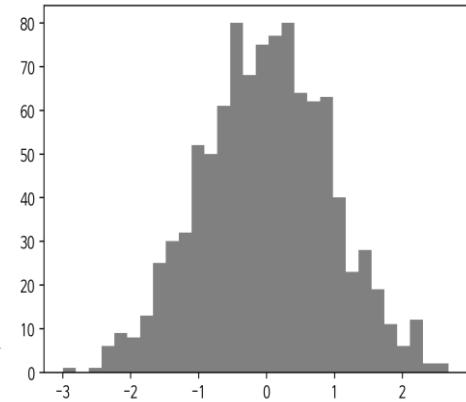


카운트 데이터

```
In [33]: rnd = np.random.RandomState(0)
X_org = rnd.normal(size=(1000, 3))
w = rnd.normal(size=3)

X = rnd.poisson(10 * np.exp(X_org))
y = np.dot(X_org, w)
print(X[:10, 0])
```

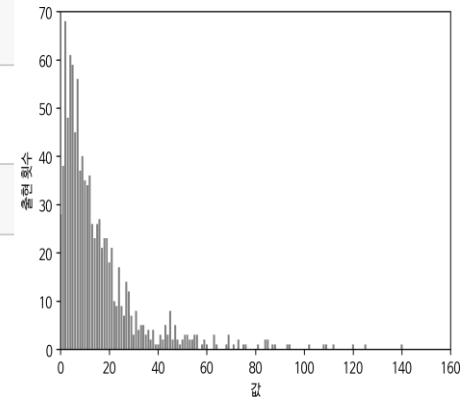
[56 81 25 20 27 18 12 21 109 7]



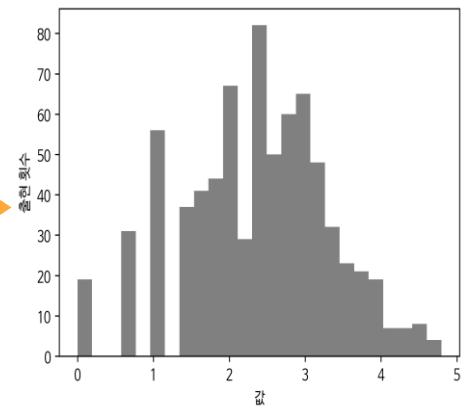
```
In [34]: print("특성 출현 횟수:\n{}".format(np.bincount(X[:, 0].astype('int'))))
```

특성 출현 횟수:

```
[28 38 68 48 61 59 45 56 37 40 35 34 36 26 23 26 27 21 23 23 18 21 10 9 17
 9 7 14 12 7 3 8 4 5 5 3 4 2 4 1 1 3 2 5 3 8 2 5 2 1
 2 3 3 2 2 3 3 0 1 2 1 0 0 3 1 0 0 0 1 3 0 1 0 2 0
 1 1 0 0 0 0 1 0 0 2 2 0 1 1 0 0 0 0 1 1 0 0 0 0
 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0]
```



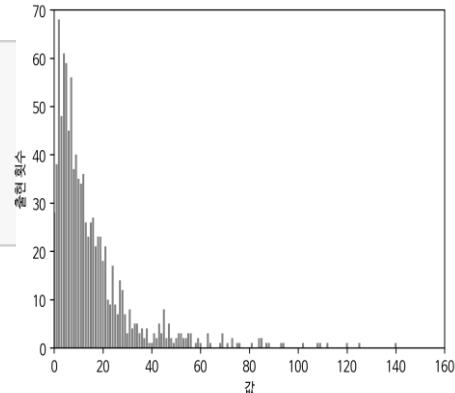
```
In [37]: X_train_log = np.log(X_train + 1)
X_test_log = np.log(X_test + 1)
```



카운트 데이터 + Ridge

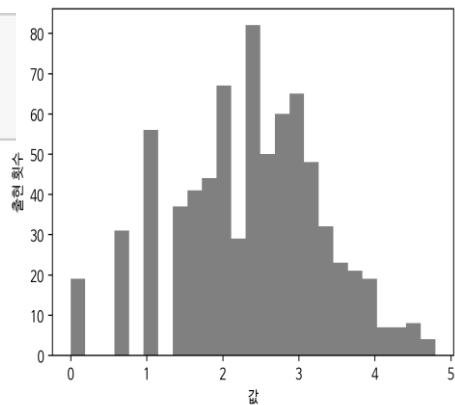
```
In [36]: from sklearn.linear_model import Ridge  
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)  
score = Ridge().fit(X_train, y_train).score(X_test, y_test)  
print("테스트 점수: {:.3f}".format(score))
```

