


# Python Tutorial 1

Overview, Install, Type, Flow

Haesun Park, [haesunrpark@gmail.com](mailto:haesunrpark@gmail.com),

# Overview

# How

doing with  play for

# History

- 네덜란드 개발자 귀도 반 로섬(Guido van Rossum)이 1989년에 개발 시작
- 1994년 1.0 Release, 2000년에 2.0 Release, 2010년에 2.7 Release
- 2008년에 3.0 Release, 2015년에 3.5 Release
- Python 2.7 버전은 2020년 까지만 지원



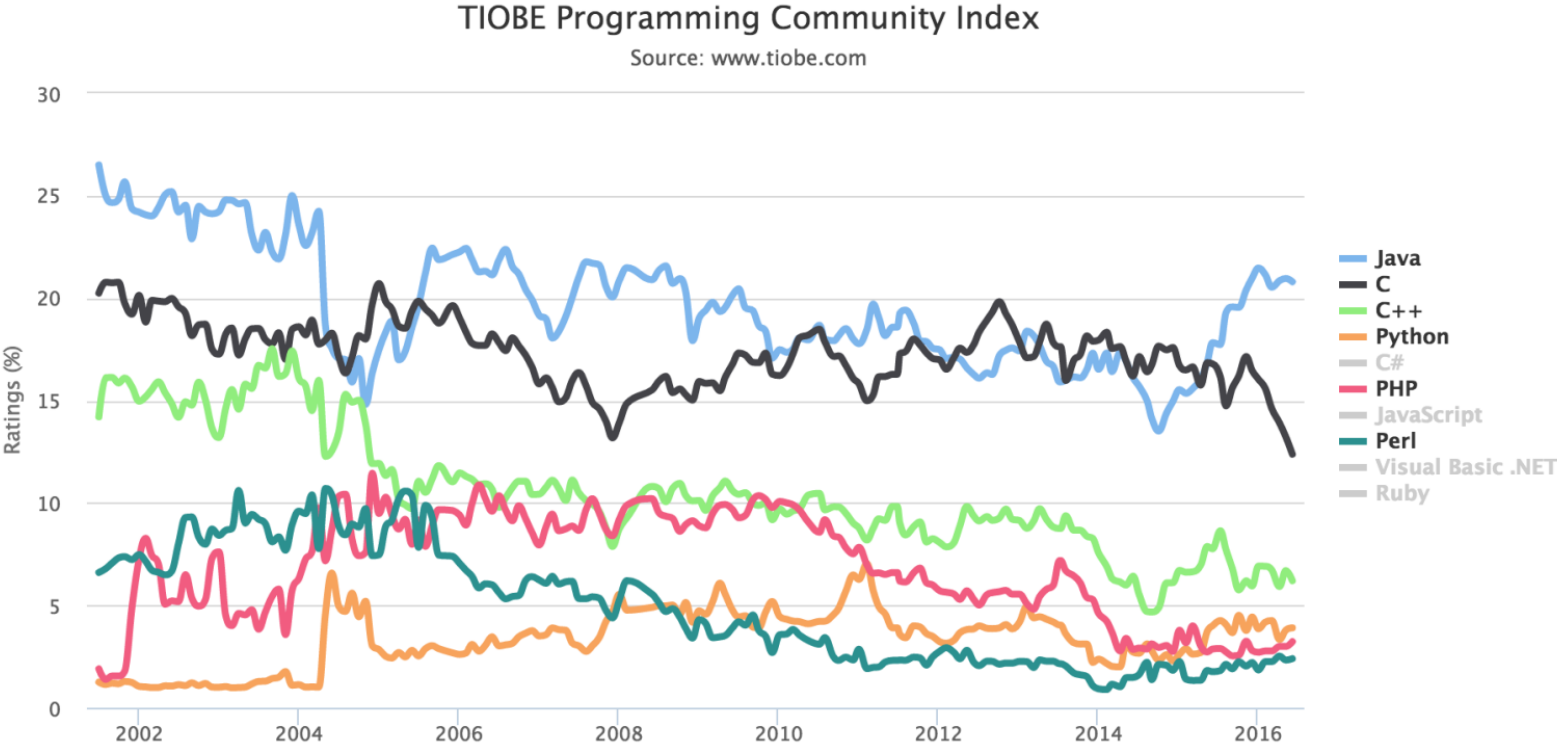
Guido van Rossum,  
Dropbox 2012~  
출처: wikipedia

# Python 2 vs 3

어떤 것을 써야할 지 잘 모르겠다면  
파이썬 3이 적절합니다.

