Welcome to Python
2015 Summer Data Mining Workshop

August 20, 2015

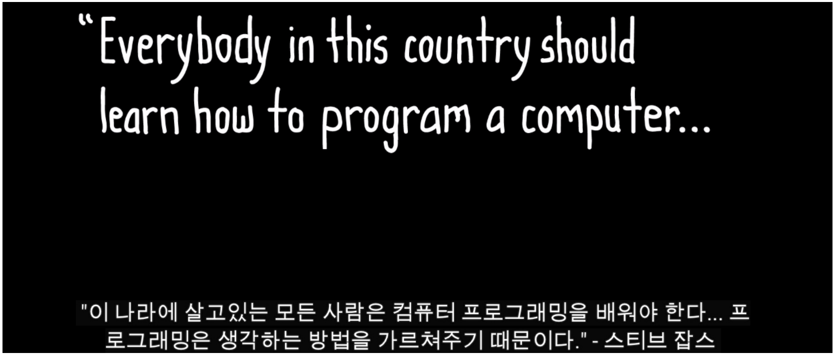
Kyunghoon Kim
`kyunghoon@unist.ac.kr`

Department of Mathematical Sciences
Ulsan National Institute of Science and Technology
Republic of Korea

https://xkcd.com/353/

Contents

왜 프로그래밍을 배워야 하나?



"Everybody in this country should learn how to program a computer...

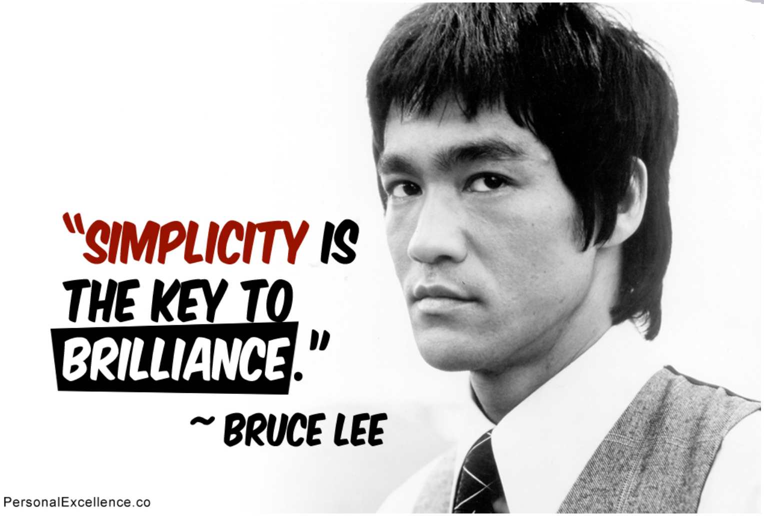
"이 나라에 살고있는 모든 사람은 컴퓨터 프로그래밍을 배워야 한다... 프로그래밍은 생각하는 방법을 가르쳐주기 때문이다." - 스티브 잡스

https://www.youtube.com/watch?v=nKIu9yen5nc

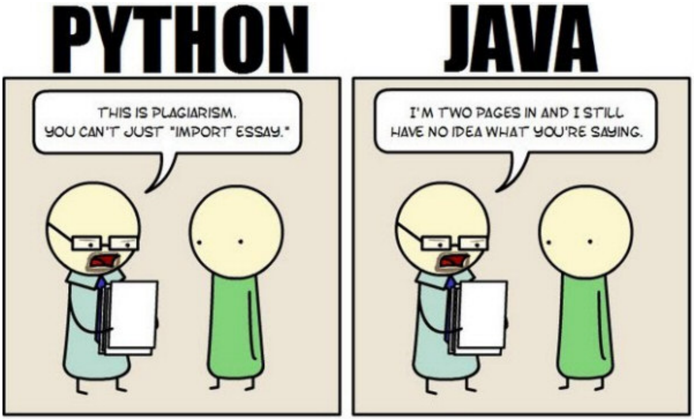
왜 파이썬을 사용하나?

http://www.collegeteacher.org/csci101/resource_programming/prog03.php

왜 파이썬을 사용하나?

