

# 바로 하는 파이썬 세미나

강성훈

<http://j.mp/2010sparcpython>

시작하기 전에:

왜?

- C
- C++
- 자바

- C
- C++
- 자바
- 파이썬?

# 스크립팅 언어



빠르게 짜고  
빠르게 확인하고  
빠르게 고친다

```
print "Hello, world!"
```

<http://python.org/>

[ABOUT >>](#)[NEWS >>](#)[DOCUMENTATION >>](#)[DOWNLOAD >>](#)[COMMUNITY >>](#)[FOUNDATION >>](#)[CORE DEVELOPMENT >>](#)[LINKS >>](#)

### Help

#### Quick Links (2.6.4)

- » [Documentation](#)
- » [Windows Installer](#)
- » [Source Distribution](#)
- » [Package Index](#)

## Python Programming Language -- Official Website

**Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs.**

Python runs on Windows, Linux/Unix, Mac OS X, and has been ported to the Java and .NET virtual machines.

Python is free to use, even for commercial products, because of its OSI-approved [open source license](#).

The [Python Software Foundation](#) holds the intellectual property rights behind Python, underwrites the [PyCon conference](#), and funds other projects in the Python community.

[Read more](#), -or- [download Python now](#)

[EuroPython 2010 registration and talk submissions](#)



... join  
Rack  
Magic  
and r

# 계산기

>>> 3 + 4 \* (5 - 6)

-1

>>> 8/5

1

>>> 8/5

1

>>> 8%5

3





>>> 8.0 / 5

1.600000000000000000000001

>>> 8 / 5.0

1.600000000000000000000001

5 → 5.0 (0)

5 ← 5.0 (X)

# 자료형

```
>>> "foo" + 'bar'  
'foobar'
```

```
>>> "foo" * 4  
'foofoofoofoo'
```

```
>>> 1 + 1 == 2
```

True

```
>>> 1 + 1 == 3
```

False

- 참과 거짓: True, False
- 정수: 1, 2, -3, -4, ...
- 실수: 1.0, -2.34, ...
- 문자열: 'x', "Hello", ...
- 기타 등등



```
>>> 16000000 + (3.0 - 2.05) * 63000000  
75850000.0000000000009
```

내가 니 학점을  
어찌 아니?

```
>>> grade = 2.05
```

```
>>> 16000000 + (3.0 - grade) * 6300000
```

```
7585000.000000000009
```

```
>>> grade = 2.05
```

```
>>> 16000000 + (3.0 - grade) * 63000000  
75850000.0000000000009
```

```
>>> grade = 2.89
```

```
>>> 16000000 + (3.0 - grade) * 63000000  
22929999.999999999991
```

변수

```
>>> grade = 3.5
```

```
>>> 16000000 + (3.0 - grade) * 63000000
```

```
-15500000.0
```

```
>>> grade = 3.5
>>> if grade < 3.0:
...     16000000 + (3.0 - grade) * 63000000
... else:
...     16000000
...
16000000
```

```
>>> grade = 1.86
>>> if grade < 3.0:
...     16000000 + (3.0 - grade) * 63000000
... else:
...     16000000
...
8782000.0
```



```
>>> grade = 1.86
>>> if grade < 2.0:
...     16000000 + (3.0 - 2.0) * 63000000
... elif grade < 3.0:
...     16000000 + (3.0 - grade) * 63000000
... else:
...     16000000
...
79000000.0
```

```
>>> grade = 1.86
>>> if grade < 3.0:
...     if grade < 2.0:
...         1600000 + (3.0-2.0) * 6300000
...     else:
...         1600000 + (3.0-grade) * 6300000
... else:
...     1600000
...
7900000.0
```

```
>>> for i in range(13):  
...     print i+1, '인의 아해가 거리를 질주하오.'  
...  
1 인의 아해가 거리를 질주하오.  
2 인의 아해가 거리를 질주하오.  
(생략)  
13 인의 아해가 거리를 질주하오.
```

```
for i in range(1, 101):  
    one = (i%10==3 or i%10==6 or i%10==9)  
    ten = (i/10==3 or i/10==6 or i/10==9)  
    if one and ten:  
        print '짝짝'  
    elif one or ten:  
        print '짝'  
    else:  
        print i
```

너무 복잡하네

# Refactoring?

```
# x가 3, 6, 9이면 True 아니면 False  
def is369(x):  
    return x==3 or x==6 or x==9
```

```
def is369(x):  
    return x==3 or x==6 or x==9  
  
for i in range(1,101):  
    one = is369(i%10)  
    ten = is369(i/10)  
    if one and ten: print ' 짹짹 '  
    elif one or ten: print ' 짹 '  
    else: print i
```



```
def is369(x):  
    return x==2 or x==3 or x==5 or x==7  
  
for i in range(1,101):  
    one = is369(i%10)  
    ten = is369(i/10)  
    if one and ten: print ' 짹짹 '  
    elif one or ten: print ' 짹 '  
    else: print i
```

```
def needsclap(x):  
    return x==2 or x==3 or x==5 or x==7  
  
for i in range(1,101):  
    one = needsclap(i%10)  
    ten = needsclap(i/10)  
    if one and ten: print '짝짝'  
    elif one or ten: print '짝'  
    else: print i
```

“함수”

단순히 코드를 묶는 것이 아닌

range도 함수인가요?

```
>>> range(10)
```

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

```
>>> [1, 2, 3]
```

```
[1, 2, 3]
```

```
>>> [4, 5] + ['hello?']
```

```
[4, 5, 'hello?']
```

```
>>> [41, 42] * 3
```

```
[41, 42, 41, 42, 41, 42]
```

```
>>> []
```

```
[]
```

```
>>> (1, 2, 3)
```

```
(1, 2, 3)
```

```
>>> (4, 5) + ('hello?',)
```

```
(4, 5, 'hello?')
```

```
>>> (41, 42) * 3
```

```
(41, 42, 41, 42, 41, 42)
```

```
>>> ()
```

```
()
```

```
>>> a = [1,2,3]
```

```
>>> a[0] + a[1] + a[2]
```

```
6
```

```
>>> a[1] = 5
```

```
>>> a
```

```
[1, 5, 3]
```



**a**



a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8]  
a[-9] a[-8] a[-7] a[-6] a[-5] a[-4] a[-3] a[-2] a[-1]

```
>>> b = (1,2,3)
```

```
>>> b[0] + b[1] + b[2]
```

```
6
```

```
>>> b[1] = 5
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: 'tuple' object does not  
support item assignment
```

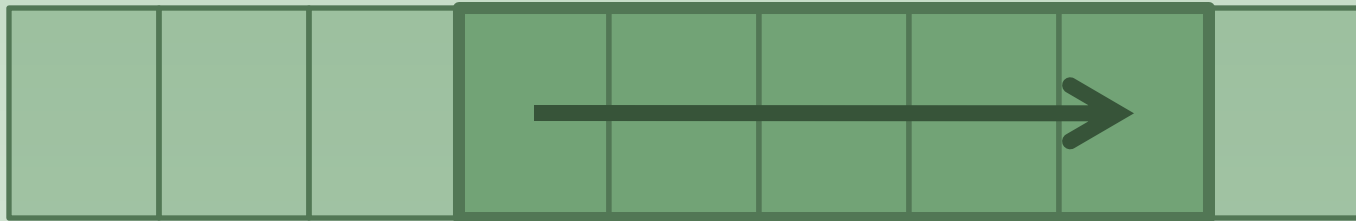
```
>>> b = (1,2,3)
```

```
>>> b = b[:1] + (5,) + b[2:]
```

```
>>> b
```

```
(1, 5, 3)
```