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

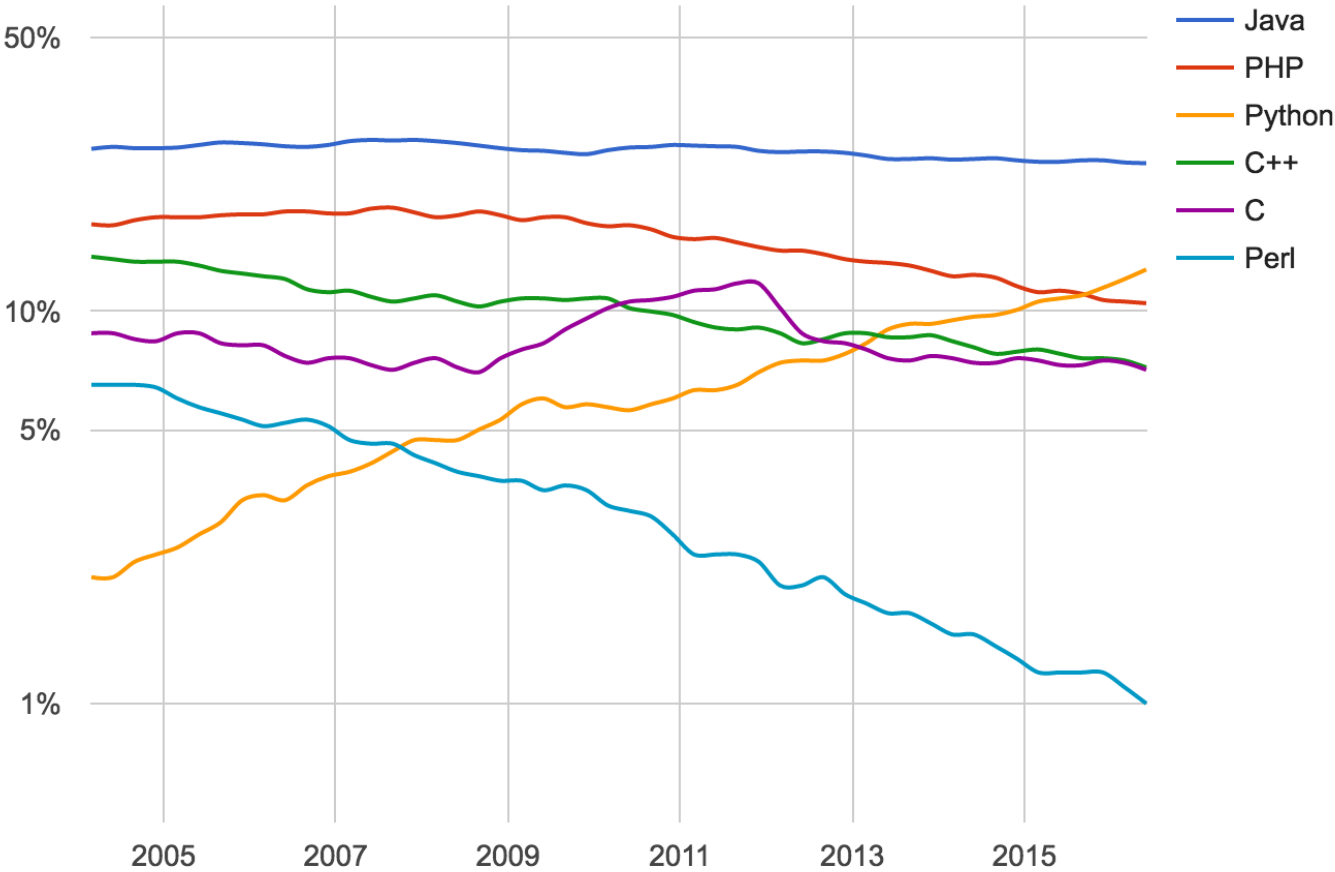
# Trend



	Jun 2016	Jun 2015	Change	Programming Language	Ratings	Change
1	1			Java	20.794%	+2.97%
2	2			C	12.376%	-4.41%
3	3			C++	6.199%	-1.56%
4	6		▲	Python	3.900%	-0.10%
5	4		▼	C#	3.786%	-1.27%

