

# 빅데이터 분석 실무과정

## Part III : 파이썬 프로그래밍

# | Part III. 파이썬 프로그래밍 - 목차

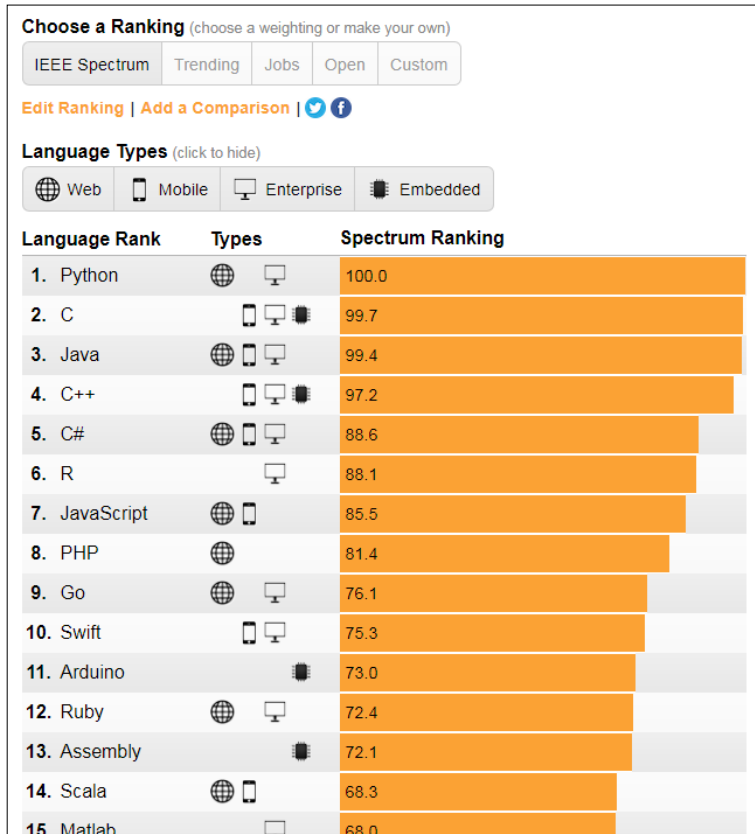
1. 파이썬이란
2. 파이썬 기초
3. 파이썬 고급
  - Numpy
  - Matplotlib
  - Scikit-learn

# 1. 파이썬이란

▶ 개념, 설치, 실행

# 왜 파이썬인가?

## ▶ The Top Programming Languages 2017 (IEEE.org)



<https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>

- 4가지 분류 중 3군데에서 1등  
(Jobs 분야에서 Java → C → Python)
- 놀랍게도 파이썬은 현재 가장 인기있고 많이 사용되는 프로그래밍 언어이다!
- 왜 그럴까?
  - 많은 사람들이 써보면 안다고 한다
  - 자바와 C에 지친 사람들이 파이썬으로 물려들고 있다
  - 양복을 벗고 청바지로 출근한 느낌이다
- 그래서 이유는? 간단히..
  - 쉽고, 편하다
  - 재미있다
  - 다른 언어에서 가능한 건 파이썬에서 다 가능하다
  - 복잡한 규칙은 벗어던지고 알고리즘에만 집중한다.
  - 개발이 빨리 끝나고 결과가 예쁘다

# | 파이썬은 이런 언어이다

- ▶ 불필요하고 복잡한 문법과 절차는 생략되어 있다
- ▶ 코딩 → 테스트 → 코딩 → 테스트 → ... → 프로토타입
  - 테스트 주도 (Trial & Error)
  - 프로토타입 우선
  - 알고리즘 중심
- ▶ 데이터분석과 수치계산에 특화되어 있다
  - numpy, matplotlib, pandas, scipy 등 강력하고 편리한 라이브러리
  - 머신러닝(scikit-learn), 딥러닝(tensorflow), 빅데이터(pyspark) 개발을 위한 최신 라이브러리

# 참고 자료

## ▶▶ The Python Standard Library

- <https://docs.python.org/3/library/>



## ▶▶ 점프 투 파이썬

- <https://wikidocs.net/book/1>

## ▶▶ 러닝 파이썬



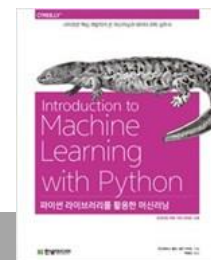
## ▶▶ 파이썬 라이브러리를 활용한 데이터 분석

## ▶▶ 파이썬 데이터 사이언스 핸드북

- <https://jakevdp.github.io/PythonDataScienceHandbook/>

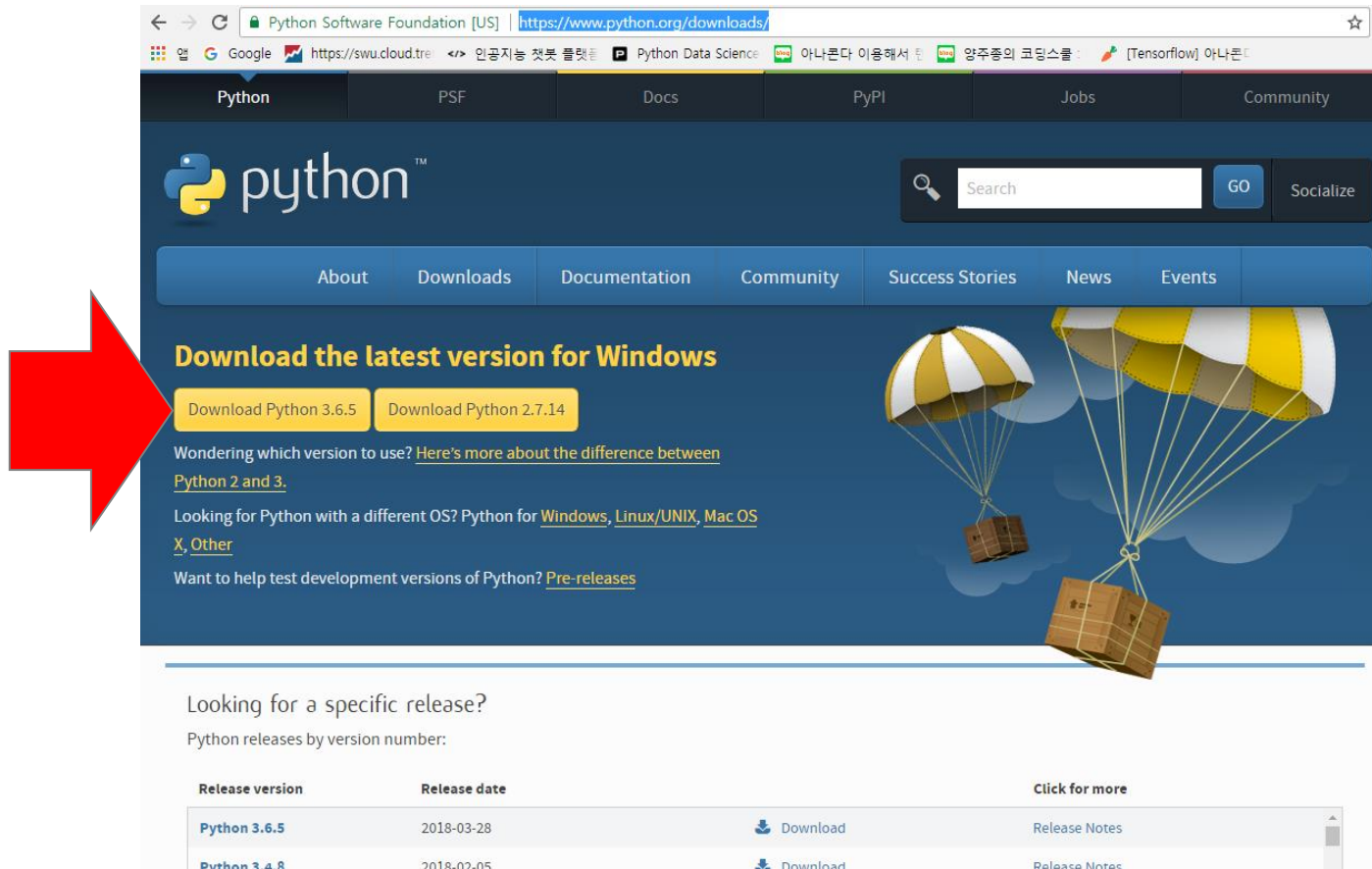


## ▶▶ 파이썬 라이브러리를 활용한 머신러닝



# 파이썬 다운로드

▶ <https://www.python.org/downloads/>



The screenshot shows the Python.org website's download page. A large red arrow points to the 'Download Python 3.6.5' and 'Download Python 2.7.14' buttons. The page features a dark blue header with the Python logo and navigation links. Below the header, there's a section titled 'Download the latest version for Windows' with two yellow buttons for downloading Python 3.6.5 and Python 2.7.14. To the right of these buttons is an illustration of two parachutes carrying boxes. Below the buttons, there are links for more information about Python versions and for downloading Python on other operating systems. At the bottom, there's a table listing Python releases by version number and date, with links to download and release notes for each version.

**Download the latest version for Windows**

[Download Python 3.6.5](#) [Download Python 2.7.14](#)

Wondering which version to use? [Here's more about the difference between Python 2 and 3.](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Pre-releases](#)

Looking for a specific release?

Python releases by version number:

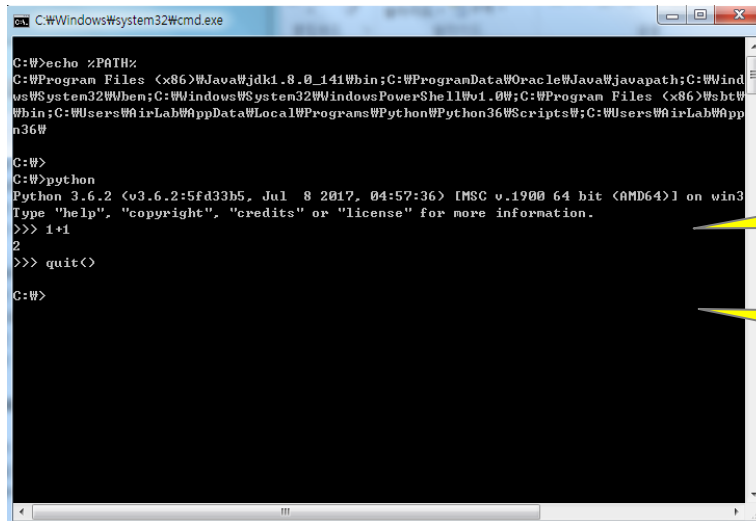
Release version	Release date		Click for more
<a href="#">Python 3.6.5</a>	2018-03-28	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.4.8</a>	2018-02-05	<a href="#">Download</a>	<a href="#">Release Notes</a>

# 파이썬 설치

## ▶▶ python-3.6.5.exe 실행

- Path 에 아래 경로가 추가됨 (명령창에서 “echo %PATH%” 로 확인)  
“C:\Users\AirLab\AppData\Local\Programs\Python\Python36\Scripts\;C:\Users\AirLab\AppData\Local\Programs\Python\Python36\”

## ▶▶ 명령창에서 “python” 실행



```
C:\Windows\system32\cmd.exe
C:\>echo %PATH%
C:\Program Files (x86)\Java\jdk1.8.0_141\bin;C:\ProgramData\Oracle\Java\javapath;C:\Windows\System32\cmd;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Webt...
C:\>
C:\>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 1+1
2
>>> quit()
C:\>
```

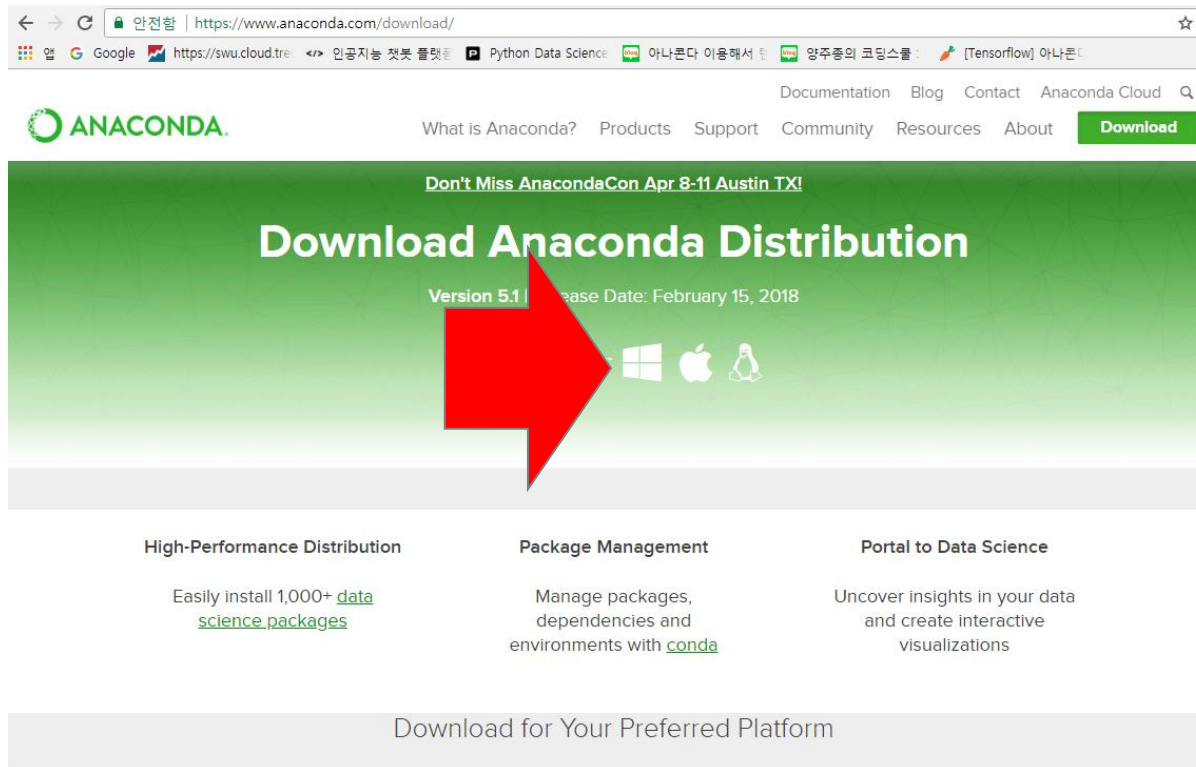
기본경로는 “python” 을 실행한 디렉토리가 됨

종료는 “quit()” 실행



# | 아나콘다 다운로드

▶▶ <https://www.anaconda.com/download/>



# 아나콘다 설치

- ▶ Anaconda3-5.1.0-Windows-x86\_64.exe 실행
- ▶ 시작메뉴 > Anaconda3 > Anaconda Prompt
- ▶ python 또는 ipython 실행

```
Anaconda Prompt

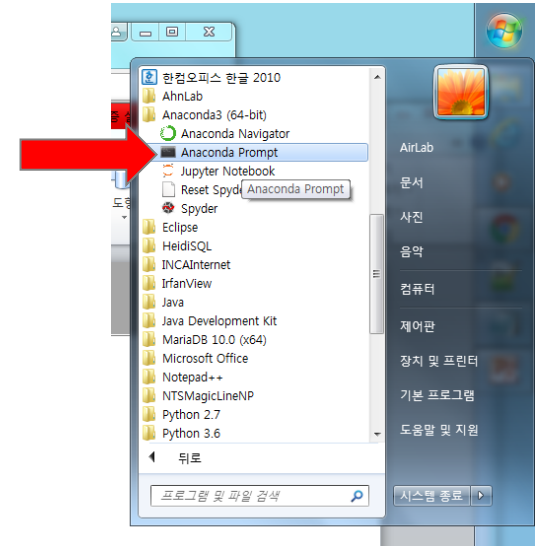
(base) C:\khh\develop>
(base) C:\khh\develop>echo %PATH%
C:\Anaconda3;C:\Anaconda3\Library\mingw-w64\bin;C:\Anaconda3\Library\usr\bin;C:\Anaconda3\Library\bin;C:\Anaconda3\Scripts;C:\Anaconda3\bin;C:\Program Files (x86)\Java\jdk1.8.0_141\bin;C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\sbt\bin;C:\spark-2.3.0-bin-hadoop2.7\bin;C:\Python27\Scripts

(base) C:\khh\develop>ipython
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: 1+1
Out[1]: 2

In [2]: quit()

(base) C:\khh\develop>
```

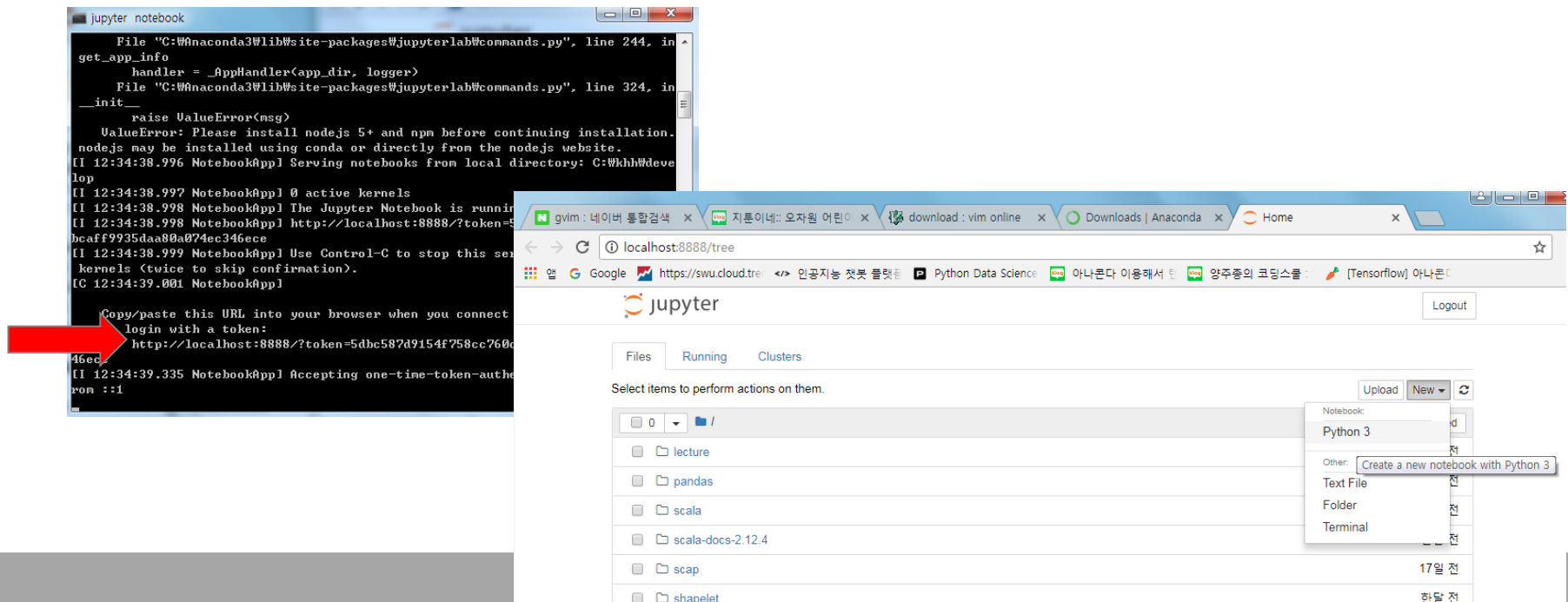


# Jupyter notebook

## ▶▶ 시작메뉴 &gt; Anaconda3 &gt; Anaconda Prompt

## ▶▶ 명령창에서 “jupyter notebook” 실행

- 접속 URL 뜨고 (<http://localhost:8888>), 기본 브라우저 자동으로 뜸
- 기본 경로는 “jupyter notebook” 을 실행한 디렉토리가 됨
- 종료는 명령창에서 “Ctrl-c” 입력
- 시작메뉴에서도 실행 가능함



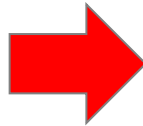
# | Hello, world!