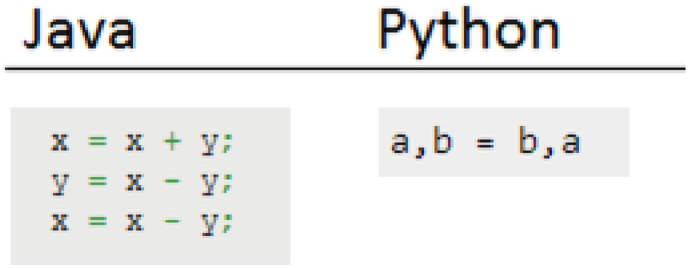

"SIMPLICITY IS THE KEY TO BRILLIANCE."
~ BRUCE LEE
PersonalExcellence.co

왜 파이썬을 사용하나?


PYTHON
THIS IS PLAGIARISM. YOU CAN'T JUST "IMPORT ESSAY."
JAVA
I'M TWO PAGES IN AND I STILL HAVE NO IDEA WHAT YOU'RE SAYING.

http://www.quora.com/Why-should-I-learn-Python-if-I-already-know-Java

왜 파이썬을 사용하나?



Java

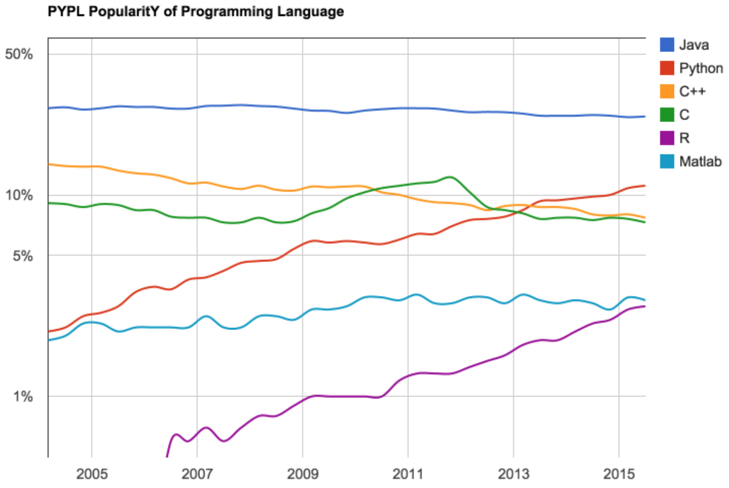
```java
x = x + y;
y = x - y;
x = x - y;
```

Python

```python
a,b = b,a
```

http://www.quora.com/Why-should-I-learn-Python-if-I-already-know-Java

왜 파이썬을 사용하나?



http://pypl.github.io/PYPL.html

왜 파이썬을 사용하나?

Python is a programming language that lets you work quickly and integrate systems more effectively.

`https://www.python.org/`

어떻게 파이썬을 배우나?

- English : `http://code.tutsplus.com/articles/the-best-way-to-learn-python--net-26288`
- Korean : `https://nolboo.github.io/blog/2014/08/10/the-best-way-to-learn-python/`

어떻게 파이썬을 배우나?

Web Tutorial(links)

- Codecademy python-ko
- Python Visualization
- ipython Tutorial
- Learn Python Online

어떻게 파이썬을 배우나?

Slide(links)

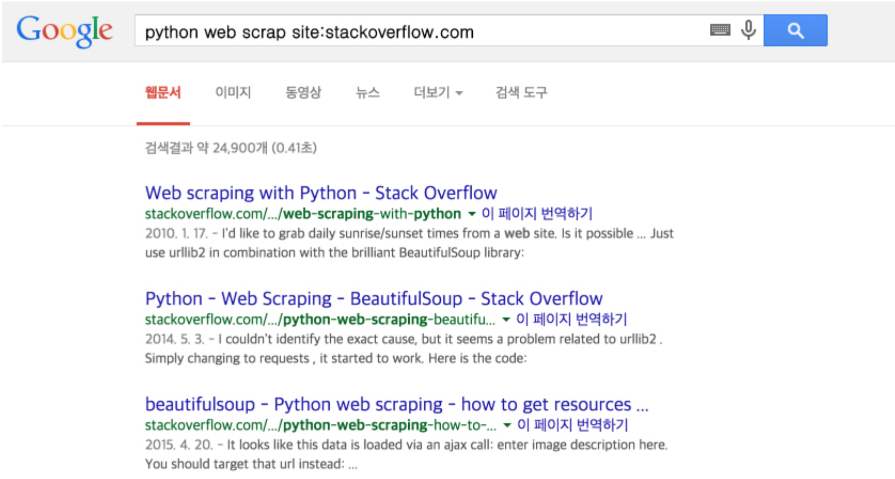
- Learn 90% of Python in 90 Minutes
- 산업공학과를 위한 프로그래밍 입문

어떻게 파이썬을 배우나?

