

Writing Python GUI program

Contents

Introduction	2
Dependancy	2
Installing Python 2.7	2
Installing wxPython	2
Installing GNU Radio	2
Installing RTL SDR	3
Installing RTL SDR Python wrapper	3
Installing matplotlib	3
Installing pip	3
Installing dateutil module	4
Installing pyparsing module	4
Installing pyserial	4
Installing numpy module	4
Installing SciPy	5
Python-wxPython Seminar	5
차례	6
Do it yourself	7
Python 문법	8
Python의 개념적인 계층	9
Core Object Types	10
Escape characters	11
List	12
Dictionary	13
Tuple	14
Getting help	14
Statements	17
Conditional statements	18

while loops	19
for loops	20
Function	21
Generator	22
Scope	23
Class object	24
Class method	25
Class attribute overloading	26
클래스 상속	27
Static and Class method	28
Decorator	30

Introduction

이 문서는 윈도에서 wxPython을 이용하여 GNU Radio application을 작성하는 방법에 대해서 정리합니다.

Dependancy

wxPython을 이용한 GNU Radio GUI 응용 프로그램을 개발하기 위해서 미리 설치해야 하는 소프트웨어들이 있습니다.

Installing Python 2.7

파이썬은 [여기](#)에서 미리 빌드된 바이너리를 다운 받아서 설치할 수 있습니다. 파이썬 2.7.x 버전과 3.x 버전이 있는데, 우리는 편의를 위해 파이썬 2.7.x 버전을 사용하겠습니다. 파이썬 설치 후 시스템 환경 변수의 Path에 C:\Python27\을 추가하고 PYTHONPATH라는 변수 이름으로 C:\Python27\Lib를 등록합니다.

Installing wxPython

wxPython은 [여기](#)에서 다운 받을 수 있습니다. 파이썬 2.7.x를 위한 wxPython을 찾아서 설치해야만 합니다.

Installing GNU Radio

파이썬과 wxPython가 모두 설치되었으면 다음은 GNU Radio를 설치합니다. 이것으로 wxPython을 import 할 모듈이 설치됩니다. 윈도를 위해 미리 빌드된 바이너리는 [여기](#)에서 다운 받을 수 있습니다. latest_stable 디렉토리에서 gnuradio_x.x.x.x_Win32.exe 파일을 다운로드 받으시면 됩니다.

다운받은 파일을 실행시키면 라이선스 확인을 한 뒤, 시스템 패쓰 설정을 합니다. 시스템의 모든 사용자를 위해 패쓰를 추가한다~Add GNU Radio to the system PATH for all users~고 체크 한 뒤 다음으로 진행하겠습니다. 설치 폴터를 정하고 (저는 C:\gnuradio로 하겠습니다.) 설치합니다. 설치가 완료되면 C:\gnuradio\lib\site-packages 디렉토리 안의 모든 디렉토리를 C:\Python27\Lib\site-packages에 복사합니다.

Installing RTL SDR

우리가 사용할 디바이스는 RTL이므로 RTL-SDR 소프트웨어도 설치합니다. [여기](#)에서 미리 빌드된 바이너리를 다운받을 수 있습니다. 다운로드 받은 파일의 압축을 C:\rtl-sdr-release에 풁니다.

압축을 푼 뒤, 시스템 환경 변수에 32비트 시스템이면 C:\rtl-sdr-release\x32를, 64비트 시스템이면 C:\rtl-sdr-release\x64를 Path에 추가합니다.

Installing RTL SDR Python wrapper

파이썬에서 RTL-SDR을 호출할 것이기 때문에 앞서 설치한 RTL-SDR 라이브러리의 파이썬 wrapper가 필요합니다. [여기](#)에서 다운받을 수 있습니다.