## ▶▶ python 콘솔

```
c:>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, world!")
Hello, world!
>>>
```

## ▶▶ hello.py 실행

```
# c:\hello.py
print('Hello, world!')
```



```
c:>python hello.py
Hello, world!
c:>
```

## ▶▶ help()

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

## ▶▶ dir()

```
>>> dir(print)
['__call__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__form
at__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_s
ubclass__', '__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__', '
__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__self__', '__setattr
__', '__sizeof__', '__str__', '__subclasshook__', '__text_signature__']
```

## 2. 파이썬 기초

▶ 자료형, 문법, ...

# | 기본 자료형

## ▶ 파이썬이 기본으로 제공하는 자료형은?

- 정수 : 1, -1, 0, -999, 36473
- 실수 : 1.0, -1.0, 0.0, -11.1, 2e-3(0.002) → 1.0 과 1. 은 같음
- 문자열 : '1', "-1", 'python is good', 'a' → 한문자형은 없음
- 튜플 : (1,'a'), (1,2,3,4) → 값 변경 안됨
- 리스트 : [1,'a'], [1,2,3,4] → 값 변경/추가/삭제 됨
- 딕셔너리(사전형) : {1: 'a', 'a': 2}, {'name': 'Python', 'age': 1} → 키/값
- 집합(set) : {1,2,3}
- 기타 : Bool형(True/False), None(값없음)

# 정수

## ▶ 정수는 양수, 음수, 0 이다

```
>>> 1
1
>>> type(1)
<class 'int'>
>>> 1.0
1.0
>>> 3/2
1.5
>>> 3//2
1
>>> 3%2
1
>>> 1*2-3+4/5
-0.19999999999999996
>>> 2**3
8
>>> int(3.14), int('1.234')
(3, 1)
```

- 정수의 정식명칭은 'int' 이다
- Short/int/long 등의 구분이 없다  
(자동으로 크기를 조절한다)
- 정수의 최대값, 최소값은 신경쓸 필요가 없다  
(관심있는 사람은 <https://stackoverflow.com/questions/7604966/maximum-and-minimum-values-for-ints>)
- 정수형 변환은 int(...)
- 주의!!!> 파이썬2 에서 3/2 는 1이다.  
➔ 포팅시 주요한 에러발생 부분임



# 실수

## ▶ 실수도 양수, 음수, 0.0 이다

```
>>> 1.
1.0
>>> a=2.1
>>> type(a)
<class 'float'>
>>> 3.1/2.4, 3.1//2.4, 3.1%2.4
(1.2916666666666667, 1.0, 0.7000000000000001)
>>> a=2e3
2000.0
>>> a=2.1e-3
0.0021
>>> a**3.4
7.862227423286072e-10
>>> max_v, min_v = float('inf'), float('-inf')
>>> float(3), float('3.14')
(3.0, 3.14)
```

- 실수의 정식명칭은 'float' 이다
- Float, double 등의 구분이 없다 (자동으로 크기를 조절한다)
- 최대값, 최소값으로 초기화가 필요할때 float('inf'), float('-inf') 를 사용하자
- 실수형으로의 변환은 float(...) 사용

# 수치형 모듈

## ▶ math, random

```
>>> import math
>>> dir(math)
>>> math.pi
3.141592653589793
>>> math.sqrt(25)
5.0

>>> import random
>>> dir(random)
>>> random.random()
0.09622399721198305 # [0,1)
>>> random.randint(0,10)
7
>>> random.uniform(3,5)
3.9843029522897044
```

- 모듈을 사용할 때는 "import [모듈명]"
- 모듈의 목록 출력 : dir(모듈명)
- math → <https://docs.python.org/3/library/math.html>
- random → <https://docs.python.org/3/library/random.html>

## ▶▶ “ 또는 ” 로 묶인 문자들이다

```
>>> 'Hello, world!'
'Hello, world!'
>>> "He said, 'GOOD'"
"He said, 'GOOD'"
>>> 'I\'m good.'
"I'm good."
>>> s='''I am
... Tom'''
>>> print(s)
I am
Tom
>>> type(s)
<class 'str'>
>>> 'py'+'thon'
'python'
>>> 'python'*3
'pythonpythonpython'
>>> str(3.14**2)
'9.8596'
>>> 'python' + 3
TypeError: must be str, not int
>>> 'python' + str(3)
'python3'
```

- 문자열의 정식명칭은 'str' 이다
- 한문자를 위한 기본형은 없다 (모두 str)
- ' ', " ", "" "", "" "" "" 네가지 형태임
- 이스케이프 문자 → w', w", ww, wt, wn, wr
- 문자열로의 변환은 str(...) 사용
- 정수, 실수 등 문자열이 아닌 것과 + 안됨  
→ str() 으로 변환해야 함

## ▶ 문자열 출력 : print()

```
>>> print('Python',3)
Python 3
>>> print('Python',3,sep='-',end=' '); print('END')
Python-3END
>>> print('Python%d' % 3)
Python3
>>> print('Python %d.%.1f is %s' % (3,6.4,'good'))
Python 3.6.4 is good
>>> print('[%10.2f / %-10.2f]' % (3.1415,3.1415))
[      3.14 / 3.14      ]
>>> print('{} {}'.format(1,2))
1 2
>>> print('{1} {0}'.format(1,2))
2 1
```

- print() 안의 쉼표는 빈칸 하나이다
- print() 의 sep, end 옵션을 사용해보자
- %d(정수), %f(실수), %s(문자열)

## ▶ 문자열 처리 함수들

```
>>> len('Python')
6
>>> '-'.join('abcd')
'a-b-c-d'
>>> s='dooly,1988,seoul'
>>> s.split(',')
['dooly', '1988', 'seoul']
>>> ':'.join(_)
'dooly:1988:seoul'
>>> s=' 1:2:3:4 \n'
>>> s2=s.strip().split(':')
>>> s2='-'.join(s2)
'1-2-3-4'
>>> s='Python'
>>> s[0], s[1], s[-1], s[:3]
('P', 'y', 'n', 'Pyt')
>>> for i in s: print(i)
```

- 문자열의 길이는 len() 사용 → len() 는 리스트, 튜플, 디렉토리 등에도 사용
- 데이터파일을 읽을때, strip(), split(), join() 함수 자주 사용함
- strip(), lstrip(), rstrip() 등 참조
- 문자열은 튜플이나 리스트처럼 한 항목씩 다룰수 있다
- <https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>에서 다양한 함수들을 찾아보자

## ▶ 문자를 하나씩 처리하기 - ord(), chr()

```
>>> s='abcd'
>>> list(s)
['a', 'b', 'c', 'd']

>>> ord('a'), ord('z')
(97, 122)
>>> chr(98), chr(ord('z')-1)
('b', 'y')

>>> alphabet=[chr(ord('a')+i) for i in range(26)]

>>> for c in 'dooley':
...     print(ord(c)-ord('a'),end=' ')
...
3 14 14 11 4 24 >>>
```

# 튜플(tuple)

## ▶ 튜플은 () 로 묶은 가장 간단한 데이터 묶음이다

```
>>> 1,2
(1, 2)
>>> (1,2)
(1, 2)
>>> 1,'2',(3,4)
(1, '2', (3, 4))
>>> len(_), type(_)
(3, <class 'tuple'>)
>>> a,b=1,2
>>> (c,d)=(1,2)
>>> a,b,c,d
(1, 2, 1, 2)
>>> v=(1,2)
>>> v
(1, 2)
>>> v[0], v[1], v[-1]
(1, 2, 2)
>>> v[0]=11
TypeError: 'tuple' object does not support item assignment
>>> tuple([1,2,3])
(1,2,3)
```

- 튜플의 정식명칭은 'tuple' 이다.  
튜플은 () 로 묶는다
- 기본적으로, 쉼표로 구분된 값들은 튜플로 인식된다
- 튜플의 요소는 어떤 것이라도 올 수 있다. → 다른 튜플, 리스트, 기본형, 리스트의 리스트 등등
- 튜플의 길이는 len()
- 튜플의 값들은 수정/삭제 할 수 없다!!
- 튜플의 처리방식은 뒤에 나오는 리스트와 유사하다

# | 리스트(list)

## ▶ 리스트는 [ ] 로 묶은 파이썬 기본 데이터 묶음이다

```
>>> l=[1,2,3,4]
>>> l
[1,2,3,4]
>>> len(l), type(l)
(4, <class 'list'>)
>>> l2=[ 1, [2,3], [[4,'a'],[5,'b']] ]

>>> for i in l:
...     print(i)
...
1
2
3
4

>>> l[0], l[1], l[2], l[3], l[-1]
(1, 2, 3, 4, 4)
>>> l[:], l[:2], l[2:], l[0:3], l[1:-1]
([1, 2, 3, 4], [1, 2], [3, 4], [1, 2, 3], [2, 3])
>>> l[-1] = 99 # [1,2,3,99]

>>> l=[1,2,3]
>>> l2=l[::-1] # [1,3]
>>> l3=l[::-2] # [3,2,1]
```

- [ ] 로 묶는다. 값은 중복 가능하다

- len(), type()

- 리스트 요소는 어떤 것이든 올 수 있다

- 리스트 값은 수정/삭제/변경 가능하다

- 리스트의 각 요소는 반복적으로 처리할 수 있다

- 범위를 지정할 때는 : 기호를 사용한다 (슬라이싱)

- 빈리스트 생성 → l=[]



# | 리스트(list)

## ▶▶ 리스트 연산

```
>>> l=[1,2,3,4]
>>> l + 5
TypeError: can only concatenate list (not "int") to list
>>> l + [5]
[1, 2, 3, 4, 5]
>>> l*2
[1, 2, 3, 4, 1, 2, 3, 4]
>>> ll=[[1,2],[3,4]]
>>> ll[0]
[1, 2]
>>> ll[0][1]
2

>>> for l in ll:
...     for i in l:
...         print(i,end='- ')
...     print()
...
1-2-
3-4-
```

- 리스트를 기본형과 더할 수 없다  
리스트는 리스트 끼리 더한다
- `ll[0][1] == (ll[0])[1]`

# | 리스트(list)

## ▶ 리스트 처리 함수

```
>>> l=[1,2,3]
>>> l.append(4) # [1,2,3,4]
>>> del l[1] # [1,3,4]

>>> l2=[3,1,2]
>>> sorted(l2) # copied
[1, 2, 3]
>>> l2
[3, 1, 2]
>>> l2.sort() # modified
>>> l2
[1, 2, 3]

>>> l=[1,2,3]
>>> l.reverse() # l==[3,2,1]

>>> 1 in l, 5 in l, 5 not in l
(True, False, True)
```

- 가장 중요한 함수는 append() 이다.
- sorted(l) 과 l.sort() 는 다르다
- 기타 함수들 → index(), insert(), remove, pop(), count(), extend()
- <https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range> 참조

# | 리스트(list)

## ▶ 리스트 참조/복사 이슈

```
>>> l1=[1,2,3]
>>> l2=l1+[99]
>>> l2
[1, 2, 3, 99]
>>> l1
[1, 2, 3]

>>> l=[1,2,3]
>>> l2=l[:-1] # l2 is copied
>>> l2[0]=-1 # l=[1,2,3]

>>> l=[1,2,3]
>>> l1=[1,1]
>>> l1[0][-1]=99
>>> l1
[[1, 2, 99], [1, 2, 99]]
>>> l1=[l.copy(),l.copy()] # l1=[l[:],l[:]]

>>> lists = [[]] * 3
>>> lists
[[], [], []]
>>> lists[0].append(3)
>>> lists
[[3], [3], [3]]
```

- 리스트를 조합할 때는 조심해야 한다
- 슬라이싱은 새로운 리스트로 복사한다
- 하지만, 아래와 같이 리스트를 참조할 경우 원하지 않는 버그가 발생할 수 있다
- 항상! 확인하는 습관을 들이자

# | 리스트(list)

## ▶ 동적 생성 및 고급 데이터 처리

```
>>> mylist = []      # 처음에 빈 리스트를 만들고 채워 나간다
>>> for i in range(3):
...     mylist.append([i,i**2,i**3])
...
>>> mylist
[[0, 0, 0], [1, 1, 1], [2, 4, 8]]

>>> # 대용량이나 고급 데이터 처리를 위해서는 numpy 모듈을 주로 사용한다
>>> # 그러므로, 리스트 처리기능을 너무 깊게 까지는 알 필요가 없다
>>> import numpy as np
>>> myarray=np.array(mylist)
>>> myarray
array([[0, 0, 0],
       [1, 1, 1],
       [2, 4, 8]])
>>> myarray[:,0]=[100,101,102]
>>> myarray
array([[100,  0,  0],
       [101,  1,  1],
       [102,  4,  8]])
```

# | 딕셔너리(dict)

## ▶ 딕셔너리는 {키:값, ...} 형태의 데이터 묶음이다

```
>>> d={'name': 'dooley','birth': 1988, 'loc': 'Seoul'}
>>> d=dict([('name','dooley'),('birth',1988),('loc','Seoul')])
>>> len(d), type(d)
(3, <class 'dict'>)
>>> d[0]
KeyError: 0
>>> d['name']
'dooley'
>>> d['name']='Dooley'
>>> d['animal']=True
>>> d
{'name': 'Dooley', 'birth': 1988, 'loc': 'Seoul', 'animal': True}
>>> del d['animal']

>>> d.keys()
dict_keys(['name', 'birth', 'loc'])
>>> d.values()
dict_values(['Dooley', 1988, 'Seoul'])
>>> d.items()
dict_items([('name', 'Dooley'), ('birth', 1988), ('loc', 'Seoul')])
>>> list(d.items())
[('name', 'Dooley'), ('birth', 1988), ('loc', 'Seoul')]
```

- 리스트는 {} 로 묶은, 키/값 쌍이다.
- 키는 숫자, 문자열이 올 수 있다
- 값은 튜플, 리스트 등 모든 것이 올 수 있다
- 키는 중복될 수 없다

# | 딕셔너리(dict)

## ▶▶ 딕셔너리 탐색 방법

```
>>> for k in d: # d.keys() 와 같음
...     print(k,d[k])
...
name Dooley
birth 1988
loc Seoul

>>> for k,v in d.items():
...     print(k,v)
...
name Dooley
birth 1988
loc Seoul

>>> 'animal' in d # 키로 검색
False
>>> sorted(d) # 키를 정렬한 결과
['birth', 'loc', 'name']
```

# | 집합(set)

## ▶ set 은 중복을 허용하지 않고 순서가 없다

```
>>> s=set([1,2,2,3,3,4])
>>> s={1,2,2,3,3,4}
>>> s
{1, 2, 3, 4}
>>> len(s), type(s)
(4, <class 'set'>)

>>> s | {4,5,6}
{1, 2, 3, 4, 5, 6}
>>> s & {4,5,6}
{4}
>>> s - {4,5,6}
{1, 2, 3}
```

- 집합은 자주 사용되지는 않지만
- 리스트에서 중복된 값을 제거할 때 가끔 사용한다

# | 기타 기본형

## ▶▶ Bool, None

```
>>> b=True
>>> b
True
>>> 1>2, 1<2, 1==1, 1!=1, 1 in (1,2)
(False, True, True, False, True)
>>> bool(1), bool(0)
(True, False)
>>> int(True), int(False)
(1, 0)
>>> True+1, False+1
(2,1)

>>> a=None
>>> a==None
True
>>> if a==None: a=1
```

- True, False 는 산술식에서 1, 0 으로 계산함
- 변수 초기화가 필요할 때 None 사용함



# 변수

## ▶ 변수는 “실체”를 가리키는 화살표이다

```
>>> a=1          # 1 이라는 실체가 만들어지고 a란 변수가 그 실체를 가리킴
>>> a=2          # 1 이란 실체가 사라지고, 2란 실체가 만들어짐
>>> l=[1]*1000
>>> l=[]          # 이전의 큰 실체가 메모리에서 사라짐 (l=None 도 방법임)
>>> l=3.14        # 변수는 정해진 타입이 없어 할당이 자유롭다

>>> a=[1,2,3]
>>> b=a           # b는 a가 가리키는 실체인 [1,2,3]을 가리킴
>>> a[0]=-1
>>> a
[-1, 2, 3]
>>> b             # b가 가리키는 실체의 값이 변했다
[-1, 2, 3]

>>> a,b,c=(1,2,3)
>>> a,b,c=[1,[2,3],[4,5,6]] # a=1, b=[2,3], c=[4,5,6]
>>> d=c.copy()    # 연관되지 않고 값을 얻는 가장 안전한 방법 (* d=c[:])

>>> sum=0
>>> sum([1,2,3])  # 에러! 예약어를 변수명으로 사용하면 곤란해짐
```

# 블럭, 인덴트

## ▶ 파이썬에는 블럭을 감싸는 {} 등이 없다

```
>>> if True:
...     if True:                # 일반적으로 tab으로 구분함
...         print('a')
a
```

```
>>> if True:
...     print('a')              # 공백 한개로 인덴트
...     if True:               # 공백 한개로 인덴트
...         print('b')         # 공백 한개와 탭으로 인덴트
a
b
```

```
'''
```

편집기에 따라 탭을 공백으로, 공백을 탭으로 자동 변환하는 경우가 있다.  
소스를 편집기에서 작성하는 경우 인덴트가 일치하도록 주의하여야 한다.

```
'''
```

# | if/elif/else

## ▶▶ 조건문

```
>>> n=99
>>> if n<0:                # 블록의 시작에는 : 를 써줘야 한다
...     print('음수')
... elif n==0:
...     print('영')
... else:
...     print('양수')
...
양수

>>> if n<0:
...     print('음수')
... else:
...     if n==0:
...         print('영')
...     else:
...         print('양수')

>>> if n>0: print('양수'); print(n) # 한줄에 적을수 있다 (좋은 방법은 아님)

>>> m = 0 if n==0 else 1      # else 가 없으면 에러임
>>> m = 0 if n==0 else (-1 if n<0 else 1) # m==1
```

