

Probability and statistics

Python

❖ 파이썬이란?

- 초보자들이 처음 프로그래밍을 배울 때 추천되는 언어.

```
if 4 in [1, 2, 3, 4]: print("4가 있습니다")
```

- 오픈소스 기반의 강력한 언어

- 만들고자 하는 프로그램의 대부분을 파이썬으로 만들 수 있음.
- 다른 언어로 만든 프로그램을 파이썬 프로그램에 포함시킬 수 있음.
- 가독성이 뛰어나.

- 파이썬 사용 예

➤ 시스템 유틸리티, GUI 프로그래밍, C/C++와의 결합, 웹프로그래밍, 수치연산, 데이터베이스 프로그래밍, 데이터 분석, 사물 인터넷.

❖ 파이썬 설치하기

- **파이썬 버전.**

- 현재 파이썬은 2.x와 3.x 두 버전이 존재함.
- 본 강의에서는 3.x 버전을 설치.

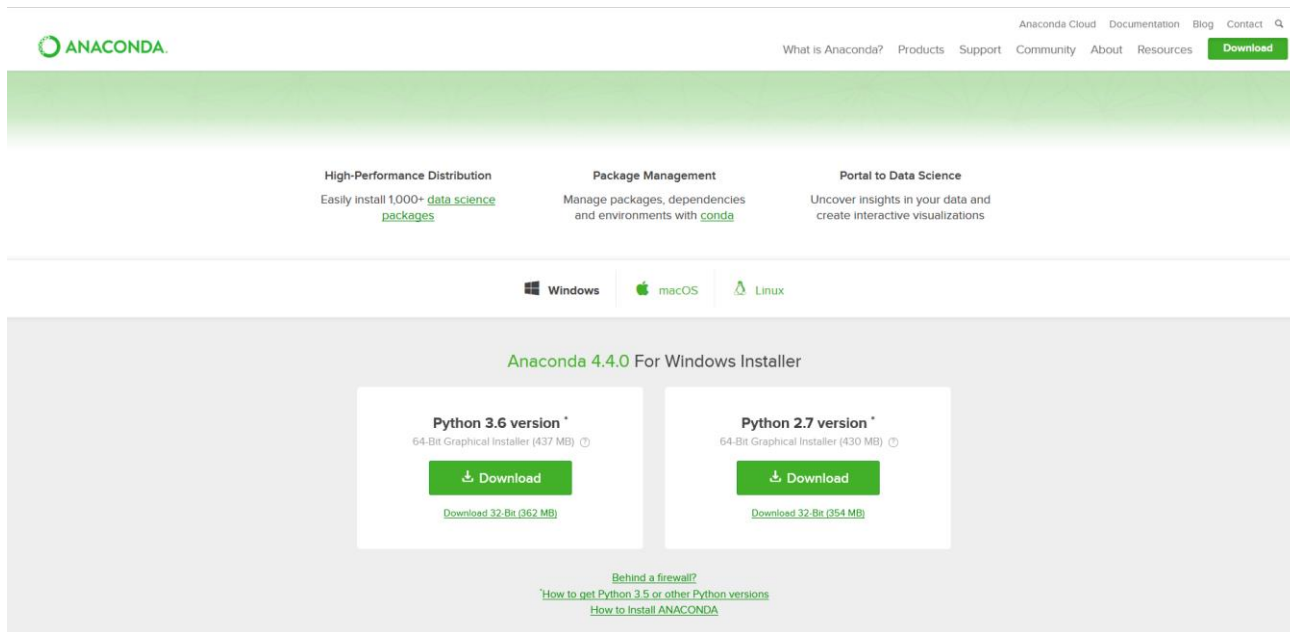
- **사용 라이브러리**

- `numpy` : 수치 계산을 라이브러리로서 수학 알고리즘과 행렬 계산을 위한 다양한 메서드를 제공함.
- `matplotlib` : 그래프를 출력하기 위한 라이브러리.

❖ 파이썬 설치하기

■ Anaconda 배포판.

- 사용자가 설치를 한 번에 할 수 있도록 필요한 라이브러리 등을 하나로 정리해둔 것.
- 데이터 분석 중점에 둔 배포판.
- Numpy와 matplotlib를 포함해 데이터 분석에 유용한 라이브러리가 포함되어 있음.



❖ 파이썬 설치하기

■ 파이썬 인터프리터.