# Trend

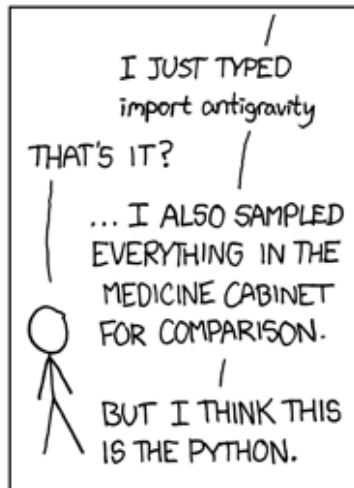
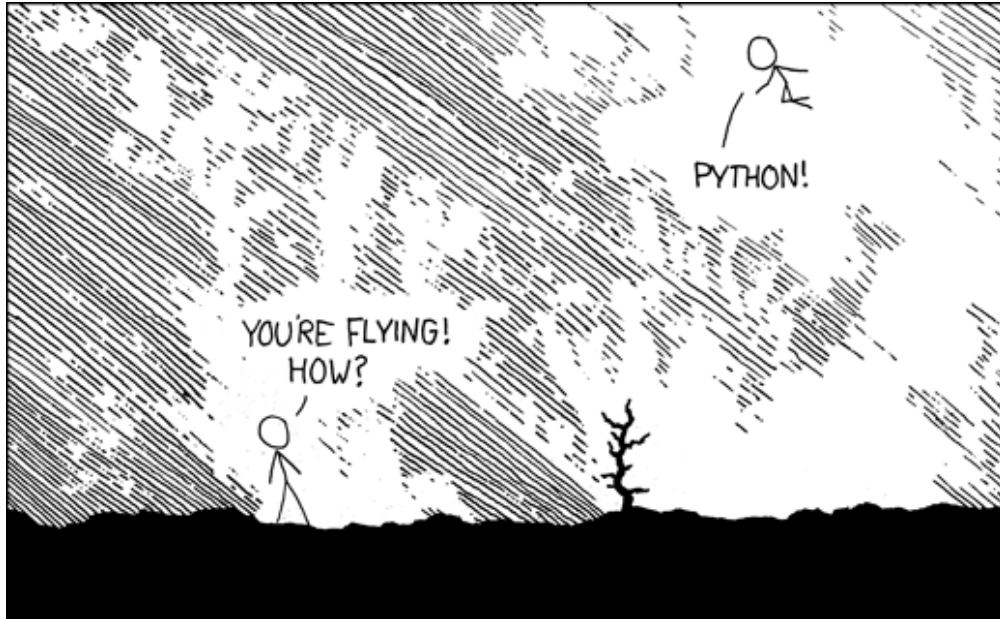
PYPL Popularity of Programming Language



Worldwide, Jun 2016 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	23.9 %	-0.5 %
2	↑	Python	12.8 %	+2.1 %
3	↓	PHP	10.5 %	-0.8 %
4		C#	8.8 %	-0.5 %
5	↑↑	Javascript	7.7 %	+0.6 %
6	↓	C++	7.2 %	-0.4 %

# Why Python



Good Readability

Whole world community

Everything is included

Used for teaching CS at MIT, UC Berkeley, ...

Django

Statistics (scipy, numpy)

Data Analysis (ipython, pandas, matplotlib)

Machine Learning (scikit-learn, nltk)

Deep Learning (Theano, TensorFlow)

and more ...



# 특징

Python	Java	Perl	Ruby
dynamic type	static type	dynamic type	dynamic type
interprete	compile	script	interprete
less develop	fast execute	unreadable code	less develop
battery included		more modules	
Flask, Django	Spring, Struts		Rails
strict indentation	android	text processing	
more object-oriented	more object-oriented	less object-oriented	more object-oriented

Install

# Implementation

CPython	Reference Implementation, 2.7.12 / 3.5.2	<a href="http://www.python.org">www.python.org</a>
PyPy	RPython, JIT compile, 2.7.10 / 3.3.5, for performance	<a href="http://www.pypy.org">www.pypy.org</a>
Jython	written java, JVM, 2.7.0, for using java classes	<a href="http://www.jython.org">www.jython.org</a>

# Installer

## Mac & Linux

- come with python 2.7 out of the box

## python.org

- windows, mac
- linux 배포판 패키지 관리툴 (e.g. yum, apt-get)

- ActiveState, Canopy

- windows, mac, linux

- Anaconda

- continuum.io
- 720+ for stats, data mining, ml, dl, nlp, ...
- include Numpy, SciPy, pandas, scikit-learn, Jupyter, NLTK, matplotlib, ...

# Installer

← → ↻ [https://www.continuum.io/downloads#\\_windows](https://www.continuum.io/downloads#_windows)

If you don't have time or disk space for the entire distribution, try **Miniconda**, which contains only conda and Python. Then install just the individual packages you want through the conda command.

## Anaconda for Windows

PYTHON 2.7	PYTHON 3.5
<div>WINDOWS 64-BIT GRAPHICAL INSTALLER</div> <div>340M</div>	<div>WINDOWS 64-BIT GRAPHICAL INSTALLER</div> <div>351M</div>
<div>Windows 32-bit Graphical Installer</div> <div>285M</div>	<div>Windows 32-bit Graphical Installer</div> <div>292M</div>
Behind a firewall? Use these <a href="#">zipped Windows installers</a> .	