# | if/elif/else

## ▶▶ 조건식

```
>>> n=1
>>> n>0 and n<10
True
>>> n>0 or n<0
True
>>> not n>0
False
>>> n in range(10)
True
>>> n not in [1,2,3,4]
False
>>> 'z' in 'xyz'
True
```

# | for

## ▶▶ for 문과 range()

```
>>> for i in range(3): # 0,1,2 (기본적으로 0부터 시작함)
...     print('%03d 번째' % (i+1))
...
001 번째
002 번째
003 번째

>>> range(3) # range() 는 리스트가 아니다 (특수한 반복자임)
range(0, 3)
>>> list(range(3))
[0, 1, 2]
>>> list(range(2,10,2)) # 2부터 10까지 2칸 간격으로 (마지막 10은 제외됨)
[2, 4, 6, 8]
>>> list(range(5,0,-1)) # 마지막 0은 제외됨
[5, 4, 3, 2, 1]

>>> # 연습문제: 100보다 작은 홀수를 모두 출력하자
```

# | for

## ▶▶ for 문과 리스트

```
>>> import random
>>> l=[random.random() for i in range(10)]
>>> l
[0.673748147980991, 0.41496215011204507, 0.5461521022064907, 0.6218571281374634,
0.14242357827173302, 0.4859597766794881, 0.23027322725524113, 0.25336541301128634,
0.41356443283298416, 0.04439825389773333]
>>> for i in range(len(l)):
...     if l[i]>0.5: print(i,l[i])
...
0 0.673748147980991
2 0.5461521022064907
3 0.6218571281374634

>>> for i in l:
...     if i>0.5: print(i)
0.673748147980991
0.5461521022064907
0.6218571281374634

>>> for i in (1,2,3): print(i)    # 튜플도 반복자임
```

# | for

## ▶ 리스트 내포 (list comprehension)

```
>>> import random
>>> l = [random.random() for i in range(10)]
>>> l = [i+1 for i in [0,1,2,3]]
[1, 2, 3, 4]

>>> l = [i for i in range(100) if i%2==0]
[0, 2, ..., 98]

>>> ll = [(i,j,i*j) for i in range(1,10) for j in range(1,10)] # 구구단
>>> ll
[(1, 1, 1), (1, 2, 2), (1, 3, 3), ..., (9, 9, 81)]

>>> ll = [ [(1 if i==j else 0) for j in range(3)] for i in range(3) ]
>>> ll
[[1, 0, 0], [0, 1, 0], [0, 0, 1]] # 대각행렬

>>> ll = [ [1 if (i*(4-i)*j*(4-j)==0) else 0 for j in range(5)] for i in range(5) ]
>>> for l in ll: print(l) # 테두리가 모두 1인 5X5 행렬
...
[1, 1, 1, 1, 1]
[1, 0, 0, 0, 1]
[1, 0, 0, 0, 1]
[1, 0, 0, 0, 1]
[1, 1, 1, 1, 1]
```

# | for

## ▶ 반복문 제어와 while

```
>>> for i in range(10):
...     if i==3: break
...     print(i)
...
0
1
2

>>> for i in range(10):
...     if i%2==0: continue
...     print(i)
...
1
3
5
7
9

>>> i=0; do_run=True
>>> while do_run:      # while 문은 되도록 사용하지 않는다. (무한루프 사용시 적용)
...     print(i)
...     i+=1
...     if i==5: do_run=False
```



# | for

## ▶ enumerate() 와 zip()

```
>>> import random
>>> l = [random.randint(10,20) for i in range(10)]
>>> for i,v in enumerate(l): # (0,v1), (1,v2), (2,v3), ...
...     if v>0.5: print(i,v)
0 12
1 15
...
8 11
9 12

>>> l1=[1,2,3]
>>> l2=['a','b','c']
>>> zip(l1,l2)
<zip object at 0x000000000294AD08>
>>> list(_)
[(1, 'a'), (2, 'b'), (3, 'c')]
>>> for i,j in zip(l1,l2):
...     print(i,':',j)
...
1 : a
2 : b
3 : c
>>> for z in zip(l1,l2):
...     print(z[0],':',z[1])
```

# | for

## ▶▶ 딕셔너리 반복문

```
>>> d={'a':1, 'b':3, 'c':2}
>>> for i in d:    # d.keys() 와 같음
...     print(i)
...
a
b
c
>>> list(d)
['a', 'b', 'c']

>>> d2=[(v,k) for k,v in d.items()]; print(d2)
[(1, 'a'), (3, 'b'), (2, 'c')]
>>> [(v,k) for k,v in sorted(d2)]
[('a', 1), ('c', 2), ('b', 3)]

>>> dict(sorted(d.items(),key=lambda x: x[1]))
{'a': 1, 'c': 2, 'b': 3}
```

## ▶ 함수 정의는 def

```
>>> def mysum(a,b):  
...     return a+b  
...  
>>> mysum(1,2)  
3  
>>> mysum('a','bcd')  
'abcd'  
>>> mysum([1,2],[3,4])  
[1, 2, 3, 4]  
  
>>> def mysum2(a,b,c=0,d=0): # 기본값 설정  
...     return a+b+c+d  
...  
>>> mysum2(1,2)  
3  
>>> mysum2('a','b')  
TypeError: must be str, not int  
>>> mysum2(1,2,3)  
6
```

# 함수

## ▶▶ 가변 변수

```
>>> def f(a,*b):
...     print(b,type(b))
...     return a+sum(b)      # 결과를 넘길때는 return 사용
...
>>> f(1,2,3,4)
(2, 3, 4) <class 'tuple'>
10
>>> f(*[1,2,3,4])      # *[1,2,3,4] → 1,2,3,4 로 변환
>>> f(*range(5))

>>> def f2(a,**b):
...     print(a,'=',b,type(b))
...
>>> f2('apple',color='red',fruit=True)
apple = {'color': 'red', 'fruit': True} <class 'dict'>

>>> def f3(a,b,c='###'):
...     print(a,b,c)
...
>>> f3(c='!!!',b='Trump',a='I am')
I am Trump !!
```

## ▶ 리턴값이 여러개일 때는 어떻게 처리하나?

```
>>> l=[[1,2,3],[11,22,33],[111,222,333]]
>>> def get_info(n):
...     item=l[n]
...     return item[0]-1, item[1]-2, item[2]-3 # 리턴값이 튜플에 담겨 넘어간다
...
>>> get_info(1)
(10, 20, 30)
>>> a,b,c = get_info(0) # 0,0,0
>>> t = get_info(2) # t[0]=110, t[1]=220, t[2]=330

>>> [get_info(i)[0] for i in range(len(l))]
[0, 10, 110]

>>> def get_info2(n):
...     s = [ str(i) for i in l[n] ]
...     return ','.join(s) # 문자열로 결합해서 리턴
...
>>> get_info2(1)
'11,22,33'
>>> get_info2(1).split(',')
['11', '22', '33']
```

# 함수

## ▶ 함수 안의 변수와 함수 밖의 변수

```
>>> n=99
>>> def f1(x):
...     return n+x    # 함수 안에서 외부의 변수를 읽을 수 있다
...
>>> f1(1)
100

>>> def f2(x):
...     n=0           # 함수 안에서 외부변수와 같은 이름의 변수를 만들면 지역변수가 된다
...     return n+x
...
>>> f2(1)
1
>>> n
99

>>> def f3(x):
...     global n      # global 로 지정하면 외부 변수의 값을 변경할 수 있다
...     n+=1
...     return n+x
...
>>> f3(1)
101
>>> n
100
```

- 함수 안에서는 되도록이면 외부변수를 참조하지도 변경하지도 같은 이름의 지역변수도 만들지 말자.
- 참조가 필요하면 함수의 인자로 넘겨주자

## ▶ 함수 재정의와 람다(lambda) 함수

```
>>> def f(a,b,c='$'):  
...     print(a,b,c)  
...  
>>> def myf(a): f(a,b='#')  
...  
>>> myf(1)  
1 # $  
  
>>> f = lambda a,b,c: a+b+c  
>>> f(1,2,3)  
6  
>>> sorted([1,2,3,4,5],key=lambda x: abs(3.1-x))  
[3, 4, 2, 5, 1]  
>>> def myf(n,f):  
...     return f(n)  
...  
>>> myf(3,lambda x: x**2-1)  
8  
>>> import math  
>>> myf(3,math.sqrt)  
1.7320508075688772
```

- 람다함수는 임시로 또는 간단하게 함수를 만들때 사용한다
- lambda [인수들]: 리턴값
- 함수의 인자로 함수를 넘길 수 있다

# 파일

## ▶ 파일 읽기/쓰기 기초

```
>>> f=open('temp.txt','w')
>>> f.write('안녕하세요\n반가워요')
5
>>> f.close()      # open() 한 후에는 반드시 close() 로 파일을 닫아준다
>>> f=open('temp.txt')      # f=open('temp.txt','r')
>>> f.read()
'안녕하세요\n반가워요'
>>> f.close()

>>> import os
>>> os.getcwd()      # 명령창에서 "python" 을 실행한 경로 출력
'C:\\develop\\python'
>>> f=open('c:/develop/python/temp.txt')
>>> f=open('c:\\develop\\python\\temp.txt')
>>> f=open(os.getcwd()+'/temp.txt')

>>> f=open('some.txt',encoding='utf-8')
```

- 파일에 한글이 들어있을 때 open() 에서 에러가 발생하는 경우가 있다.
- 이런 경우는 한글이 'utf-8' 인코딩 방식으로 저장되어 있는 경우가 대부분이다.
- 이 경우, **encoding='utf-8'** 옵션을 추가하자



# 파일

## ▶ 파일 읽는 방법

```
>>> chars=[['둘리','동물',1988,0],['또치','외계인',1111],['고길동','사람',1969]]
>>> f=open('둘리와 친구들.txt','w')
>>> for c in chars:
...     f.write('%s,%s,%d\n' % (c[0],c[1],c[2]))    # '\n' 개행문자 주목 (CSV 파일)
>>> f.close()

>>> f=open('둘리와 친구들.txt')
>>> for line in f:                                # f 자체가 반복자이다 (한 라인씩 처리함)
...     print('#',line,'$')
...
# 둘리,동물,1988                                # 읽어온 라인에 개행문자가 포함됨을 알 수 있다
$
# 또치,외계인,1111
$
# 고길동,사람,1969
$
>>> f.close()

>>> for line in f:
...     print('#',line.strip().split(',') ['$')    # strip() 으로 개행문자 제거
...
# ['둘리', '동물', '1988'] $                    # 모든 항목이 문자열임에 주목 ('1988')
# ['또치', '외계인', '1111'] $
# ['고길동', '사람', '1969'] $
```

## ▶ 파일 읽는 방법 (계속)

```
>>> f = open('둘리와 친구들.txt')
>>> ll = []
>>> for line in f:
...     l = line.strip().split(',')
...     l[1] = 0 if l[1]=='사람' else (1 if l[1]=='동물' else 2)
...     l[2] = int(l[2])
...     ll.append(l)
...
>>> ll
[['둘리', 1, 1988], ['또치', 2, 1111], ['고길동', 0, 1969]]
>>> f.close()

>>> # 만일 모든 데이터가 숫자라면
>>> for line in f:
...     l = [int(i) for i in line.strip().split(',')]
...     ll.append(l)
```

- CSV 파일에서 데이터를 읽어들이기 때,  
고급기능이 필요한 경우 아래의 함수들을  
주로 사용한다.
  - ➔ numpy 모듈의 loadtxt()
  - ➔ pandas 모듈의 read\_csv()

# 파일

## ▶▶ with 문

```
>>> with open('둘리와 친구들.txt') as f:  
...     f.read()  
...  
'둘리, 동물, 1988\n또치, 외계인, 1111\n고길동, 사람, 1969\n'
```

with 사용시 close() 를  
호출할 필요가 없다

## ▶▶ 파이썬 객체 저장 - pickle

```
>>> import pickle  
>>> l=[1,2,3,4,5]  
>>> with open('1.pkl','wb') as f:  
...     pickle.dump(l,f)  
...  
>>> with open('1.pkl','rb') as f:  
...     l2=pickle.load(f)  
...  
>>> l2  
[1, 2, 3, 4, 5]
```

'wb' 바이트 형식으로 열음  
pickle.dump() 로 객체 저장

'rb' 바이트 형식으로 열음  
pickle.load() 로 객체 읽음

# 소스파일

## ▶ 파이썬의 소스파일은 “\*.py” 이다

```
# c:\python\hello.py
```

주석

```
print('Hello, world!')
```

메모장/Notepad++  
등에서 작성

```
n=1
```

```
for i in range(3):
```

```
    print(i,':',i*i)
```

탭

```
c:\python>python hello.py
```

```
Hello, world!
```

```
1 : 1
```

```
2 : 4
```

```
3 : 9
```

```
c:\python> cd \
```

```
c:\>python python\hello.py
```

```
Hello, world!
```

```
1 : 1
```

```
2 : 4
```

```
3 : 9
```

```
c:\>
```

# 소스파일

## ▶ 주석 처리

```
# c:\python\hello.py
```

```
import math
```

math 모듈을  
불러옴

```
print('Hello, world!')
```

```
...
```

```
n=1
```

```
for i in range(3):
```

```
    print(i,':',i*i)
```

```
...
```

실행하지 않을 부분을  
''' ''' 또는 """ """ 로 감싼다

```
# l=[1,2,3]
```

한 라인 주석

```
l=list(range(5))
```

```
l=[ math.square(i) for i in l ]
```

```
print(l)
```

```
c:\python>python hello.py
```

```
Hello, world!
```

```
[0.0, 1.0, 1.4142135623730951, 1.7320508075688772, 2.0]
```

```
c:\python>
```

# 소스파일

## ▶ 인자 넘기기 / ipython

```
# arg.py  
import sys  
arg=sys.argv  
print(arg)  
print(len(arg),type(arg))  
  
arg_list=[ int(i) for i in arg[1:] ]  
print(arg_list)
```

인자들의 리스트

문자열을 정수로 바꿈

```
c:\python>python arg.py 1 2 3  
['arg.py', '1', '2', '3']  
4 <class 'list'>  
[1, 2, 3]
```

```
C:\python>ipython  
In [1]: run arg.py 1 2  
['arg.py', '1', '2']  
1 <class 'list'>  
[1, 2]
```

ipython 사용시, 코딩창에서  
run 명령 사용 가능

# 소스파일

## ▶ 명시적인 실행절차

```
# arg.py

import sys

def main():
    arg=sys.argv
    print(arg)
    print(len(arg),type(arg))

    arg_list=[ int(i) for i in arg[1:] ]
    for i in arg_list: print(i,end=' ')

if __name__ == "__main__":
    main()
```

main() 함수 정의

명령창에서 실행했음을 확인

```
>>> import arg          # arg.py 를 불러온다 → main() 부분이 실행되지 않음
>>> arg
<module 'arg' from 'C:\\khh\\develop\\lecture\\arg.py'>
>>> arg.main()
['']
1 <class 'list'>
```

이와 같이 소스파일을 모듈형태로 불러올때 원치않는 실행을 막을 수 있다

# | 클래스(class)

## ▶▶ 클래스에 대해서는 자세히 설명하지 않는다. 이유는...

- 파이썬은 객체지향 언어이며, 클래스의 기능도 자바에 못지않다.
- 하지만, 파이썬을 쓰는 주요 목적은 빠른 시제품을 만들고 알고리즘을 테스트하는 것이다.
- 클래스를 사용하면 커다란 장점도 많지만, 클래스간의 의존성을 주의깊게 체크해야 하고 가독성을 떨어뜨릴 가능성이 크다.
- 일단 클래스에 대해서는 기본적인 기능만 익히고, 향후 여러 개발자와 협력해야 하거나 개발 마지막 단계에서 패키징 할 때 상세한 내용을 공부하는 것이 좋을 것이다.