- “python --version”을 입력하여 파이썬 버전 확인.

```
(D:\Anaconda3) C:\Users\whhault>python --version  
Python 3.6.1 :: Anaconda 4.4.0 (64-bit)
```

- “python”을 입력하여 파이썬 인터프리터 시작.

```
(D:\Anaconda3) C:\Users\whhault>python  
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

❖ 파이썬 설치하기

- 파이썬 에디터.
 - 에디트 플러스.



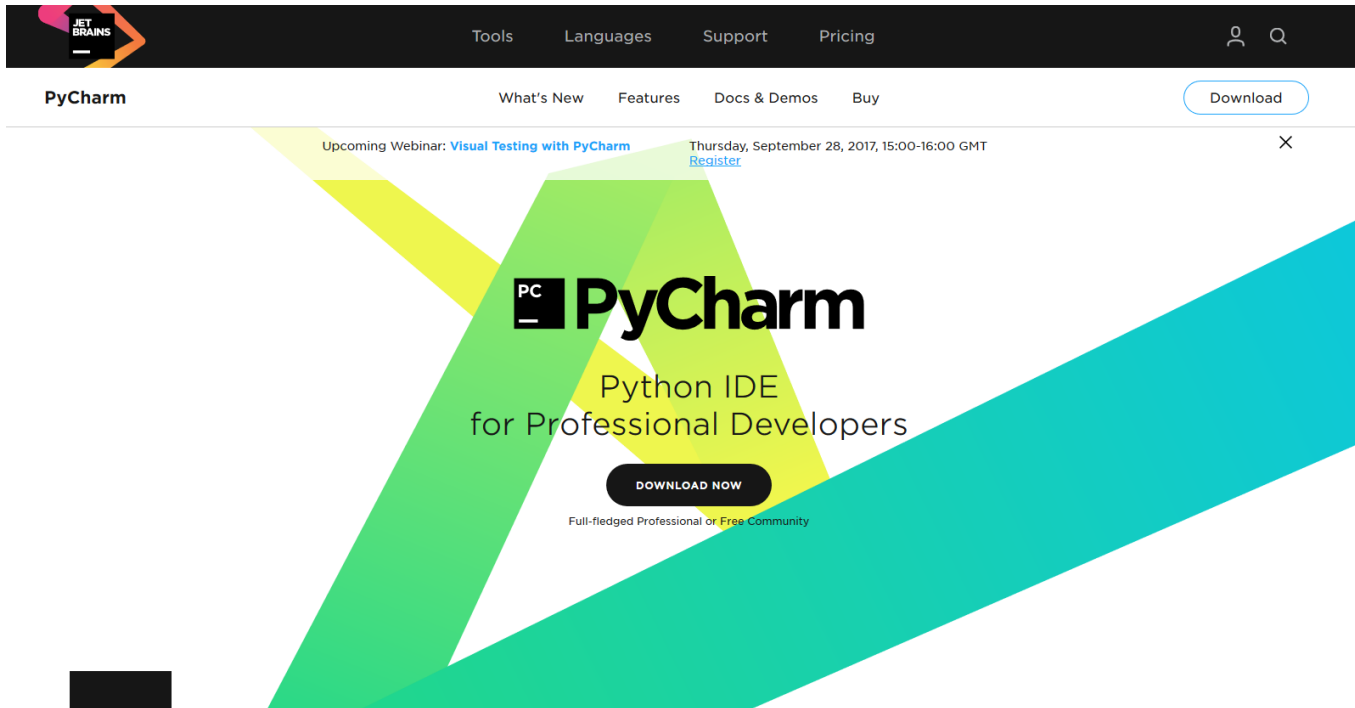
The screenshot shows the Edit+ website, an "INTERNET-READY TEXT EDITOR". The page features a navigation bar with links like "시작", "구입하기", "다운로드", "새로운 사항", "기능", "스크린샷", "사용자 파일", and "제품 지원". Below the navigation bar, there's a section titled "에디트플러스 문서 편집기" with a sub-header "에디트플러스 홈페이지에 잘 오셨습니다.". A list of links includes "구입하기" and "에디트플러스 문서 편집기 4.3 다운로드 (2017-05-12) New!". A paragraph describes the editor as an "인터넷 환경에서 편리하게 사용할 수 있는 윈도우용 문서 편집기로서, HTML 편집기, PHP 편집기, 자바 편집기, Hex 뷰어 기능을 지원하고, 메모장을 대신할 뿐 아니라 웹문서나 프로그램 개발을 쉽게 할 수 있도록 도와주는 많은 기능들을 지원합니다.". Below this is a preview of the editor's interface, which includes a toolbar with icons for file operations and editing, a file explorer on the left showing the "C:\BOOTCAMP" directory, and a main text area displaying HTML code. The code includes a style definition for a table cell background color and a link with a name attribute. At the bottom, a list of supported languages and features is provided.