테스트 점수: 0.622



```
In [39]: score = Ridge().fit(X_train_log, y_train).score(X_test_log, y_test)  
print("테스트 점수: {:.3f}".format(score))
```

테스트 점수: 0.875



특성 자동선택

일변량 통계

유용한 특성인지를 판단하는 방법: 일변량 통계, 모델 기반 선택, 반복적 선택
분산 분석(ANOVA_{analysis of variance})은 클래스별 평균을 비교합니다.

$$F = \frac{SS_{between}/(k - 1)}{(SS_{tot} - SS_{between})/(n - k)}$$

$$SS_{between} = \sum_{j=1}^k n_j (\bar{x}_j - \bar{x})^2, SS_{tot} = \sum_{i=1}^n (x_i - \bar{x})^2$$

SelectKBest나 SelectPercentile에서 분류는 f_classif, 회귀는 f_regression로 지정

cancer + noise

```
In [40]: from sklearn.datasets import load_breast_cancer
        from sklearn.feature_selection import SelectPercentile, f_classif
        from sklearn.model_selection import train_test_split

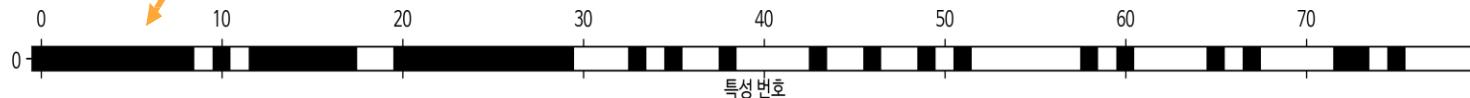
cancer = load_breast_cancer()

# 고정된 난수를 발생시킵니다
rng = np.random.RandomState(42)
noise = rng.normal(size=(len(cancer.data), 50))
# 데이터에 노이즈 특성을 추가합니다
# 처음 30개는 원본 특성이고 다음 50개는 노이즈입니다
X_w_noise = np.hstack([cancer.data, noise])

X_train, X_test, y_train, y_test = train_test_split(
    X_w_noise, cancer.target, random_state=0, test_size=.5)
# f_classif(기본값)과 SelectPercentile을 사용하여 특성의 50%를 선택합니다
select = SelectPercentile(score_func=f_classif, percentile=50)
select.fit(X_train, y_train)
# 훈련 세트에 적용합니다
X_train_selected = select.transform(X_train)

print("X_train.shape: {}".format(X_train.shape))
print("X_train_selected.shape: {}".format(X_train_selected.shape))
```

X_train.shape: (284, 80)
X_train_selected.shape: (284, 40)



SelectPercentile + LogisticRegression

성능 향상 + 모델 해석↑

```
In [42]: from sklearn.linear_model import LogisticRegression

# 테스트 데이터 변환
X_test_selected = select.transform(X_test)

lr = LogisticRegression()
lr.fit(X_train, y_train)
print("전체 특성을 사용한 점수: {:.3f}".format(lr.score(X_test, y_test)))
lr.fit(X_train_selected, y_train)
print("선택된 일부 특성을 사용한 점수: {:.3f}".format(
    lr.score(X_test_selected, y_test)))
```

전체 특성을 사용한 점수: 0.930

선택된 일부 특성을 사용한 점수: 0.940

모델 기반 선택

feature_importances_(결정트리)나 coef_(선형모델) 값을 사용합니다.

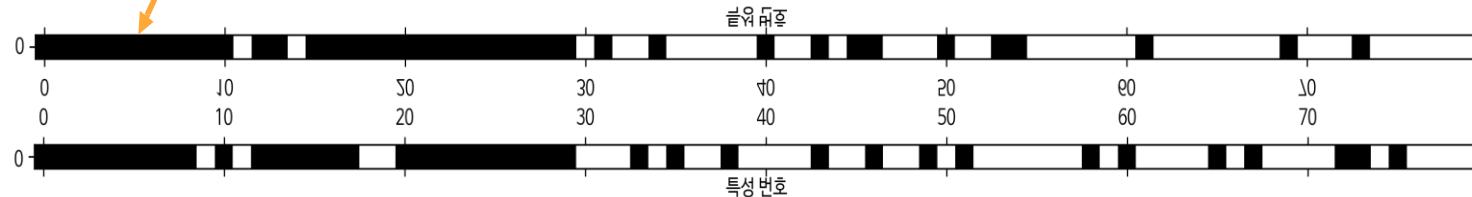
기본 임계값 : L1 페널티(라쏘)가 있는 경우 10^{-5} , 그외는 평균값