# | 클래스(class)

## ▶ class 를 정의할 때, self 지시자를 주목하자

```
# myclass.py
```

```
class MyClass:
```

```
    def __init__(self, name, count):
```

```
        self.name = name
```

```
        self.count = count
```

```
    def info(self):
```

```
        return ('MyClass: name=%s, count=%d' % (self.name, self.count))
```

```
    def __str__(self):
```

```
        return self.info()
```

```
    def __repr__(self):
```

```
        return 'MyClass: ' + self.info()
```

```
c=MyClass('apple',3)
```

```
print(c.info())
```

```
print(c)
```

초기화 함수 \_\_init\_\_

사용자정의 함수

print() 문에서 출력 내용

인터프리터에서 정보 출력

```
>>> import myclass
```

```
MyClass: name=apple, count=3
```

```
MyClass: name=apple, count=3
```

```
>>> myclass.c
```

```
MyClass: MyClass: name=apple, count=3
```

# | 클래스(class)

## ▶ 클래스를 정의한 소스파일을 불러오기

```
# myclass.py

class MyClass:
    def __init__(self, name, count):
        self.name = name
        self.count = count

    def info(self):
        return ('MyClass: name=%s, count=%d' % (self.name, self.count))
```

```
# runclass.py

import myclass

c1=myclass.MyClass('apple',3)
c2=myclass.MyClass('orange',2)
print(c1.count + c2.count)
```

myclass.py 를 불러온다

myclass 에 있는 MyClass 생성

```
c:\python>python runclass.py
5
```

## ▶ 파이썬 기본 모듈 불러오기

```
# module.py
```

```
import sys
```

```
import os as window
```

```
from math import sqrt, pi
```

```
from random import *
```

```
print(sys.argv)
```

```
print(window.getcwd())
```

```
print(sqrt(2*pi))
```

```
l=list(range(10)); shuffle(l)
```

```
print(l)
```

모듈 참조이름 변경

sqrt(), pi 는 모듈명 없이 사용

random 모듈에 있는 모든 변수, 함수  
모듈명 없이 사용

```
C:\python>python module.py
```

```
['module.py']
```

```
C:\khh\develop\lecture
```

```
2.5066282746310002
```

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

## ▶ 사용자 소스파일 다시 불러오기

```
# module2.py  
  
import myclass as mc  
  
import imp  
imp.reload(mc)  
  
c=mc.MyClass('apple',3)  
print(c)
```

실행환경에 따라 소스의 변경이  
반영되지 않을 수 있다.  
명시적으로 다시 불러들이는 방법이다.

```
C:\python>ipython  
In [1]: run module2.py  
MyClass: name=apple, count=3  
  
In [2]: run module2.py  
### MyClass ### name=apple, count=3
```

두번째 실행전에 module2.py의  
출력문을 바꾸었을때

# 파이썬 내장 함수

▶▶ <https://docs.python.org/3/library/functions.html>

Python Software Foundation [US] | <https://docs.python.org/3/library/functions.html>

앱 Google <https://swu.cloud.tr> 인공지능 챗봇 플랫폼 Python Data Science 아나콘다 이용해서 딥러닝 공부하기 양주종의 코딩스쿨 [Tensorflow] 아나콘다

Python » English 3.6.5 Documentation » The Python Standard Library »  Quick search Go | previous | next | modules | index

Previous topic  
1. Introduction

Next topic  
3. Built-in Constants

This Page  
Report a Bug  
Show Source

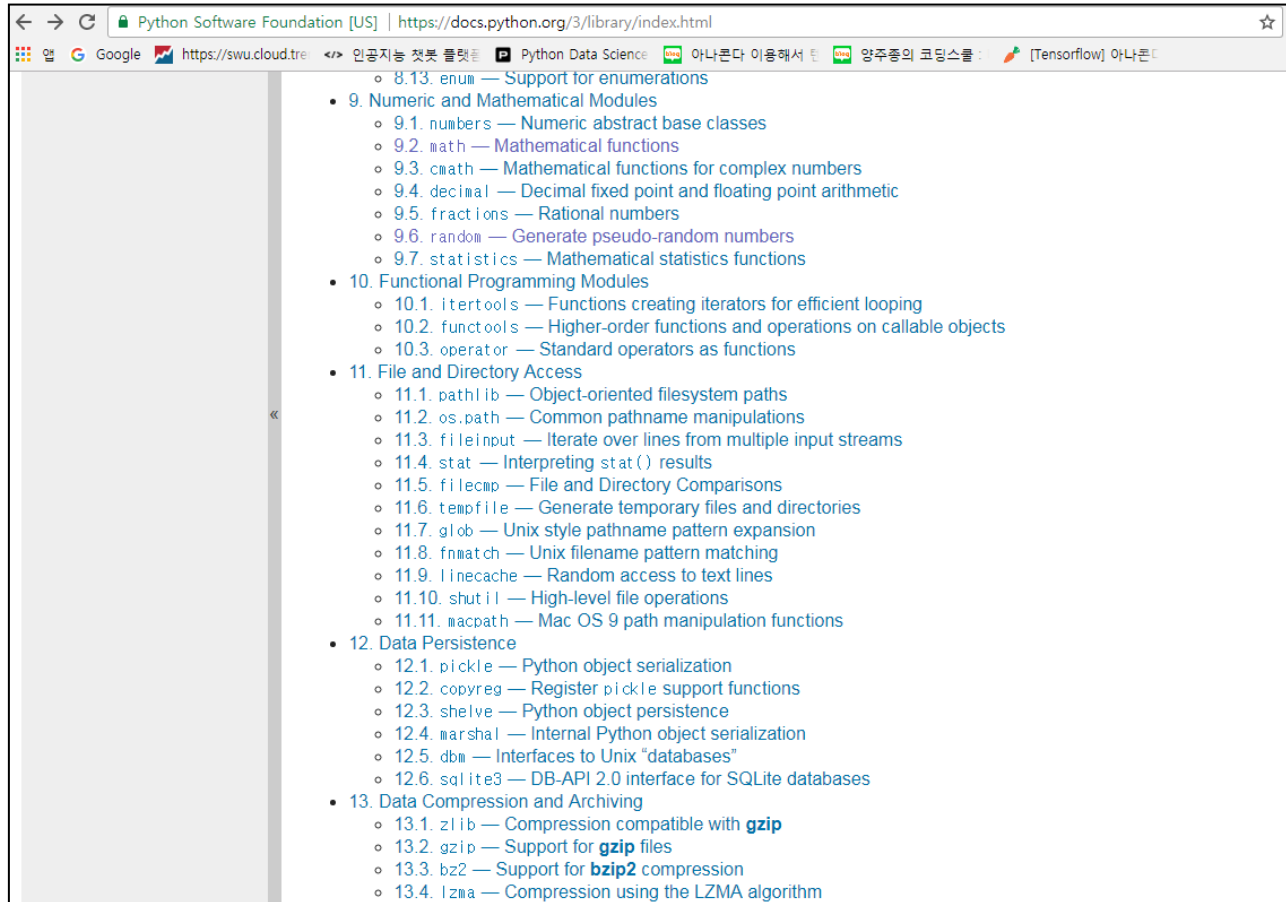
## 2. Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

Built-in Functions				
<a href="#">abs()</a>	<a href="#">dict()</a>	<a href="#">help()</a>	<a href="#">min()</a>	<a href="#">setattr()</a>
<a href="#">all()</a>	<a href="#">dir()</a>	<a href="#">hex()</a>	<a href="#">next()</a>	<a href="#">slice()</a>
<a href="#">any()</a>	<a href="#">divmod()</a>	<a href="#">id()</a>	<a href="#">object()</a>	<a href="#">sorted()</a>
<a href="#">ascii()</a>	<a href="#">enumerate()</a>	<a href="#">input()</a>	<a href="#">oct()</a>	<a href="#">staticmethod()</a>
<a href="#">bin()</a>	<a href="#">eval()</a>	<a href="#">int()</a>	<a href="#">open()</a>	<a href="#">str()</a>
<a href="#">bool()</a>	<a href="#">exec()</a>	<a href="#">isinstance()</a>	<a href="#">ord()</a>	<a href="#">sum()</a>
<a href="#">bytearray()</a>	<a href="#">filter()</a>	<a href="#">issubclass()</a>	<a href="#">pow()</a>	<a href="#">super()</a>
<a href="#">bytes()</a>	<a href="#">float()</a>	<a href="#">iter()</a>	<a href="#">print()</a>	<a href="#">tuple()</a>
<a href="#">callable()</a>	<a href="#">format()</a>	<a href="#">len()</a>	<a href="#">property()</a>	<a href="#">type()</a>
<a href="#">chr()</a>	<a href="#">frozenset()</a>	<a href="#">list()</a>	<a href="#">range()</a>	<a href="#">vars()</a>
<a href="#">classmethod()</a>	<a href="#">getattr()</a>	<a href="#">locals()</a>	<a href="#">repr()</a>	<a href="#">zip()</a>
<a href="#">compile()</a>	<a href="#">globals()</a>	<a href="#">map()</a>	<a href="#">reversed()</a>	<a href="#">__import__()</a>
<a href="#">complex()</a>	<a href="#">hasattr()</a>	<a href="#">max()</a>	<a href="#">round()</a>	
<a href="#">delattr()</a>	<a href="#">hash()</a>	<a href="#">memoryview()</a>	<a href="#">set()</a>	

# 파이썬 제공 모듈

▶▶ <https://docs.python.org/3/library/index.html>



# 3. 파이썬 고급

▶ 데이터분석, 시각화, 머신러닝

# 핵심 모듈 설치

## ▶▶ 핵심 모듈

- Numpy : 행렬형 데이터 수치계산
- Matplotlib : 데이터 시각화
- Scikit-learn : 머신러닝

## ▶▶ python3 에서 설치

- 명령창에서 “pip install [모듈명]” 과 같이 실행  
→ “pip install numpy”, “pip install matplotlib”

## ▶▶ 아나콘다는 위의 세 모듈을 포함하고 있어 새로 설치할 필요가 없음



# 데모

## ▶ numpy 와 matplotlib 예시

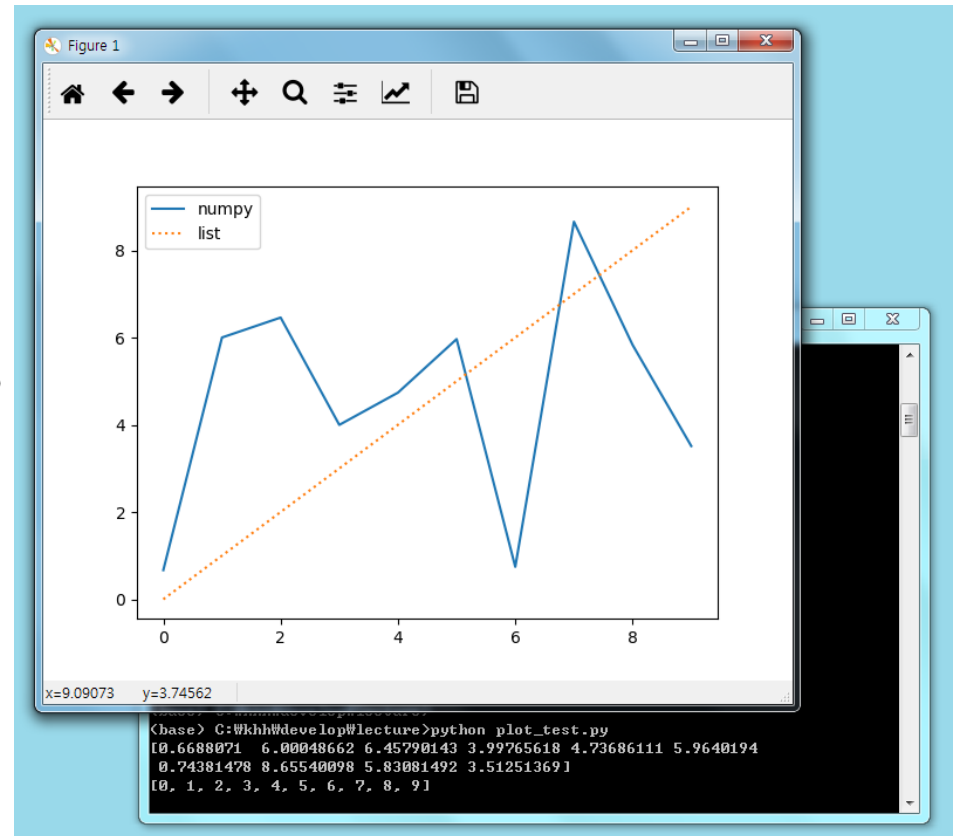
```
# plot_test.py

import numpy as np
import matplotlib.pyplot as plt

data1=np.random.random(10)*10
print(data1)

data2=list(range(10))
print(data2)

plt.plot(data1, label='numpy')
plt.plot(data2, ':', label='list')
plt.legend()
plt.show()
```



# Numpy – 메뉴얼과 레퍼런스

▶▶ 메뉴얼 ➔ <https://docs.scipy.org/doc/numpy-1.14.0/user/index.html>

The screenshot shows the 'NumPy User Guide' page. The header includes the SciPy.org logo and 'Sponsored By ENTHOUGHT'. Navigation links for 'SciPy.org', 'Docs', and 'NumPy v1.14 Manual' are present, along with 'index', 'next', and 'previous' buttons. The main title is 'NumPy User Guide'. A paragraph states: 'This guide is intended as an introductory overview of NumPy. For detailed reference contained in the package, see the NumPy Reference.' A list of links includes: Setting up, Quickstart tutorial, NumPy basics, Miscellaneous, NumPy for Matlab users, Building from source, and Using NumPy C-API. The footer contains copyright information: '© Copyright 2008-2017, The SciPy community. Last updated'.

The screenshot shows the 'NumPy Reference' page. The header is identical to the previous page. Navigation links for 'SciPy.org', 'Docs', and 'NumPy v1.14 Manual' are present, along with 'index', 'next', and 'previous' buttons. The main title is 'NumPy Reference'. It shows 'Release: 1.14' and 'Date: January 08, 2018'. A paragraph states: 'This reference manual details functions, modules, and objects included in NumPy, describing what they are and what they do. For learning how to use NumPy, see also NumPy User Guide.' A list of links includes: Array objects (with sub-links: The N-dimensional array (ndarray), Scalars, Data type objects (dtype), Indexing, Iterating Over Arrays, and Standard array subclasses). On the right, there is a 'Table Of Contents' with a link to 'NumPy Reference' and 'Acknowledgements'. Below that, 'Previous topic' is 'Beyond the Basics' and 'Next topic' is 'Array objects'.

▶▶ 레퍼런스 ➔ <https://docs.scipy.org/doc/numpy-1.14.0/reference/index.html>

# Numpy – 어레이(ndarray)

## ▶ Numpy 는 ndarray 형태의 자료구조를 처리하는 모듈이다

- ndarray 는 다차원 배열인데 2차원에서는 직사각형, 3차원에서는 직육면체 형태이다
- ndarray 는 고급 수치계산을 위해 list 기능을 확장한 것으로 볼 수 있다

```
import numpy as np

l=[1,2,3]
a=np.array(l)
print(l)
print(a)
print(type(a),len(a))
print(a.shape, a.dtype, a.ndim)

a2=np.array([[1,2,3],[4,5,6]])
print(a2)
print(len(a2), a2.size)
print(a2.shape, a2.dtype, a2.ndim)
```

기존 list 를 어레이로 변환하려면  
np.array() 함수를 사용한다

```
[1, 2, 3]
[1 2 3]
<class 'numpy.ndarray'> 3
(3,) int32 1
[[1 2 3]
 [4 5 6]]
2 6
(2, 3) int32 2
```

리스트와 어레이의  
출력형태가 다름에 주목

# Numpy – 사용이유

## ▶ 항목별로 계산할 수 있다

```
>>> import numpy as np
>>> l=[1,2,3]
>>> [i+10 for i in l]
[11, 12, 13]
>>> a=np.array([1,2,3])
>>> a + 10
array([11, 12, 13])
>>> np.sin(a)          # [math.sin(i) for i in l]
array([0.84147098, 0.90929743, 0.14112001])
>>> a + a              # [i+i for i in l]
array([2, 4, 6])
```

## ▶ 다차원 배열 계산이 용이하다

```
>>> l=[[1,2,3],[11,12,13]]
>>> a=np.array(l)
>>> a[:,0]          # [i[0] for i in l]
array([1, 11])
```

## ▶ 복잡한 수치계산을 간단하게 처리할 수 있다

```
>>> a[a.mean(axis=1)>10]+[1,2,3]
array([[12, 14, 16]])
>>> np.argsort([3,1,2,4])[::-1]
array([3, 0, 2, 1], dtype=int64)
```

## ▶ 소스가 C 로 작성되어 있어 아주 빠르다

# Numpy – dtype

## ▶ Numpy 어레이는 dtype 으로 데이터 타입을 지정한다

- 한개의 어레이는 한가지 dtype 을 가진다
- 어레이의 기본형은 float 으로 생각하자
- 사용비율 : float > int >> bool >> str

```
>>> a=np.array([1,2,3])      # np.array([1,2,3], dtype=int)
>>> a.dtype
dtype('int32')
>>> a=np.array([1.0,2,3]) # np.array([1,2,3], dtype=float)
>>> a.dtype
dtype('float64')
>>> a=np.array([1,2,3],dtype=float)
>>> a.dtype
dtype('float64')
>>> a=np.array([1.1,2.2,3.3],dtype=int)
>>> a
array([1, 2, 3])

>>> a=np.array(['a','b','c']) # dtype = str
>>> a=np.array([1,2,3],dtype='str') # array(['1', '2', '3'], dtype='<U1')
>>> a=np.array(['1','2','3'],dtype=int) # array([1, 2, 3])
>>> a=np.array([0,1,2],dtype=bool) # array([False,  True,  True])
```

# Numpy – shape

## ▶▶ 3\*4 행렬이라고 할때의 (3,4) 가 shape 이다

- 어레이는 행렬과 유사한 구조로 생각할 수 있다
- 어레이의 차원은 ndim 인데, 3차원 이상은 복잡하여 거의 사용되지 않는다

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

>>> a=np.array([1])    # a.shape==(1,), a.ndim==1
>>> a=np.array(1)      # a.shape==(), a.ndim==0

>>> a=np.array([[1],[2,3]]); a
array([list([1]), list([2, 3])], dtype=object)
```

갯수가 맞지 않을 경우  
원하지 않는 결과가 나온다

# Numpy – 기본자료형 으로 부터 생성

## ▶ 에러이는 list 와 tuple, range 로 부터 생성할 수 있다

- tuple 도 가능하나 일반적으로는 list 를 사용한다
- dict 와 set 은 사용하지 않는다

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

```
>>> a=np.array({1,2,3})
>>> a
array({1, 2, 3}, dtype=object)
>>> a=np.array({1: 11, 2: 22, 3: 33})
>>> a
array({1: 11, 2: 22, 3: 33}, dtype=object)
```

```
>>> a=np.array(range(10))    # np.arange(10) 과 같음
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

dict 와 set 을 사용하면  
원하지 않는 결과가 된다

# Numpy – 기존 어레이의 dtype 변경

## ▶▶ np.array() 함수를 사용하여 새로 생성한다

```
>>> a=np.array([1,2,3])    # [1,2,3], int32
>>> b=np.array(a, dtype=float) # [1., 2., 3.], float64
>>> c=np.array(b, dtype=str)   # ['1.0', '2.0', '3.0'], str

>>> a=np.array(a, c.dtype)    # ['1', '2', '3'], str
```

a 가 중복 사용됨을 주목

## ▶▶ a.astype() 과 np.asarray() 함수도 사용 가능하나, 되도록이면 array() 를 사용하자



# Numpy – 어레이 생성 함수(1)

## ▶ np.zeros(), np.ones(), np.eye()

```
>>> np.zeros(10)
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
>>> np.zeros(2,3)
TypeError: data type not understood
>>> np.zeros((2,3))
array([[0., 0., 0.],
       [0., 0., 0.]])
>>> np.zeros([2,3], dtype=int)
array([[0, 0, 0],
       [0, 0, 0]])

>>> np.ones([2,3,4])
array([[[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]],

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

>>> np.eye(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

실수형으로  
생성됨

대각 행렬

# Numpy – 어레이 생성 함수(2)

## ▶ np.arange()

```
>>> np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> np.arange(1,10,2)
array([1, 3, 5, 7, 9])
>>> np.arange(1,10,2,dtype=float)
array([1., 3., 5., 7., 9.])
```

range() 함수와  
사용법은 동일

```
>>> np.arange(8).reshape(2,2,2)
array([[[0, 1],
        [2, 3]],
       [[4, 5],
        [6, 7]]])
```

reshape() 함수와  
결합하면 유용함

```
>>> np.arange(8).reshape(2,-1)
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