## Windows Anaconda Installation

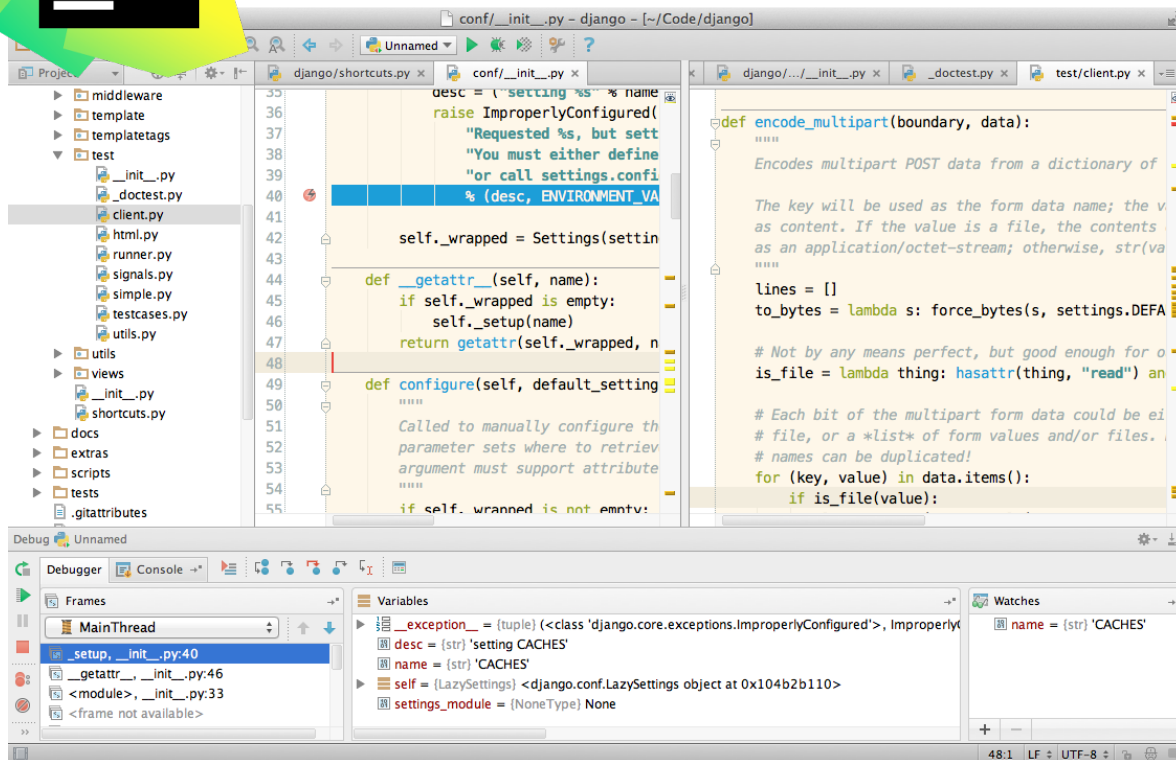
1. Download the graphical installer.
2. Optional: **Verify data integrity with SHA-256**.
3. Double-click the .exe file to install Anaconda and follow the instructions on the screen.

# IDE

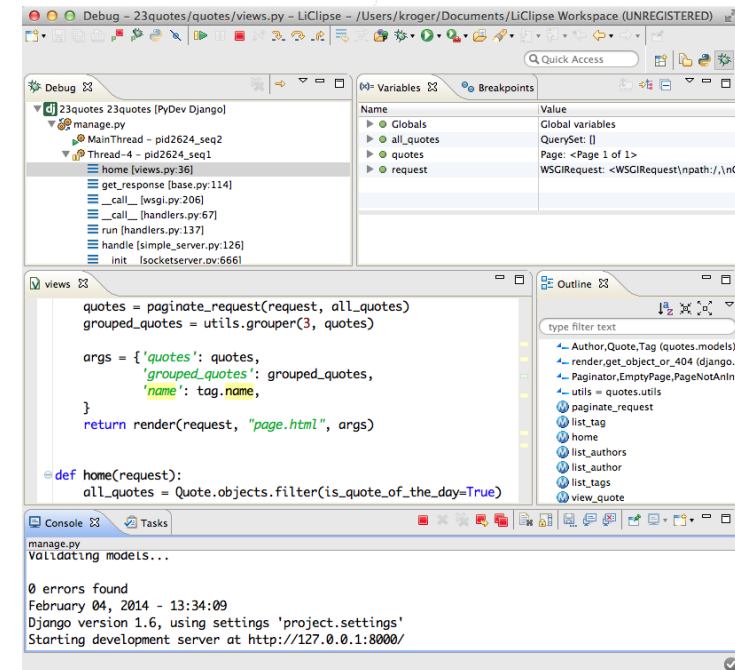
Text Editor is Good.  
e.g. Sublime Text, TextMate



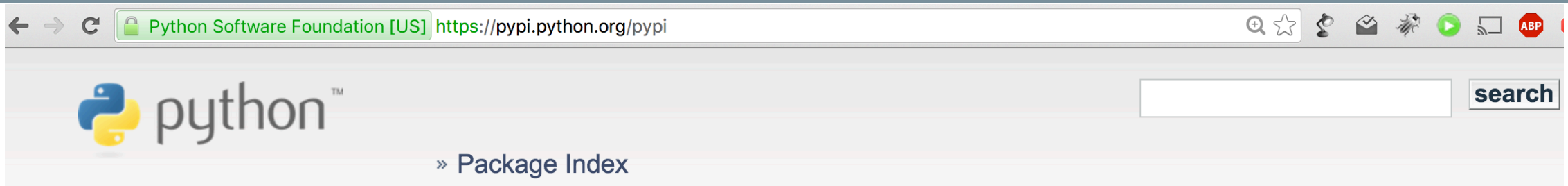
<https://www.jetbrains.com/pycharm/>



<http://www.pydev.org/>



# Package



## PACKAGE INDEX >>

- [Browse packages](#)
- [Package submission](#)
- [List trove classifiers](#)
- [List packages](#)
- [RSS \(latest 40 updates\)](#)
- [RSS \(newest 40 packages\)](#)
- [Python 3 Packages](#)
- [PyPI Tutorial](#)
- [PyPI Security](#)
- [PyPI Support](#)
- [PyPI Bug Reports](#)
- [PyPI Discussion](#)
- [PyPI Developer Info](#)

ABOUT >>

NEWS >>

DOCUMENTATION >>

DOWNLOAD >>

## PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **83597** packages here. To contact the PyPI admins, please use the [Support](#) or [Bug reports](#) links.