설치는 github에서 소스를 다운 받은 뒤, 아래와 같이 설치할 수 있습니다.

```
> python setup.py install
running install
running build
running build_py
creating build
creating build\lib
creating build\lib\rtlsdr
copying rtlsdr\helpers.py -> build\lib\rtlsdr
copying rtlsdr\librtlsdr.py -> build\lib\rtlsdr
copying rtlsdr\rtlsdr.py -> build\lib\rtlsdr
copying rtlsdr\__init__.py -> build\lib\rtlsdr
running install_lib
creating C:\Python27\Lib\site-packages\rtlsdr
copying build\lib\rtlsdr\helpers.py -> C:\Python27\Lib\site-packages\rtlsdr
copying build\lib\rtlsdr\librtlsdr.py -> C:\Python27\Lib\site-packages\rtlsdr
copying build\lib\rtlsdr\rtlsdr.py -> C:\Python27\Lib\site-packages\rtlsdr
copying build\lib\rtlsdr\__init__.py -> C:\Python27\Lib\site-packages\rtlsdr
byte-compiling C:\Python27\Lib\site-packages\rtlsdr\helpers.py to helpers.pyc
byte-compiling C:\Python27\Lib\site-packages\rtlsdr\librtlsdr.py to librtlsdr.pyc
byte-compiling C:\Python27\Lib\site-packages\rtlsdr\rtlsdr.py to rtlsdr.pyc
byte-compiling C:\Python27\Lib\site-packages\rtlsdr\__init__.py to __init__.pyc
running install_egg_info
Writing C:\Python27\Lib\site-packages\pyrtlsdr-0.1.1-py2.7.egg-info
```

설치가 완료되면 시스템 환경 변수의 Path에 C:\Python27\Lib\site-packages\rtlsdr를 추가해 줍니다.

[msvcr100.dll](#) 또는 [msvcp100.dll](#)이 없어서 오류가 발생할 수 있습니다. 그러면 이 파일들을 C:\Windows\system32에 넣으시면 됩니다.

Installing matplotlib

각종 그래프를 그리기 위해 [matplotlib](#)를 설치합니다. 링크로 연결된 곳에 가서 python 2.7.x.x에 해당하는 matplotlib를 다운받아 설치합니다.

Installing pip

파이썬 모듈 설치를 위해 제공되는 툴입니다.

먼저 [get-pip.py](#)를 다운받습니다. 저는 C:\Python27에 다운받았습니다.

```
PS C:\Python27> python get-pip.py
Downloading/unpacking pip
Downloading/unpacking setuptools
Installing collected packages: pip, setuptools
Successfully installed pip setuptools
Cleaning up...
```

다운로드 한 뒤에는 위와 같이 get-pip.py를 python 명령으로 실행하여 설치할 수 있습니다.

이제 pip를 이용하여 파이썬 모듈을 설치 할 수 있습니다.

Installing dateutil module

파이썬 모듈 중 하나인 dateutil을 설치합니다.

```
PS C:\Python27> python -m pip install python-dateutil
Downloading/unpacking python-dateutil
  Running setup.py (path:c:\users\b10013~1.myt\appdata\local\temp\pip_build_b100
133\python-dateutil\setup.py) egg_info for package python-dateutil

Downloading/unpacking six (from python-dateutil)
  Downloading six-1.7.3-py2.py3-none-any.whl
Installing collected packages: python-dateutil, six
  Running setup.py install for python-dateutil

Successfully installed python-dateutil six
Cleaning up...
```

Installing pyparsing module

파이썬 모듈 중 하나인 pyparsing을 설치합니다.

```
PS C:\Python27\Tools> python -m pip install pyparsing
Downloading/unpacking pyparsing
  Running setup.py (path:c:\users\b10013~1.myt\appdata\local\temp\pip_build_b100133\pyparsing\setup.py)
    package pyparsing

Installing collected packages: pyparsing
  Running setup.py install for pyparsing

Successfully installed pyparsing
Cleaning up...
```

Installing pyserial

RTL SDR Scanner를 테스트 해보기 위해 필요한 종속적인 모듈 라이브러리입니다.

Installing numpy module

파이썬 모듈 중 하나인 numpy를 설치합니다.

```
PS C:\Python27\Tools> python -m pip install numpy
```

이렇게 pip로 설치할 경우 시간이 오래 걸릴 수도 있습니다. 그것이 싫은 사람은 [numpy](#)를 아래 사이트에서 받아서 설치 할 수도 있습니다. windows version은 바로 python에 lib 폴더로 들어가도록 되어 있습니다.

Installing SciPy

[여기](#)에서 파이썬 2.7.x.x에 맞는 바이너리를 다운로드 받아서 설치합니다. 저는 [Scipy-stack](#)을 다운로드 받아서 설치했습니다.