에디트플러스는 인터넷 환경에서 편리하게 사용할 수 있는 윈도우용 문서 편집기로서, HTML 편집기, PHP 편집기, 자바 편집기, Hex 뷰어 기능을 지원하고, 메모장을 대신할 뿐 아니라 웹문서나 프로그램 개발을 쉽게 할 수 있도록 도와주는 많은 기능들을 지원합니다.

- HTML, PHP, 자바, C/C++, CSS, ASP, Perl, 자바스크립트, VB스크립트, Paython, Ruby on Rails 파일에서 구문 강조 기능을 지원합니다. 사용자가 작성한 구문 파일을 추가하여 다른 프로그래밍 언어도 지원할 수 있습니다.
- HTML 문서의 내용을 확인해 볼 수 있는 내장 브라우저와, 작성한 문서를 FTP 서버로 업로드할 수 있는 FTP (sftp, FTPS 포함) 기능을 지원합니다.
- 이 밖에 Hex 뷰어, HTML 도구모음, 사용자 도구, 줄 번호, 눈금자, URL 강조, 자동 완성, 클립보드, 칸 단위 선택, 강력한 찾기과 바꾸기, 다중 실행 취소 및 재실행, 사용자 정의 단축키 등 다양한 기능들을 쉽고 편리하게 사용할 수 있습니다.

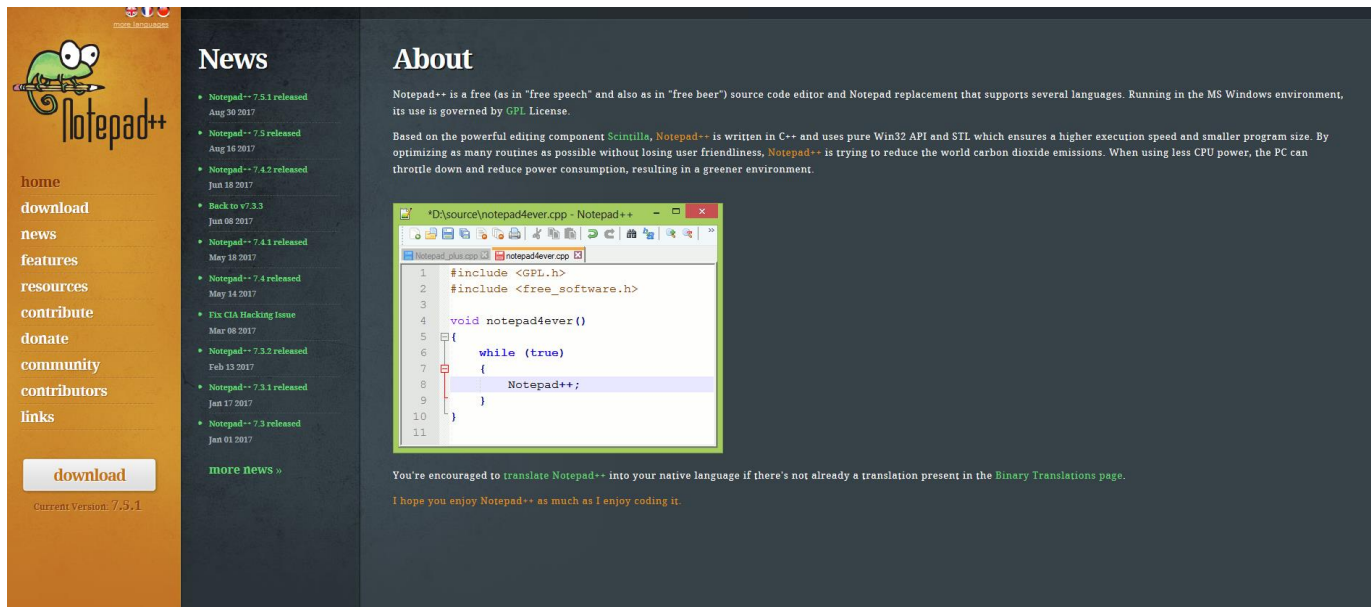
❖ 파이썬 설치하기

- 파이썬 에디터.
 - 파이참.



❖ 파이썬 설치하기

- 파이썬 에디터.
 - 노트패드++.



The screenshot displays the Notepad++ website on the left and a code editor window on the right. The website has an orange sidebar with a green cartoon frog logo and a list of navigation links: home, download, news, features, resources, contribute, donate, community, contributors, and links. A 'download' button is visible, with the text 'Current Version: 7.5.1' below it. The main content area is dark grey and divided into 'News' and 'About' sections. The 'News' section lists several releases of Notepad++ with dates. The 'About' section contains text about the software's license and goals, along with a code editor window showing a C++ program. The code editor window has a title bar that reads '*D:\source\notepad4ever.cpp - Notepad++'. The code inside the editor is as follows:

```
1 #include <GPL.h>
2 #include <free_software.h>
3
4 void notepad4ever()
5 {
6     while (true)
7     {
8         Notepad++;
9     }
10 }
11
```

Below the code editor, the 'About' section text reads: 'Notepad++ is a free (as in "free speech" and also as in "free beer") source code editor and Notepad replacement that supports several languages. Running in the MS Windows environment, its use is governed by [GPL License](#). Based on the powerful editing component [Scintilla](#), Notepad++ is written in C++ and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size. By optimizing as many routines as possible without losing user friendliness, Notepad++ is trying to reduce the world carbon dioxide emissions. When using less CPU power, the PC can throttle down and reduce power consumption, resulting in a greener environment.'

At the bottom of the 'About' section, there is a link to the [Binary Translations page](#) and a statement: 'I hope you enjoy Notepad++ as much as I enjoy coding it.'

❖ 자료형

■ 숫자형.

- 파이썬은 동적언어로 분류되는 프로그래밍 언어.
- 변수의 자료형을 상황에 맞게 자동으로 결정.

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
복소수	1+2j, -3j
8진수	0o34, 0o25
16진수	0x2A, 0xFF

■ 사용 예

```
A = 123  
B = 1.2  
C = 4.24e10
```

❖ 자료형

■ 문자열.

- 문자, 단어 등으로 구성된 문자들의 집합을 의미.
- "(큰따옴표), '(작은따옴표), ""(큰 따옴표 3개), "('작은따옴표) 3개 를 사용하여 생성
- 문자열 연결 시 '+' 연산자, 문자열 반복 시 '*'

■ 사용 예

```
"Hello world"  
'Python is fun'  
"python's favorite food is perl"  
"""python is very easy." he says.'  
'python\'s favorite food is perl'
```

```
head = "python"  
tail = "is fun"  
Head + tail  
Head*2
```

❖ 자료형

- 문자열.
 - 문자열 접근 시 인덱스를 통하여 접근.
- 사용 예

```
a = "Life is too short"  
a[3]    # 3번째 문자 접근  
a[-1]   # 뒤에서 첫번째 문자 접근  
a[0:4]  # 0부터 4까지 문자 접근  
a[3:]   # 3부터 끝까지 접근  
a[:]
```

❖ 자료형

■ 리스트.

- 숫자 또는 문자의 집합으로 어떠한 자료형도 포함시킬 수 있음.
- 인덱스를 통하여 리스트 내의 요소를 접근할 수 있음.

리스트명 = [요소 1, 요소 2, 요소 3, ...]

■ 사용 예

```
a = []  
b = [1, 2, 3]  
c = ['life', 'is', 'to', 'short']  
d = [1, 2, 'life', 'is']  
e = [1, 2, ['life', 'is']]
```

```
a = [1, 2, 3, [a, b, c]]  
a[0]  
a[0] + b[2]  
a[3][:2]
```

❖ 자료형

■ 튜플.

- 몇가지 점을 제외하고 리스트와 유사함.

- (,)를 사용.

- 리스트는 값 생성, 삭제가 가능하지만 튜플은 값을 변경할 수 없음.

■ 사용 예

```
t1 = ()  
t2 = (1, )    #1개의 요소만을 가질 때는 요소 뒤에 반드시逗를 붙여야함.  
t3 = (1, 2, 3)  
t4 = 1, 2, 3  # 괄호를 생략해도 무방
```

❖ 자료형

▪ 딕셔너리.

- 대응 관계를 나타낼 수 있는 자료형
- 리스트나 튜플처럼 순차적으로 해당 요소값을 구하지 않고 Key를 통해 Value를 얻음.

{Key1:Value1, Key2:Value2, Key3:Value3,...}

- key 값이 중복일 경우 하나를 제외한 나머지는 무시됨.
- 리스트를 key로 사용 불가

▪ 사용 예