-1 로 지정하면  
자동 할당됨

# Numpy – 어레이 생성 함수(3)

## ▶▶ np.random

- random 함수들은 테스트를 위해 자주 사용된다
- <https://docs.scipy.org/doc/numpy-1.14.0/reference/routines.array-creation.html> 참조

```
>>> np.random.rand(3,3)          # 0~1 사이의 실수
array([[0.35466016, 0.35276403, 0.7184818 ],
       [0.57738826, 0.02513179, 0.05267243],
       [0.18576663, 0.06791933, 0.17513957]])
>>> np.random.randn(2,5)         # 평균 0, 표준편차 1인 정규분포
array([[ -0.26995186,  1.78314123,  0.95798007,  0.88460443,  0.1507183 ],
       [-1.29304688,  0.46715146,  1.10038887, -1.40327091,  0.48194161]])
>>> np.random.randint(10,size=(3,3)) # 0~9 사이의 정수
array([[1, 5, 9],
       [6, 5, 5],
       [3, 5, 2]])
>>> np.random.randint(1,4,size=(3,3)) # 1~3 사이의 정수

>>> np.random.uniform(100,101,size=(2,2)) # 100~101 사이의 실수
array([[100.95531059, 100.70547873],
       [100.89672494, 100.99164855]])
>>> np.random.normal(10,2,size=(2,2))      # 평균 10, 표준편차 2인 정규분포
array([[8.23806369, 8.30636293],
       [9.20605656, 8.15973884]])

>>> np.random.choice([0,1],size=10)        # 0과 1에서 10개를 골라낸다
array([1, 1, 1, 0, 0, 1, 0, 0, 0, 1])
>>> np.random.choice(['dog','cat'],size=(2,5))
array([[ 'dog', 'dog', 'cat', 'dog', 'cat'],
       [ 'cat', 'cat', 'dog', 'dog', 'dog']], dtype='<U3')
```

# Numpy – 기본연산

## ▶ Numpy 는 list 와는 다르게 항목별로 계산한다

```
>>> a=np.arange(4).reshape(2,2)
>>> b=np.arange(10,14).reshape(2,2)
>>> a
array([[0, 1],
       [2, 3]])
>>> b
array([[10, 11],
       [12, 13]])

>>> a+b
array([[10, 12],
       [14, 16]])
>>> a-b
array([[-10, -10],
       [-10, -10]])
>>> a*b
array([[ 0, 11],
       [24, 39]])
>>> a/b
array([[0.          , 0.09090909],
       [0.16666667, 0.23076923]])
>>> a%b
array([[0, 1],
       [2, 3]], dtype=int32)
```

```
>>> a+1
array([[1, 2],
       [3, 4]])
>>> 1+a
array([[1, 2],
       [3, 4]])
>>> 1/(a+1)
array([[1.          , 0.5          ],
       [0.33333333, 0.25          ]])
>>> a**2
array([[0, 1],
       [4, 9]], dtype=int32)
```

# Numpy – 수학함수

## ▶ np.sqrt(), np.exp() 등의 수학함수 들은 항목별로 적용된다

```
>>> a=np.arange(4).reshape(2,2); a
array([[0, 1],
       [2, 3]])

>>> np.sqrt(a)
array([[0.          , 1.          ],
       [1.41421356, 1.73205081]])
>>> np.exp(a)
array([[ 1.          ,  2.71828183],
       [ 7.3890561 , 20.08553692]])
>>> np.sin(a)
array([[0.          , 0.84147098],
       [0.90929743, 0.14112001]])
>>> np.ceil(np.sin(a))
array([[0., 1.],
       [1., 1.]])
```

```
>>> a=np.random.randn(3,4)
>>> a
array([[ -0.91702662,  0.09610722, -
 0.98262525,  0.9476858 ],
       [ -0.90056783, -0.35983828, -
 0.61186193,  1.35130327],
       [ 1.71191651, -0.13786827, -
 0.45417859, -1.07308241]])
>>> np.sign(a)
array([[ -1.,  1., -1.,  1.],
       [ -1., -1., -1.,  1.],
       [ 1., -1., -1., -1.]])
>>> np.floor(a)
array([[ -1.,  0., -1.,  0.],
       [ -1., -1., -1.,  1.],
       [ 1., -1., -1., -2.]])
>>> np.abs(a)
array([[ 0.91702662,  0.09610722,
 0.98262525,  0.9476858 ],
       [ 0.90056783,  0.35983828,
 0.61186193,  1.35130327],
       [ 1.71191651,  0.13786827,
 0.45417859,  1.07308241]])
```

# Numpy – 색인과 슬라이싱

## ▶ Numpy 는 색인과 슬라이싱(범위설정) 기능에 큰 강점을 가진다

- 다차원 행렬을 다양하게 잘라내어 다양한 작업을 할 수 있다
- 슬라이싱은 복사되지 않고 뷰(참조) 를 가진다

```
>>> a=np.arange(10)
>>> a[0], a[1], a[-1], a[-2]
(0, 1, 9, 8)
>>> a[:-1]
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
>>> a[3:7]
array([3, 4, 5, 6])
>>> a[3:7] = -1
>>> a
array([0, 1, 2, -1, -1, -1, -1, 7, 8, 9])

>>> a=np.arange(10)[::-1]; a
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
>>> b=a[:,2]; b      # b는 a 일부분의 뷰
array([9, 7, 5, 3, 1])
>>> b[0]=999
>>> a
array([999, 8, 7, 6, 5, 4, 3, 2, 1, 0])
>>> a[1:4].copy()    # 뷰를 원하지 않을때
```

```
>>> a=np.arange(12).reshape(3,4); a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> a[0], a[-1]
(array([0, 1, 2, 3]),
 array([ 8,  9, 10, 11]))
>>> a[1][1], a[1,1]  # a[행, 열] 구조임
(5, 5)

>>> a[:,2:]
array([[0, 1],
       [4, 5]])
>>> a[:,2:]
array([[2, 3],
       [6, 7]])
>>> a[:,1]
array([1, 5, 9])
>>> a[:,1::2]      # 두번째, 네번째 열
```

# Numpy – 색인에서 항목별 선택

## ▶ 이런 경우를 팬시 색인이라고 한다

- 원하는 항목을 리스트에 담아 색인에 넣으면 된다
- 뒷장에 나오는 불리언 색인과 결합하여, 고급 계산에서 자주 사용됨

```
>>> a=np.arange(12).reshape(3,4)
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

>>> a[ [0,2] ]      # 첫번째, 세번째 행
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])

>>> a[:, [0,2] ]    # 첫번째, 세번째 열
array([[ 0,  2],
       [ 4,  6],
       [ 8, 10]])

>>> a[1]
array([4, 5, 6, 7])
>>> a[ [1] ] # 원 구조를 유지하면서 골라냄
array([[4, 5, 6, 7]])
>>> a[ [1,1] ]
array([[4, 5, 6, 7],
       [4, 5, 6, 7]])
```

```
>>> a=np.eye(3)
>>> a
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
>>> l=[2,0,1,0,1,2,1]
>>> a[l]      # One-Hot 인코딩 사례임
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.]])

>>> a=np.arange(12).reshape(4,3)
>>> a[ [1,3], [0,2] ] # 쌍으로 선택됨
array([ 3, 11])
>>> a[[1,3]][:,[0,2]]
array([[ 3,  5],
       [ 9, 11]])
```

# Numpy – 불리언 색인

## ▶ 사칙연산을 하듯이, 조건식을 적용하면 어떻게 될까?

- 각 항목별로 True, False 가 할당된다
- 어레이 간의 조건식에는, '|' (or) 와 '&' (and) 가 사용된다

```
>>> a=np.random.seed(1)
>>> a=np.random.randn(3,3)
>>> a
array([[ 1.62434536, -0.61175641, -
0.52817175],
       [-1.07296862,  0.86540763, -
2.3015387 ],
       [ 1.74481176, -0.7612069 ,
0.3190391 ]])
>>> a>0
array([[ True, False, False],
       [False,  True, False],
       [ True, False,  True]])
>>> (a>1) | (a<-1) # 반드시 괄호 사용할것
array([[ True, False, False],
       [ True, False,  True],
       [ True, False, False]])
>>> (a>-0.5) & (a<-.5)
array([[False, False, False],
       [False, False, False],
       [False, False, False]])
```

```
>>> a[ [True,False,True] ]
array([[ 1.62434536, -0.61175641, -
0.52817175],
       [ 1.74481176, -0.7612069 ,
0.3190391 ]])
>>> a[:, [True,False,True] ]
array([[ 1.62434536, -0.52817175],
       [-1.07296862, -2.3015387 ],
       [ 1.74481176,  0.3190391 ]])
>>> a[ a>0 ]
array([1.62434536, 0.86540763, 1.74481176,
0.3190391 ])
>>> a[ a>0 ] = 0
>>> a
array([[0., -0.61175641, -0.52817175],
       [-1.07296862, 0., -2.3015387],
       [ 0., -0.7612069, 0.]])
```



# Numpy – 행 골라내기

## ▶ 불리언 색인을 이용하여, SQL 문 처럼 원하는 행을 골라내 보자

```
>>> np.random.seed(55)
>>> a=np.random.randn(100,3)
>>> b=a.mean(axis=1)
>>> b.shape
(100,)

>>> a[ b>0 ]
array([[ 2.62653840e-01,  2.59952678e-01,
        -3.81086383e-01],
       [-2.28986282e-03,  3.41615180e-01,
        8.97572246e-01],
       ...,
       [-6.89713383e-01,  2.09307686e+00,
        1.24858567e+00]])
```

```
>>> np.random.seed(55)
>>> a=np.random.randn(100,3)
>>> b = np.random.choice
( ['one','two','three'], 100 )
>>> b.shape
(100,)

>>> a[ b=='one' ]
array([[ 1.66642853e+00, -2.00343926e+00,
        -4.77872715e-01],
       [ 8.88381844e-01,  7.22219807e-01,
        -1.38210273e+00],
       ...,
       [-1.30892478e-01, -3.27712055e-01,
        -1.07651014e-01]])
>>> a[ b=='one' ].shape
(33, 3)
```

# Numpy – 축 바꾸기

## ▶▶ N\*M 행렬을 M\*N 행렬로 바꾸자

```
>>> a=np.arange(5)
>>> a.T
array([0, 1, 2, 3, 4])
>>> a=a.reshape(1,5)
>>> a
array([[0, 1, 2, 3, 4]])
>>> a.T
array([[0],
       [1],
       [2],
       [3],
       [4]])

>>> a.reshape(5,1)
array([[0],
       [1],
       [2],
       [3],
       [4]])
```

```
>>> a=np.arange(12).reshape(3,4)
>>> a.shape
(3, 4)
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

>>> b=a.T
>>> b.shape
(4, 3)
>>> b
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

# Numpy – 통계 함수

## ▶ 평균, 표준편차, 최소, 최대 등을 구해보자

- 다차원 어레이인 경우 axis 옵션을 적용한다

```
>>> a=np.arange(12).reshape(3,4)
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> a.sum()
66
>>> a.sum(axis=0)
array([12, 15, 18, 21])
>>> a.sum(axis=1)
array([ 6, 22, 38])

>>> a.std(axis=0)
array([3.26598632, 3.26598632, 3.26598632,
       3.26598632])
>>> a.min()
0
>>> a.min(axis=1)
array([0, 4, 8])
>>> a.max(axis=1)
array([ 3,  7, 11])
>>> a.cumsum(axis=0) # 누적 합계
array([[ 0,  1,  2,  3],
       [ 4,  6,  8, 10],
       [12, 15, 18, 21]], dtype=int32)
```

```
>>> a.argmin(axis=0)
array([0, 0, 0, 0], dtype=int64)
>>> a.argmax(axis=1)
array([3, 3, 3], dtype=int64)

>>> np.random.seed(10)
>>> a=np.random.rand(10,3)
>>> b = a.mean(axis=1)>0.5
>>> b
array([False, False, False,  True, False,
       True,  True, False, False,
        True])
>>> np.where(b) # 조건에 맞는 인덱스 출력
(array([3, 5, 6, 9], dtype=int64),)
>>> np.where(b)[0]
array([3, 5, 6, 9], dtype=int64)
>>> a[a.mean(axis=1) > 0.5 ]

>>> (a>0.5).any(axis=1) # 1개 이상
array([ True,  True,  True,  True,  True,
        True,  True,  True,  True,
         True])
>>> (a>0.5).all(axis=1) # 모두
array([False, False, False, False, False,
       False,  True, False, False,
         True])
```

# Numpy – 정렬

## ▶▶ sort(), argsort()

```
>>> np.random.seed(1)
>>> data=np.random.randint(10,size=(3,4))
>>> data
array([[5, 8, 9, 5],
       [0, 0, 1, 7],
       [6, 9, 2, 4]])

>>> a=data.copy(); a.sort(); a
array([[5, 5, 8, 9],
       [0, 0, 1, 7],
       [2, 4, 6, 9]])
>>> a=data.copy(); a.sort(axis=0); a
array([[0, 0, 1, 4],
       [5, 8, 2, 5],
       [6, 9, 9, 7]])

>>> b=np.sort(data); b
array([[5, 5, 8, 9],
       [0, 0, 1, 7],
       [2, 4, 6, 9]])
>>> c=np.sort(data,axis=0); c
array([[0, 0, 1, 4],
       [5, 8, 2, 5],
       [6, 9, 9, 7]])
```

```
>>> data
array([[5, 8, 9, 5],
       [0, 0, 1, 7],
       [6, 9, 2, 4]])

>>> # 첫번째 칼럼값으로 정렬하자 (5,0,6)

>>> np.argsort([5,0,6]) # 위치를 반환
array([1, 0, 2], dtype=int64)

>>> sorter=np.argsort(data[:,0])
>>> sorter
array([1, 0, 2], dtype=int64)
>>> data[sorter]
array([[0, 0, 1, 7],
       [5, 8, 9, 5],
       [6, 9, 2, 4]])
```

# Numpy – 어레이 저장과 불러오기

## ▶ np.save(), np.load()

```
>>> a=np.random.randn(1000,100)
>>> a.shape
(1000, 100)
>>> a[0,0]
1.74481176421648

>>> np.save('1.npy', a)
>>> aa=np.load('1.npy')

>>> aa.shape
(1000, 100)
>>> aa[0,0]
1.74481176421648

>>> a=np.random.randn(10,10)
>>> b=np.random.randn(10,10)
>>> np.save('1.npy', [a,b])
>>> aa,bb = np.load('1.npy')
```

# Numpy – 텍스트 파일 불러오기 (1)

▶ <https://github.com/gubosd/lecture/blob/master/iris.csv>

```
"sepal.length","sepal.width","petal.length","petal.width","variety"  
5.1,3.5,1.4,.2,"Setosa"  
4.9,3,1.4,.2,"Setosa"  
4.7,3.2,1.3,.2,"Setosa"  
4.6,3.1,1.5,.2,"Setosa"  
...  
6.2,3.4,5.4,2.3,"Virginica"  
5.9,3,5.1,1.8,"Virginica"
```

iris.csv  
150라인

```
import numpy as np  
  
f=open('iris.csv')  
  
line=f.readline() # 레이블 이름인 첫번째 줄 읽기  
header=line.strip().split(',')  
header=[i.strip('\"') for i in header]  
  
data=[]  
label=[]  
  
for line in f: # 나머지 라인 읽기  
    l=line.strip().split(',')  
    dl=[float(i) for i in l[:4]] # 4번째 칼럼 까지 읽기  
  
    data.append(dl)  
    label.append(l[4].strip('\"')) # 마지막 칼럼 읽기  
  
X=np.array(data); print(X.shape) # (150,4)  
y=np.array(label); print(y.shape) # (150,)  
  
f.close()
```

iris.py

# Numpy – 텍스트 파일 불러오기(2)

## ▶ np.loadtxt()

```
>>> a=np.loadtxt('iris.csv', delimiter=',', skiprows=1, usecols=(0,1,2,3))
>>> a.shape
(150, 4)
>>> a[0]
array([5.1, 3.5, 1.4, 0.2])

>>> name={'Setosa':0, 'Versicolor':1, 'Virginica':2}
>>> a=np.loadtxt('iris.csv', delimiter=',', skiprows=1,
                 converters={4: lambda x: name[x.strip('"')]} , encoding='utf-8')
>>> a.shape
(150, 5)
>>> X=a[:, :-1]
>>> y=a[:, -1]
>>> X.shape
(150, 4)
>>> y.shape
(150,)
```

5번째 칼럼을  
실수로 변환한다

# | Numpy – 텍스트 파일로 저장하기

## ▶▶ iris2.csv 로 저장

```
header=...
X=...
y=...

f=open('iris2.csv','w')
f.write(','.join(header) + '\n')
for i,j in zip(X,y):
    f.write('%f,%f,%f,%f,%s\n' % (i[0],i[1],i[2],i[3],j))
f.close()
```



# Numpy – 어레이 합치기

## ▶ hstack(), vstack(), c\_[ ], r\_[ ] 의 4가지 함수를 기억하자

```
>>> a=np.arange(4).reshape(2,2)
>>> b=np.arange(4).reshape(2,2)
>>> b=np.arange(10,14).reshape(2,2)
>>> a
array([[0, 1],
       [2, 3]])
>>> b
array([[10, 11],
       [12, 13]])
>>> np.hstack([a,b])
array([[ 0,  1, 10, 11],
       [ 2,  3, 12, 13]])
>>> np.vstack([a,b])
array([[ 0,  1],
       [ 2,  3],
       [10, 11],
       [12, 13]])
>>> np.c_[a,b]
array([[ 0,  1, 10, 11],
       [ 2,  3, 12, 13]])
>>> np.r_[a,b]
array([[ 0,  1],
       [ 2,  3],
       [10, 11],
       [12, 13]])
```

[a,b] 와 같이  
[ ] 로 묶어야함

( ) 가 아니라  
[ ] 임

# Numpy – 브로드캐스팅(1)

## ▶▶ 브로드캐스팅은 이해하기는 쉽지 않지만, 아주 유용하다

- 각 칼럼의 평균값으로 각 칼럼의 값들을 빼는 경우를 생각해 보자

```
>>> np.random.seed(11)
>>> a=np.random.randint(10,size=(3,2))
>>> a
array([[9, 0],
       [1, 7],
       [1, 7]])

>>> b=a.mean(axis=0)
>>> b
array([3.66666667, 4.66666667])

>>> a - b
array([[ 5.33333333, -4.66666667],
       [-2.66666667,  2.33333333],
       [-2.66666667,  2.33333333]])
```

1차원 배열은  
아래로 확장됨을  
알수있다

```
>>> c=np.arange(4).reshape(2,2)
>>> c
array([[0, 1],
       [2, 3]])
>>> c-[1,2]
array([[ -1,  -1],
       [ 1,   1]])
>>> c-[[1,2]]
array([[ -1,  -1],
       [ 1,   1]])
>>> c-[[1],[2]]
array([[ -1,   0],
       [ 0,   1]])

>>> a-a.mean(axis=1)
ValueError: operands could not be
broadcast together with shapes (3,2) (3,)
>>> a - a.mean(axis=1).reshape(3,1)
array([[ 4.5, -4.5],
       [-3. ,  3. ],
       [-3. ,  3. ]])
```

# Numpy – 브로드캐스팅(2)

## ▶ 기타 브로드캐스팅 기능

```
>>> a=np.arange(3).reshape(3,1)
>>> a
array([[0],
       [1],
       [2]])
>>> a+[1,2,3]
array([[1, 2, 3],
       [2, 3, 4],
       [3, 4, 5]])
```

행벡터와  
열벡터간  
계산

```
>>> a=np.arange(16).reshape(4,4)
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
>>> a[1:-1,1:-1] = [99,101]
>>> a
array([[ 0,  1,  2,  3],
       [ 4, 99, 101,  7],
       [ 8, 99, 101, 11],
       [12, 13, 14, 15]])
```

값 할당

행별로(옆으로)  
정규화 예제

```
>>> np.random.seed(1)
>>> a=np.random.randint(5,size=(3,5))
>>> a
array([[3, 4, 0, 1, 3],
       [0, 0, 1, 4, 4],
       [1, 2, 4, 2, 4]])
>>> a_mean=a.mean(axis=1).reshape(3,1)
>>> a_mean
array([[2.2],
       [1.8],
       [2.6]])
>>> a_std=a.std(axis=1).reshape(3,1)
>>> a_std
array([[1.46969385],
       [1.83303028],
       [1.2         ]])
>>> (a-a_mean)/a_std
array([[ 0.54433105,  1.22474487, -
 1.4969104 , -0.81649658,  0.54433105],
       [-0.98198051, -0.98198051, -
 0.43643578,  1.2001984 ,  1.2001984 ],
       [-1.33333333, -0.5         ,
 1.16666667, -0.5         ,  1.16666667]])
```

# Numpy – 기타 기능

## ▶▶ ravel(), linspace() 등

```
>>> a=np.arange(10).reshape(2,5)
>>> a
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
>>> a.ravel()
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

펼쳐준다

```
>>> a=np.linspace(0,1,10)
>>> a
array([0.          , 0.11111111, 0.22222222, 0.33333333, 0.44444444,
       0.55555556, 0.66666667, 0.77777778, 0.88888889, 1.          ])
>>> a=np.linspace(0,1,11)
>>> a
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ])
```

시작값, 끝값,  
나누는 갯수

# | Numpy – 심화학습

## ▶▶ iris.csv 파일을 읽어와 어떻게 분류할지 분석해 보자

- sample : 150개
- 클래스(3개) : Setosa, Versicolor, Virginica
- 속성 (4개) : sepal.length, sepal.width, petal.length, petal.width
- 분석방법 : 속성별/클래스별 평균/표준편차/최대값/최소값 등

# Matplotlib – 데이터 시각화

▶▶ <https://matplotlib.org/tutorials/index.html>



Fork me on GitHub

[home](#) | [examples](#) | [tutorials](#) | [pyplot](#) | [docs](#) » [User's Guide](#) »

[previous](#) | [next](#) | [modules](#) | [index](#)

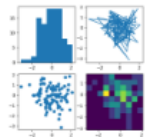
## Tutorials

This page contains more in-depth guides for using Matplotlib. It is broken up into beginner, intermediate, and advanced sections, as well as sections covering specific topics.

