

---

# Python 개요

---

Chungbuk National University, Korea  
Intelligent Robots Lab. (IRL)

Prof. Gon-Woo Kim

# Python 소개

---

## ■ 파이썬 언어

### ■ 공식 소개글

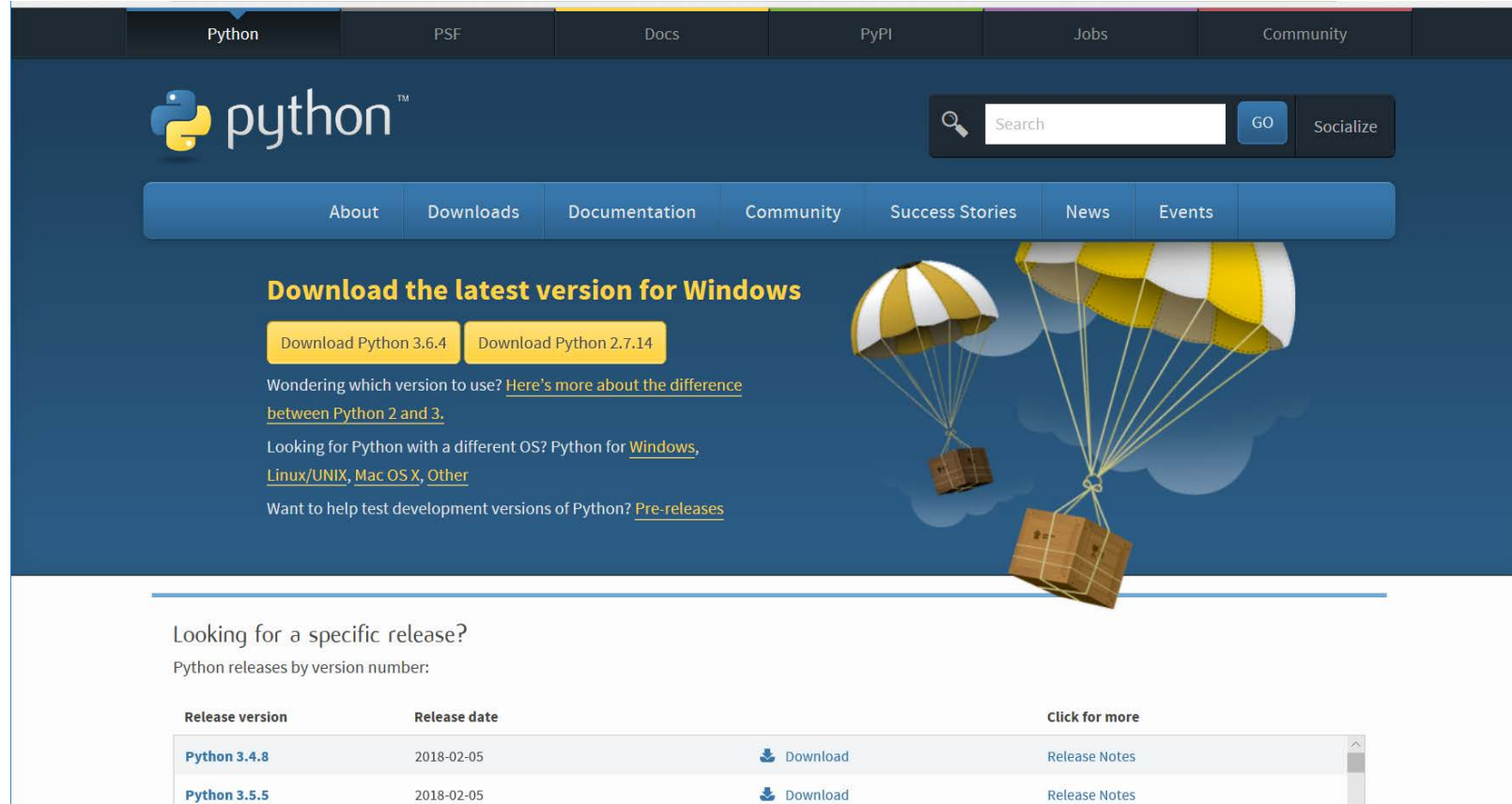
- 파이썬은 배우기 쉽고, 강력한 프로그래밍 언어입니다. 파이썬은 효율적인 고수준 데이터 구조를 갖추고 있으며, 간단하지만 효과적인 객체 지향 프로그래밍 접근법 또한 갖추고 있습니다. 우아한 문법과 동적 타이핑, 그리고 인터프리팅 환경을 갖춘 파이썬은 다양한 분야, 다양한 플랫폼에서 사용될 수 있는 최적의 스크립팅, RAD(rapid application development - 빠른 프로그램 개발) 언어입니다.

### ■ 특징

- 단순함
- 배우기 쉬움
- 자유, 오픈 소스 소프트웨어
- 고수준 언어
- 이식성
- 객체 지향 언어
- 확장성
- 확장 가능한 라이브러리

# Python 설치

- 파이썬 설치
  - 다운로드 사이트
    - <https://www.python.org/downloads/>



The screenshot shows the Python.org website. At the top, there's a navigation bar with links: Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar. A secondary navigation bar contains links: About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a large banner with the text "Download the latest version for Windows" and two buttons: "Download Python 3.6.4" and "Download Python 2.7.14". Below these buttons, there's text explaining the difference between Python 2 and 3, and links for other operating systems (Windows, Linux/UNIX, Mac OS X, Other) and pre-releases. To the right of the text is an illustration of two parachutes carrying boxes. Below the banner, there's a section titled "Looking for a specific release?" with a link to "Python releases by version number:". This leads to a table of recent releases.