Online Book(links)

- 점프 투 파이썬
- Dive into Python(번역본)
- 컴퓨터 과학자 같이 생각하는 법(파이썬 버전)
- Problem Solving with Algorithms and Data Structures
- 파이썬 문서고

어떻게 파이썬을 배우나?



검색결과 약 24,900개 (0.41초)

Web scraping with Python - Stack Overflow
stackoverflow.com/.../web-scraping-with-python ▾ 이 페이지 번역하기
2010. 1. 17. - I'd like to grab daily sunrise/sunset times from a web site. Is it possible ... Just use urllib2 in combination with the brilliant BeautifulSoup library:

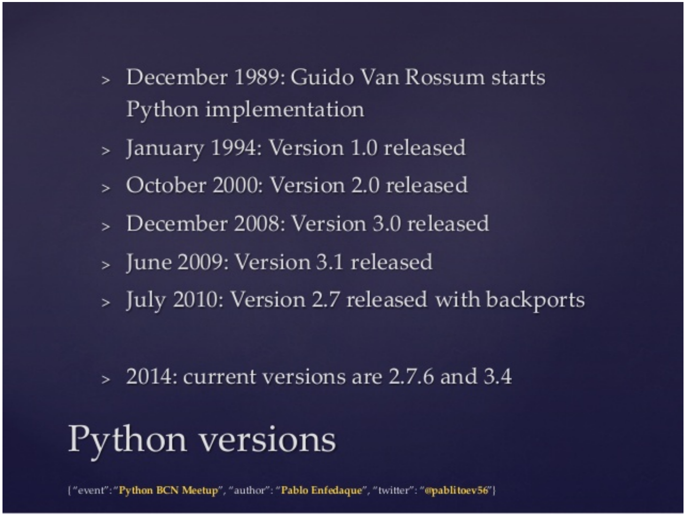
Python - Web Scraping - BeautifulSoup - Stack Overflow
stackoverflow.com/.../python-web-scraping-beautifu... ▾ 이 페이지 번역하기
2014. 5. 3. - I couldn't identify the exact cause, but it seems a problem related to urllib2 . Simply changing to requests , it started to work. Here is the code:

beautifulsoup - Python web scraping - how to get resources ...
stackoverflow.com/.../python-web-scraping-how-to-... ▾ 이 페이지 번역하기
2015. 4. 20. - It looks like this data is loaded via an ajax call: enter image description here. You should target that url instead: ...

파이썬2냐 파이썬3냐

http://b.ssut.me/64

파이썬2냐 파이썬3냐



- December 1989: Guido Van Rossum starts Python implementation
- January 1994: Version 1.0 released
- October 2000: Version 2.0 released
- December 2008: Version 3.0 released
- June 2009: Version 3.1 released
- July 2010: Version 2.7 released with backports

- 2014: current versions are 2.7.6 and 3.4

Python versions

{"event":"Python BCN Meetup", "author": "Pablo Enfedaque", "twitter": "@pablitoev56"}

http://www.slideshare.net/pablito56/python-2-vs-python-3

파이썬 코드와 실행

hello.py

```
print("hello world")
```

Command prompt

```
$ python hello.py
```

객체 Object

Everything in Python is an **object**

- identity(id)
- value
  - mutable
    - list
    - dictionary
  - immutable
    - string
    - integer
    - tuple

help(), dir()

정의된 객체에 관한 문서를 얻을 때 사용하는 명령어.

1. help()

객체에 대해 뭔가 알고 싶다

```
>>> help(dir)
Help on built-in function dir in module __builtin__:

dir(...)
    dir([object]) -> list of strings
```

2. dir()

객체의 속성에 관한 목록을 얻고 싶다

```
>>> dir(help)
[‘__call__’, ‘__class__’, ‘__delattr__’, ‘__dict__’, ...]
```

변수의 자료형

변수는 자료형이 없어서 선언할 필요가 없다.

```
>>> a = 3 # integer
>>> b = 3.14 # float
>>> c = "string" # string
>>> d = ‘string’
>>> e = "3"
```

```
>>> type(a)
int
>>> type(b)
float
>>> type(c)
str
```

정수의 자료형

고정 소수점 Fixed point 방식

int : 마이크로프로세서의 기본 비트의 길이

```
>>> import sys
>>> print(sys.maxint)
9223372036854775807
```

long : 정수의 범위를 넘어서는 큰 숫자(only python2)

```
>>> print(sys.maxint+1)
9223372036854775808L
```