mean
1.3*median

```
In [43]: from sklearn.feature_selection import SelectFromModel
         from sklearn.ensemble import RandomForestClassifier
         select = SelectFromModel(
             RandomForestClassifier(n_estimators=100, random_state=42),
             threshold="median")
```

```
In [44]: select.fit(X_train, y_train)
         X_train_l1 = select.transform(X_train)
         print("X_train.shape: {}".format(X_train.shape))
         print("X_train_l1.shape: {}".format(X_train_l1.shape))
```

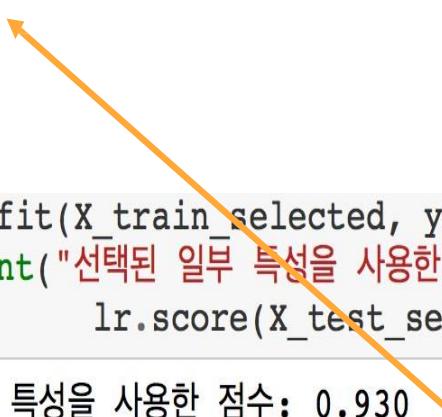
X_train.shape: (284, 80)
X_train_l1.shape: (284, 40)



SelectFromModel + LogisticRegression

```
In [46]: X_test_11 = select.transform(X_test)
score = LogisticRegression().fit(X_train_11, y_train).score(X_test_11, y_test)
print("Test score: {:.3f}".format(score))
```

Test score: 0.951



```
lr.fit(X_train_selected, y_train)
print("선택된 일부 특성을 사용한 점수: {:.3f}".format(
    lr.score(X_test_selected, y_test)))
```

전체 특성을 사용한 점수: 0.930
선택된 일부 특성을 사용한 점수: 0.940

반복적 특성선택

특성을 하나씩 추가하면서 모델을 만들거나 모든 특성에서 하나씩 제거하면서 모델을 만듭니다.

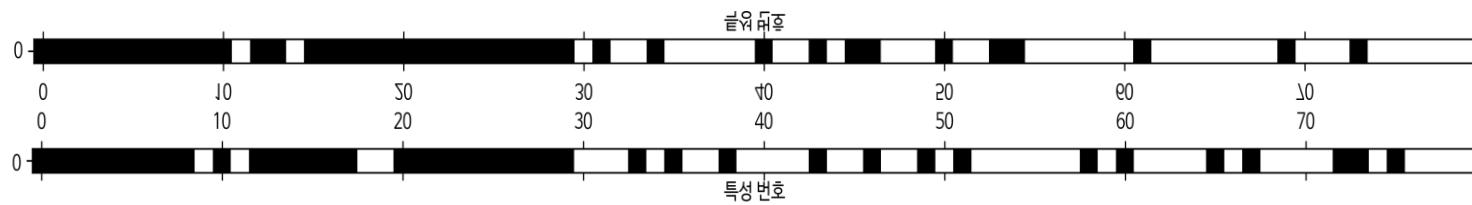
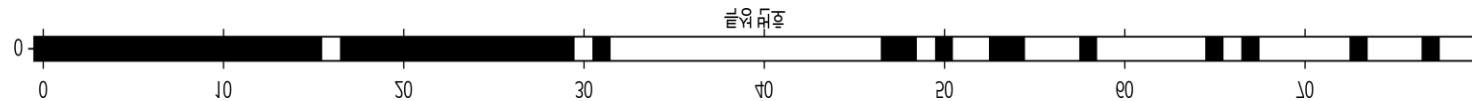
여러개의 모델을 만들기 때문에 계산 비용이 많이 듭니다.

재귀적 특성 제거(RFE recursive feature elimination)은 전체 특성을 포함한 모델에서 특성 중요도가 가장 낮은 특성을 지정된 개수만큼 남을 때 까지 계속 제거합니다.

`feature_importances_`(결정트리)나 `coef_`(선형모델) 값을 사용합니다.