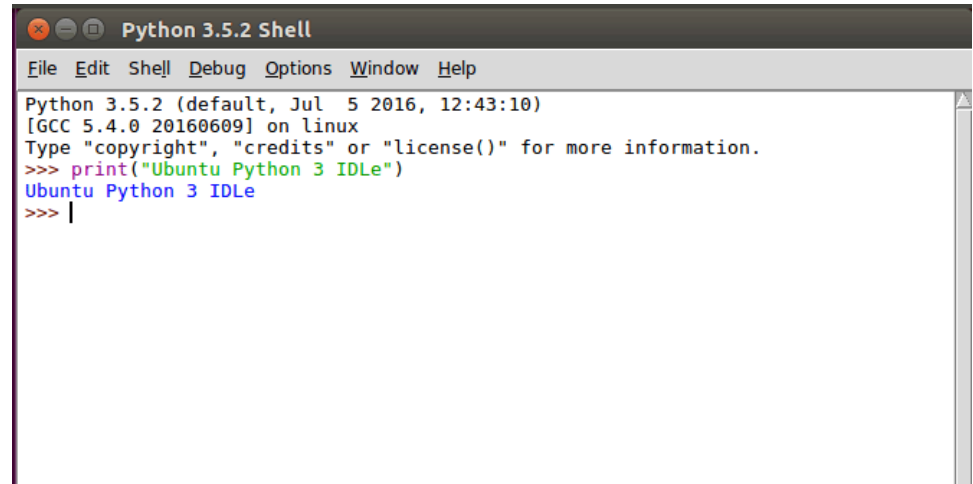
Release version	Release date		Click for more
<a href="#">Python 3.4.8</a>	2018-02-05	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.5.5</a>	2018-02-05	<a href="#">Download</a>	<a href="#">Release Notes</a>

# Python 설치

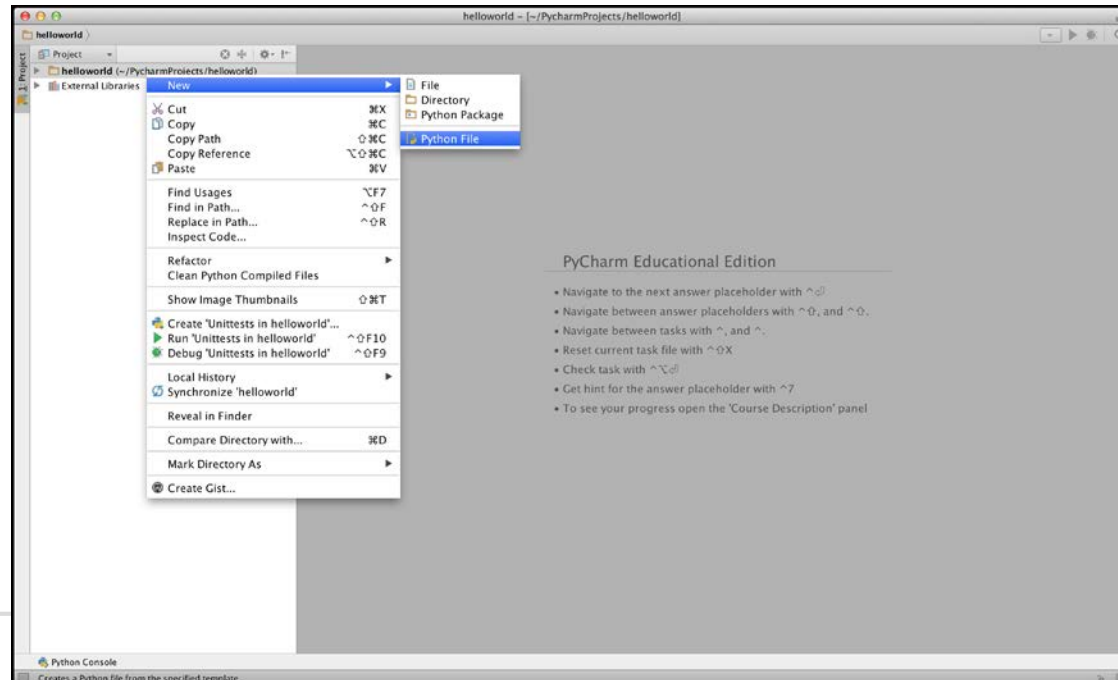
- 파이썬 실행
  - 커맨드라인 실행
    - 대화식 인터프리터 실행
  - 프로그램 파일 작성 실행
    - 일반적으로 '.py' 확장자 사용