### Get Packages

To use a package from this index either "[pip install package](#)" ([get pip](#)) or download, unpack and "[python setup.py install](#)" it.

### Package Authors

Submit packages with "[python setup.py upload](#)". The index [hosts package docs](#). You may also use the [web form](#). You must [register](#). Testing? Use [testpypi](#).

## Not Logged In

- [Login](#)
- [Register](#)
- [Lost Login?](#)
- Use [OpenID](#) 
- [Login with Google](#) 

## Status

[Nothing to report](#)

# Package

\$ pip search *package\_name*

\$ pip install *package\_name*

\$ pip install --upgrade *package\_name*

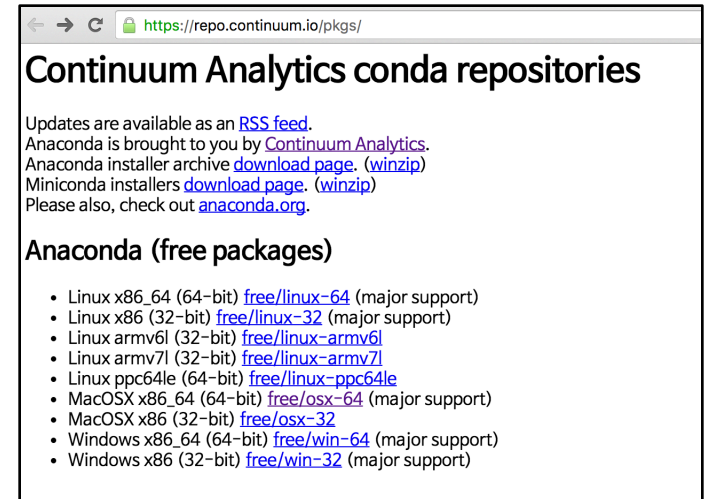
\$ pip uninstall *package\_name*

\$ conda search *package\_name*

\$ conda install *package\_name*

\$ conda update *package\_name*

\$ conda uninstall *package\_name*





# Hello, World!

## Demo

script, interactive

Type

# bool

```
<True, False>
```

```
$ python ↵
```

```
>>> a = bool
```

```
>>> type(a)
```

```
<class 'bool'>
```

```
>>> True.__class__
```

```
<class 'bool'>
```

변수 이름은 대소문자(a~z, A~Z), 숫자, 언더스코어로 구성할 수 있으나 숫자로 시작할 수 없습니다.

# number

```
$ python ↵
```

```
>>> a = 1
```

```
>>> type(a)
```

```
<class 'int'>
```

```
>>> a.__class__
```

```
<class 'int'>
```

```
>>> 3.5.__class__
```

```
<class 'float'>
```

+ 덧셈

- 뺄셈

\* 곱셈

/ 나눗셈

// 몫

% 나머지

\*\* 제곱

+=, -=, \*=, /=,

//=, %=, \*\*=

a += 1 , a -= 1

# number

- 2진수는 숫자 앞에 '0b', '0B' 를 붙입니다.
- 8진수는 숫자 앞에 '0o', '0O' 를 붙입니다.
- 16진수는 숫자 앞에 '0x', '0X' 를 붙입니다.
- int 와 float 간의 형변환은 int() 함수와 float() 함수를 사용합니다.
- 숫자의 크기에 제한이 없습니다.

# number

## Demo

## jupyter notebook

```
$ ipython notebook ↵
```

[github link](#)

# string

- 홑따옴표나 쌍따옴표를 이용하여 문자열을 만들 수 있습니다.  
e.g. 'Korea', "It's great!"
- 세개의 따옴표를 연속으로 사용하면 여러줄을 쓸 수 있습니다.  
e.g. `"""Python is  
a programming language"""`
- string 클래스는 +, \* 연산자를 지원하고 다양한 메소드를 가지고 있습니다.
- string은 배열처럼 인덱스 번호로 문자열의 위치를 참조할 수 있습니다.  
e.g. `some_string[9]`