실수의 자료형

부동 소수점 Floating point 방식
: 소수점의 위치를 고정하지 않고 그 위치를 나타내는 수(exponent)를 따로 적음

$$[mantissa] * [base]^{[exponent]}$$

정밀도 precision 문제 발생

```
>>> (1234.567+0.001)+0.0004
1234.5683999999999
```

```
>>> 1234.567+(0.001+0.0004)
1234.5684
```

정밀도 설정

decimal 객체를 생성하여 10진수를 정확하게 나타낼 수 있음.

```
>>> from decimal import Decimal, getcontext # Module import
>>> getcontext().prec = 12 # precision
>>> Decimal(1234.567)+Decimal(0.001)+Decimal(0.0004)
Decimal('1234.56840000')
>>> getcontext().prec = 24
>>> (Decimal(1234.567)+Decimal(0.001))+Decimal(0.0004)
Decimal('1234.56840000000000727600')
>>> Decimal(1234.567)+(Decimal(0.001)+Decimal(0.0004))
Decimal('1234.56840000000000727600')
>>> Decimal(10)**600
Decimal('1.00000000000000000000000E+600')
```

하지만.. 긴 시간의 연산에서는 느리다.

정수의 나눗셈

```
>>> 3/4
0
>>> 3/4.
0.75
```

Python3에서는 문제가 없다.

복소수

실수, 허수 부분은 64비트 부동 소수점 숫자로 저장됨.

```
>>> a = 1-2j
>>> a.real
1.0
>>> a.imag
-2.0
>>> abs(a)
2.23606797749979
>>> a
(1-2j)
```

문자열

```
>>> a = ‘apple’
>>> b = "apple"
>>> c = """apple"""
>>> print a, b, c
apple apple apple
```

String Escaping

```
>>> print "I’m happy"
I’m happy
>>> print ‘I\’m happy’
I’m happy
>>> print """\"I’m happy\""""
"I’m happy"
```

문자열 포맷

```
>>> a = "apple"
>>> b = "banana"
>>> "%s and %s" % (a, b)
‘apple and banana’
>>> "0 and 1".format(a, b)
‘apple and banana’
>>> print a, "and", b
apple and banana
```

문자열 방법

Dunder(Double under) Methods and String Methods

```
>>> dir("apple")
['__add__', '__class__', '__contains__', '__delattr__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
...
'capitalize', 'center', 'count', 'decode', 'encode',
'endswith', 'expandtabs', 'find', 'format', 'index',
'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'partition', 'replace', 'rfind', 'rindex', 'rjust',
'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

주석 comment

```
# 한 줄 주석

"""
여러 줄 주석
"""
```

그 외 자료형들

- None
- Booleans
- Sequences
  - list
  - tuple
  - set
- Dictionary

리스트 list

```
>>> a = ['apple', 'banana', 'kiwi']
>>> a[0]
'apple'
>>> a[2]
'kiwi'
>>> a[-1] # a[len(a)-1]
'kiwi'
>>> len(a)
3
```

리스트 list

```
>>> dir([])
[..., 'append', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']

>>> a.append('melon')
>>> a
['apple', 'banana', 'kiwi', 'melon']
>>> a.index('kiwi')
2
>>> a.remove('banana')
>>> a
['apple', 'kiwi', 'melon']
>>> a.pop(2)
'melon'
>>> a
['apple', 'kiwi']
```

리스트 list

```
>>> range(10) # half-open interval
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(3, 10) # length = end - start
[3, 4, 5, 6, 7, 8, 9]
>>> range(1, 10, 2)
[1, 3, 5, 7, 9]
>>> a = range(10)
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> a[3:5]
[3, 4]
>>> a[::-1]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

사전 dictionary

Hashmap, Associative array

```
>>> a = {} # a = dict()
>>> a['apple'] = 500
>>> a['banana'] = 200
>>> a
{'apple': 500, 'banana': 200}
```

```
>>> dir({})
[...,'clear', 'copy', 'fromkeys', 'get', 'has_key', 'items',
'iteritems', 'iterkeys', 'itervalues', 'keys', 'pop',
'popitem', 'setdefault', 'update', 'values', 'viewitems',
'viewkeys', 'viewvalues']
```