- 프로그래밍 환경

- IDLE
- PyCharm
- Anaconda



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (default, Jul 5 2016, 12:43:10)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>> print("Ubuntu Python 3 IDLe")
Ubuntu Python 3 IDLe
>>> |
```



# Python 설치

## ■ Anaconda 설치

- 데이터 분석에 필요한 많은 Package들을 기본으로 포함하고 있어서 의존성 관리에 용이함
- 다운로드: <https://www.continuum.io/downloads>

The image shows a screenshot of the Anaconda website. The main heading is "Download Anaconda Distribution" with the version "5.1" and release date "February 15, 2018". Below this, there are icons for Windows, macOS, and Linux. The page is divided into two main sections: "Anaconda 5.1 For Windows Installer" and "Anaconda 5.1 For Linux Installer". Each section has two columns for "Python 3.6 version" and "Python 2.7 version", each with a "Download" button. Below the buttons, there are links to specific installers: "64-Bit (x86) Installer (551 MB)", "64-Bit (Power8) Installer (286 MB)", and "32-Bit Installer (450 MB)" for Windows; and "64-Bit (x86) Installer (533 MB)", "64-Bit (Power8) Installer (267 MB)", and "32-Bit Installer (431 MB)" for Linux. At the bottom, there are links for "How to get Python 3.5 or other Python versions" and "How to Install ANACONDA".

ANACONDA

What is Anaconda? Products Support Community About Resources [Download](#)

### Download Anaconda Distribution

Version 5.1 | Release Date: February 15, 2018

Download For:

High-Performance Distribution  
Easily install 1,000+ [data science packages](#)

Package Management  
Manage packages, dependencies and environments with [conda](#)

Portal to  
Uncover insight, create interactive visualizations

[packages](#) and environments with [conda](#)

Windows macOS Linux

#### Anaconda 5.1 For Windows Installer

Python 3.6 version [Download](#)

Python 2.7 version [Download](#)

#### Anaconda 5.1 For Linux Installer

Python 3.6 version [Download](#)

64-Bit (x86) Installer (551 MB) ⓘ  
64-Bit (Power8) Installer (286 MB)  
32-Bit Installer (450 MB)

Python 2.7 version [Download](#)