Python-wxPython Seminar



Telechips Inc. reserves the right to revise this publication and to make changes to its content at any time, without obligation to notify any person or entity of such revisions or changes.

이 장에서는 제가 회사에서 세미나를 했던 내용을 정리합니다.

차례



Index

- I. Python 문법
- II. Python programming
- III. wxPython GUI programming

세미나의 순서는 파이썬 문법, 파이썬 프로그래밍 실습, wxPython을 이용한 GUI 프로그래밍에 대해서 기술합니다.

Do it yourself

Do it yourself



9GAG.COM/GAG/4826111

3 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential TeleChips

파이썬을 공부하기 시작한지 3주가 지난 지금, 파이썬은 아주 어렵지도 그렇다고 아주 쉽지도 않은 그런 프로그래밍 언어라는 생각이 들었습니다. 그래서 줋어 먹기로 이 스크립트 언어를 배우겠구나 생각하는 것은 잘못되었다는 것을 알았고, 또 해보지도 않고 지레 겁 먹을 필요도 없다고 생각했습니다. 결국 직접 해보는 수밖에 없습니다.

이 세미나는 여러분이 파이썬을 직접 다룰 수 있게 3주 동안 제가 삽질하며 얻은 지식들을 공유하는 세미나가 될 것 입니다. 그래서 “Python in a nutshell”과 같이 파이썬을 깊이 파고들지 않고 제가 했던 혼자서 파이썬을 공부하는 방법에 대한 논의를 하고자 합니다. 물론 저와 다른 방식으로 공부하실 수도 있고, 제가 한 방식이 비효율적일 수도 있습니다. 다만 여러분이 여러분의 방식으로 파이썬을 공부할 때 제가 삽질한 경험이 도움이 되길 바랍니다.

I. Python 문법

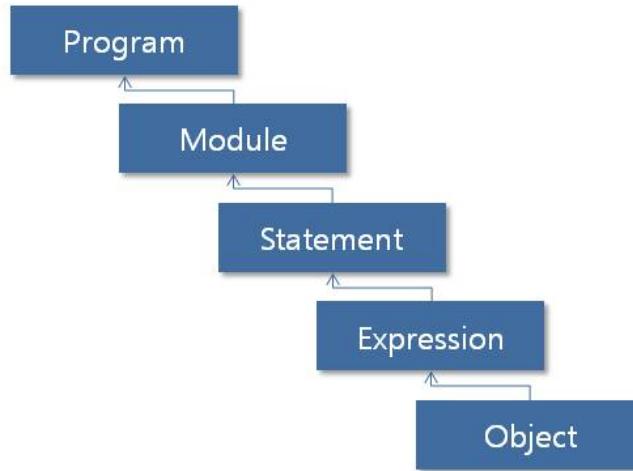
I. Python Syntax

4 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

그럼 먼저 파이썬 문법에 대해서 정리를 하겠습니다.

I. Python의 개념적인 계층



5 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential **TeleChips**

파이썬은 프로그램, 모듈, 구문(statement), 표현(expression), 객체(object)의 개념이 계층적으로 구성되어 있습니다. 프로그램은 모듈들의 집합이고, 모듈은 구문들로 구성되어 있으며, 구문(statement)은 표현(expression)으로 이루어져 있고, 표현(expression)은 객체(object)를 생성하고 처리합니다.

여기에서 객체(object)라는 개념에 대한 설명이 혼란스러운데, “Learning Python 5E(오라일리)” 책의 p.93에 있는 내용에 있는 것을 일부만 그대로 가져와서 그렇습니다. 제가 이해하기로는 일반적인 C/C++ 개발자들을 배려해서 “객체=자료구조”로서 설명하는 것 같습니다.