사전 dictionary

```
>>> apple = dict()
>>> apple = {"price": 500, "type": ["fruit", "red"],
    "property": {"a": 1}}
>>> apple['price']
500
>>> apple['type']
['fruit', 'red']
>>> apple['type'][0]
'fruit'
>>> apple['property']
{'a': 1}
>>> apple['property']['a']
1
>>> apple.get("price", None)
500
>>> apple.get("where", None)
>>>
```

그 외 유용한 자료형들

```
>>> values = [1,1,1,2,3,3,4,5,6,7,7]
>>> set(values) # 집합 set
set([1, 2, 3, 4, 5, 6, 7])
>>> list(set(values))
[1, 2, 3, 4, 5, 6, 7]
```

```
>>> a = (11, 23, "Monday") # 튜플 tuple
>>> (mon, day, text) = a
>>> print mon, day, text
11 23 Monday
```

함수 Function

$$f(x) = 4x * (1 - x)$$

```
>>> def f(x):
        return 4*x*(1-x)
>>> f(0.2)
0.6400000000000001
```

함수 Function

```
>>> from decimal import Decimal, getcontext
>>> getcontext().prec = 24
>>> def f(a, x):
        """ Logistic Function """
        x = Decimal(x)
        return a*x*(1-x)
>>> f(4, 0.2)
Decimal('0.640000000000000026645353')

>>> help(f)
Help on function f in module __main__:

f(a, x)
    Logistic Function
```

반복문 Loop

```
>>> for i in ['apple', 'banana', 'kiwi']:
...     print i
...
apple
banana
kiwi

>>> for i in range(0, 10, 2):
...     print i
...
0
2
4
6
8
```

반복문 Loop

```
>>> x = 0.1
>>> for i in range(10):
        print x
        x = f(4, x) # logistic function
    print x
0.1
0.36000000000000017763568
0.92160000000000019895195
0.28901375999999932897486
0.82193922612264948673834O
0.58542053873419824930112 1
0.97081332624943734621213 1
0.11333924730376348296114 9
0.40197384929751930007619 5
0.96156349511381817032339 6
0.14783655991326540003640 6
```

조건문 Conditional statement

```
>>> x = 0.1
>>> for i in range(100):
        print x
        x = f(2, x)
    print x
0.1
0.18000000000000008881784
0.29520000000000011368684
0.41611392000000009313226
0.485926251164467203125000
0.499603859187428678487954
0.499999686144913230666241
0.499999999999802989969018
0.500000000000000000000000
0.500000000000000000000000
0.500000000000000000000000
0.500000000000000000000000
```

조건문 Conditional statement

```
>>> x = Decimal(0.1)
>>> for i in range(100):
        print x
        new_x = f(2, x)
        if new_x - x < 0.0000001:
            print "Converged"
            break
        else:
            x = new_x
    print x
0.1000000000000000055511151231257827021181583404541015625
0.1800000000000000008881784
0.2952000000000000011368684
0.4161139200000000009313226
0.4859262511644672031250000
0.4996038591874286784879540
0.4999996861449132306662410
0.4999999999998029899690180 Converged
```

조건문 Conditional statement

```
>>> A = 85
>>> if A>90:
...     print "A"
... elif A>80:
...     print "B"
... elif A>70:
...     print "C"
... else:
...     print "D"
...
B
```

반복문과 조건문 예

```
>>> animals = ['cat', 'dog', 'cock', 'rabbit']
>>> for index, value in enumerate(animals):
...     print index, value
...
0 cat
1 dog
2 cock
3 rabbit

>>> for index, value in enumerate(animals):
...     if value[0] == 'c':
...         print index, value
...
0 cat
2 cock
```

클래스 Class

- object
- constructor(dunder init)
- All method take **self** as first parameter.

```
>>> class Animal(object):
...     def __init__(self, name):
...         self.name = name
...     def talk(self):
...         print "Hello"
...
>>> animal = Animal('thing')
>>> animal.talk()
Hello
```

서브클래스 Subclass

```
>>> class Cat(Animal):
...     def talk(self):
...         print "%s is cat name." % (self.name)
...
>>> cat = Cat("Persia")
>>> cat.talk()
Persia is cat name.
```

Thank you for attention!

In case you have any comments, suggestions or have found a bug, please do not hesitate to contact me.

Kyunghoon Kim
`kyunghoon@unist.ac.kr`
Graduate Student in
Computational Mathematical Sciences Laboratory
`http://amath.unist.ac.kr`

https://xkcd.com/353/