64-Bit (x86) Installer (533 MB) ⓘ  
64-Bit (Power8) Installer (267 MB)  
32-Bit Installer (431 MB)


[How to get Python 3.5 or other Python versions](#)  
[How to Install ANACONDA](#)

# Python 설치

## ■ Anaconda 실행

### ■ 설치시, 환경변수를 자동으로 잡아주는 등 편리한 기능 탑재

FileHelp

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments


Projects (beta)

Learning

Community

Applications onbase (root)Channels


Refresh



jupyterlab0.31.5

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.


Launch



jupyter notebook5.4.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.


Launch



qtconsole4.3.1

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.


Launch



spyder3.2.6

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features


Launch



vscode1.21.0

Streamlined code editor with support for development operations like debugging, task running and version control.


Launch



glueviz0.12.4

Multidimensional data visualization across files. Explore relationships within and among related datasets.


Install



orange33.4.1

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Install



rstudio1.1.383




A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Install

Documentation

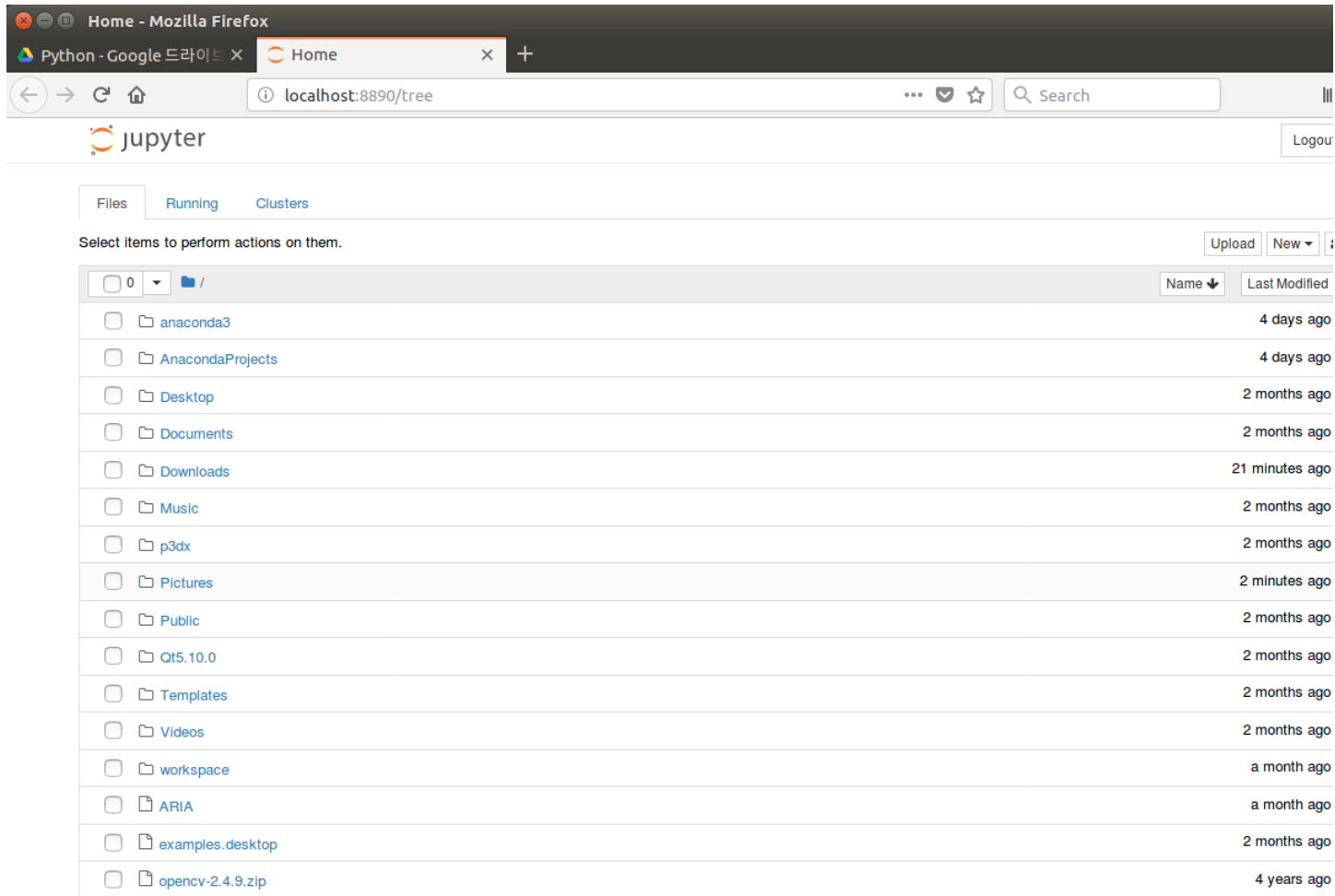
Developer Blog

Feedback



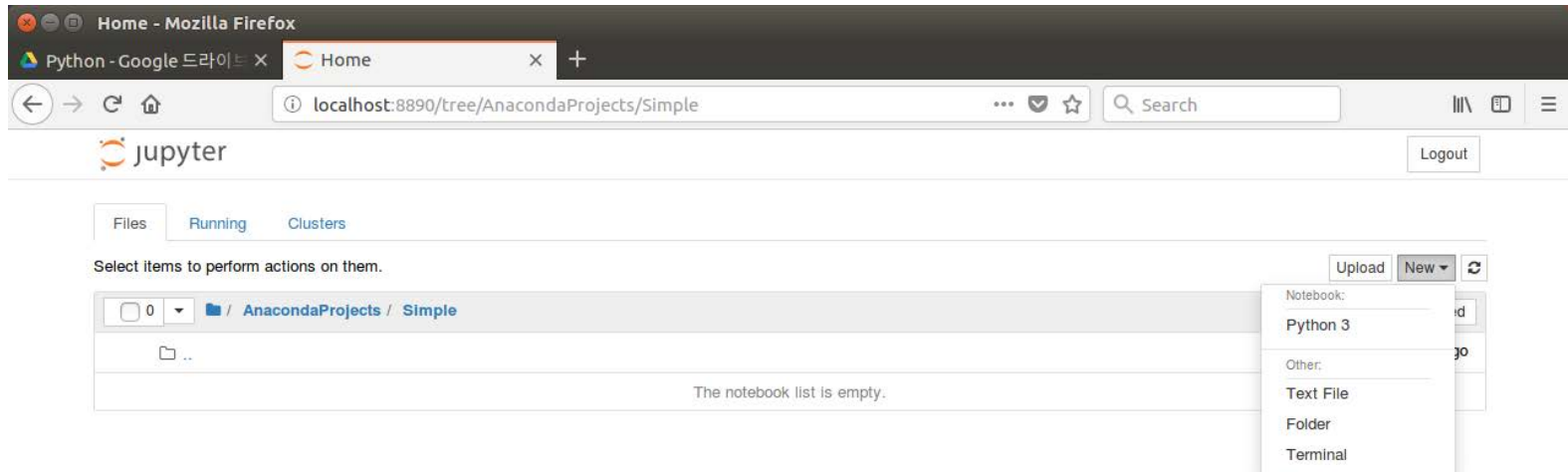
# Python 설치

- Jupyter 노트북
  - Web기반 Notebook Interface



# Python 설치

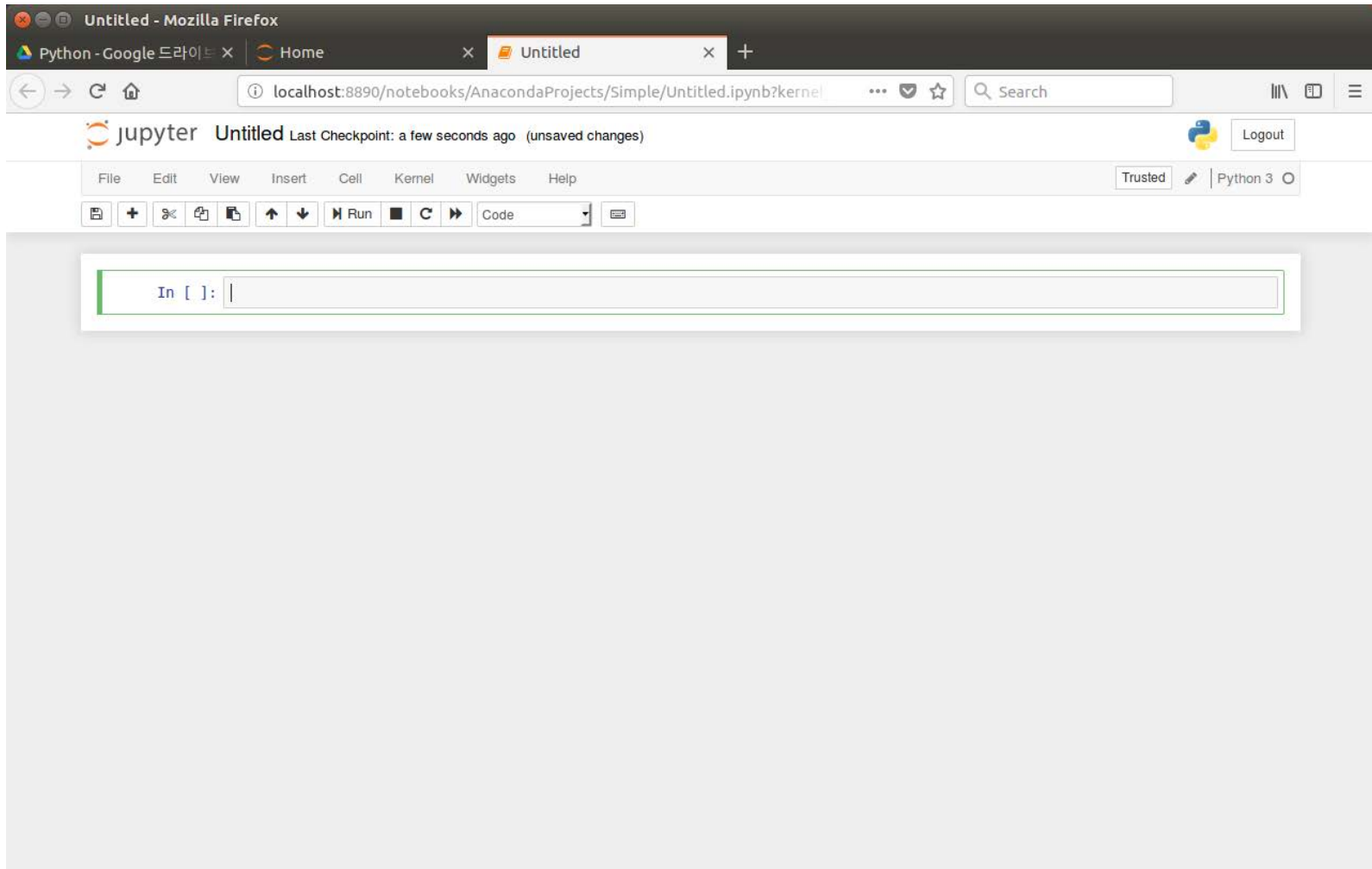
- Jupyter 노트북
  - Web기반 Notebook Interface





# Python 설치

- Jupyter 노트북
  - Web기반 Notebook Interface



# Python 프로그래밍

## ■ 데이터 타입

종류	설명	예제
수치형	정수형(Integer), 실수형(float), 복소수(Complex number) 등의 숫자를 나타내는 자료형	1,2,-12,-321.0, 321.0, 34-e3
문자형	문자들의 모임	"Hello World" 'helloworld' """This is long text, This is second line."""
리스트	파이선에서 가장 많이 사용되는 자료형, 순서를 가지는 자료의 집합. Array의 한 종류	[1,2,3,4,5] ['a','b','c','d']
튜플	리스트와 비슷한 형식으로 값의 변경 불가	(1,2,3,4,5) ( 'a','b','c','d')
딕셔너리	Key-value형식의 자료구조 리스트와 유사하지만 오프셋이 아닌 key로 항목을 선택	{ 'a': 'apple', 'b': 'banana', 'c': 'cheese' }

# Python 프로그래밍

## ■ 데이터 타입 예제

### ■ 수치형