**a[3:8]**



a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8]  
a[-9] a[-8] a[-7] a[-6] a[-5] a[-4] a[-3] a[-2] a[-1]

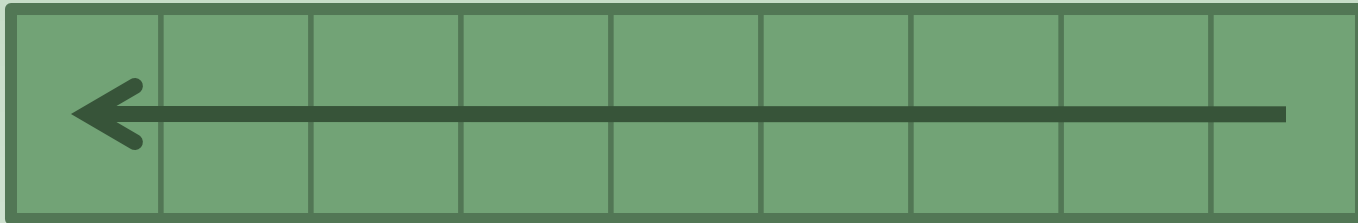
**a[-3:0:-2]**



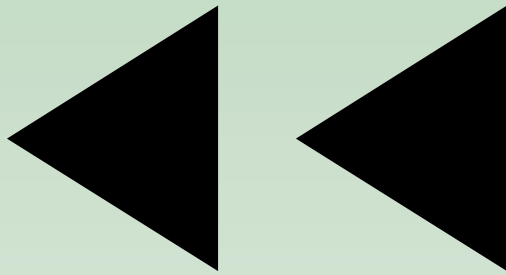
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8]  
a[-9] a[-8] a[-7] a[-6] a[-5] a[-4] a[-3] a[-2] a[-1]

사실 이런 거  
몰라도 돼요

**a[:: -1]**



a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8]  
a[-9] a[-8] a[-7] a[-6] a[-5] a[-4] a[-3] a[-2] a[-1]





# 리스트와 튜플을 함께 쓸 이유

```
birthdays = [  
    ('강성훈', 1987, 9, 13),  
    ('정재성', 1987, 2, 23),  
    ('김준기', 1987, 5, 12),  
]
```

```
for entry in birthdays:  
    name, year, month, day = entry  
print name, month, '월', day, '일생'
```

```
for name, year, month, day in birthdays:  
    print name, month, '월', day, '일생'
```

```
for name, _, month, day in birthdays:  
    print name, month, '월', day, '일생'
```

```
for name, _, month, day in birthdays:  
    print '%s - %d월 %d일생' % \  
        (name, month, day)
```

```
for name, _, month, day in birthdays:  
    print ' {0} - {1}월 {2}일생'.format(  
        name, month, day)
```

메소드



# 자료형이나 값과 연관된 함수

```
formatfun = '{0} - {1}월 {2}일생'.format
for name, _, month, day in birthdays:
    print formatfunc(name, month, day)
```

자료형 = 값

```
>>> type(4)
<type 'int'>
>>> type('hello?')
<type 'str'>
>>> type([1,2,3])
<type 'list'>
```

```
>>> type(4) == int
```

```
True
```

```
>>> type('hello?') == str
```

```
True
```

```
>>> type([1,2,3]) == list
```

```
True
```

```
>>> str.format
```

```
<method 'format' of 'str' objects>
```

```
fmt = '{0} - {1}월 {2}일생'  
for name, _, month, day in birthdays:  
    # 엄밀하게는...  
    print str.format(fmt, name,  
                      month, day)
```

```
>>> 'hello'.upper()
```

```
'HELLO'
```

```
>>> _.lower()
```

```
'hello'
```

```
>>> ' '.join(['we', 'are', 'the', 'world'])
```

```
'we are the world'
```

```
>>> _.split(' ')
```

```
['we', 'are', 'the', 'world']
```



```
>>> len('hello')
```

```
5
```

```
>>> str(42)
```

```
'42'
```

```
>>> '%d' % 42
```

```
'42'
```

```
>>> print str(1/5.0)
```

```
0.2
```

```
>>> 1/5.0
```

```
0.200000000000000000000001
```

```
>>> print repr(1/5.0)
```

```
0.200000000000000000000001
```

```
def names(birthdays):  
    result = []  
    for name, __, __, _ in birthdays:  
        result.append(name)  
    result.sort()  
    return result
```

```
>>> names(birthdays)
```

```
[' \xb0\xad\xbc\xba\xc8\xc6',  
 ' \xb1\xe8\xc1\xd8\xb1\xe2',  
 ' \xc1\xa4\xc0\xe7\xbc\xba']
```

```
>>> for name in names(birthdays):  
...     print name
```

```
...
```

```
강성훈
```

```
김준기
```

```
정재성
```

원래 세상  
다 이런 거지

```
def printbirthdays(birthdays):  
    for name, _, month, day in birthdays:  
        print '%s - %d월 %d일생' % \  
            (name, month, day)
```

```
def filterbyyear(birthdays, targetyear):  
    result = []  
    for entry in birthdays:  
        _, year, _, _ = entry  
        if year == targetyear:  
            result.append(entry)  
    return result
```



```
def filterbyyear(birthdays, targetyear):  
    def func(entry):  
        _, year, _, _ = entry  
        return year == targetyear  
    return filter(func, birthdays)
```

```
def filterbyyear(birthdays, targetyear):  
    def func((name, year, month, day)):  
        return year == targetyear  
    return filter(func, birthdays)
```

```
def filterbyyear(birthdays, targetyear):  
    return filter(  
        lambda (n,y,m,d): y==targetyear,  
        birthdays)
```

```
def filterbyyear(birthdays, targetyear):  
    return [(name, year, month, day)  
            for name, year, month, day  
            in birthdays  
            if year == targetyear]
```



스크립팅 언어?