# string

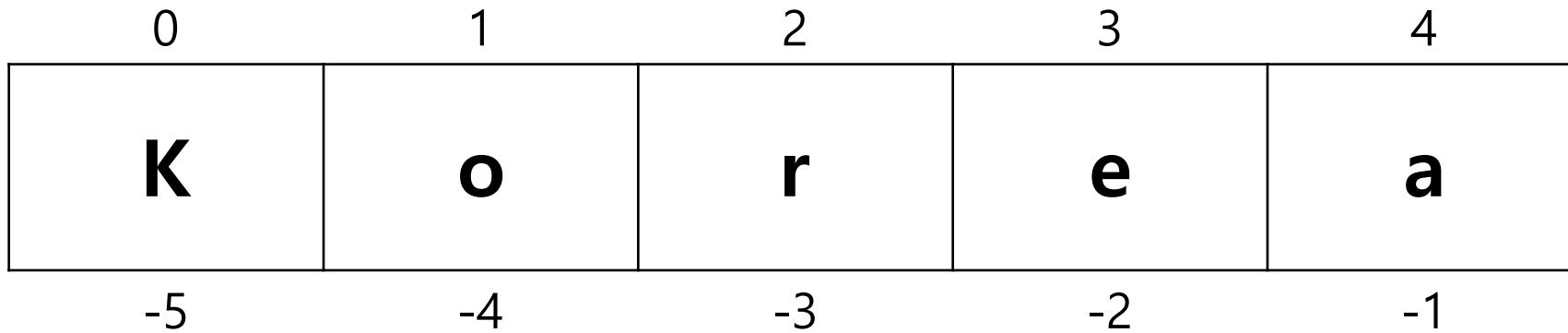
```
s = 'Korea'
```

```
s[0]    # K
```

```
s[4]    # a
```

```
s[-2]   # e
```

```
s[-5]   # K
```





# string

```
$ ipython ↵
```

```
] : s = 'a'
```

```
] : s + s
```

```
'aa'
```

```
] : s * 2
```

```
'aa'
```

```
] : s.\t
```

```
s.capitalize
```

```
s.endswith
```

```
s.index
```

```
s.isidentifier
```

```
s.istitle
```

```
s.lstrip ...
```

```
] : s.find?
```

```
...
```

string

Demo

jupyter notebook

# list

- 딕셔너리(dictionary)와 함께 가장 많이 쓰이는 데이터 타입입니다.
- 튜플(tuple)은 수정이 불가능한 리스트(list) 입니다.
- 리스트는 대괄호를 이용하여 생성하거나 list() 함수를 이용합니다.

e.g. `lst = [], lst = list()`

- 리스트의 요소는 배열 인덱스로 참조할 수 있습니다.

e.g. `lst = [0, 1, 2]`  
`print(lst[1])`    # 1

- 이종의 데이터 타입을 포함할 수 있습니다.

e.g. `lst = [0, 1, 'Korea', 3, 4]`

- 다차원 배열을 표현할 수 있습니다.

e.g. `lst = [ [0, 1, 2], [3, 4, 5] ]`

list

Demo

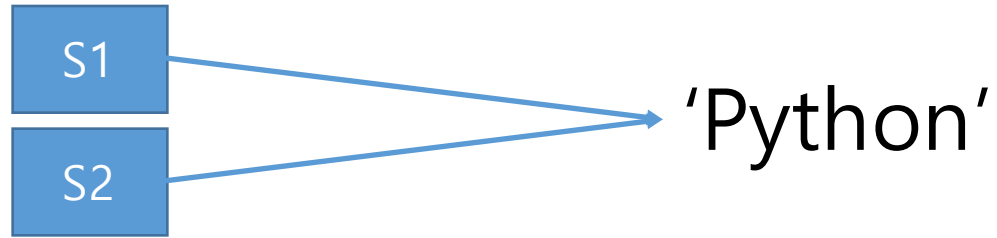
jupyter notebook

# Object

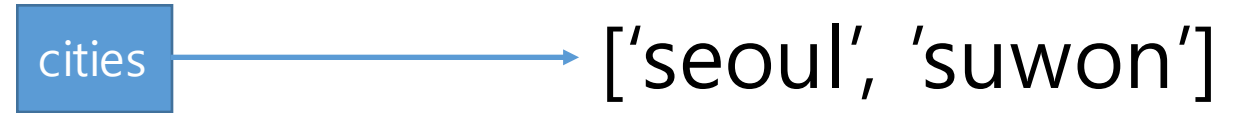
```
s1 = 'Python'
```



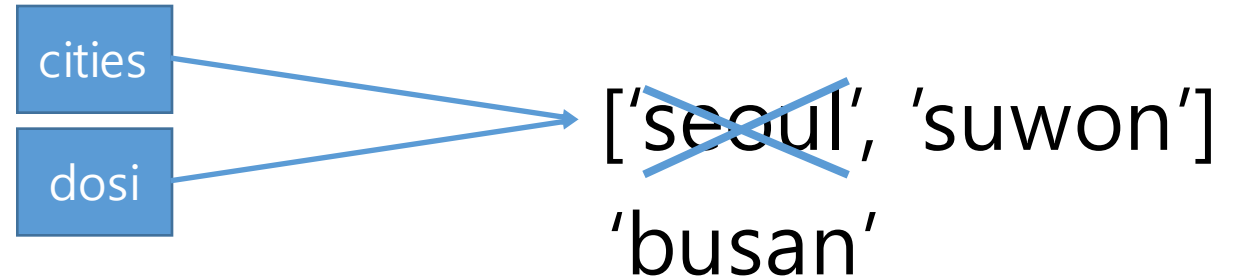
```
s2 = s1
```



```
cities = ['seoul', 'suwon']
```



```
dosi = cities
```



```
cities[0] = 'busan'
```