```
>>> a = 7                # 변수
>>> print(a)
7
>>> b = a
>>> print(b)
7
>>> type(a)              # 데이터 타입
<class 'int'>
>>> type(b)
<class 'int'>
>>> type(58)
<class 'int'>
>>> type(99.9)
<class 'float'>
>>> type('abc')
<class 'str'>
>>> a = 95
>>> a -= 3
>>> a
92
>>> 10                   # 진수
10
>>> 0b10
2
>>> 0x10
16
```

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

< 파이썬 예약어 >

Operator	Description	Example	Result
+	addition	5 + 8	13
-	subtraction	90 - 10	80
*	multiplication	4 * 7	28
/	floating point division	7 / 2	3.5
//	integer (truncating) division	7 // 2	3
%	modulus (remainder)	7 % 3	1
**	exponentiation	3 ** 4	81

< 연산자 >



Intelligent Robots Lab.  
Chungbuk National University

# Python 프로그래밍

## ■ 데이터 타입 예제

### ■ 수치형

```
>>> int(True)          # 형변환
1
>>> float(False)
0.0
>>> int(98.6)
98
>>> int(1.0e4)
10000
>>> int('99')
99
>>> int('-23')
-23
>>> int('+12')
12
>>> float(98)
98.0
>>> float('99')
99.0
```

### 문자형

```
>>> 'Snap'
'Snap'
>>> "Crackle"
'Crackle'
>>> '''Boom!'''
'Boom!'
>>> """Eek!"""
'Eek!'
>>> print(99, 'bottles', 'would be enough.')
99 bottles would be enough.
>>> bottles = 99
>>> base = ''
>>> base += 'current inventory: '
>>> base += str(bottles)
>>> base
'current inventory: 99'
>>> str(98.6)          # 형변환
'98.6'
>>> str(1.0e4)
'10000.0'
>>> str(True)
'True'
>>> palindrome = 'A man,\nA plan:' # 이스케이프 문자
>>> print(palindrome)
A man,
A plan:
```

# Python 프로그래밍

## ■ 데이터 타입 예제: 리스트

### ■ 리스트 생성: [] or list()

```
>>> empty_list = [ ] # []을 이용한 생성
>>> weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
>>> big_birds = ['emu', 'ostrich', 'cassowary']
>>> first_names = ['Graham', 'John', 'Terry', 'Terry', 'Michael']
>>> another_empty_list = list() # list()를 이용한 생성
>>> another_empty_list
[]
```

### ■ 다른 데이터 타입을 리스트로 변환

```
>>> list('cat')
['c', 'a', 't']
>>> a_tuple = ('ready', 'fire', 'aim')
>>> list(a_tuple)
['ready', 'fire', 'aim']
>>> birthday = '1/6/1952'
>>> birthday.split('/')
['1', '6', '1952']
```

### ■ [offset]으로 항목 얻기

```
>>> marxes = ['Groucho', 'Chico', 'Harpo']
>>> marxes[0]
'Groucho'
>>> marxes[1]
'Chico'
>>> marxes[2]
'Harpo'
```

```
>>> marxes = ['Groucho', 'Chico', 'Harpo']
>>> marxes[2] = 'Wanda'
>>> marxes
['Groucho', 'Chico', 'Wanda']
```



# Python 프로그래밍

## ■ 데이터 타입 예제

### ■ 슬라이스로 항목 얻기

```
>>> marxses = ['Groucho', 'Chico', 'Harpo']
>>> marxses[0:2]
['Groucho', 'Chico']
```

### ■ 리스트 병합: extend() or +=

```
>>> marxses = ['Groucho', 'Chico', 'Harpo', 'Zeppo']
>>> others = ['Gummo', 'Karl']
>>> marxses.extend(others)
>>> marxses
['Groucho', 'Chico', 'Harpo', 'Zeppo', 'Gummo', 'Karl']
```

OR      >>> marxses += others

### ■ 존재여부 확인하기: in

```
>>> words = ['a', 'deer', 'a', 'female', 'deer']
>>> 'deer' in words
True
```

### ■ 정렬하기: sort()

```
>>> marxses.sort()
>>> marxses
['Chico', 'Groucho', 'Harpo']
>>> numbers = [2, 1, 4.0, 3]
>>> numbers.sort(reverse=True)
>>> numbers
[4.0, 3, 2, 1]
```



# Python 프로그래밍

## ■ 데이터 타입 예제: 튜플

\* 튜플은 리스트와 다르게 데이터 변경이 불가함

### ■ 튜플 생성: ()