Core Object Types

I. Core Object Types

Object type	Example literals/creation
Numbers	1234, 3.1415, 3+4j, 0b111, Decimal(), Fraction()
Strings	'spam', "Bob's", b'a\x01c', u'xp\xc4m'
Lists	[1, [2, 'three'], 4.5], list(range(10))
Dictionaries	{'food': 'spam', 'taste':'yum'}, dic(hours=10)
Tuples	(1, 'spam', 4, 'U'), tuple('spam'), namedtuple
Files	open('eggs.txt'), open(r'C:\ham.bin', 'wb')
Sets	set('abc'), {'a', 'b', 'c'}
Other core types	Booleans, types, None
Program unit types	Functions, modules, classes
Implementation-related types	Compiled code, stack traceback

© The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

파이썬의 핵심 객체 타입은 이 표와 같습니다. 숫자, 문자열, 그리고 몇몇 데이터 타입들과 사용자 정의 객체들이 있습니다. 각 객체들은 C++을 설명하는 많은 책에서 말하는 메소드와 멤버 변수들을 속성으로 갖고 있습니다. 속성들은 파이썬 언어에서 built-in으로 제공되는 속성도 있고, 써드 파티나 개별 개발자들이 확장한 속성도 있습니다. 속성에 접근하기 위해서 ”.”를 사용합니다.

객체이름.속성-메소드(인자)

속성을 이용하는 방법은 위와 같습니다.

Escape characters

I. Escape characters

Escape	Meaning
\newline	Ignored (continuation line)
\\\	Backslash (stores one \)
\'	Single quote (stores ')
\"	Double quote (stores ")
\a	Bell
\b	Backspace
\f	Formfeed
\n	Newline(linefeed)
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\xhh	Character with hex value hh (exactly 2 digits)
\ooo	Character with octal value ooo (up to 3 digits)
\0	Null: binary 0 character (doesn't end string)
\N{id}	Unicode database ID
\uhhhh	Unicode character with 16-bit hex value
\uhhhhhhhh	Unicode character with 32-bit hex value
\other	Not an escape (keeps both '\' and 'other')

7 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential **TeleChips**

파이썬에서도 특수 문자가 있습니다. 위의 도표는 파이썬에서 사용하는 특수 문자들의 목록을 보입니다.

List

I. List

List는 객체들의 순서열이다.

```
>>> L = [123, 'spam', 1.23]          # A list of three different-type objects
>>> len(L)                           # Number of items in the list
3
>>> L[0]                             # Indexing by position
123
>>> L[:-1]                           # Slicing a list returns a new list
[123, 'spam']
>>> L+[4, 5, 6]                      # Concatenate make new lists too
[123, 'spam', 1.23, 4, 5, 6]
>>> L*2                             # Repeat make new lists too
[123, 'spam', 1.23, 123, 'spam', 1.23]
>>> L.append('NI')                  # Growing: add object at end of list
>>> L
[123, 'spam', 1.23, 'NI']
>>> L.pop(2)                         # Shrinking: delete an item in the middle
>>> L
[123, 'spam', 'NI']
>>> L[99]
...error text omitted...
IndexError: list index out of range
>>> M = [[1,2,3], [4,5,6], [7,8,9]]      # A 3X3 matrix, as nested lists
>>> M[1]
[4,5,6]
>>> M[1][2]
6
```

8 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

리스트는 C/C++에서 배열과 비슷합니다. 다만, 배열은 같은 타입의 데이터만 저장이 가능한 반면 리스트는 그러한 제한이 없습니다. 리스트 객체에도 속성이 있습니다. 이 속성을 이용하여 리스트의 기능을 이용할 수 있습니다.

Dictionary

I. Dictionary

Dictionary는 키로 색인되는 객체들을 담는 연관 배열 혹은 해시 테이블이다.

```
>>> D = { 'food':'Spam', 'quantity':4, 'color':'pink' }
>>> D['food']
'Spam'
>>> D['quantity'] += 1
>>> D
{'color':'pink', 'food':'Spam', 'quantity':5}
>>> D = {}
>>> D['name'] = 'Bob'
>>> D['job'] = 'dev'
>>> D['age'] = 40
>>> D
{'age':40, 'job':'dev', 'name':'Bob'}
>>> rec = {'name':{'first':'Bob', 'last':'Smith'}, 'jobs':['dev', 'mgr'], 'age':40.5}
>>> rec['name']['last']
'Smith'
>>> rec['jobs'][-1]
'mgr'
>>> rec['jobs'].append('guitar list')
>>> rec
{'age':40.5, 'jobs':['dev', 'mgr', 'guitar list'], 'name':{'last':'Smith',
'first':'Bob'})
```

9 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

Dictionary는 ‘키’:‘값’의 목록입니다. Dictionary의 ‘값’은 리스트처럼 순서를 가리키는 숫자로 찾지 않고 ‘키’로 찾습니다.