```
# 입력 예
dic = {'name':'pay', 'phone':'01046060630', 'birth':'1118'}
a = {1:'hi'}
a = {'a' : [1, 2, 3]}
# 삭제 예
del a[1]
# 사용 예
grade = {'pay':10, 'julliet':99}
grade['pey']
```

❖ 자료형

■ 집합.

- 집합에 관련된 것들을 쉽게 처리하기 위해 만들어진 자료형.
- 중복을 허용하지 않고 순서가 없음.
- 리스트를 key로 사용 불가
- 교집합 : &, 합집합 : |, 차집합 : - 연산자

■ 사용 예

```
s1 = set([1, 2, 3])  
s2 = set("Hello")  
# {'e', 'l', 'o', 'H'}  
  
s1 = set([1, 2, 3])  
l1 = list(s1) # 리스트로 변환  
  
s1 = set([1, 2, 3])  
s1.add(4) #1개의 값 추가  
s1.update([4, 5, 6]) #여러 개의 값 추가  
s1.remove(2) #특정 값 제거
```

❖ 자료형

▪ 자료형의 참과 거짓.

자료형	값	참 or 거짓
문자열	"python"	참
	""	거짓
리스트	[1, 2, 3]	참
	[]	거짓
튜플	()	거짓
딕셔너리	{}	거짓
숫자형	0이 아닌 숫자	참
	0	거짓
	None	거짓

▪ 사용 예

```
a = [1, 2, 3, 4]
while a:
    a.pop()
# 4, 3, 2, 1
```


❖ 자료형

■ 변수.

- 변수를 만들 때는 =(assignment) 기호를 사용함.
- C언어나 JAVA 처럼 변수의 자료형을 함께 쓸 필요는 없음.(파이썬은 변수에 저장된 값을 스스로 판단하여 자료형을 알아냄.)
- 파이썬에서 사용하는 변수는 객체를 가리키는 것으로 말할 수 있음.

변수명 = 변수에 저장할 값

■ 사용 예

```
a = 3 # 변수 a는 객체가 저장된 메모리의 위치를 가리키는 레퍼런스라고 할 수 있음.  
b = 3
```

```
a is b # a와 b가 동일한 객체를 가리키고 있는지 아닌지에 대해 판단.
```

```
import sys  
sys.getrefcount(3) #자료형에 대한 참조 개수를 알려줌.
```

```
del(a) #객체를 가리키는 변수를 없앴.
```

❖ 제어문

▪ if 문.

if 조건문:

수행할 문장 1

수행할 문장 2

else:

수행할 문장 A

수행할 문자 B

비교 연산자	설명
$x < y$	x가 y보다 작다.
$x > y$	x가 y보다 크다.
$x == y$	x와 y가 같다.
$x != y$	x와 y가 같지 않다.
$x >= y$	x가 y보다 크거나 같다.
$x <= y$	x가 y보다 작거나 같다.

▪ 사용 예

```
money = 2000
if money >= 2000:
    print("택시를 타고 가라.")
else:
    print("걸어 가라")
```

❖ 제어문

▪ if 문.

연산자	설명
x or y	x와 y 둘 중에 하나만 참이면 참이다.
x and y	x와 y 모두 참이어야 참이면 참이다.
not x	x가 거짓이면 참이다.

▪ 사용 예

```
money = 2000
card = 1
if money >= 2000 or card:
    print("택시를 타고 가라.")
else:
    print("걸어 가라")
```

❖ 제어문

▪ if 문.

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

▪ 사용 예

```
1 in [1, 2, 3] # True
1 not in [1, 2, 3] # False
'a' in ('a', 'b', 'c') # True
'l' not in 'Python' # True
```

```
pocket = ['paper', 'cellphone', 'money']
if 'money' in pocket:
    print("택시를 타고 가라") #아무 일도 하지 않게 할려면 pass
elif card:
    print("택시를 타고 가라")
else:
    print("걸어 가라")
```

❖ 제어문

▪ while 문.

while 조건문:

수행할 문장 1

수행할 문장 2

수행할 문장 3

▪ 사용 예

```
treeHit = 0;
while treeHit < 10:
    treeHit = treeHit + 1
    print("나무를 %d번 찍었습니다." % treeHit)
    if treeHit == 10:
        print("나무 넘어갑니다.")
```

❖ 제어문

▪ while 문

▪ 사용 예(break)