스크립팅 언어

“스크립팅할때 쓰면  
스크립팅 언어”



# 목표

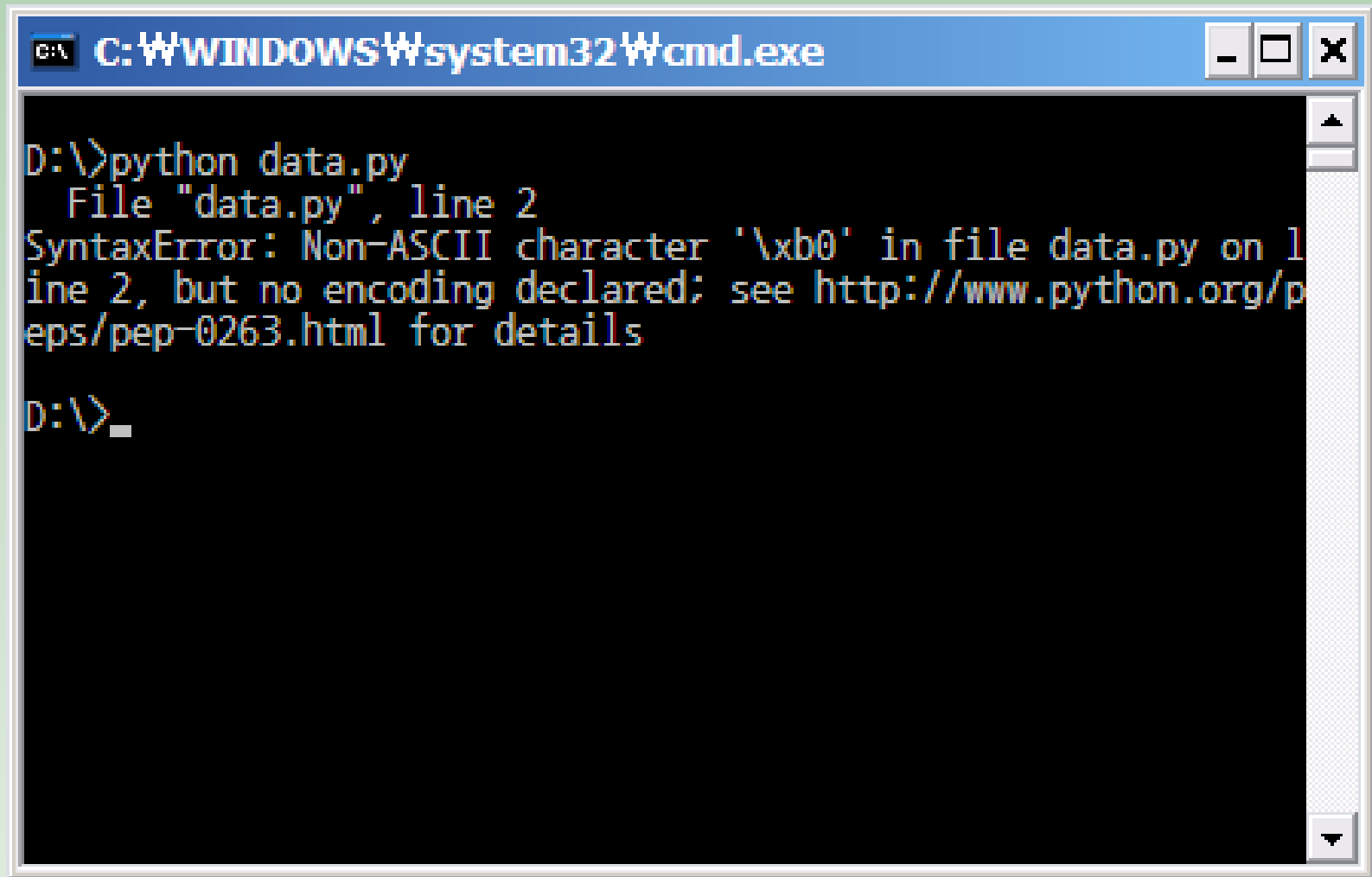
# 생년월일 검색하는 프로그램

- 보통은 이름과 생년월일을 모두 보여 준다
- 이름만 보여 줄 수도 있다
- 이름으로 검색할 수 있다
- 생년별 목록도 볼 수 있다

```
birthdays = [  
    ('강성훈', 1987, 9, 13),  
    ('정재성', 1987, 2, 23),  
    ('김준기', 1987, 5, 12),  
    ('안병욱', 1989, 10, 14),  
    ('강철', 1990, 3, 11),  
    ('유수형', 1991, 3, 13),  
    ('조유정', 1990, 4, 18),  
    ('김도국', 1990, 3, 11),  
]
```

데이터는 data.py  
프로그램은 birthday.py

출력할 때는  
꼭 `print` 명령어를  
써야 한다

A screenshot of a Windows command prompt window. The title bar at the top is blue and contains the text "C:\WINDOWS\system32\cmd.exe" along with standard window control buttons (minimize, maximize, close). The main area of the window is black with white text. The text shows a command being executed: "D:\>python data.py". This is followed by an error message: "File 'data.py', line 2", "SyntaxError: Non-ASCII character '\xb0' in file data.py on line 2, but no encoding declared; see http://www.python.org/peps/pep-0263.html for details". The prompt "D:\>" is shown again at the end of the output.

```
C:\WINDOWS\system32\cmd.exe
D:\>python data.py
File "data.py", line 2
SyntaxError: Non-ASCII character '\xb0' in file data.py on line 2, but no encoding declared; see http://www.python.org/peps/pep-0263.html for details
D:\>
```

```
# coding=cp949 ← utf-8이어야 할 수도 있음
```

```
birthdays = [  
    ('강성훈', 1987, 9, 13),  
    ('정재성', 1987, 2, 23),  
    ('김준기', 1987, 5, 12),  
    ('안병욱', 1989, 10, 14),  
    ('강철', 1990, 3, 11),  
    ('유수형', 1991, 3, 13),  
    ('조유정', 1990, 4, 18),  
    ('김도국', 1990, 3, 11),  
]
```



원래 세상  
다 이런 거지  
(2)

```
>>> import data
```

```
>>> print '%d명' % len(data.birthdays)
```

```
8명
```

```
# coding=cp949  
import data  
print '%d명' % len(data.birthdays)
```

```
# coding=cp949  
from data import birthdays  
print '%d명' % len(birthdays)
```

모름

birthday.py



data.py

**import data**

\_\_main\_\_



data

persons.py

\_\_main\_\_

**import** birthday

birthday.py

birthday

**import** data

data.py

data

```
# coding=cp949  
from data import birthdays  
  
def main():  
    print '%d명' % len(birthdays)  
  
if __name__ == '__main__': main()
```



birthday.py



data.pyc

**import data**

\_\_main\_\_



data

```
def main():  
    choice = showmenu()  
    if choice == 1:  
        printnames(birthdays)  
    elif choice == 2:  
        printbirthdays(birthdays)  
    elif choice == 3:  
        printbyname(birthdays)  
    elif choice == 4:  
        printbyyear(birthdays)
```

```
def printnames(birthdays):  
    pass # 이름만 출력할 것  
  
def printbirthdays(birthdays):  
    pass # 이름과 생년월일을 출력할 것  
  
def printbyname(birthdays):  
    pass # 이름을 입력받아서  
        # 해당하는 생년월일을 출력  
  
def printbyyear(birthdays):  
    pass # 생년을 입력받아서  
        # 해당하는 목록을 출력
```

```
def printnames(birthdays):  
    print 'TODO: 이름 목록을 출력'  
def printbirthdays(birthdays):  
    print 'TODO: 이름과 생년월일 출력'  
def printbyname(birthdays):  
    print ('TODO: 이름을 입력받아 '  
           '해당하는 생년월일을 출력')  
def printbyyear(birthdays):  
    print ('TODO: 생년을 입력받아 '  
           '해당하는 목록을 출력')
```

```
def showmenu():  
    print '----- 메뉴 -----'  
    print '1. 이름 보기'  
    print '2. 이름과 생년월일 보기'  
    print '3. 이름으로 찾기'  
    print '4. 생년으로 찾기'  
    return input('>>> ')
```