```
>>> empty_tuple = ()
>>> empty_tuple
()
>>> marx_tuple = 'Groucho', 'Chico', 'Harpo'
>>> marx_tuple
('Groucho', 'Chico', 'Harpo')
```

```
>>> marx_tuple = ('Groucho', 'Chico', 'Harpo')
>>> marx_tuple
('Groucho', 'Chico', 'Harpo')
```

### ■ 한 번에 여러 변수 할당

```
>>> marx_tuple = ('Groucho', 'Chico', 'Harpo')
>>> a, b, c = marx_tuple
>>> a
'Groucho'
>>> b
'Chico'
>>> c
'Harpo'
```

### ■ 튜플 변환: tuple()

```
>>> marx_list = ['Groucho', 'Chico', 'Harpo']
>>> tuple(marx_list)
('Groucho', 'Chico', 'Harpo')
```

# Python 프로그래밍

## ■ 데이터 타입 예제: 딕셔너리

### ■ 딕셔너리 생성: {} → 키:값 쌍으로 생성

```
>>> empty_dict = {}
>>> empty_dict
{}
>>> bierce = {
...     "day": "A period of twenty-four hours, mostly misspent",
...     "positive": "Mistaken at the top of one's voice",
...     "misfortune": "The kind of fortune that never misses",
... }
>>>
```

### ■ 딕셔너리로 변환하기: dict()

```
>>> lol = [ ['a', 'b'], ['c', 'd'], ['e', 'f'] ]
>>> dict(lol)
{'c': 'd', 'a': 'b', 'e': 'f'}
```

### ■ 항목 추가/변경: [key]

```
>>> pythons = {
...     'Chapman': 'Graham',
...     'Cleese': 'John',
...     'Idle': 'Eric',
...     'Jones': 'Terry',
...     'Palin': 'Michael',
... }
>>> pythons
```

```
{'Cleese': 'John', 'Jones': 'Terry', 'Palin': 'Michael',
'Chapman': 'Graham', 'Idle': 'Eric'}
```

```
>>> pythons['Gilliam'] = 'Gerry'
```

```
>>> pythons
```

```
{'Cleese': 'John', 'Gilliam': 'Gerry', 'Palin': 'Michael',
'Chapman': 'Graham', 'Idle': 'Eric', 'Jones': 'Terry'}
```





# Python 프로그래밍

## ■ 데이터 타입 예제: 딕셔너리

### ■ 딕셔너리 결합하기: update()

```
>>> others = { 'Marx': 'Groucho', 'Howard': 'Moe' }
>>> pythons.update(others)
>>> pythons
{'Cleese': 'John', 'Howard': 'Moe', 'Gilliam': 'Terry',
 'Palin': 'Michael', 'Marx': 'Groucho', 'Chapman': 'Graham',
 'Idle': 'Eric', 'Jones': 'Terry'}
```

### ■ 항목/모든 항목 삭제: del, clear()

```
>>> del pythons['Marx']
>>> pythons
{'Cleese': 'John', 'Howard': 'Moe', 'Gilliam': 'Terry',
 'Palin': 'Michael', 'Chapman': 'Graham', 'Idle': 'Eric',
 'Jones': 'Terry'}
>>> pythons.clear()
>>> pythons
{}
```

### ■ 키 멤버십 테스트 및 항목 얻기: in & [key], get()

```
>>> pythons = {'Chapman': 'Graham', 'Cleese': 'John', 'Jones': 'Terry', 'Palin': 'Michael'}
>>> 'Chapman' in pythons
True
>>> 'Gilliam' in pythons
False
>>> pythons['Cleese']
'John'
```

```
>>> pythons.get('Cleese')
'John'
```

# Python 프로그래밍

## ■ 코드 구조

### ■ 주석 처리: #

```
>>> # 60 sec/min * 60 min/hr * 24 hr/day
>>> seconds_per_day = 86400
```

### ■ 조건문: if, elif, else

```
>>> disaster = True
>>> if disaster:
...     print("Woe!")
... else:
...     print("Whee!")
...
Woe!
>>>
```

### ■ 반복문: for, while

```
>>> rabbits = ['Flopsy', 'Mopsy', 'Cottontail', 'Peter']
>>> for rabbit in rabbits:
...     print(rabbit)
...
Flopsy
Mopsy
Cottontail
Peter
```

## 비교연산자

equality	==
inequality	!=
less than	<
less than or equal	<=
greater than	>
greater than or equal	>=
membership	in ...

```
>>> x = 7
>>> x == 5
False
>>> 5 < x and x < 10
True
>>> 5 < x or x < 10
True
>>> 5 < x and x > 10
False
>>> 5 < x and not x > 10
True
```

```
>>> count = 1
>>> while count <= 5:
...     print(count)
...     count += 1
...
1
2
3
4
5
>>>
```

# Python 프로그래밍

## ■ 코드 구조

- 중단하기, 건너뛰기: break, continue
  - for, while 문 동일
- 숫자 시퀀스 생성하기: range()
  - 특정 범위 내에서 숫자 스트림을 반환
  - range(start, stop, step) 형식: 기본값은 start=0, step=1

```
>>> for x in range(0,3):
...     print(x)
...
0
1
2
>>> list( range(0, 3) )
[0, 1, 2]
>>> for x in range(2, -1, -1):    # step이 -1이면 거꾸로 진행
...     print(x)
...
2
1
0
>>> list( range(2, -1, -1) )
[2, 1, 0]
```

# Python 프로그래밍

## ■ 코드 구조: 함수

### ■ 함수

#### ■ 함수 정의와 호출: 매개변수 이용

```
>>> def do_nothing():  
...     Pass
```

```
>>> def make_a_sound():  
...     print('quack')  
...  
>>> make_a_sound()  
quack
```