```
coffee = 10
while True:
    money = int(input("돈을 넣어 주세요: "))
    if money == 300:
        print("커피를 줍니다.")
        coffee = coffee - 1
    elif money > 300:
        print("거스름돈 %d를 주고 커피를 줍니다." %(money - 300))
        coffee = coffee - 1
    else:
        print("돈을 다시 돌려주고 커피를 주지 않습니다.")
        print("남은 커피의 양은 %d개입니다 " %coffee)
    if not coffee:
        print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
        break
```

❖ 제어문

▪ while 문

▪ 사용 예(continue)

```
a = 0
while a < 10:
    a = a+1
    if a % 2 == 0: continue
    print(a)
```

▪ 사용 예(무한루프)

```
while True:
    print("Ctrl + C를 눌러야 while문을 빠져나갈 수 있습니다.")
```

❖ 제어문

▪ for 문

for 변수 in 리스트(또는 튜플, 문자열)
수행할 문장 1
수행할 문장 2

▪ 사용 예

```
test_list = ["one", "two", "three"]  
for i in test_list:  
    print(i)
```

```
a = [(1, 2), (3, 4), (5, 6)]  
for (first, last) in a: # a리스트의 요소들이 자동으로 (first, last)라는 변수에 대입  
    print(first + last)
```


❖ 제어문

▪ for 문

▪ 사용 예(range 함수)

```
a = range(10) # 0부터 10 미만의 숫자를 포함하는 range 객체를 생성.
```

```
sum = 0
for i in range(1, 11): # 1부터 10까지 더하기
    sum = sum + 1
print(sum)
```

```
mark = [90, 25, 67, 45, 80]
for number in range(len(mark)):
    if mark[number] < 60: continue
    print("%d번 학생 축하합니다. 합격입니다." %(number + 1))
```

❖ 함수

▪ 파이썬 함수 구조

- def는 함수를 만들 때 사용하는 예약어이며, 함수명은 만드는 사람이 임의로 만들 수 있음.
- 함수명 뒤 괄호 안의 입력 인수는 이 함수에 입력될 값이란 뜻.

def 함수명(입력 인수):

수행할 문장 1

수행할 문장 2

▪ 사용 예

```
def sum(a, b):  
    return a + b
```

```
c = sum(1, 2)
```

❖ 함수

- 입력 값이 몇 개가 될지 모를 때

```
def 함수명(*입력 인수):  
    수행할 문장 1  
    수행할 문장 2
```

- 사용 예

```
def sum_many(*args):  
    sum = 0  
    for i in args:  
        sum = sum + i  
    return sum
```

```
result = sum_many(1, 2, 3)  
print(result)
```

❖ 클래스

- 클래스를 정의하면 독자적인 자료형을 만들거나 클래스의 메서드와 속성을 정의할 수 있음.

```
class 클래스 이름:  
    def __init__(self, 인수, ...):  
    def 메서드 이름 1(self, 인수, ...):  
    def 메서드 이름 2(self, 인수, ...):
```

- 사용 예

```
class Man:  
    def __init__(self, name):  
        self.name = name  
        print("Initialized!")  
    def hello(self):  
        print("Hello " + self.name + "!")  
    def goodbye(self):  
        print("Good-Bye " + self.name + "!")  
m = Man("David")  
m.hello()  
m.goodbye()
```

❖ 넘파이

- 기본적으로 array라는 자료를 생성하고 이를 바탕으로 수치 배열 데이터를 편하게 다룰 수 있는 외부 라이브러리로서 상당부분 C나 포트란으로 작성되어 실행속도가 빠름.
- 라이브러리를 사용하기 위해 'import' 문을 이용.

```
import numpy as np
```

❖ 넘파이

- 넘파이 배열을 만들 때는 `np.array()` 메서드를 이용.
- `np.array()` 파이썬의 리스트를 인수로 받아 넘파이의 배열 형태로 반환함.

- 사용 예

```
x = np.array([1.0, 2.0, 3.0])  
print(x)  
#[1, 2, 3]  
type(x)  
# <class 'numpy.ndarray'>
```

❖ 넘파이

▪ 사용 예 (산술 연산)

```
x = np.array([1.0, 2.0, 3.0])  
y = np.array([2.0, 4.0, 6.0])  
x + y  
# ([3. , 6. , 9.])  
x - y  
# ([-1. , -2. , -3.])  
x * y  
# array([2. , 8., 18.])  
x / y  
# array([0.5, 0.5, 0.5])
```

▪ 연산 시 배열의 원소 수가 같아야 함.

❖ 넘파이

■ 사용 예 (N차원 배열)

```
A = np.array([1, 2], [3, 4])  
print(A)  
A.shape # 행렬의 형상  
A.dtype # 행렬 원소의 자료형
```