---- 메뉴 ----

1. 이름 보기
2. 이름과 생년월일 보기
3. 이름으로 찾기
4. 생년으로 찾기

>>> 2

TODO: 이름과 생년월일 출력

---- 메뉴 ----

1. 이름 보기
2. 이름과 생년월일 보기
3. 이름으로 찾기
4. 생년으로 찾기

>>> 5

---- 메뉴 ----

1. 이름 보기
2. 이름과 생년월일 보기
3. 이름으로 찾기
4. 생년으로 찾기

>>> 3\*4-11

TODO: 이름 목록을 출력



---- 메뉴 ----

1. 이름 보기
2. 이름과 생년월일 보기
3. 이름으로 찾기
4. 생년으로 찾기

>>> **end**

Traceback (most recent call last):

(생략)

File "<string>", line 1, in <module>

NameError: name 'end' is not defined

---- 메뉴 ----

1. 이름 보기
2. 이름과 생년월일 보기
3. 이름으로 찾기
4. 생년으로 찾기

>>> [엔터]

Traceback (most recent call last):

(생략)

File "<string>", line 0

^

SyntaxError: unexpected EOF while parsing

input으로는  
안 된다

```
>>> dir()
```

```
['__builtins__', '__doc__', '__name__',  
 '__package__']
```

```
>>> dir(__builtins__)
```

```
['ArithmeticError', 'AssertionError',  
 (...생략...), 'xrange', 'zip']
```

```
>>> [x for x in dir(__builtins__)  
...   if 'input' in x]  
['input', 'raw_input']
```

```
>>> help(raw_input)
```

```
Help on built-in function raw_input in module __builtin__:
```

```
raw_input(...)
```

```
raw_input([prompt]) -> string
```

Read a string from standard input. The trailing newline is stripped. If the user hits EOF (Unix: Ctl-D, Windows: Ctl-Z+Return), raise EOFError. On Unix, GNU readline is used if enabled. The prompt string, if given, is printed without a trailing newline before reading.

모르는 게 있으면  
우선 help를!

```
def inputnum(prompt):  
    return input(prompt)
```

```
def showmenu():  
    print '---- 메뉴 ----'  
    print '1. 이름 보기'  
    print '2. 이름과 생년월일 보기'  
    print '3. 이름으로 찾기'  
    print '4. 생년으로 찾기'  
    return inputnum('>>>')
```



```
def inputnum(prompt):  
    return int(raw_input(prompt))
```

```
def inputnum(prompt):  
    while True: # 무한반복 (사용에 주의!)  
        line = raw_input(prompt)  
        try:  
            return int(line)  
        except:  
            print '숫자를 입력하시죠.'
```

예외 (exception)

“일반적이지 않다”

```
>>> 3/0
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ZeroDivisionError: integer division or  
modulo by zero
```

```
>>> asdf
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
NameError: name 'asdf' is not defined
```

```
>>> int('notanumber')
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ValueError: invalid literal for int()  
with base 10: 'notanumber'
```

---- 메뉴 ----

1. 이름 보기
2. 이름과 생년월일 보기
3. 이름으로 찾기
4. 생년으로 찾기

>>> **exit**

숫자를 입력하시죠.

>>> **quit**

숫자를 입력하시죠.

>>> **나 좀 나가게 해 줘 서버**

숫자를 입력하시죠.

- 나가는 메뉴가 없다
- Ctrl-C 같은 강제 종료키도 try~except가 먹어버린다



```
def showmenu():  
    print '----- 메뉴 -----'  
    print '1. 이름 보기'  
    print '2. 이름과 생년월일 보기'  
    print '3. 이름으로 찾기'  
    print '4. 생년으로 찾기'  
    print '0. 끝내기'  
    return inputnum('>>>')
```

```
def inputnum(prompt):  
    while True:  
        line = raw_input(prompt)  
        try:  
            return int(line)  
        except ValueError:  
            print '숫자를 입력하시죠.'
```

# 방어적 프로그래밍

```
def names(birthdays):  
    result = []  
    for name, _, _, _ in birthdays:  
        result.append(name)  
    result.sort()  
    return result  
  
def printnames(birthdays):  
    print ', '.join(names(birthdays))
```

```
def printbirthdays(birthdays):  
    for name, _, month, day in birthdays:  
        print '%s - %d월 %d일생' % \  
            (name, month, day)
```

```
def printbyname(birthdays):  
    name = raw_input('이름을 입력하세요: ')  
    for n, y, m, d in birthdays:  
        if n == name:  
            print '%s - %d월 %d일생' % \  
                (n, m, d)
```

```
def printbyname(birthdays):  
    name = raw_input('이름을 입력하세요: ')  
    count = 0  
    for n, y, m, d in birthdays:  
        if n == name:  
            print '%s - %d월 %d일생' % \  
                (n, m, d)  
            count += 1  
    if count == 0:  
        print '그런 사람이 없습니다.'
```

똑같은 기능에  
똑같은 성능이면  
간결한 게 낫다



```
def filterbyname(birthdays, targetname):  
    return [(name, year, month, day)  
            for name, year, month, day  
            in birthdays  
            if name == targetname]
```

```
def printbyname(birthdays):  
    name = raw_input('이름을 입력하세요: ')  
    filtered = filterbyname(birthdays,  
                             name)  
    printbirthdays(filtered)
```

```
def filterbyyear(birthdays, targetyear):  
    return [(name, year, month, day)  
             for name, year, month, day  
             in birthdays  
             if year == targetyear]
```

```
def printbyyear(birthdays):  
    year = inputnum('생년을 입력하세요: ')  
    filtered = filterbyyear(birthdays,  
                             year)  
    printbirthdays(filtered)
```

---- 메뉴 ----

1. 이름 보기
2. 이름과 생년월일 보기
3. 이름으로 찾기
4. 생년으로 찾기
0. 끝내기

>>> 4

생년을 입력하세요: **1990**

강철 - 3월 11일생

조유정 - 4월 18일생

김도국 - 3월 11일생

새 기능을 만들려면?

```
def main():
    choice = showmenu()
    if choice == 1:
        printnames(birthdays)
    elif choice == 2:
        printbirthdays(birthdays)
    elif choice == 3:
        printbyname(birthdays)
    elif choice == 4:
        printbyyear(birthdays)
```

여기다  
새 조건을 추가

```
def showmenu():
    print '---- 메뉴 ----'
    print '1. 이름 보기'
    print '2. 이름과 생년월일 보기'
    print '3. 이름으로 찾기'
    print '4. 생년으로 찾기'
    print '0. 끝내기'
    return inputnum('>>> ')
```