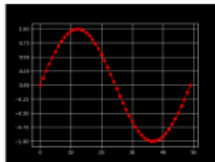
For shorter examples, see our [examples page](#). You can also find [external resources](#) and a [FAQ](#) in our [user guide](#).

## Introductory

These tutorials cover the basics of creating visualizations with Matplotlib, as well as some best-practices in using the package effectively.



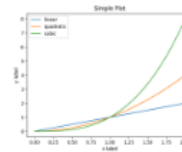
Sample plots in  
Matplotlib



Customizing  
matplotlib



Image tutorial



Usage Guide

Quick search

Go

Table Of Contents

### Tutorials

- [Introductory](#)
- [Intermediate](#)
- [Advanced](#)
- [Colors](#)
- [Text](#)
- [Toolkits](#)

Related Topics

### Documentation overview

- [User's Guide](#)
  - [Previous: Installing](#)
  - [Next: Sample plots in Matplotlib](#)

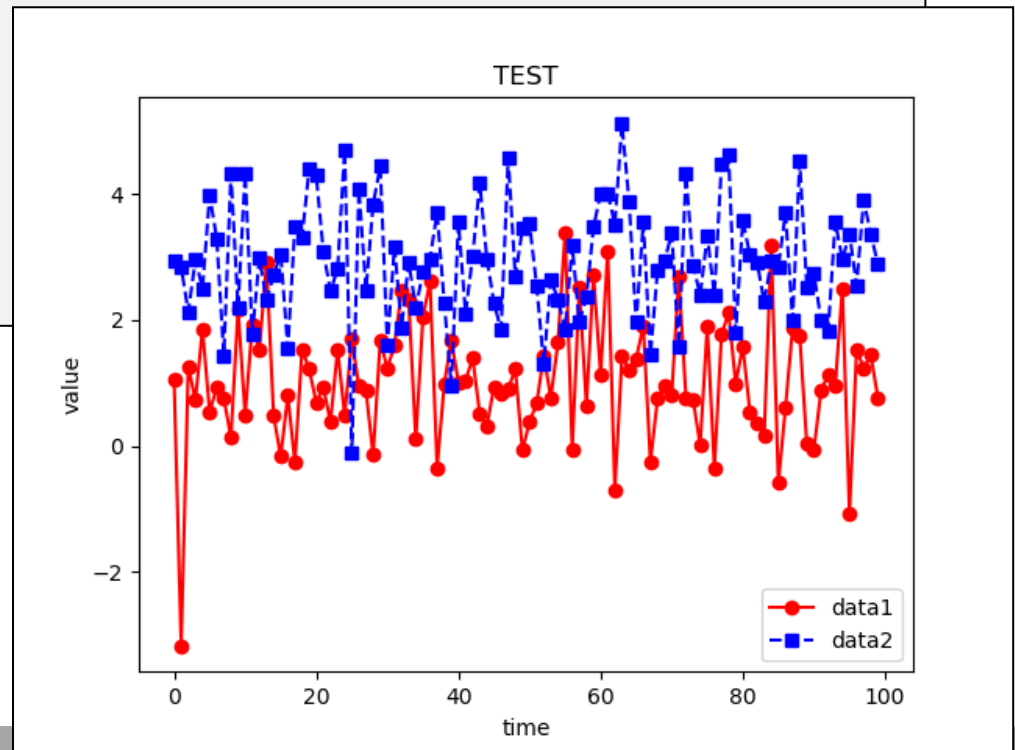
# Matplotlib - 맛보기

▶▶ plt.plot() 으로 그리고 plt.show() 로 보여준다

```
import numpy as np
import matplotlib.pyplot as plt

data1=np.random.normal(1,1,size=100)
data2=np.random.normal(3,1,size=100)

plt.plot(data1,'ro-')
plt.plot(data2,'bs--')
plt.title('TEST')
plt.legend(['data1','data2'])
plt.xlabel('time')
plt.ylabel('value')
plt.show()
```



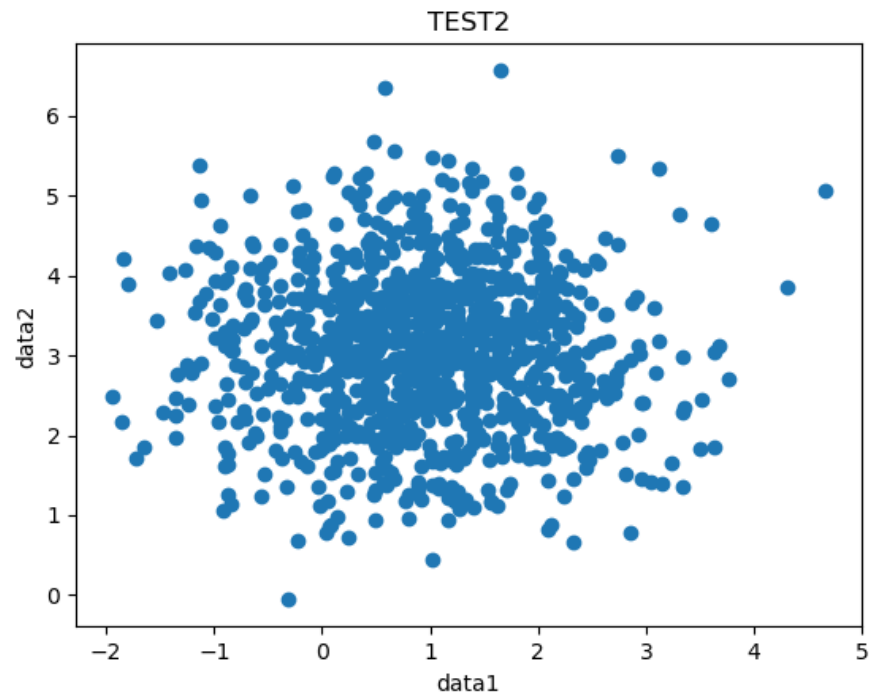
# Matplotlib – 맛보기2

## ▶▶ plt.scatter()

```
import numpy as np
import matplotlib.pyplot as plt

data1=np.random.normal(1,1,size=1000)
data2=np.random.normal(3,1,size=1000)

plt.scatter(data1,data2)
plt.title('TEST2')
plt.xlabel('data1')
plt.ylabel('data2')
plt.show()
```



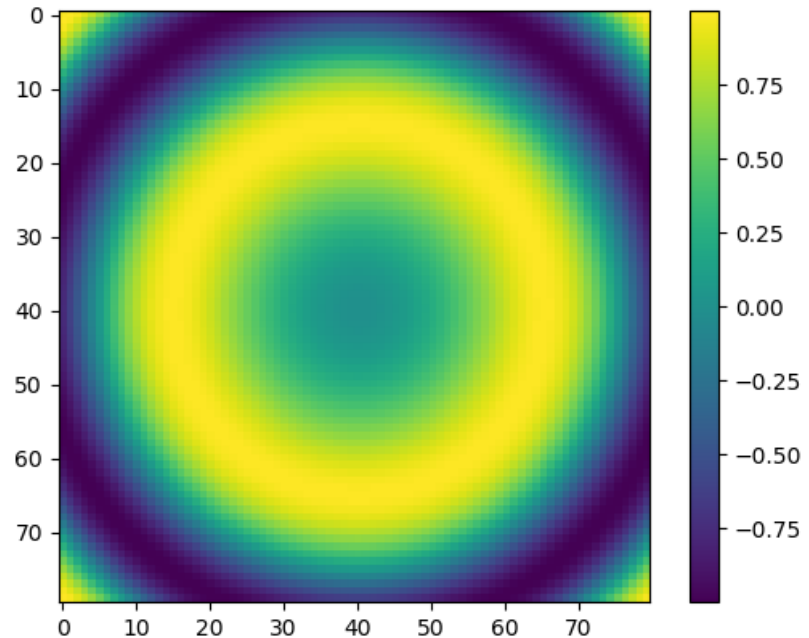


# Matplotlib – 맛보기3

## ▶ plt.imshow()

```
import numpy as np
import matplotlib.pyplot as plt

data=[ [np.sin(i**2+j**2) for j in np.arange(-2,2,0.05)]
        for i in np.arange(-2,2,0.05) ]
plt.imshow(data)
plt.colorbar()
plt.show()
```



# Matplotlib – iris 데이터

## ▶ iris 데이터 읽어와 그래프 그리기

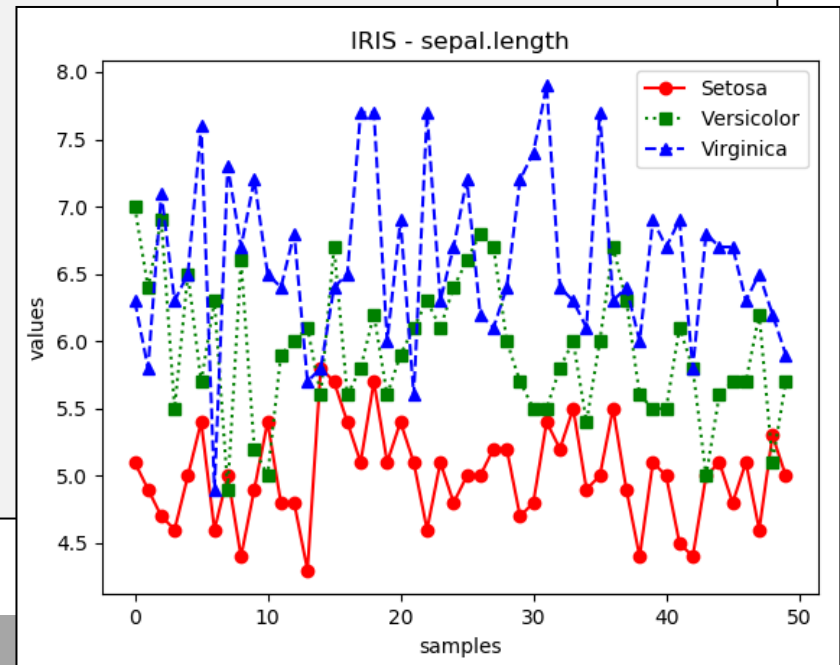
```
import numpy as np
import matplotlib.pyplot as plt

# load iris data

columns=["sepal.length","sepal.width","petal.length","petal.width"]
name={'Setosa':0, 'Versicolor':1, 'Virginica':2}
X=np.loadtxt('iris.csv', delimiter=',', skiprows=1, converters={4: lambda x:
        name[x.strip('')]}, encoding='utf-8')
X1=X[X[:,-1]==0]
X2=X[X[:,-1]==1]
X3=X[X[:,-1]==2]

# plot

plt.title('IRIS - sepal.length')
plt.plot(X1[:,0],'ro-')
plt.plot(X2[:,0],'gs:')
plt.plot(X3[:,0],'b^--')
plt.legend(name.keys())
plt.xlabel('samples')
plt.ylabel('values')
plt.show()
```



# Matplotlib –plot()

## ▶▶ plt.plot()

```
>>> plt.plot(range(10,40), X1[10:40,0], color='red', marker='o', linestyle='solid',  
linewidth=2, markersize=20, label='Setosa')  
  
>>> plt.plot(range(10,40), X1[10:40,0], 'ro-', linewidth=2, markersize=20, label='Setosa')  
  
>>> plt.plot(X2[:,1], 'gs')
```

### Colors

The following color abbreviations are supported:

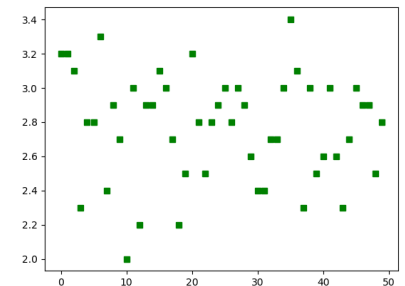
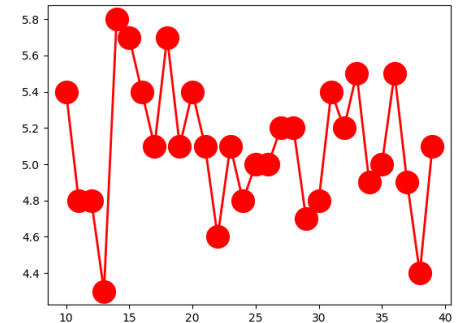
character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

### Line Styles

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

### Markers

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'*'	tri_down marker

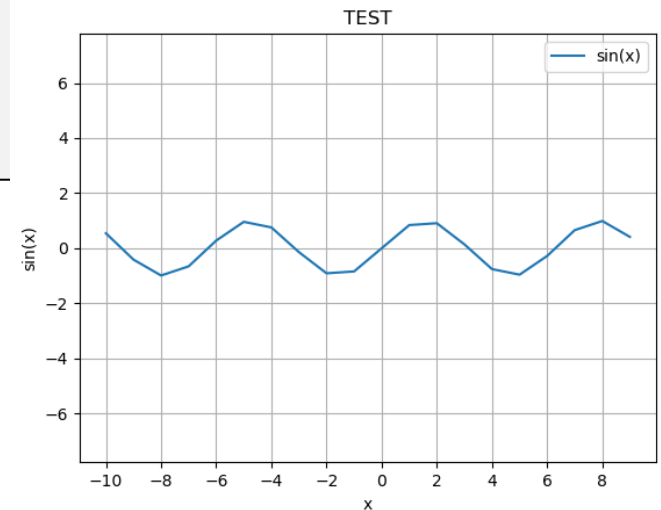


[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot)

# Matplotlib – 설정 기능들

## ▶ 축, 그리드, 레이블 등

```
plt.title('TEST')
plt.plot(range(-10,10),[np.sin(i) for i in range(-10,10)],
         label='sin(x)')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.xticks(range(-10,10,2))
plt.axis('equal')
plt.grid()
plt.legend()
plt.show()
```

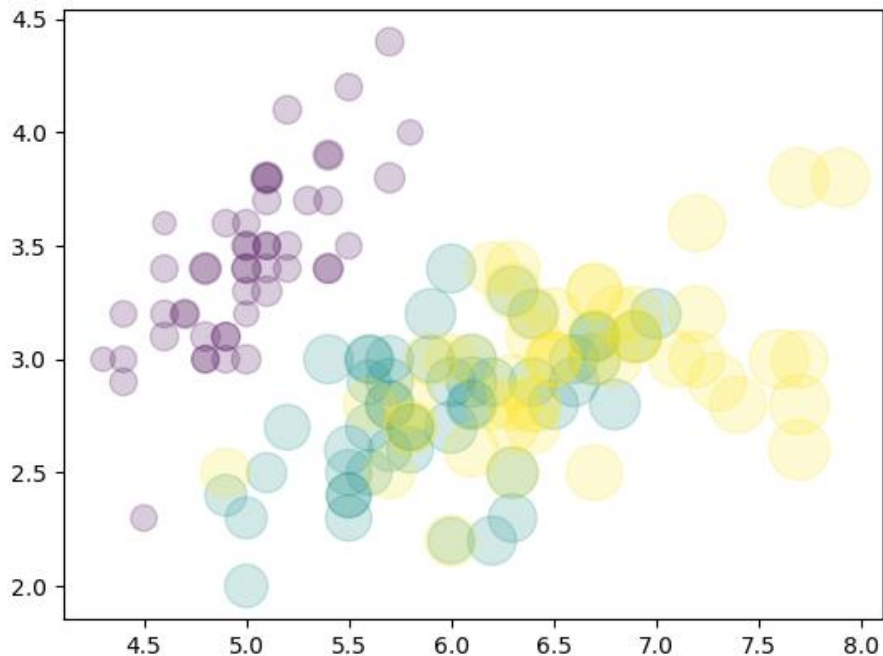


[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.html](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html)

# Matplotlib – scatter()

## ▶▶ 2차원 점들로 표시

```
plt.scatter(X[:,0], X[:,1], c=X[:,2], s=X[:,3]*100, alpha=0.2)  
plt.show()      # c: color, s: size
```