Tuple

I. Tuple

Tuple은 읽기 전용 List입니다.

```
>>> T = (1, 2, 3, 4)      # A 4-item tuple
>>> len(T)                # Length
4
>>> T+(5, 6)              # Concatenation
(1, 2, 3, 4, 5, 6)
>>> T[0]                  # Indexing
1
>>> T[0] = 2               # Tuples are immutable
...error text omitted...
TypeError: 'tuple' object does not support item assignment
>>> T = (2,) + T[1:]       # Make a new tuple for a new value
>>> T
(2, 2, 3, 4)
>>> T = 'spam', 3.0, [11, 22, 33]
>> T[1]
3.0
>>> T[2][1]
22
>>> T.append(4)
AttributeError: 'tuple' object has no attribute 'append'
```

10 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

파이썬에는 읽기 전용 리스트인 튜플이라는 데이터 형이 있습니다.

Getting help

객체들의 속성들에 대해서는 직접 찾아서 확인할 수 있습니다.

Getting help (cont')

dir을 이용한 객체의 메소드 확인



11 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

dir 함수의 이름은 **directory**(목록)에서 왔습니다. dir 함수를 이용하여 객체의 속성을 볼 수 있습니다. dir 함수는 built-in 함수로 파일 언어에 내장되어 있습니다.

Getting help

help를 이용한 메소드의 도움말 보기

```
help(object.__class__)
Help on class type in module __builtin__:
class type(object)
| type(object) -> the object's type
| type(name, bases, dict) -> a new type

| Methods defined here:
|   __call__(...)
|     x.__call__(...) <==> x(...)
|   __delattr__(...)
|     x.__delattr__('name') <==> del x.name
|   __eq__(...)
|     x.__eq__(y) <==> x==y
|   ...
| -----
| Data descriptors defined here:
|   __abstractmethods__
|   __base__
|   __bases__
|   __basicsize__
|   __dict__
|   __dictoffset__
|   __flags__
|   __itemsize__
|   __mro__
|   __weakrefoffset__
| -----
| Data and other attributes defined here:
|   __new__ = <built-in method __new__ of type object>
|     T.__new__(S, ...) -> a new object with type S, a subtype of T
```

1.2 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

help 함수는 dir 함수로 확인한 메소드의 도움말을 볼 수 있습니다.

Statements

I. Statements

Statement	Role	Example
Assignment	Creating references	<code>A, b = 'good', 'bad'</code>
Calls and other expressions	Running functions	<code>log.write("spam, ham")</code>
<code>print</code> calls	Printing objects	<code>print('The Killer', joke)</code>
<code>if/elif/else</code>	Selecting actions	<code>if "python" in text: print(text)</code>
<code>for/else</code>	Iteration	<code>for x in mylist: print(x)</code>
<code>while/else</code>	General loops	<code>while X > Y: print('hello')</code>
<code>pass</code>	Empty placeholder	<code>while True: pass</code>
<code>break</code>	Loop exit	<code>while True: if exittest(): break</code>
<code>continue</code>	Loop continue	<code>while True: if skiptest(): continue</code>
<code>def</code>	Functions and methods	<code>def f(a, b, c=1, *d): print(a+b+c+d[0])</code>
<code>return</code>	Functions results	<code>def f(a, b, c=1, *d): return a+b+c+d[0]</code>
<code>yield</code>	Generator functions	<code>def gen(n): for i in n: yield i*2</code>
<code>global</code>	Namespaces	<code>x = 'old' def function(): global x, y; x='new'</code>

1.3 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

I. Statements (cont')

Statement	Role	Example
<code>nonlocal</code>	Namespaces (3.X)	<code>def outer(): x = 'old' def function(): nonlocal x; x = 'new'</code>
<code>import</code>	Module access	<code>import sys</code>
<code>from</code>	Attribute access	<code>from sys import stdin</code>
<code>class</code>	Building objects	<code>class Subclass(Superclass): staticData=[] def method(self):pass</code>
<code>try/except/finally</code>	Catching exceptions	<code>try: action() except: print('action error')</code>
<code>raise</code>	Triggering exceptions	<code>raise EndSearch(location)</code>
<code>assert</code>	Debugging checks	<code>assert X>Y, 'X too small'</code>
<code>with/as</code>	Context managers (3.X, 2.6+)	<code>with open('data') as my_file: process(myfile)</code>
<code>del</code>	Deleting references	<code>del data[k] del data[i:j] del obj.attr del variable</code>

파이썬의 statement는 코드를 이루는 뼈대와 살 중 살과 같습니다. 위의 슬라이드에서는 파이썬의 statement 들에 대해서 정리하고 있습니다.