여기다  
새 메뉴를 추가

하나의 함수에  
하나의 기능을

```
def showmenu():  
    print '---- 메뉴 ----'  
    print '1. 이름 보기'  
    print '2. 이름과 생년월일 보기'  
    print '3. 이름으로 찾기'  
    print '4. 생년으로 찾기'  
    print '0. 끝내기'  
    choice = inputnum('>>> ')  
    if choice == 1: return printnames  
    elif choice == 2: return printbirthdays  
    elif choice == 3: return printbyname  
    elif choice == 4: return printbyyear
```

```
CHOICES = {1: printnames, 2: printbirthdays,  
           3: printbyname, 4: printbyyear}  
  
def showmenu():  
    print '---- 메뉴 ----'  
    print '1. 이름 보기'  
    print '2. 이름과 생년월일 보기'  
    print '3. 이름으로 찾기'  
    print '4. 생년으로 찾기'  
    print '0. 끝내기'  
    choice = inputnum('>>> ')  
    if choice in CHOICES:  
        return CHOICES[choice]
```



```
def main():  
    routine = showmenu()  
    if routine: routine(birthdays)
```

```
def main():  
    while True:  
        routine = showmenu()  
        if not routine: return  
        routine(birthdays)  
        print
```

- `main`: 시작 함수
- `showmenu`: 메뉴를 보여줌
- `inputnum`: 숫자를 입력 받음
- `names`: 목록에서 이름만 반환
- `filterby...`: 목록을 조건에 따라 걸러냄
- `printbirthdays`: 목록을 출력
- `print...`: 각 메뉴를 구현함

여전히 중복이  
있지 않나요?

```
for name,_,_,_ in birthdays: ...
```

```
for name,_,month,day in birthdays: ...
```

```
[ (name,year,month,day)
```

```
  for name,year,month,day in birthdays  
  if ... ]
```

순서를 하나라도  
빼놓았다면?

# 새 자료형을 만들기

```
class person(object):  
    def __init__(self, name, year,  
                month, day):  
        self.name = name  
        self.year = year  
        self.month = month  
        self.day = day
```



자료형과 연관된  
함수?

# 즉 그냥 함수

(첫 인자만 빼고)

```
class person(object):  
    def __init__(you, name, year,  
                month, day):  
        you.name = name  
        you.year = year  
        you.month = month  
        you.day = day
```

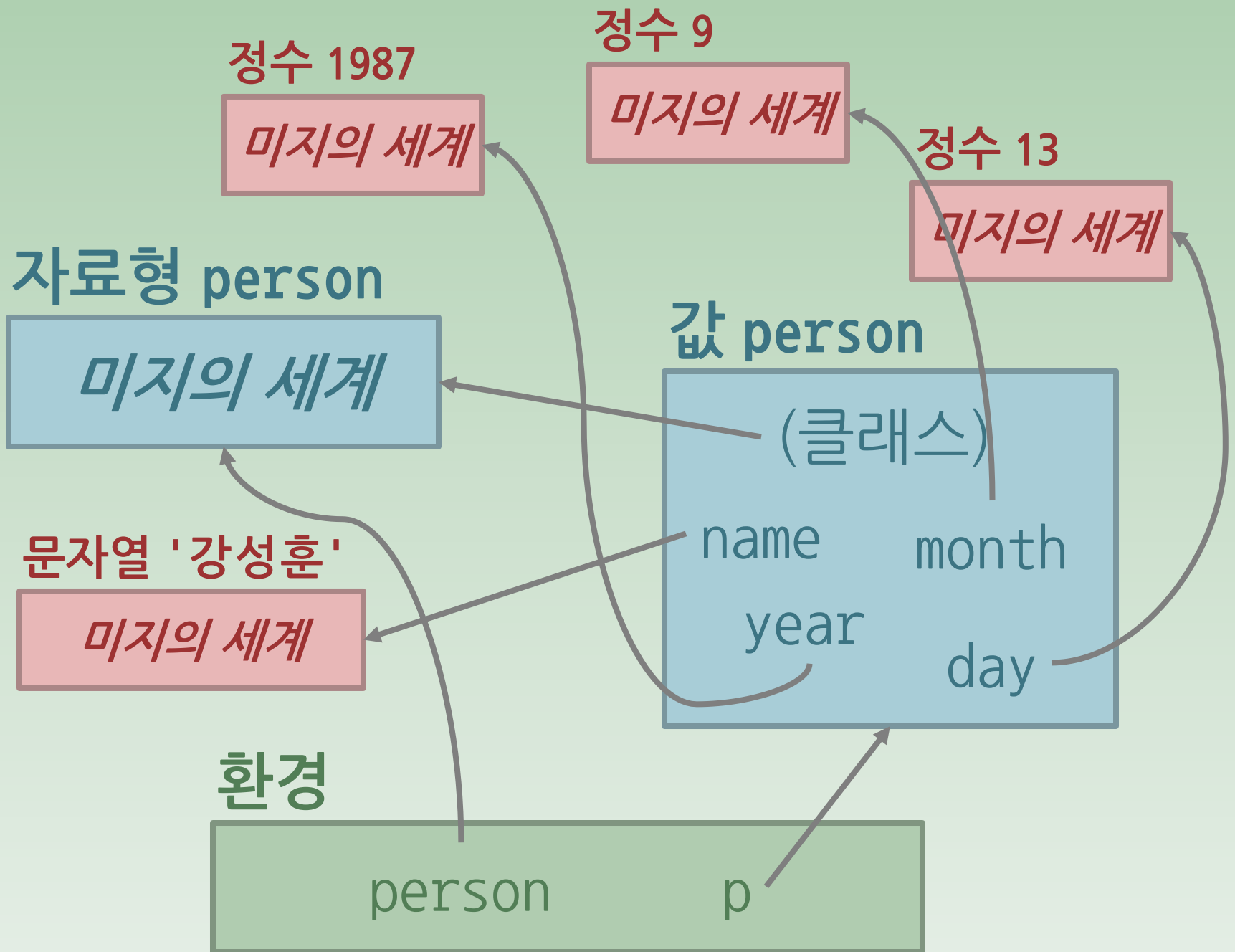
```
>>> p = person('강성훈', 1987, 9, 13)
>>> p
<__main__.person object at 0x00B12110>
>>> print p.name
강성훈
>>> p.year, p.month, p.day
(1987, 9, 13)
```

```
class person(object):  
    def __init__(self, name, year,  
                month, day):  
        self.name = name  
        self.year = year  
        self.month = month  
        self.day = day  
  
    def __str__(self):  
        return '%s - %d월 %d일생' % \  
                (self.name, self.month,  
                 self.day)
```