A blue arrow points from the text 'busan' to the first element of the list in the previous diagram.

```
print(dosi[0]) # busan
```

# Object

## Demo

### jupyter notebook

# tuple

- 튜플(tuple)은 생성이된 후 수정이 불가능합니다.
- 튜플은 소괄호를 이용하여 생성하거나 tuple() 함수를 이용합니다.

e.g. `tuple = (0, 1, 2), tuple = tuple((0, 1, 2))`

- 튜플의 요소는 배열 인덱스로 참조할 수 있습니다.

e.g. `tuple = (0, 1, 2)`  
`print(tuple[1]) # 1`

- 이종의 데이터 타입을 포함할 수 있습니다.

e.g. `tuple = (0, 1, 'Korea', 3, 4)`

- 다차원 배열을 표현할 수 있습니다.

e.g. `tuple = ( (0, 1, 2), (3, 4, 5) )`  
`tuple[0][1] # 1`

tuple

Demo

jupyter notebook



# dictionary

- {key: value} 구조를 가지는 데이터 타입으로 다른 언어에서는 연관배열 (associative array) 또는 해시(hash)라고도 부릅니다.

- 딕셔너리는 중괄호를 이용하여 생성하거나 dict() 함수를 이용합니다.

e.g. `dct = {}`, `dct = dict()`

- 딕셔너리의 요소는 키 값으로 참조할 수 있습니다.

e.g. `dct = {'age': 20, 'height': 170}`  
`print(dct['age'])`    # 20

- 이종의 데이터 타입을 포함할 수 있습니다.

e.g. `dct = {'age': 20, 'height': 170, 2015: 'Paris'}`

- 다차원 구조를 표현할 수 있습니다.

e.g. `dct = {'age': 20, 2015: ['Paris', 'Tokyo', 'NYC']}`  
`print(dct[2015][2])`    # NYC

dictionary

Demo

jupyter notebook

# set

- value가 없는 딕셔너리와 같습니다.
- 셋은 중괄호를 이용하여 생성하거나 `set()` 함수를 이용합니다.  
e.g. `st = {}`, `st = set()`
- 딕셔너리와 마찬가지로 셋안의 키는 고유해야 합니다.  
e.g. `st = {'red', 'blue', 'yellow'}`
- 이종의 데이터 타입을 포함할 수 있습니다.  
e.g. `st = {'red', 'blue', 'yellow', 2016}`
- 딕셔너리와 마찬가지로 순서가 없으며 키 값으로 스트링, 정수, 실수, 불리언값이 사용됩니다.
- 다양한 교집합, 합집합 연산자가 제공됩니다.

set

Demo

jupyter notebook

Flow

# comment

- # 으로 시작하는 뒷 부분은 모두 주석이 됩니다.

e.g.     a = 1   # 주석입니다.

- 홀따옴표나 겹따옴표 세개로 감싸진 문단은 주석이 됩니다.

e.g.     """

프로그램 설명이나 API 설명 등  
여러줄의 주석을 적을때 사용합니다.

"""

# if

- 비교문에 괄호는 선택사항입니다.
- if, elif, else 절의 구분은 ':' 과 들여쓰기로 합니다.

```
e.g.  if a > 1:
        print('ok')
      elif a < 1:
        print('no')
      else:
        print('equal')
```

- False 인 것들: None, 0, 0.0, '', [], (), {}, set(), False
- ==, !=, <, <=, >, >=, not, is, in, and, or
- == 은 값을 비교 is 는 오브젝트를 비교

# for

- for A in B : 구문을 사용하여 반복문을 만듭니다.
- B 에는 순회 가능한(iterable) 객체가 들어가며 A 는 요소를 하나씩 전달 받습니다.
- 루프안에서 빠져나오려면 break 명령을 쓰고 루프의 처음으로 돌아가려면 continue 명령을 사용합니다.
- for 루프의 구분은 ':' 과 들여쓰기로 합니다.  
e.g.    `for i in range(10):`  
          `print(i)      # 0~9`
- 순회 가능한 객체: 스트링, 리스트, 튜플, 딕셔너리, 셋



# while

- while A : 구문을 사용하여 반복문을 만듭니다.
- A 에는 비교문이 들어가며 참일 경우 계속 반복을 실행합니다.
- 루프안에서 빠져나오려면 break 명령을 쓰고 루프의 처음으로 돌아가려면 continue 명령을 사용합니다.
- while 루프의 구분은 ':' 과 들여쓰기로 합니다.

```
e.g.  while True:
        if a > 1:
            break
        elif a < 1:
            continue
        print('ok')
```

if, for, while

Demo

jupyter notebook

# list comprehension

- 리스트 컴프리헨션은 리스트나 딕셔너리 생성할 때 사용할 수 있는 for 반복문을 사용한 간결한 표현식입니다.
- 리스트, 딕셔너리, 셋을 만들 수 있습니다.

```
e.g.  num_list = list(range(10))    # 0~9
      num_list = [i for i in range(10)]
      num_dict = {i: chr(i) for i in range(65, 68)}
                      # {65: 'A', 66: 'B', 67: 'C'}
      num_set = {chr(i) for i in range(65, 68)}
                      # {'A', 'B', 'C'}
```

- 많은 중첩은 좋지 않습니다.