```
>>> def commentary(color):  
...     if color == 'red':  
...         return "It's a tomato."  
...     elif color == "green":  
...         return "It's a green pepper."  
...     elif color == 'bee purple':  
...         return "I don't know what it is, but only bees can see it."  
...     else:  
...         return "I've never heard of the color " + color + "."  
...  
>>>  
>>> comment = commentary('blue')  
>>> print(comment)  
I've never heard of the color blue.
```

```
>>> def agree():  
...     return True  
...  
>>> if agree():  
...     print('Splendid!')  
... else:  
...     print('That was unexpected.')  
...  
Splendid!
```

# Python 프로그래밍

---

- 모듈(Module)과 패키지(Package)

- 커맨드 라인 인자

- 커맨드 라인 프로그램 실행 시 인자(argument)를 받아서 처리할 수 있음

<test.py 코드>

```
import sys
print('Program arguments:', sys.argv)
```

<표준 셸 프로그램에서 코드 실행>

```
$ python test2.py
Program arguments: ['test2.py']
$ python test2.py tra la la
Program arguments: ['test2.py', 'tra', 'la', 'la']
```

# Python 프로그래밍

---

## ■ 모듈(Module)과 패키지(Package)

### ■ 모듈과 import문

- 모듈(Module): 파이썬 코드로 구성된 파일
- import 문을 이용하여 다른 모듈 코드를 참조: 임포트한 모듈의 코드와 변수를 프로그램에서 사용할 수 있도록 해줌

```
import math  
angle = math.cos(math.pi)
```

```
import numpy as np  
x = np.array([1.0, 2.0, 3.0])
```

### ■ 패키지

- 파이썬 어플리케이션을 좀 더 확장 가능하게 만들기 위해 모듈을 패키지라는 파일 계층구조로 구성할 수 있음
- pip를 이용하여 대부분의 패키지 설치 가능

```
pip install numpy
```

# Python 프로그래밍

## ■ 클래스(class)

### ■ 클래스 선언하기: class

```
>>> class Person():  
...     Pass  
>>> someone = Person()
```

### ■ 객체 초기화 메서드: \_\_init\_\_

```
>>> class Person():  
...     def __init__(self):  
...         Pass
```

- \_\_init\_\_은 특별한 메서드 이름  
- 클래스에서 \_\_init\_\_() 정의 시 첫 번째 매개변수는 self이어야 함

```
>>> class Person():  
...     def __init__(self, name):  
...         self.name = name  
...  
>>>  
>>> hunter = Person('Elmer Fudd')
```

### ■ 코드 동작

- Person 클래스 정의 찾기
- 새 객체를 메모리에 초기화(생성) 함
- 객체의 \_\_init\_\_ 메서드 호출: 인자('Elmer Fudd')를 name에 전달
- 객체에 name 값 저장
- 새로운 객체를 반환하여 hunter에 연결

# Python 프로그래밍

---

## ■ 클래스(class)

### ■ 상속

```
>>> class Car():
...     def exclaim(self):
...         print("I'm a Car!")
...
>>> class Yugo(Car):
...     pass
...
>>> give_me_a_car = Car()
>>> give_me_a_yugo = Yugo()
>>> give_me_a_car.exclaim()
I'm a Car!
>>> give_me_a_yugo.exclaim()
I'm a Car!
```



# Python 프로그래밍

---

- 클래스(class)

- 메서드 오버라이드

- 메서드에 대한 재정의

```
>>> class Car():
...     def exclaim(self):
...         print("I'm a Car!")
...
>>> class Yugo(Car):
...     def exclaim(self):
...         print("I'm a Yugo! Much like a Car, but more Yugo-ish.")
...
>>> give_me_a_car = Car()
>>> give_me_a_yugo = Yugo()
>>> give_me_a_car.exclaim()
I'm a Car!
>>> give_me_a_yugo.exclaim()
I'm a Yugo! Much like a Car, but more Yugo-ish.
```

# Python 프로그래밍

## ■ 클래스(class)

### ■ 메서드 추가

```
>>> class Car():
...     def exclaim(self):
...         print("I'm a Car!")
...
>>> class Yugo(Car):
...     def exclaim(self):
...         print("I'm a Yugo! Much like a Car, but more Yugo-ish.")
...     def need_a_push(self):
...         print("A little help here?")
...
>>> give_me_a_car = Car()
>>> give_me_a_yugo = Yugo()
>>> give_me_a_yugo.need_a_push()
A little help here?
>>> give_me_a_car.need_a_push()
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AttributeError: 'Car' object has no attribute 'need_a_push'
```

# Python 프로그래밍

## ■ 수학 및 통계를 위한 표준 라이브러리

### ■ math

- 파이썬은 표준 math 모듈에서 다양한 수학 함수를 제공

```
>>> import math
>>> math.pi
>>> 3.141592653589793
>>> math.e
>>> 2.718281828459045
>>> math.fabs(98.6)
>>> 98.6
>>> math.fabs(-271.1)
>>> 271.1
>>> math.floor(98.6)
>>> 98
>>> math.floor(-271.1)
>>> -272
>>> math.ceil(98.6)
>>> 99
>>> math.ceil(-271.1)
>>> -271
>>> math.factorial(2)
>>> 2
>>> math.factorial(3)
>>> 6
>>> math.log(1.0)
>>> 0.0
>>> math.log(math.e)
>>> 1.0
```

```
>>> math.log(8, 2)
>>> 3.0
>>> math.pow(2, 3)
>>> 8.0
>>> math.sqrt(100.0)
>>> 10.0
>>> math.radians(180.0)
>>> 3.141592653589793
>>> math.degrees(math.pi)
>>> 180.0
```

# Python 프로그래밍

---

## ■ 주요 외부 라이브러리

### ■ NumPy

- 고성능의 과학계산 컴퓨팅과 데이터 분석에 필요한 기본 패키지
- 빠른 다차원 수 배열을 제공하기 위해 작성됨

### ■ SciPy

- 과학, 분석용 라이브러리로 NumPy 위에서 작성된 수학 및 통계 함수 제공
- 포함된 모듈: 최적화, 통계, 보간(interpolation), 선형 회귀(linear regression), 이미지 처리, 신호처리 등

### ■ matplotlib

- 파이썬과 NumPy에서 plotting을 위해 사용되며 주로 2D 도표를 위한 패키지

# Python 프로그래밍

## ■ NumPy

### ■ 배열 만들기: array()