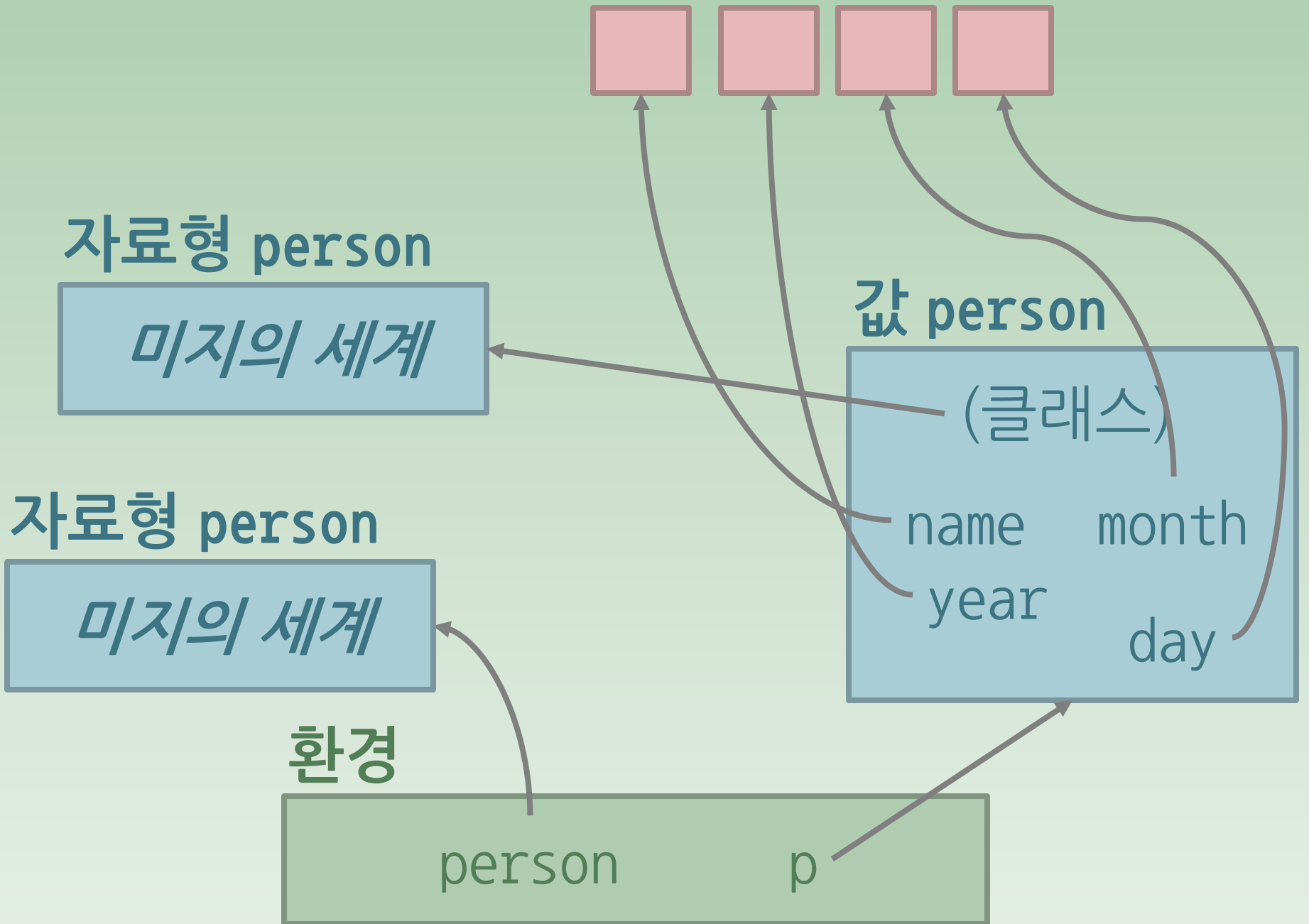
```
>>> print str(p)
```

```
<__main__.person object at 0x00B12110>
```

# 파이썬 메모리의 구조







모든 것은



요 레퍼런스로

```
>>> a = [[]] * 3
>>> a
[[], [], []]
>>> a[0].append(4)
>>> a
[[4], [4], [4]]
```

```
>>> a = [[] for i in range(3)]
```

```
>>> a
```

```
[[], [], []]
```

```
>>> a[0].append(4)
```

```
>>> a[1].append(5)
```

```
>>> a[2].append(6)
```

```
>>> a
```

```
[[4], [5], [6]]
```

제자리에서 변경이  
가능한가?

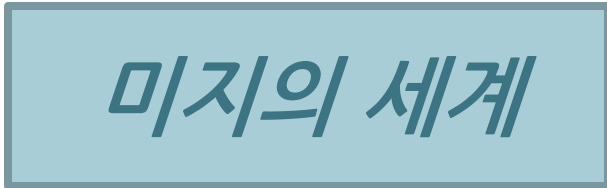
- 그런 것: 리스트, 사전
- 안 그런 것: 문자열, 튜플

하여튼...

망한 값들



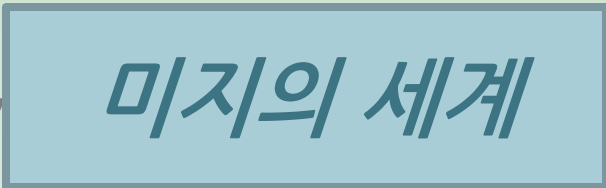
자료형 person



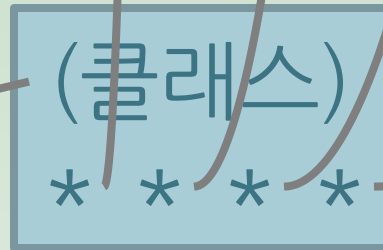
값 person



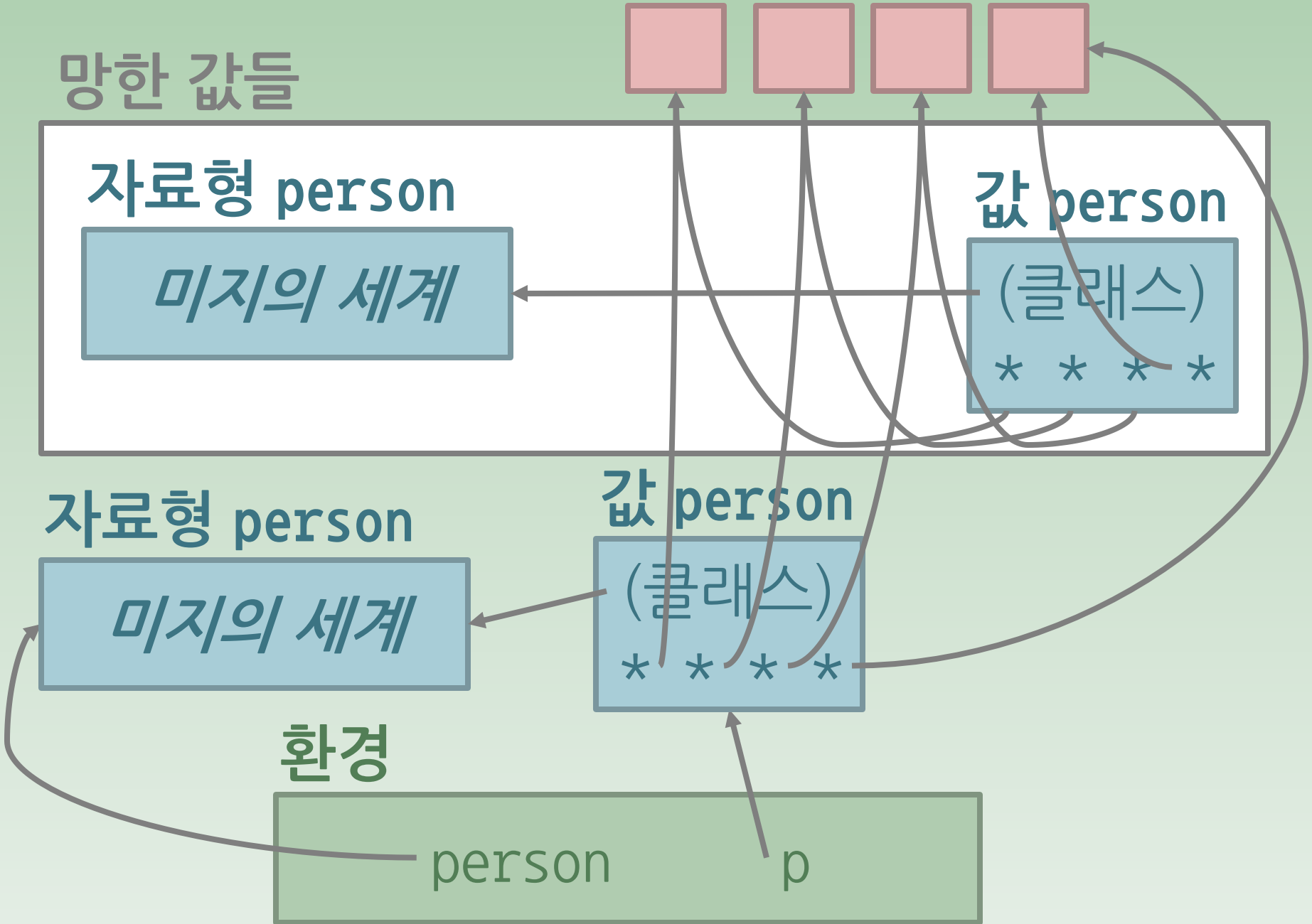
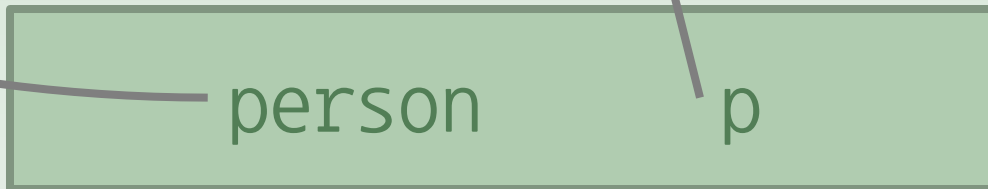
자료형 person