RFE

```
In [47]: from sklearn.feature_selection import RFE  
select = RFE(RandomForestClassifier(n_estimators=100, random_state=42),  
               n_features_to_select=40)  
  
select.fit(X_train, y_train)  
# 선택된 특성을 표시합니다  
mask = select.get_support()
```



RFE + LogisticRegression

RFE에 사용한 모델을 이용해 평가에 사용할 수 있습니다.

```
In [48]: X_train_rfe = select.transform(X_train)
X_test_rfe = select.transform(X_test)

score = LogisticRegression().fit(X_train_rfe, y_train).score(X_test_rfe, y_test)
print("테스트 점수: {:.3f}".format(score))
```

테스트 점수: 0.951

```
In [49]: print("테스트 점수: {:.3f}".format(select.score(X_test, y_test)))
```

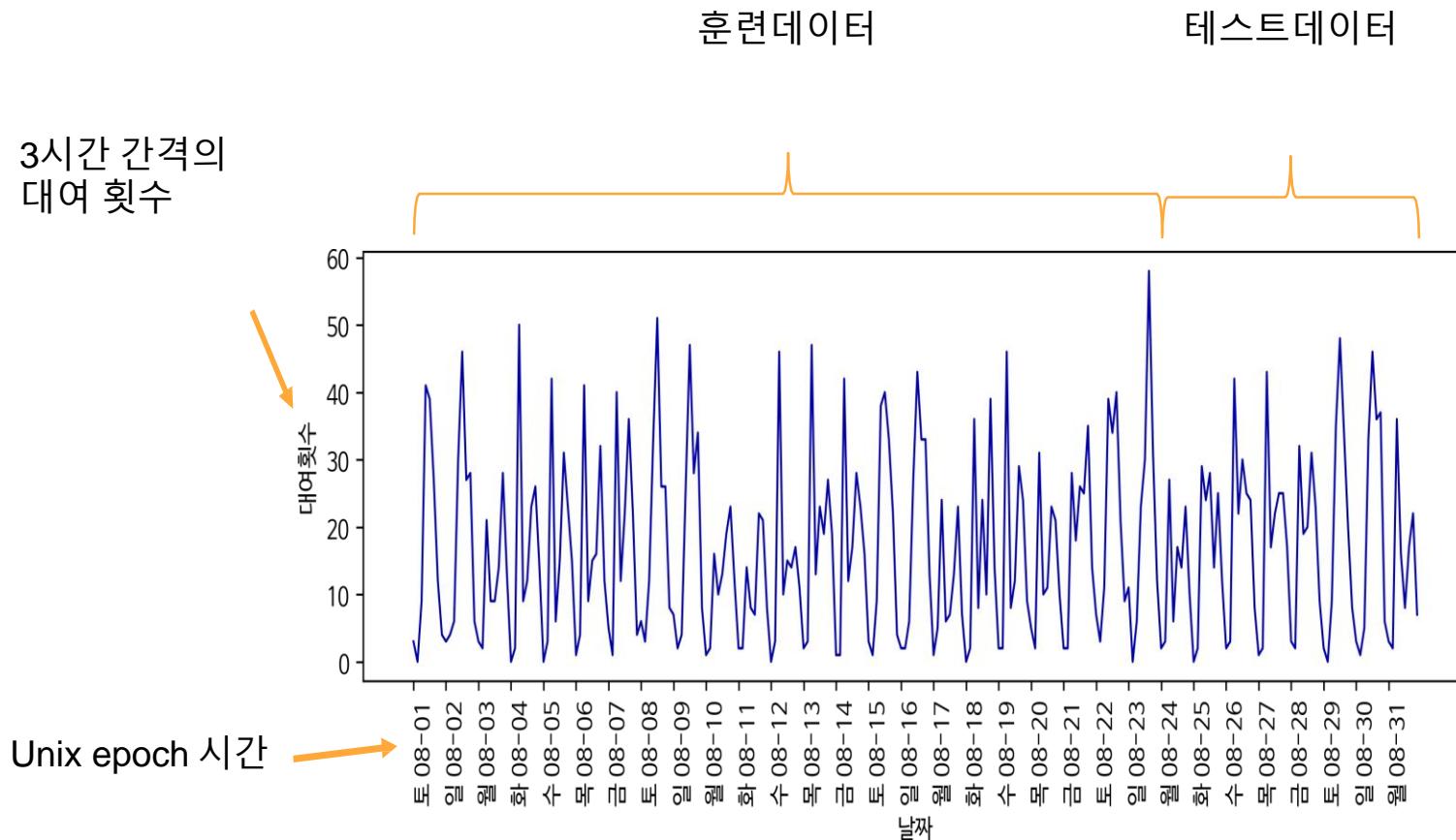
테스트 점수: 0.951

로지스틱 회귀와 랜덤 포레스트의 성능이 비슷함

특성 공학과 도메인지식

항공료를 잘 예측하려면 날짜, 항공사, 출발지, 도착지외에 음력 공휴일이나 방학 기간을 알아야 합니다.

뉴욕의 시티 바이크 대여 예측



Unix Time + RandomForestRegressor

```
In [55]: regressor = RandomForestRegressor(n_estimators=100, random_state=0)  
eval_on_features(X, y, regressor)
```

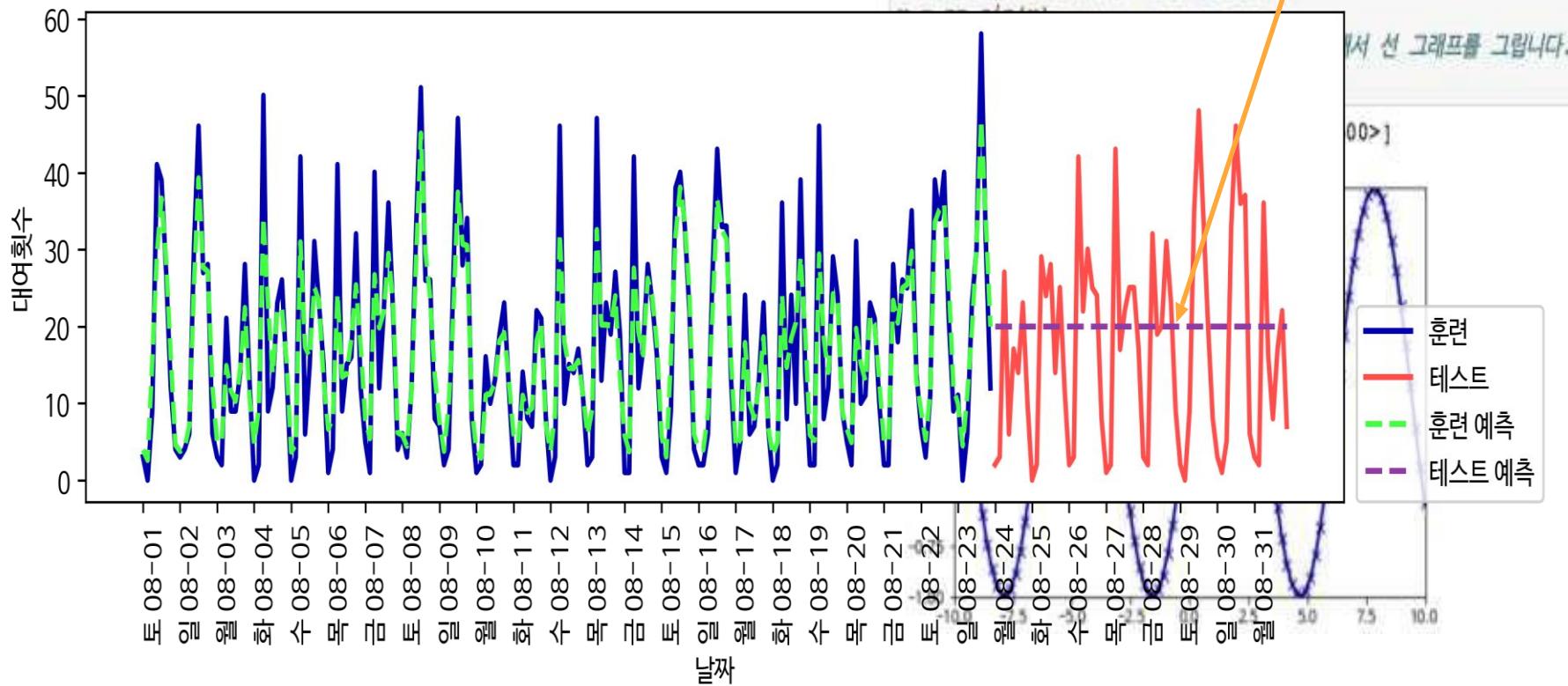
테스트 세트 R^2 : -0.04

[248, 1]

x = np.linspace(-10, 10, 100)

사이킷 런처를 사용하여 y 배열을 생성합니다.