Conditional statements

I. Condition statements

```
>>> branch = { 'spam':1.25 ,  
...           'ham' :1.99 ,  
...           'eggs':0.99 }  
>>> print(branch.get('spam' ,  
'Bad choice'))  
1.25  
  
>>> print(branch.get('bacon' ,  
'Bad choice'))  
  
Bad choice
```

```
>>> branch = { 'spam':1.25 ,  
...           'ham' :1.99 ,  
...           'eggs':0.99 }  
>>> choice = 'bacon'  
>>> if choice in branch:  
...     print(branch[choice])  
... else:  
...     print('Bad choice')  
...  
Bad choice
```

15 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

파이썬 프로그래밍에서 조건 제어의 방법은 여러가지가 있을 수 있습니다. 속성을 이용한 선택 제어가 있을 수 있고, 다른 프로그래밍 언어와 같이 **if/elif/else** statement를 사용할 수도 있습니다.

while loops

I. while Loops

Syntax

```
while test:      #loop test
    statements #loop body
else:           #optional else
    statements #run if didn't exit loop with break
```

Example

```
>>> x = 1
>>> y = 1
>>> while x < 5
...     while x > y-1:
...         print('*'),
...         y += 1
...     print('\n')
...     x += 1
...     y = 1
...
*
*
*
*
*
```

16 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

위의 슬라이드 예제에서 `print('*')`,을 보면 문장 마지막에 ','가 있습니다. 이것은 `print` 함수를 호출할 때 자동으로 붙는 개행 문자를 없애주는 것을 의미합니다.

for loops

I. for Loops

Syntax

```
for target in object:  
    statements  
    if test: break  
    if test: continue  
else:  
    statements
```

Assign object items to target
Exit loop now, skip else
Go to top of loop now
If we didn't hit a 'break'

Example

```
>>> for x in range(0, 5):  
...     for y in range(0, x):  
...         print('*'),  
...     print()  
...  
*  
* *  
* * *  
* * * *
```

1.7 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

위의 슬라이드 예제에서 `range` 함수는 아래와 같은 도움말을 가지고 있습니다.

```
help(range)  
Help on built-in function range in module __builtin__:  
  
range(...)  
    range(stop) -> list of integers  
    range(start, stop[, step]) -> list of integers  
  
    Return a list containing an arithmetic progression of integers.  
    range(i, j) returns [i, i+1, i+2, ..., j-1]; start (!) defaults to 0.  
    When step is given, it specifies the increment (or decrement).  
    For example, range(4) returns [0, 1, 2, 3]. The end point is omitted!  
    These are exactly the valid indices for a list of 4 elements.
```

설명에 따르면 `range` 함수는 정수의 리스트를 리턴합니다. 그래서 `for statement`에서 `in` 뒤의 리스트 객체가 오면 그 객체의 개수만큼 반복한다는 것을 알 수 있습니다.

Function

Function

Syntax

```
def name(arg1, arg2, ... argN):  
    statements  
    return value
```

Example

```
>>> def times(x, y):  
...     return x*y  
...  
>>> times(2, 4)  
  
8
```

18 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

함수를 정의할 때는 **def** 키워드를 사용합니다.

Generator

Generator

`yield`를 사용하는 함수를 generator라고 부릅니다.

```
>>> def countdown(n):
...     print "Counting down!"
...     while n>0:
...         yield n
...         n -= 1
...
>>> c = countdown(5)
>>> c.next()
Counting down!
5
>>> c.next()
4
>>> c.next()
3
>>> c.next()
2
>>> c.next()
1
>>> c.next()
-----
StopIteration                                     Traceback (most recent call last)
<ipython-input-8-50b4dad19337> in <module>()
----> 1 c.next()

StopIteration:
```

19 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

Generator는 조금 독특한 느낌이 드는 기능입니다. 어떤 키워드를 말하는 것도 아니고, 어떤 함수에 속한 기능을 이야기 합니다. 다시 말해서 Generator 함수를 Iterator 함수라고도 부르는 것 같습니다.¹

위 슬라이드의 예제에서 보이다 시피, `yield`로 지정된 변수를 생성하여 정적으로 다루어지는 경우 Generator라고 부르고 그런 기능을 포함한 함수를 Generator 함수라고 부릅니다. 그리고 이 기능의 동작이 마치 iterator와 같아서 iterator 함수로도 부르는 것 같습니다.