```
>>> import numpy as np
>>> b = np.array( [2, 4, 6, 8] )
>>> b
array([2, 4, 6, 8])
>>> b.ndim
1
>>> b.size
4
>>> b.shape
(4,)
```

```
>>> x = np.array( [1.0, 2.0, 3.0] )
>>> print(x)
[1. 2. 3.]
>>> type(x)
```

### ■ 배열 만들기: arange()

```
>>> a = np.arange(10)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> a = np.arange(7, 11)
>>> a
array([ 7, 8, 9, 10])
>>> f = np.arange(2.0, 9.8, 0.3)
>>> f
array([ 2. , 2.3, 2.6, 2.9, 3.2, 3.5, 3.8, 4.1, 4.4, 4.7, 5. ,
 5.3, 5.6, 5.9, 6.2, 6.5, 6.8, 7.1, 7.4, 7.7, 8. , 8.3,
 8.6, 8.9, 9.2, 9.5, 9.8])
```

- 하나의 num 정수 인자로 arrange()를 호출하면 0 ~ num-1까지 배열을 반환
- 두 인자인 경우 첫 번째 인자 ~ 두 번째 인자 - 1까지의 배열 생성
- 세 번째 인자는 스텝 크기

# Python 프로그래밍

## ■ NumPy

### ■ 배열 만들기: zeros(), ones(), random()

```
>>> a = np.zeros((3,))
>>> a
array([ 0., 0., 0.])
>>> a.ndim
1
>>> a.shape
(3,)
>>> a.size
3
```

```
>>> b = np.zeros((2, 4))
>>> b
array([[ 0., 0., 0., 0.],
       [ 0., 0., 0., 0.]])
>>> b.ndim
2
>>> b.shape
(2, 4)
>>> b.size
8
```

```
>>> k = np.ones((3, 5))
>>> m = np.random.random((3, 5))
```

### ■ 배열 모양 바꾸기: reshape(), flatten()

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

```
>>> a = a.reshape(5, 2)
>>> a = a.flatten()
```

튜플값을 할당하여 배열 모양 변경

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

# Python 프로그래밍

## ■ NumPy

### ■ 산술연산

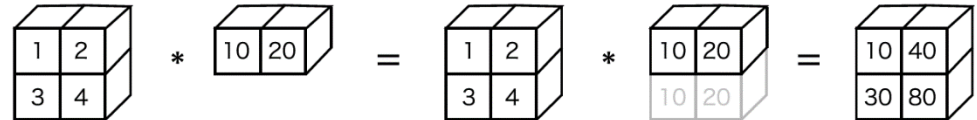
```
>>> x = np.array( [1.0, 2.0, 3.0] )
>>> x = np.array( [1.0, 2.0, 3.0] )
>>> x + y      # 원소별 덧셈
>>> x - y
>>> x * y      # 원소별 곱셈
>>> x / y
>>> x / 2.0    # 브로드캐스트
```

### ■ 행렬곱: @

```
>>> A = np.array([[1, 2], [3, 4]] )
>>> B = A @ B
```

### ■ 브로드캐스트

```
>>> A = np.array([[1, 2], [3, 4]] )
>>> B = np.array([10, 20] )
>>> A * B
```



### ■ 원소 접근: []

```
>>> a = np.arange(10)
>>> a[7]
7
>>> a[-1]
9
```

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

# Python 프로그래밍

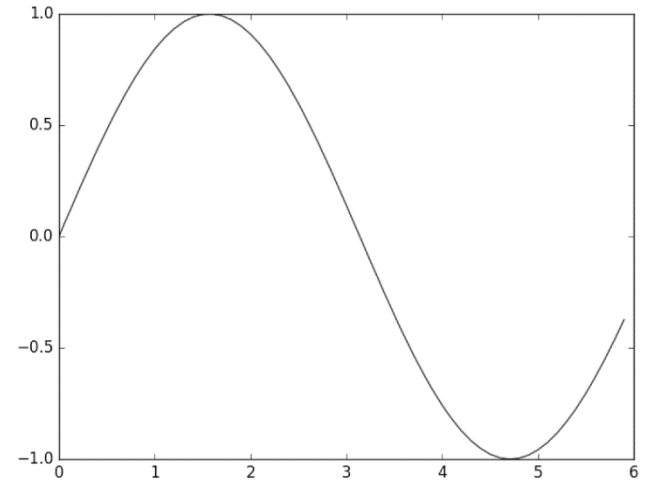
## ■ matplotlib

### ■ 그래프 그리기: pyplot

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

```
x = np.arange(0, 6, 0.1)
y = np.sin(x)
```

```
plt.plot(x, y)
plt.show()
```



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

```
x = np.arange(0, 6, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)
```

```
# 그래프 그리기
```

```
plt.plot(x, y1, label="sin")
plt.plot(x, y2, linestyle = "--", label="cos") # cos 함수는 점선으로 그리기
plt.xlabel("x") # x축 이름
plt.ylabel("y") # y축 이름
plt.title('sin & cos')
plt.legend()
plt.show()
```



# Python 프로그래밍

## ■ matplotlib

### ■ 이미지 표시하기: imshow(), imread()

```
import matplotlib.pyplot as plt  
from matplotlib.image import imread
```

```
img = imread('../dataset/lena.png') # 이미지 저장 경로 작성
```

```
plt.imshow(img)  
plt.show()
```