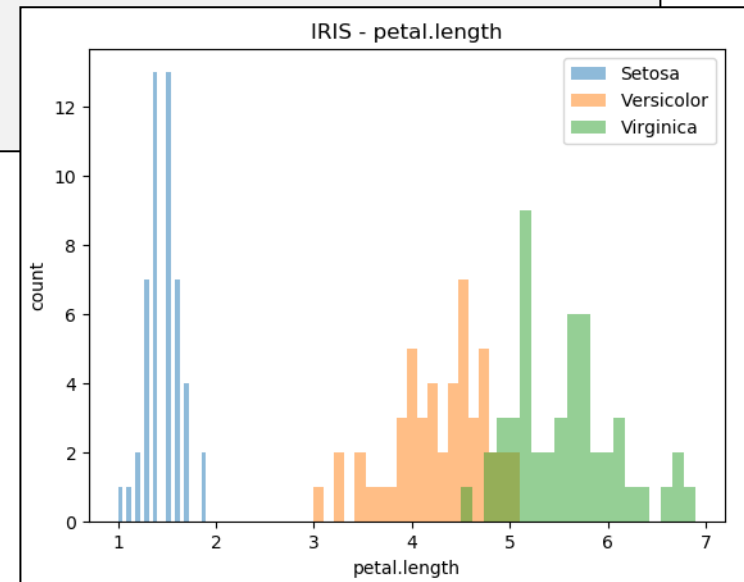


[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.scatter.html#matplotlib.pyplot.scatter](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html#matplotlib.pyplot.scatter)

# Matplotlib – hist()

## ▶ 히스토그램

```
plt.title('IRIS - petal.length')
plt.hist(X1[:,2],bins=20,alpha=0.5,label='Setosa')
plt.hist(X2[:,2],bins=20,alpha=0.5,label='Versicolor')
plt.hist(X3[:,2],bins=20,alpha=0.5,label='Virginica')
plt.xlabel('petal.length')
plt.ylabel('count')
plt.legend()
plt.show()
```

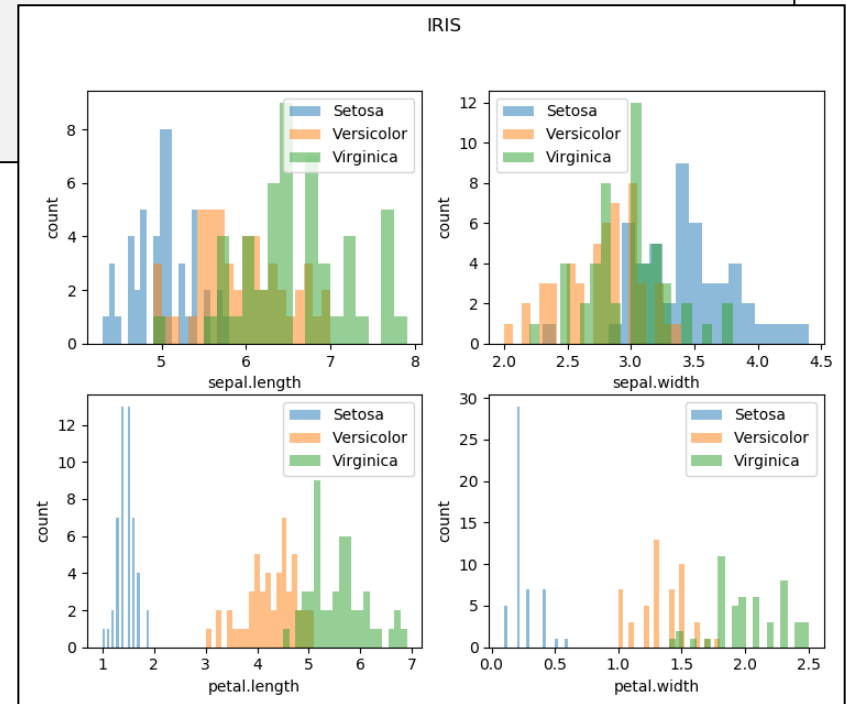


[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.hist.html#matplotlib.pyplot.hist](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib.pyplot.hist)

# Matplotlib – 서브플롯

## ▶ 여러개의 플롯을 한 장에 그려보자 → subplot()

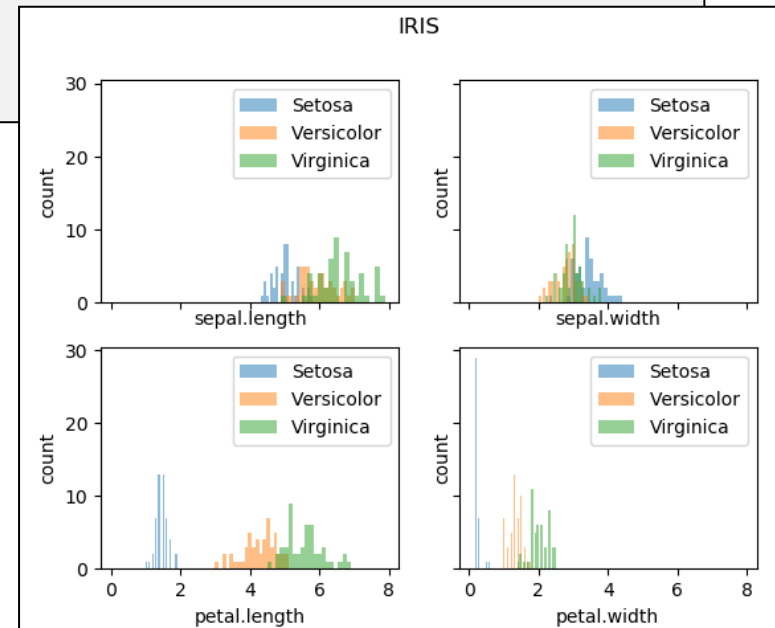
```
fig=plt.figure()
fig.suptitle('IRIS')
for col in range(4):
    plt.subplot(2,2,col+1) # 1 ~ 4
    plt.hist(X1[:,col],bins=20,alpha=0.5,label='Setosa')
    plt.hist(X2[:,col],bins=20,alpha=0.5,label='Versicolor')
    plt.hist(X3[:,col],bins=20,alpha=0.5,label='Virginica')
    plt.xlabel(columns[col])
    plt.ylabel('count')
    plt.legend()
plt.show()
```



# Matplotlib – 서브플롯2

## ▶▶ subplots()

```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
fig.suptitle('IRIS')
for col in range(4):
    ax = axes[col//2, col%2]    # axes.shape==(2,2)
    ax.hist(X1[:, col], bins=20, alpha=0.5, label='Setosa')
    ax.hist(X2[:, col], bins=20, alpha=0.5, label='Versicolor')
    ax.hist(X3[:, col], bins=20, alpha=0.5, label='Virginica')
    ax.set_xlabel(columns[col])
    ax.set_ylabel('count')
    ax.legend()
plt.show()
```





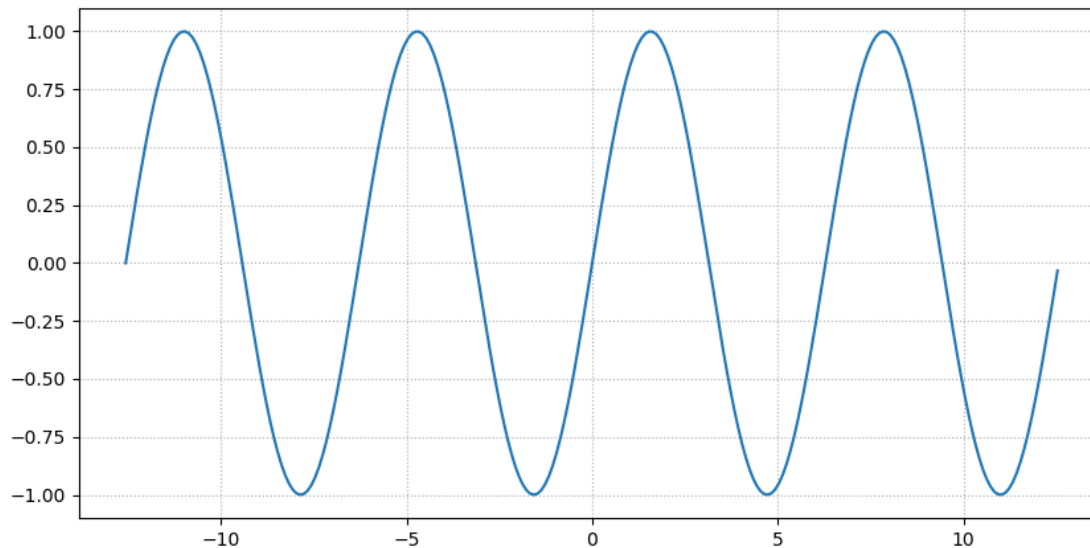
# Matplotlib – 파일로 저장

## ▶▶ savefig() 사용

```
import numpy as np
import matplotlib.pyplot as plt

x=np.arange(-4*np.pi,4*np.pi,0.1)

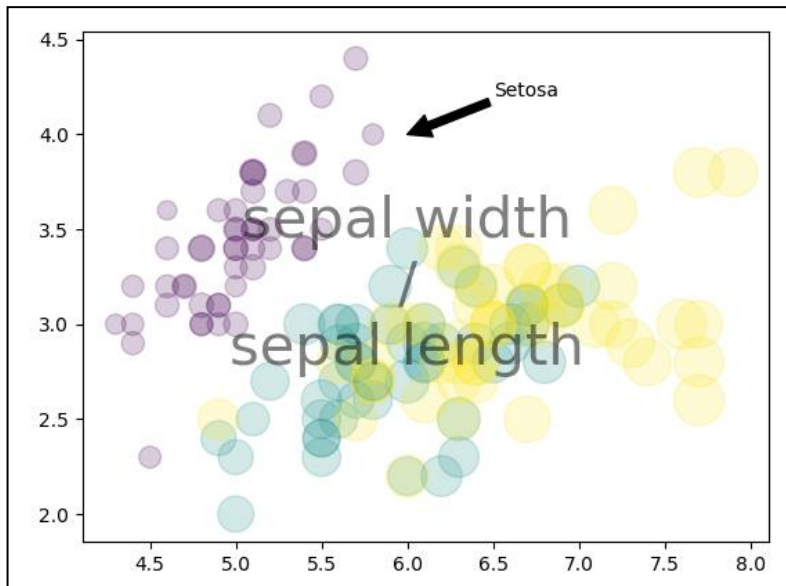
fig=plt.figure(figsize=(10,5))
plt.plot(x,np.sin(x))
plt.grid(linestyle=':')
plt.savefig('sin.png') # 경우에 따라서 fig.savefig('sin.png') 사용
```



# Matplotlib – 기타

## ▶ text(), annotate()

```
plt.scatter(X[:,0],X[:,1],c=X[:,-1],s=X[:,2]*100, alpha=0.2)
plt.text(6,3.2,'sepal width\n/\nsepal length', fontsize=30, alpha=0.5,
        ha='center', va='center')
plt.annotate('Setosa',xy=(6,4),xytext=(6.5,4.2),arrowprops=dict(facecolor='black'))
plt.show()
```

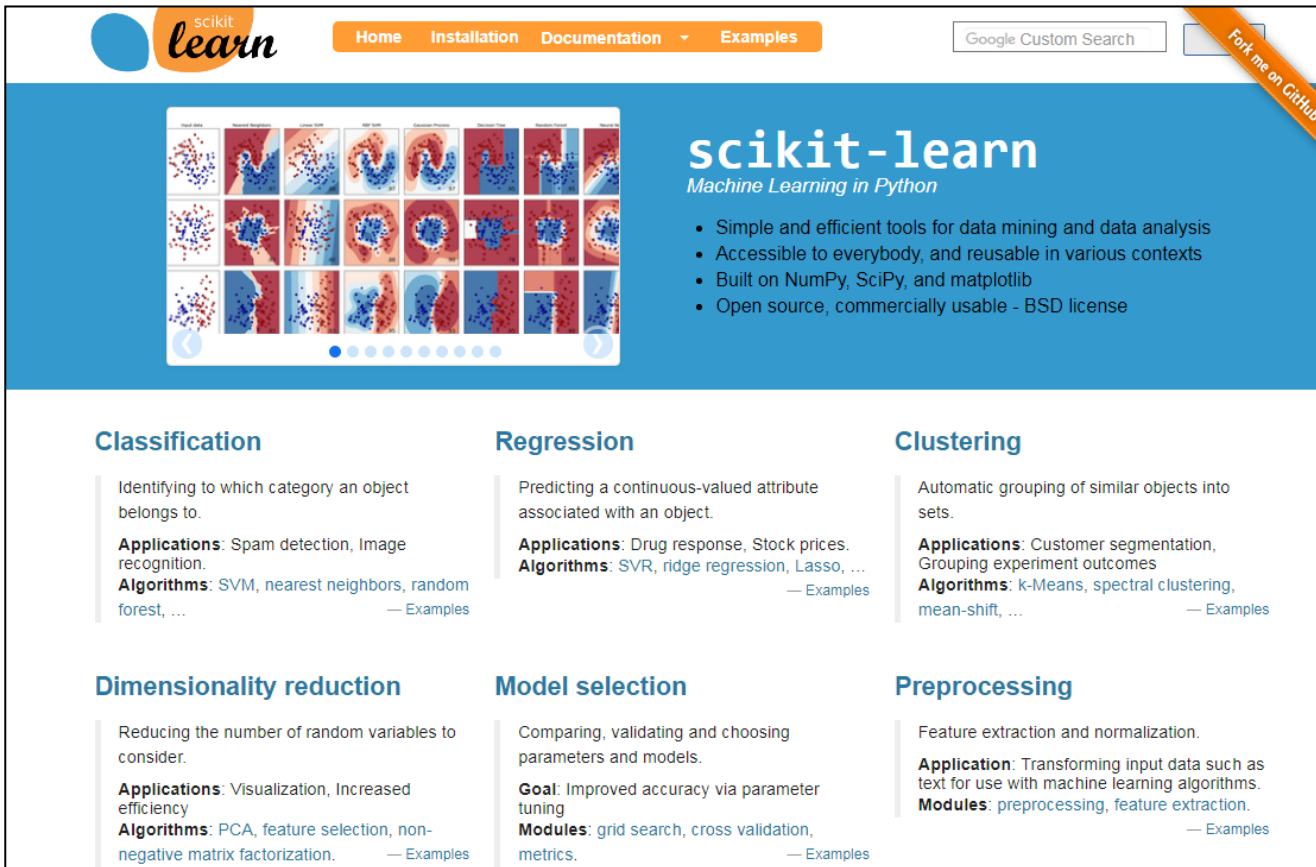


[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.text.html#matplotlib.pyplot.text](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.text.html#matplotlib.pyplot.text)  
[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.annotate.html#matplotlib.pyplot.annotate](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.annotate.html#matplotlib.pyplot.annotate)

# Scikit-learn – Python 머신러닝 모듈

## ▶▶ scikit-learn.org

- API reference → <http://scikit-learn.org/stable/modules/classes.html>



The screenshot shows the scikit-learn website homepage. At the top, there is a navigation bar with links for Home, Installation, Documentation, and Examples. A Google Custom Search bar is also present. The main header features the scikit-learn logo and the tagline "Machine Learning in Python". Below this, a grid of 12 small plots illustrates various machine learning concepts. To the right of the grid, a list of features is provided: Simple and efficient tools for data mining and data analysis, Accessible to everybody, and reusable in various contexts, Built on NumPy, SciPy, and matplotlib, and Open source, commercially usable - BSD license. The page is divided into six sections: Classification, Regression, Clustering, Dimensionality reduction, Model selection, and Preprocessing. Each section includes a brief description, applications, and algorithms.

**scikit-learn**  
*Machine Learning in Python*

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification	Regression	Clustering
Identifying to which category an object belongs to.	Predicting a continuous-valued attribute associated with an object.	Automatic grouping of similar objects into sets.
<b>Applications:</b> Spam detection, Image recognition.	<b>Applications:</b> Drug response, Stock prices.	<b>Applications:</b> Customer segmentation, Grouping experiment outcomes
<b>Algorithms:</b> SVM, nearest neighbors, random forest, ...	<b>Algorithms:</b> SVR, ridge regression, Lasso, ...	<b>Algorithms:</b> k-Means, spectral clustering, mean-shift, ...
— Examples	— Examples	— Examples

Dimensionality reduction	Model selection	Preprocessing
Reducing the number of random variables to consider.	Comparing, validating and choosing parameters and models.	Feature extraction and normalization.
<b>Applications:</b> Visualization, Increased efficiency	<b>Goal:</b> Improved accuracy via parameter tuning	<b>Application:</b> Transforming input data such as text for use with machine learning algorithms.
<b>Algorithms:</b> PCA, feature selection, non-negative matrix factorization.	<b>Modules:</b> grid search, cross validation, metrics.	<b>Modules:</b> preprocessing, feature extraction.
— Examples	— Examples	— Examples

# | Scikit-learn – 데이터 준비

## ▶▶ Iris 데이터셋 사용

```
import numpy as np
import matplotlib.pyplot as plt

# load iris data

columns=["sepal.length","sepal.width","petal.length","petal.width"]
name={'Setosa':0, 'Versicolor':1, 'Virginica':2}

data=np.loadtxt('iris.csv', delimiter=',', skiprows=1, converters={4: lambda x:
name[x.strip('')] }], encoding='utf-8')

# data for machine-learning

X=data[:, :-1] # (150,4)
y=data[:, -1] # (150,) label/target

X1=X[y==0] # (50,4) Setosa
X2=X[y==1] # (50,4) Versicolor
X3=X[y==2] # (50,4) Virginica
```

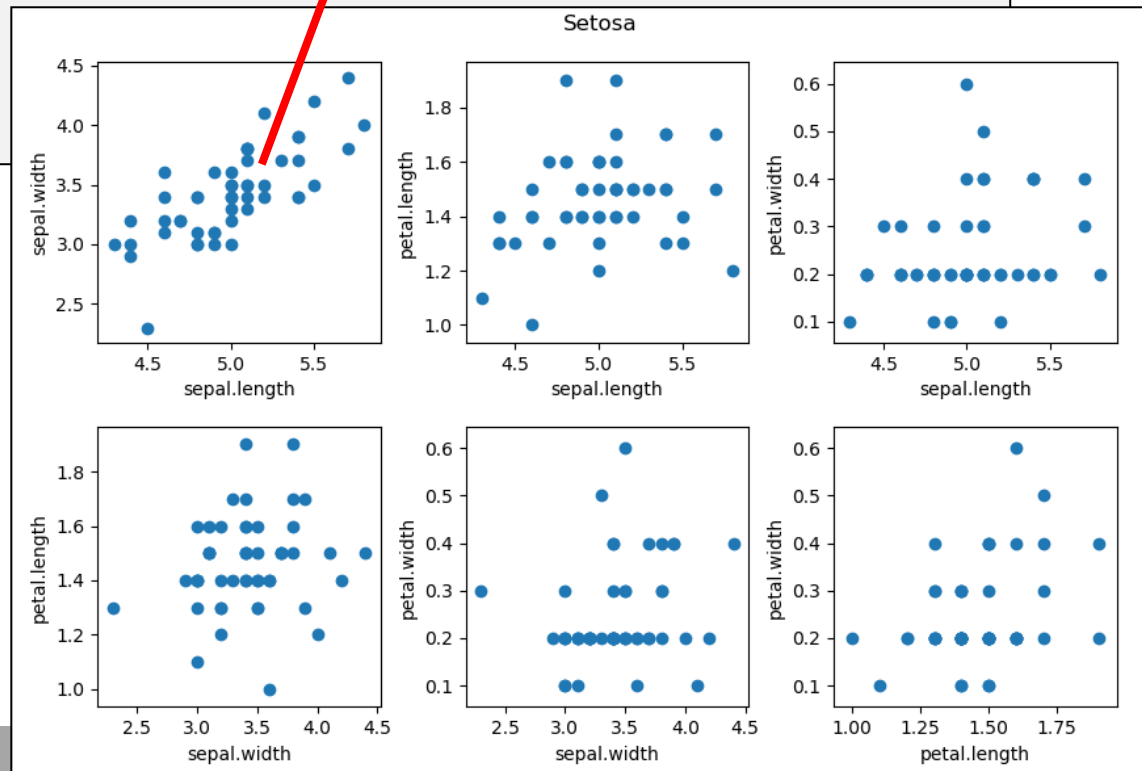
# Scikit-learn – 회귀(Regression)