¹여러 인터넷 문서와 책에서 Generator 함수와 Iterator 함수는 같은 의미로도 쓰이고 따로따로 쓰이기도 하는 것 같다고 느꼈습니다.

Scope

Built-in (Python)

Names pre-assigned in the built-in names module: open, range, SyntaxError...

Global (module)

Names assigned at the top-level of a module file, or declared global in a *def* within the file

Enclosing function locals

Names in the local scope of any and all enclosing functions (*def* or *lambda*), from inner to outer

Local (function)

Names assigned in any way within a function (*def* or *lambda*), and not declared global in that function

²범위에 대한 예제가 필요한 것 같습니다.

Class object

Class object

class 객체의 정의와 초기화

```
class Person:  
    def __init__(self, name, job=None, pay=0):  
        self.name = name  
        self.job = job  
        self.pay = pay  
  
bob = Person('Bob Smith')  
  
sue = Person('Sue Jones', job='dev', pay=100000)  
  
print(bob.name, bob.pay)  
('Bob Smith', 0)  
  
print(sue.name, sue.pay)  
('Sue Jones', 100000)
```

21 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

class 키워드를 붙여서 클래스를 정의합니다. 이 예제에서는 클래스에 init 재정의만 들어있습니다. init의 도움말을 보면 아래와 같습니다.

```
help(object.__class__.__init__)  
Help on wrapper_descriptor:  
  
__init__(...)  
    x.__init__(...) initializes x; see help(type(x)) for signature
```

init은 object 객체의 class 속성의 속성인 메소드입니다. 이 built-in 메소드를 재정의합니다. 그러면 클래스의 인스턴스가 만들어질 때 init이 호출되어 인스턴스의 멤버 변수를 초기화 할 수 있습니다.³

class의 인스턴스에 접근하기 위해서 self를 init 메소드의 첫번째 인자로 사용한 것에 주의하시길 바랍니다. 인스턴스의 멤버 변수를 클래스의 메소드에서 접근하기 위해 self를 사용하게 됩니다.

³클래스와 인스턴스에 대해서는 따로 정리하지 않습니다.

Class method

Class method

class 객체의 method 추가

```
class Person:  
    def __init__(self, name, job=None, pay=0):  
        self.name = name  
        self.job = job  
        self.pay = pay  
    def lastName(self):  
        return self.name.split()[-1]  
    def giveRaise(self, percent):  
        self.pay = int(self.pay * (1+percent))  
  
bob = Person('Bob Smith')  
sue = Person('Sue Jones', job='dev', pay=100000)  
print(bob.name, bob.pay)  
('Bob Smith', 0)  
print(sue.name, sue.pay)  
('Sue Jones', 100000)  
print(bob.lastName(), sue.lastName())  
('Smith', 'Jones')  
sue.giveRaise(.10)  
print(sue.pay)  
110000
```

22 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

클래스의 메소드 정의는 위의 예와 같이 **class** 정의 안에 함수 정의를 하면 됩니다. 클래스 메소드에 리턴하는 값이 있으면 **return** 키워드를 쓰고 그렇지 않으면 안쓰면 됩니다.

Class attribute overloading

Class attribute overloading

class 객체의 attribute overriding

```
class Person:  
    def __init__(self, name, job=None, pay=0):  
        self.name = name  
        self.job = job  
        self.pay = pay  
    def lastName(self):  
        return self.name.split()[-1]  
    def giveRaise(self, percent):  
        self.pay = int(self.pay * (1+percent))  
    def __repr__(self):  
        return '[Person: %s, %s]' % (self.name, self.pay)  
  
bob = Person('Bob Smith')  
sue = Person('Sue Jones', job='dev', pay=100000)  
print(bob)  
[Person: Bob Smith, 0]  
print(sue)  
[Person: Sue Jones, 100000]  
print(bob.lastName(), sue.lastName())  
('Smith', 'Jones')  
sue.giveRaise(.10)  
print(sue)  
[Person: Sue Jones, 110000]
```

23 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

슬라이드의 예제에서 `init`을 재정의했듯이 `repr`를 재정의합니다.