# list comprehension

Demo

jupyter notebook

# function

- `def <func name> (<params>):`  
    `<func body>`
- `return` 문으로 결과값을 함수 호출자에게 돌려 줄 수 있습니다.  
e.g. 

```
def test_func(num):  
    return num+1
```
- 사용하기 전에 먼저 정의되어야 합니다.
- 함수를 호출할 때 파라메타 이름을 사용할 수 있습니다.  
e.g. 

```
test_func(10)    # 11  
test_func(num=10) # 11
```
- 파라메타의 기본 값을 지정할 수 있습니다.  
e.g. 

```
def test_func(num = 0):  
    return num+1
```

function

Demo

jupyter notebook

# class

- `class <class name> (<parent>):`  
    <class body>
- 초기화는 `__init__` 메소드에서 수행 합니다.  
e.g.     `class Person(object):`  
          `def __init__(self):`  
          `pass`
- 클래스 안에는 여러개의 메소드를 포함할 수 있습니다.
- `self`는 자기 자신을 가리키는 오브젝트입니다.
- `super`는 부모 클래스를 가리키는 오브젝트입니다.
- 파이썬은 다중 상속(multiple inheritance)을 지원합니다.
- Private 함수일 경우 함수 이름을 `__`로 만들지만 외부에서 접근됩니다.

class

Demo

jupyter notebook



# module

- 모듈은 파이썬 코드가 있는 파일입니다.
- import 명령을 이용해 다른 파일을 포함시킬 수 있습니다.

e.g.     `import random`  
          `random.random()`     # 0.19982493613894659

- 필요한 오브젝트만, 혹은 전체를 임포트할 수 있습니다.

e.g.     `from random import random`  
          `random()`     # 0.19982493613894659  
          `from random import *`

- as 를 사용하여 임포트 오브젝트를 리네임할 수 있습니다.

e.g.     `from random import random as rand`  
          `rand()`     # 0.19982493613894659

# exception

- try: ~ except: ~ else: ~ finally: 블록을 사용합니다.

e.g.     try:  
            1/0  
          except ZeroDivisionError as e:  
              print(e)  
          else:  
              print('no error')  
          finally:  
              print('try end')

- raise 명령을 사용해서 익셉션을 발생시킬 수 있습니다.

e.g.     raise NameError('Error occurred')

exception

Demo

jupyter notebook

# IDE & debug

Demo

PyCharm CE

misc.

# Naming

<https://www.python.org/dev/peps/pep-0008/>

module	<code>all_lower_case</code>
function	<code>all_lower_case</code>
class	<code>CamelCase</code>
method	<code>all_lower_case</code>
variable	<code>all_lower_case</code>
constant	<code>ALL_UPPER_CASE</code>

# Python 2 vs 3

Python 2	Python 3
<pre>print 'Hello, World!' print('Hello, World!') # syntax not func print('a', 'b') → ('a', 'b')</pre>	<pre>SyntaxError print('Hello, World!') print('a', 'b') → a b</pre>
<pre>3 / 2 → 1 3 // 2 → 1 3 / 2.0 → 1.5 3 // 2.0 → 1.0</pre>	<pre>3 / 2 → 1.5 3 // 2 → 1 3 / 2.0 → 1.5 3 // 2.0 → 1.0</pre>
<pre>print '한글' # SyntaxError # coding:utf-8 'test' # str b'test' # str</pre>	<pre>print('한글')  'test' # str b'test' # bytes</pre>
<pre>range(3) # [0, 1, 3] xrange(3) # xrange type, sequence generation</pre>	<pre>range(3) # similar xrange of python2</pre>

# Python 2 vs 3

Python 2	Python 3
<pre>raise IOError 'file error' raise IOError('file error')</pre>	<pre>SyntaxError raise IOError('file error')</pre>
<pre>try:     pass except NameError, err: # NameError as err     pass</pre>	<pre>try:     pass except NameError as err:     pass</pre>
<pre>round(2.5)    # 3 round(3.5)    # 4 round(4.5)    # 5 # Round Half up</pre>	<pre>round(2.5)    # 2 round(3.5)    # 4 round(4.5)    # 4 # Round Half to Even, Gaussian rounding  from decimal import * print(Decimal(2.5).quantize(Decimal('1.'), rounding=ROUND_HALF_UP))    # 3</pre>



<http://pycon.kr/2016apac/>

8월 13일(토) ~ 15일(월)

강남구 코엑스

# Q&A

Any Question: [haesunrpark@gmail.com](mailto:haesunrpark@gmail.com)

github URL: <https://github.com/rickiepark/python-tutorial>

This slide available at  
<http://tensorflowkorea.wordpress.com>