```
B = np.array([3, 0], [0, 6])  
A + B #원소 별 계산  
A * B #원소 별 계산
```


❖ 넘파이

■ 사용 예 (원소 접근)

```
x = np.array([51, 55], [14, 19], [0, 4])
x[0]
# array([51, 55])
x[0][1]
# 55
```

```
for row in x
    print(row)
[51 55]
[14 19]
[0 4]
```

```
x = x.flatten() # 1차원 배열로 변환
[51 55 14 19 0 4]
x[np.array(0, 2, 4)] # 0, 2, 4 인덱스 원소 접근
array([51, 14, 0])
```

❖ matplotlib

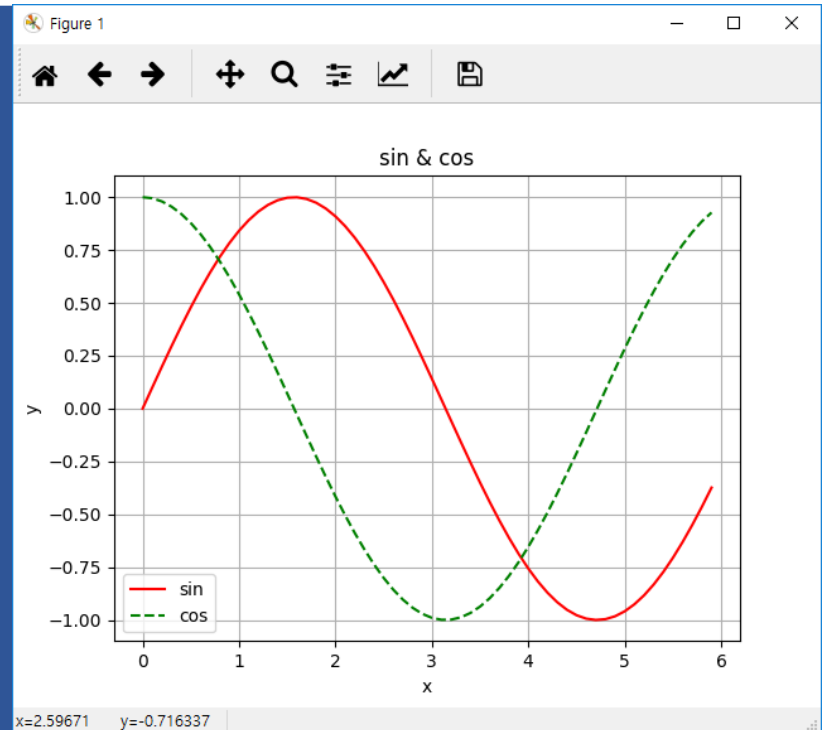
- 데이터의 시각화를 위해 그래프를 그려주는 라이브러리.

import matplotlib.pyplot as plt

- 사용 예

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 6, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)

plt.plot(x, y1, 'r', label = "sin")
plt.plot(x, y2, 'g', linestyle = "--", label = "cos")
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
plt.title('sin & cos')
plt.legend()
plt.show()
```

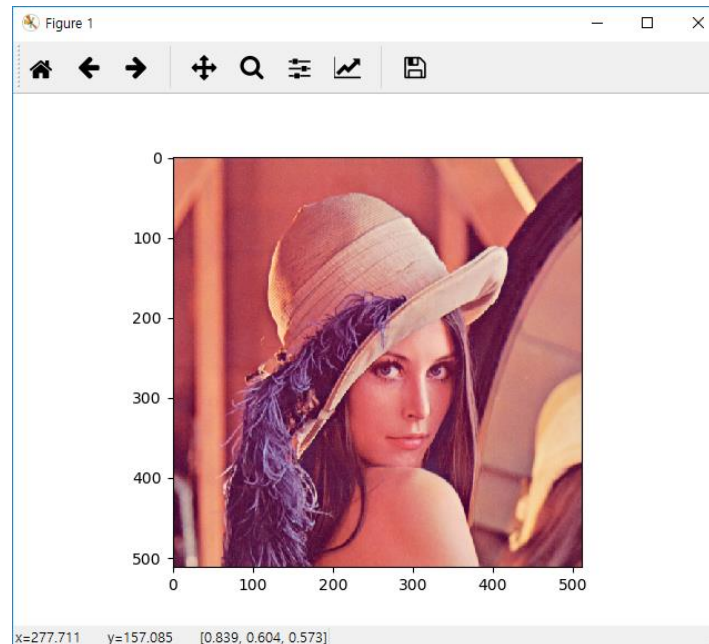


❖ matplotlib

■ 사용 예(이미지 출력하기)

```
import matplotlib.pyplot as plt
import matplotlib.image as img

img = img.imread('Lenna.png')
plt.imshow(img)
plt.show()
```