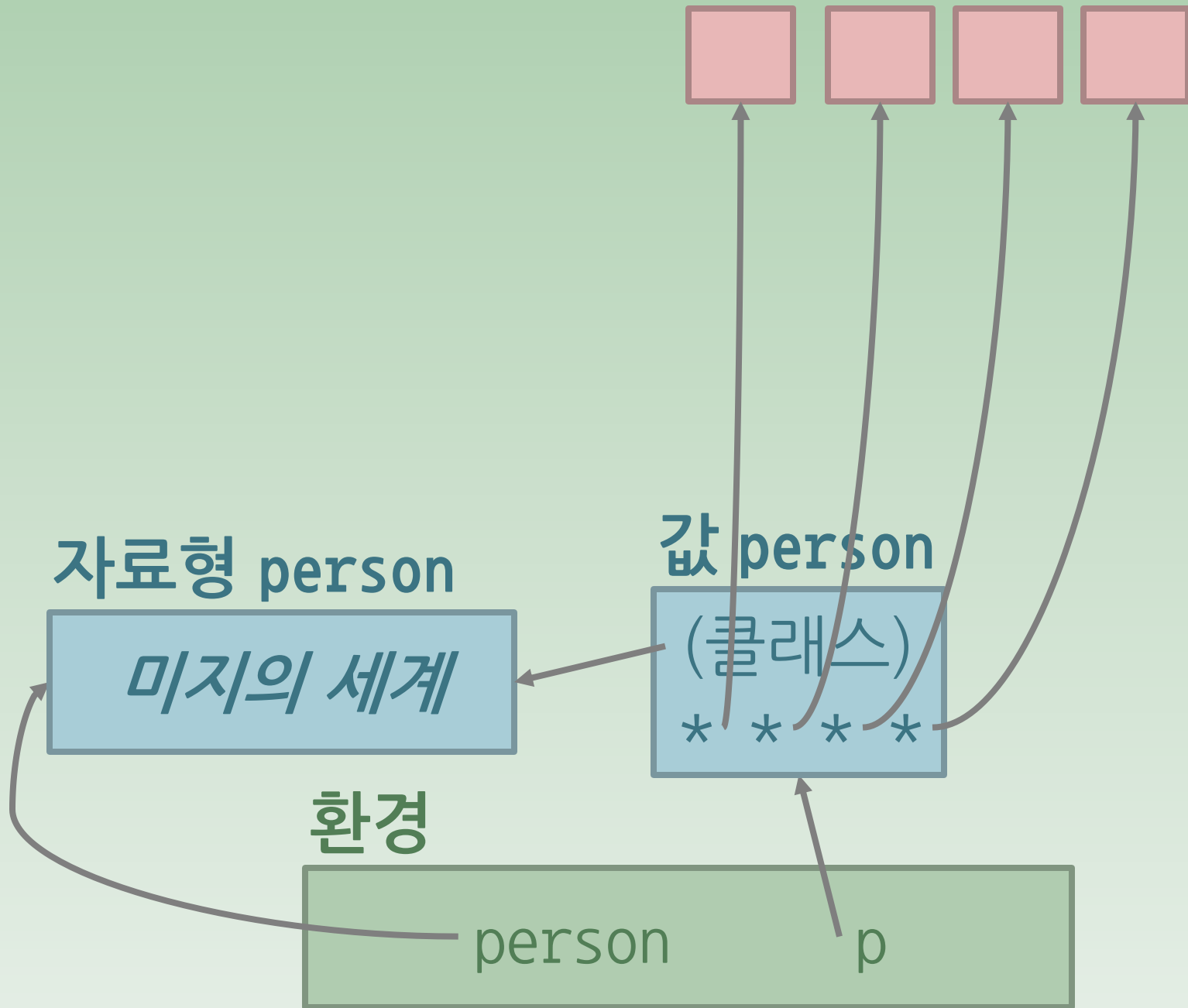
값 person



환경







가비지 컬렉션  
= 망한 값 처리하기

```
>>> p = person('강성훈', 1987, 9, 13)
```

```
>>> print p.__str__()
```

```
강성훈 - 9월 13일생
```

```
>>> print str(p)
```

```
강성훈 - 9월 13일생
```

```
>>> print p
```

```
강성훈 - 9월 13일생
```

마무리

```
# coding=cp949
class person(object):
    def __init__(self, ...): ...
    def __str__(self, ...): ...

birthdays = [
    person('강성훈', 1987, 9, 13),
    person('정재성', 1987, 2, 23),
    person('김준기', 1987, 5, 12),
    ...
]
```

```
def printnames(birthdays):  
    names = [p.name for p in birthdays]  
    names.sort()  
    print ', '.join(names)
```

```
def printbirthdays(birthdays):  
    for p in birthdays: print p
```

```
def printbyname(birthdays):  
    name = raw_input('이름을 입력하세요: ')  
    filtered = [p for p in birthdays  
                if p.name == name]  
    printbirthdays(filtered)
```

```
def printbyyear(birthdays):  
    year = inputnum('생년을 입력하세요: ')  
    filtered = [p for p in birthdays  
                if p.year == year]  
    printbirthdays(filtered)
```

더 고칠 거리를  
생각해 보세요!