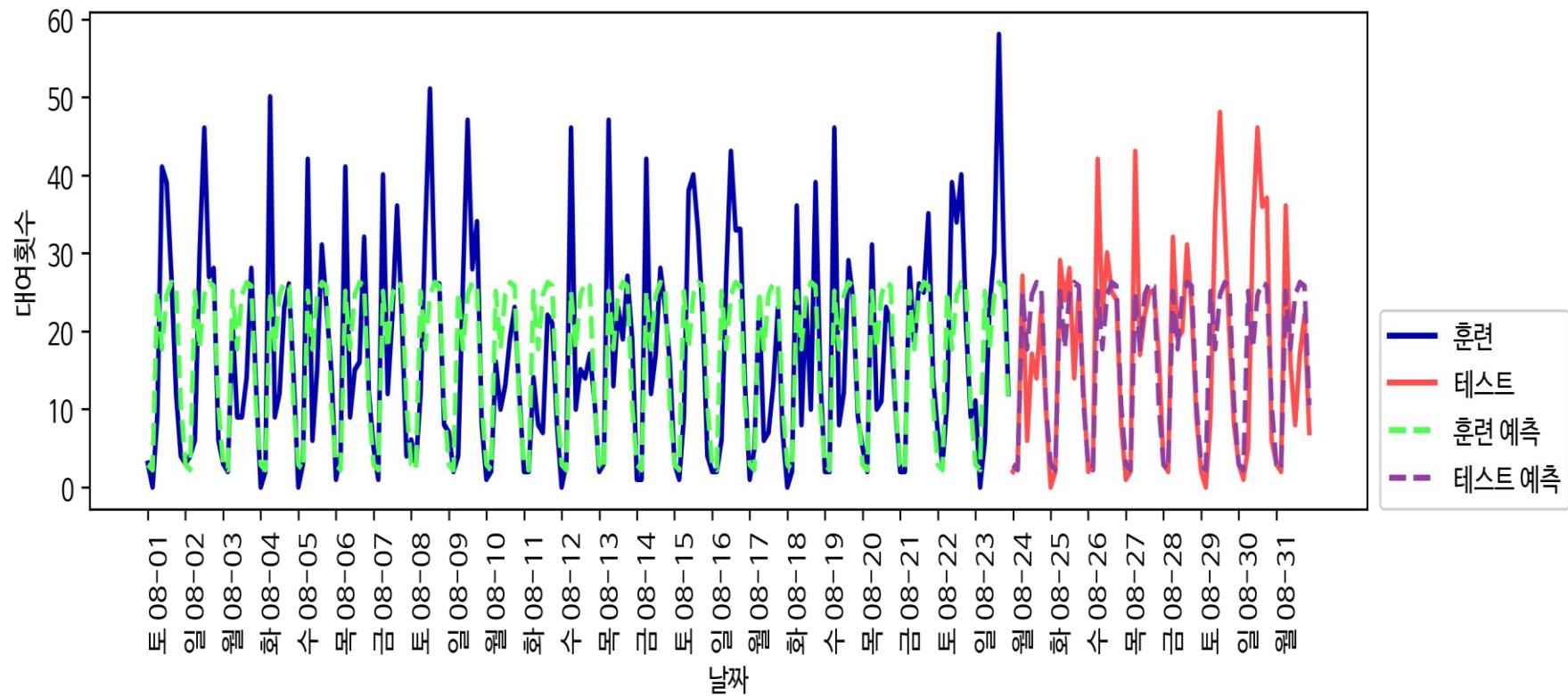
트리 모델의 특징
(외삽 불능)



Hour + RandomForestRegressor

```
In [56]: X_hour = citibike.index.hour.values.reshape(-1, 1)  
eval_on_features(X_hour, y, regressor)
```

테스트 세트 R²: 0.60



Hour,week + RandomForestRegressor

```
In [57]: X_hour_week = np.hstack([citibike.index.dayofweek.values.reshape(-1, 1),
                                citibike.index.hour.values.reshape(-1, 1)])
eval on features(X_hour_week, y, regressor)
```

테스트 세트 R²: 0.84

[248, 2]