❖ matplotlib

▪ 사용 예(그래프 출력하기)

- `numpy.random.randn(d0, d1, ..., dn)`

- generates an array of shape `(d0, d1, ..., dn)`, filled with random floats sampled from a univariate "normal" (Gaussian) distribution of mean 0 and variance 1.

```
dataNum = 2000  
trainData_c1 = np.random.randn(dataNum, 2)  
trainData_c2 = np.random.randn(dataNum, 2) + 5
```

❖ matplotlib

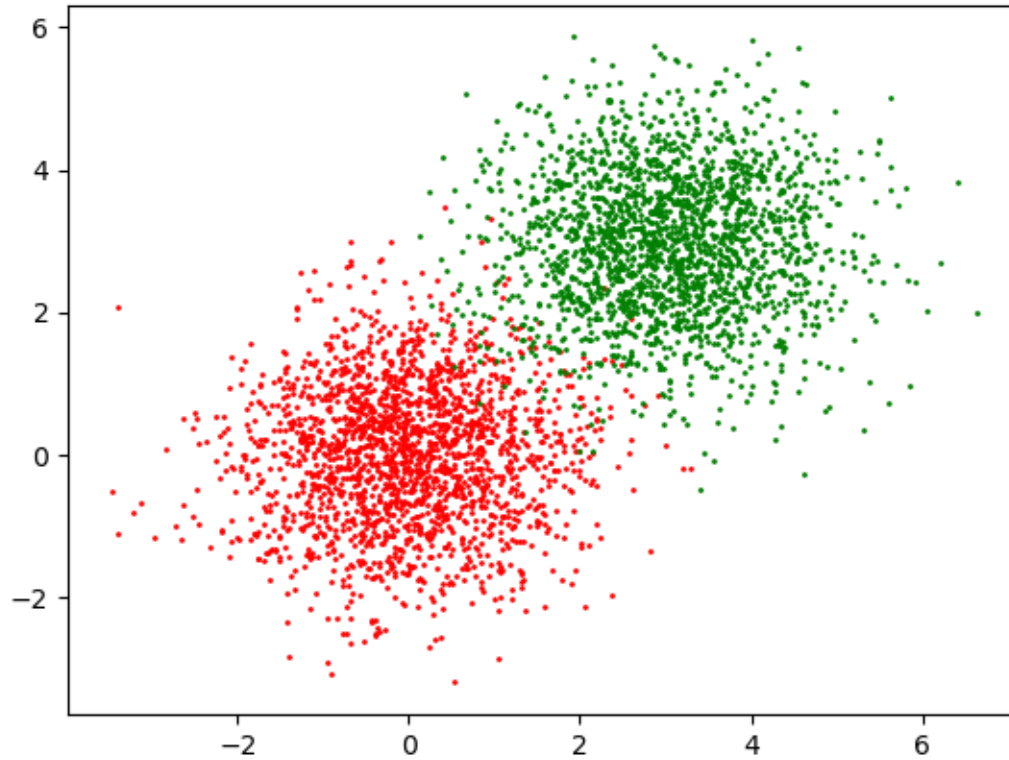
▪ 사용 예(그래프 출력하기)

```
trainData_c1 = np.random.randn(dataNum, 2)
trainData_c2 = np.random.randn(dataNum, 2) + 3

plt.plot(trainData_c1[:, 0], trainData_c1[:, 1], 'ro', markersize = 1);
plt.plot(trainData_c2[:, 0], trainData_c2[:, 1], 'go', markersize = 1);

plt.show();
```

❖ matplotlib



❖ matplotlib

- Plot에 사용되는 컬러, 선, 마커 지정자

marker	symbol	description
"."	•	point
","	.	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	⋿	tri_down
"2"	⋈	tri_up
"3"	⋊	tri_left
"4"	⋋	tri_right
"8"	◼	octagon
"s"	■	square
"p"	⬠	pentagon
"P"	⬢	plus (filled)
"*"	★	star
"h"	⬡	hexagon1
"H"	⬢	hexagon2
"+"	+	plus
"x"	×	x
"X"	⊠	x (filled)
"D"	◆	diamond
"d"	◊	thin_diamond

❖ matplotlib

- Plot에 사용되는 컬러, 선, 마커 지정자

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

❖ matplotlib

▪ Exercise 1