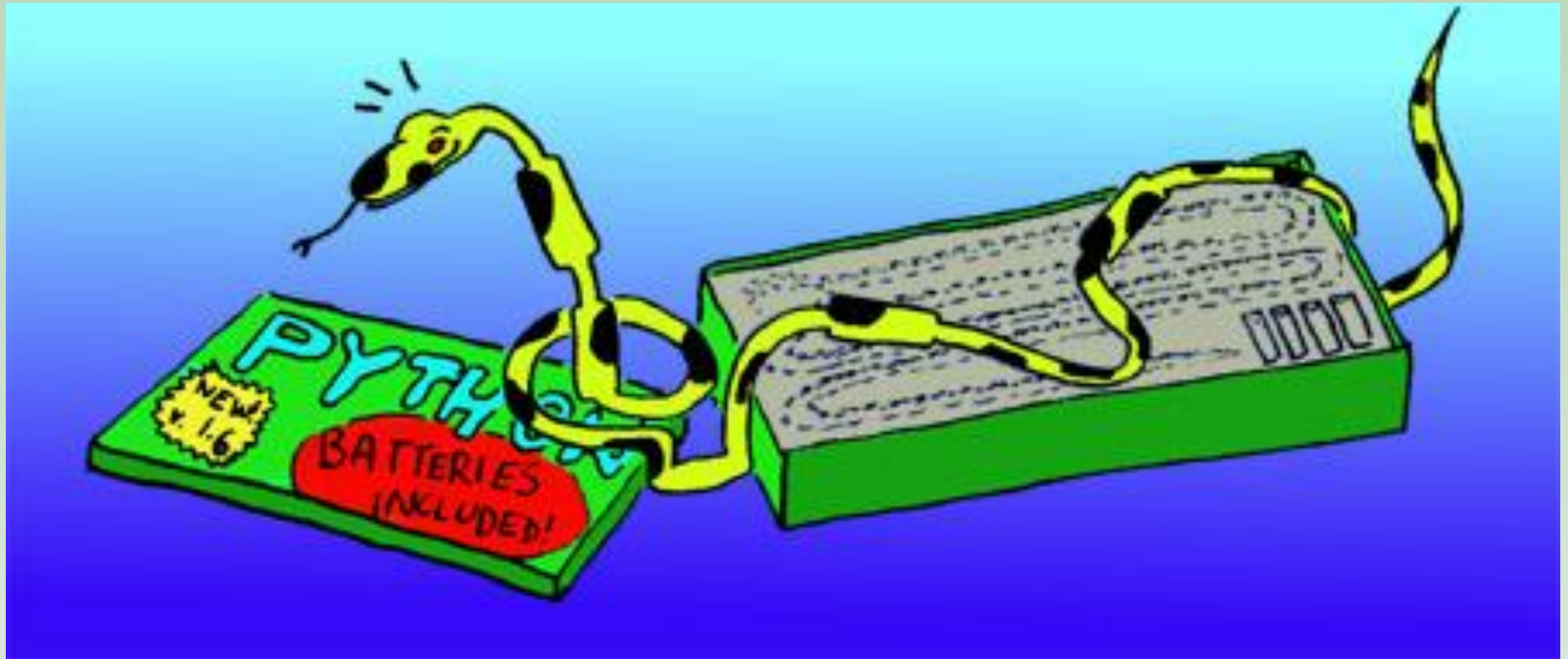
כב  
כב ...?

세상 모든 일이  
이렇게 잘 풀리진  
않겠지만...

- 리팩토링
- 테스트링
- 디버깅
- 문서화 ← 이건 안했지만

모든 것을 꼭 우리만  
만들러 법이 있나

“바퀴를 재발명하기”



```
>>> import sys
```

```
>>> sys.version
```

```
'2.6.2 (r262:71605, Apr 14 2009,  
22:40:02) [MSC v.1500 32 bit (Intel)]'
```

```
>>> import math
>>> math.sqrt(2)
1.4142135623730951
>>> 2 ** 0.5
1.4142135623730951
>>> math.factorial(16)
20922789888000L
```



```
>>> import datetime
>>> datetime.date.today()
datetime.date(2010, 3, 12)
>>> _.weekday()
4
>>> print ['월', '화', '수', '목', '금',
...        '토', '일'][_]
금
```

```
>>> import random
>>> menus = ['짜장', '짬뽕', '짜짬면']
>>> print menus[random.randint(0,
...                               len(menus)-1)]
짜장
>>> print random.choice(menus)
짬뽕
```

```
>>> import urllib
>>> for line in urllib.urlopen(
...     'http://www.census.gov/ipc/'
...     'www/popclockworld.html'):
...     if 'worldnumber' in line:
...         print line[45:-14]
```

```
6,807,870,286
```

## Parts of the documentation:

What's new in Python 2.6?  
*or all "What's new" documents since 2.0*

Tutorial  
*start here*

Using Python  
*how to use Python on different platforms*

Library Reference  
*keep this under your pillow*

Language Reference  
*describes syntax and language elements*

Python HOWTOs  
*in-depth documents on specific topics*

Extending and Embedding  
*tutorial for C/C++ programmers*

Python/C API  
*reference for C/C++ programmers*

Installing Python Modules  
*information for installers & sys-admins*

Distributing Python Modules  
*sharing modules with others*

Documenting Python  
*guide for documentation authors*

FAQs  
*frequently asked questions (with answers!)*

레퍼런스는 정독해도  
부족함이 없습니다

그 밖에

Numpy라거나

Django라거나

Pyglet이라거나...

안 다른 것들

- 클래스에 대한 자세한 내용
- set, unicode 같은 자료형들
- 외부 라이브러리 쓰기
- 패키지 다루기
- 그 밖에 파이썬을 쓰면서 필요하게 될 수많은 조언들



두 시간 안에 하긴  
좀 벅찬지라...

질문과 답변?

감사합니다.

너무 길어서  
정말로 죄송...

50

# 슬라이드 목차

- #1 시작
- #4 왜 파이썬을 배우는가?
- #13 계산기로 쓰기, 자료형
- #25 변수, 조건문, 반복문
- #36 리팩토링, 함수
- #44 리스트, 튜플
- #58 메소드, 값으로서의 자료형
- #72 내장 함수 및 메소드
- #86 아젠다 설정
- #92 모듈
- #106 생년월일 프로그램의 뼈대
- #116 내장 도움말
- #120 리팩토링 (2), 방어적 프로그래밍, 예외
- #132 생년월일 프로그램의 뼈대 (2)
- #140 리팩토링 (3)
- #148 클래스
- #159 파이썬 메모리의 구조
- #172 생년월일 프로그램의 마무리
- #180 파이썬 내장 라이브러리
- #191 안 다룬 것들

# 표기 컨벤션

- 코드 및 대화식 환경은 노란 배경으로,
- 프로그램 출력은 파란 배경으로,
- 직접 입력하지 않은 내용은 옅은 색으로,
- 이번 슬라이드에 처음 등장하는 문법·심볼은 붉은 색으로, (단, 공백 등은 별도 표시 안 함)
- 파이썬 프롬프트 및 예약어는 굵게 표기했음.