## ▶ Setosa 에서 속성들 간의 분포를 분석해 보자

```
fig=plt.figure()
fig.suptitle('Setosa')
count=0
for i in range(4):
    for j in range(i+1,4):
        count+=1
        plt.subplot(2,3,count)
        plt.scatter(X1[:,i],X1[:,j])
        plt.xlabel(columns[i])
        plt.ylabel(columns[j])
plt.show()
```

상관관계  
높음

- 회귀는 점들에 근사하는 직선이나 평면을 찾는 것이다
- Setosa 에서 sepal\_length 와 sepal\_width 간에 회귀를 적용해 보자



# Scikit-learn – 회귀(Regression)

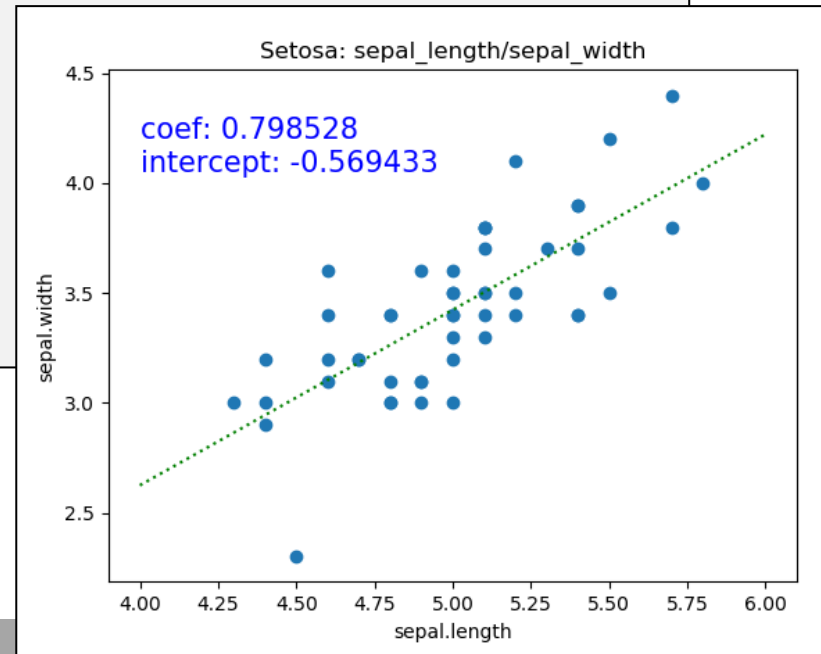
## ▶ 선형회귀 (LinearRegression)

```
from sklearn.linear_model import LinearRegression

model=LinearRegression()
model=model.fit(X1[:,0].reshape(-1,1), X1[:,1])
w=model.coef_      # 기울기
b=model.intercept_ # y 절편

plt.title('Setosa: sepal_length/sepal_width')
plt.scatter(X1[:,0],X1[:,1])
plt.xlabel(columns[0])
plt.ylabel(columns[1])
plt.plot([4,6],[4*w+b,6*w+b],'g:')
plt.text(4,4.3,'coef: %f\nintercept: %f' % (w,b),
        va='top',fontsize=15,color='b')
plt.show()

pred_y = model.predict([[0],[1],[2],[3]])
print(pred_y)
# [-0.56943267  0.22909563  1.02762393  1.82615223]
```

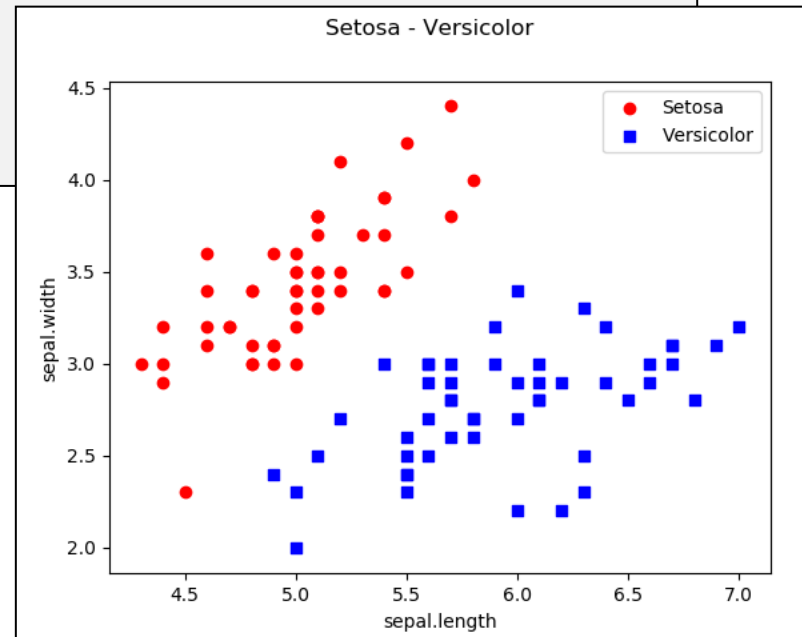


# Scikit-learn – 분류(Classification)

## ▶ KNN 분류법 → k개의 가장 가까운 이웃이라는 의미임

- 공간 상에 점을 뿌렸을 때, 가장 가까이 있는 점들로 해당 클래스를 판단한다
- Setosa 와 Versicolor 를 분류해 보자
- 속성은 sepal.length 와 sepal.width 두개만 사용한다

```
fig=plt.figure()
fig.suptitle('Setosa - Versicolor')
plt.scatter(X1[:,0],X1[:,1],marker='o',color='r')
plt.scatter(X2[:,0],X2[:,1],marker='s',color='b')
plt.xlabel(columns[0])
plt.ylabel(columns[1])
plt.legend(['Setosa','Versicolor'])
plt.show()
```



- 속성의 2개이므로 2차원 평면임
- 임의의 한점을 찍었을 때 가장 가까이 있는 점들의 클래스로 판별한다
- 속성이 4개이지만 시각화를 위해 2개의 속성으로 제한함

# Scikit-learn – KNN

## ▶ KNeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

X=np.vstack([X1,X2])[:, :2] # X1 과 X2 를 합쳐 데이터를 만든다
y=np.array([0]*50+[1]*50) # 각각 50개씩

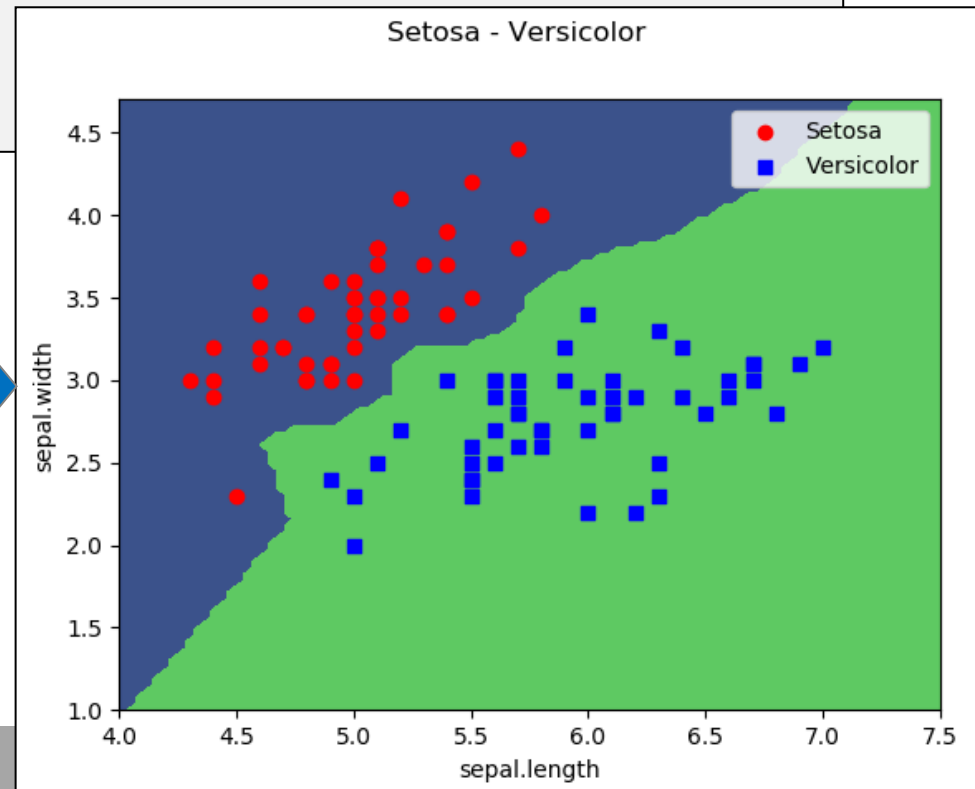
model = KNeighborsClassifier(n_neighbors=1) # 가장 가까운 1개의 이웃만 본다
model.fit(X,y)
pred_y = model.predict(X)
acc=accuracy_score(y,pred_y)
print('accuracy :',acc)
```

```
xx=np.linspace(4,7.5,100)
yy=np.linspace(1,4.7,100)
data1, data2 = np.meshgrid(xx,yy)
X_grid = np.c_[data1.ravel(), data2.ravel()]
decision_values = model.predict_proba(X_grid)[: , 1]

fig=plt.figure()
fig.suptitle('Setosa - Versicolor')

CS=plt.contourf(data1,data2,decision_values.reshape(data1.shape),levels=[-1,0,1])

plt.scatter(X1[:,0],X1[:,1],marker='o',color='r',label='Setosa')
plt.scatter(X2[:,0],X2[:,1],marker='s',color='b',label='Versicolor')
plt.xlabel(columns[0])
plt.ylabel(columns[1])
plt.legend()
plt.show()
```





# Scikit-learn – train/test 데이터 분리, 과적합

## ▶▶ train\_test\_split()

- 학습방법에 따라 학습결과가 학습한 데이터에 너무 딱 맞게 될 수가 있다
- 이러한 경우를 과적합이라고 하는데, 이렇게 되면 새로운 데이터를 제대로 평가하지 못하는 경우가 생긴다
- 그래서 데이터는 되도록이면 학습용과 테스트용으로 분리해 적용한다

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,stratify=y,random_state=1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
print(X_train[y_train==0].shape, X_train[y_train==1].shape, X_train[y_train==2].shape)
print(X_test[y_test==0].shape, X_test[y_test==1].shape, X_test[y_test==2].shape)

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,y_train)      # 학습할 때는 X_train, y_train
pred_y = model.predict(X_test)  # 평가할 때는 X_test, y_test
acc=accuracy_score(y_test,pred_y)
print('accuracy :',acc)
```

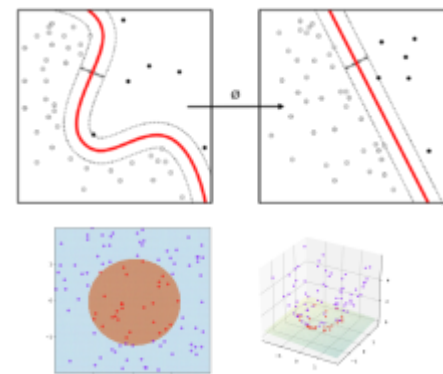
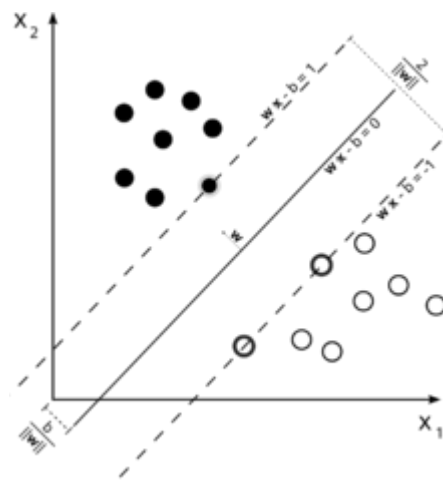
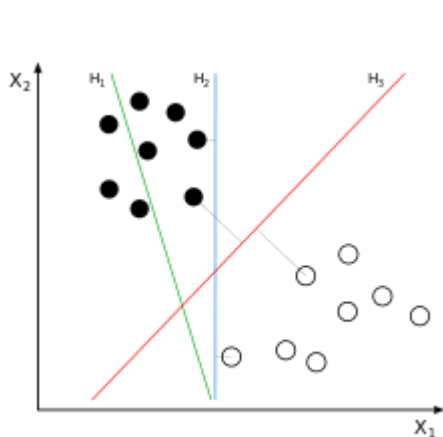
```
(120, 4) (30, 4) (120,) (30,)
(40, 4) (40, 4) (40, 4)
(10, 4) (10, 4) (10, 4)
accuracy : 0.9666666666666667
```

[http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

# Scikit-learn – 서포트 벡터 머신 (SVM)

## ▶▶ SVM → Support Vector Machine

- 두 클래스 간의 간격(gap or margin)을 가장 크게 하도록 선을 긋는다
- 경계선에 가장 가까이 있는 점들을 ‘Support Vector’ 라고 한다
- 다차원에서는 선이 아니라 면이 된다
- 커널 기법을 적용하면 직선과 평면이 아니라, 곡선과 구부러진 면이 된다 (예를 들면, 2차원 평면을 3차원으로 들어올리는 것과 같은 효과이다)
- [https://en.wikipedia.org/wiki/Support\\_vector\\_machine](https://en.wikipedia.org/wiki/Support_vector_machine) 참조



# | Scikit-learn – SVM

## ▶▶ sklearn.svm.SVC 사용

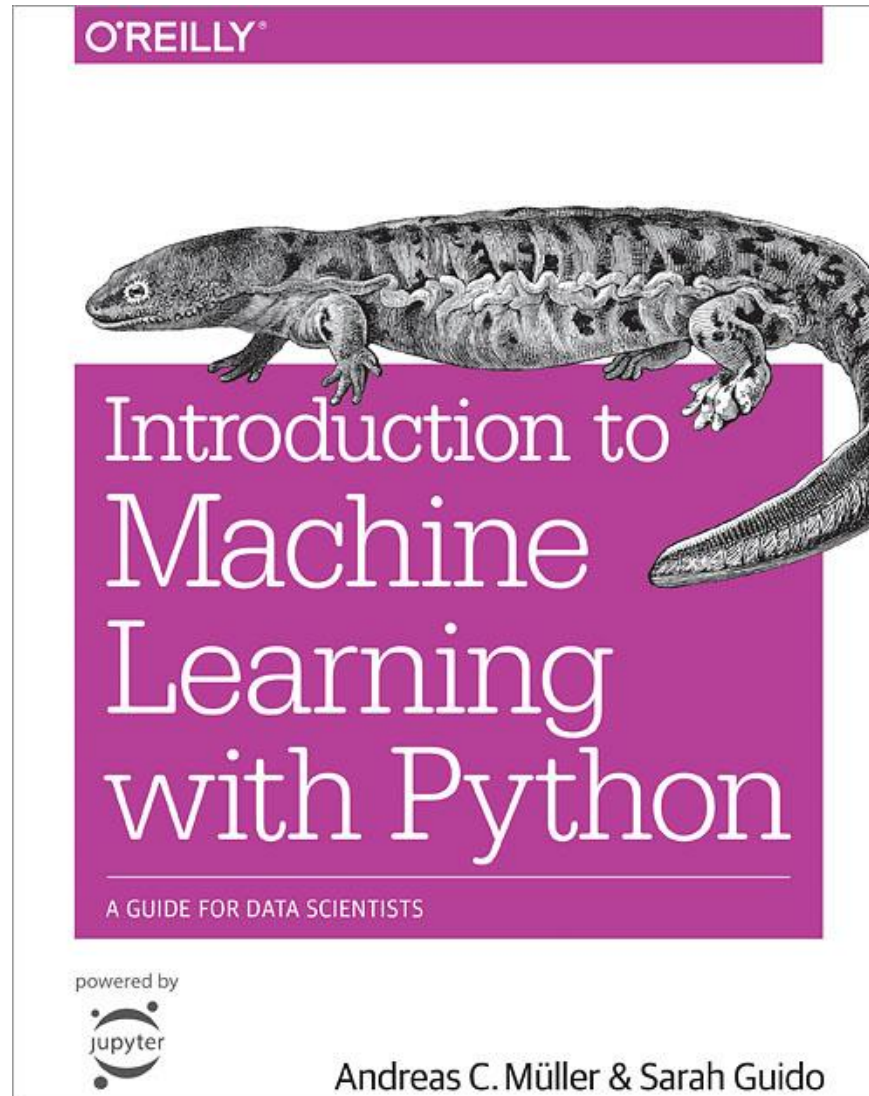
- kernel → rbf(디폴트), linear, poly, sigmoid 등
- C 값 → 값이 클수록 학습용 데이터에 최적화된다는 (과적합 경향)
- gamma 값 → 마찬가지로 클수록 과적합 경향이 나타납니다
- SVM 은 사용하지는 편리하지만 최적화된 인자값을 찾기가 어렵고, 다차원에 커널을 적용하면 의미를 이해하기가 아주 어렵습니다

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y,
                                                    random_state=1)
model = SVC() # kernel='rbf', C=1.0, gamma='auto'=1/n_feature
model.fit(X_train, y_train)

pred_y = model.predict(X_test)
acc = accuracy_score(y_test, pred_y)
print('accuracy :', acc) # accuracy : 0.9666666666666667
```

# | 머신러닝 추천 교재



# | 파이썬을 사용하자

- ▶▶ 하둡 Spark (PySpark)
- ▶▶ 머신러닝
- ▶▶ 인공지능 / 딥러닝 : 텐서플로
- ▶▶ 강화학습
- ▶▶ 금융/주식 (Pandas)
- ▶▶ 과학/IT/산업