클래스 상속

Class 상속

class 객체의 상속

```
class Person:  
    def __init__(self, name, job=None, pay=0):  
        self.name = name  
        self.job = job  
        self.pay = pay  
    def lastName(self):  
        return self.name.split()[-1]  
    def giveRaise(self, percent):  
        self.pay = int(self.pay * (1+percent))  
    def __repr__(self):  
        return '[Person: %s, %s]' % (self.name, self.pay)  
  
class Manager(Person):  
    def giveRaise(self, percent, bonus=.10):  
        Person.giveRaise(self, percent+bonus)  
  
tom = Manager('Tom Jones', 'mgr', 50000)  
tom.giveRaise(.10)  
print(tom.lastName())  
Jones  
print(tom)  
[Person: Tom Jones, 60000]
```

24 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

클래스 객체의 상속은 객체 정의에 부모 클래스의 이름을 넣기만 하면 됩니다.

Static and Class method

Static and Class methods

overriding 할 수 없는 class-level method: Static and Class method

```
Static method
class AClass(object):
    def astatic( ):
        print("a static method")
    astatic = staticmethod(staticmethod)

anInstance = AClass( )
AClass.asstatic( ) # possible to call
a static method
anInstance.asstatic( ) # possible to call
a static method
-----
Class method
class ABase(object):
    def aclassmet(cls):
        print("a class method for %s" % cls.__name__)
    aclassmet = classmethod(aclassmet)

class ADeriv(ABase):
    pass

bInstance = ABase( )
dInstance = ADeriv( )
ABase.aclassmet( )
a class method for ABase
bInstance.aclassmet( )
a class method for ABase
ADeriv.aclassmet( )
a class method for ADeriv
dInstance.aclassmet( )
a class method for ADeriv
```

25 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

static method와 class method는 class-level 메소드라고도 부릅니다. 인스턴스를 생성하지 않고도 메소드를 호출할 수 있어서 그렇습니다.

static method와 class method의 차이를 class method 예제를 보면 알 수 있습니다.

class method의 예제를 보기 전에 먼저 classmethod()의 도움말을 보겠습니다.

```
help(classmethod)
Help on class classmethod in module __builtin__:

class classmethod(object)
|   classmethod(function) -> method
|
|   Convert a function to be a class method.
|
|   A class method receives the class as implicit first argument,
|   just like an instance method receives the instance.
|   To declare a class method, use this idiom:
|
|       class C:
|           def f(cls, arg1, arg2, ...): ...
|           f = classmethod(f)
|
|   It can be called either on the class (e.g. C.f()) or on an instance
```

```

| (e.g. C().f()). The instance is ignored except for its class.
| If a class method is called for a derived class, the derived class
| object is passed as the implied first argument.

|
| Class methods are different than C++ or Java static methods.
| If you want those, see the staticmethod builtin.

|
| Methods defined here:

|
| __get__(...)
|     descr.__get__(obj[, type]) -> value

|
| __getattribute__(...)
|     x.__getattribute__('name') <==> x.name

|
| __init__(...)
|     x.__init__(...) initializes x; see help(type(x)) for signature

-----
| Data descriptors defined here:

|
| __func__

|
| -----
| Data and other attributes defined here:

|
| __new__ = <built-in method __new__ of type object>
|     T.__new__(S, ...) -> a new object with type S, a subtype of T

```

class method는 첫번째 인자로 `cls`라는 클래스 자체를 넣어주어야 합니다. `Class object` 절에서 보셨듯이 `self`를 사용하는 것이 아닙니다.

이제 class method의 예제를 보겠습니다. `aclasmeth` 메소드 함수가 `cls`를 인자로 정의되었습니다. 그런데 인스턴스의 `aclasmeth`을 호출할 때나 클래스의 이름으로 `aclasmeth`을 호출할 때나 인자로 `cls`를 넣어주지는 않습니다. 그래도 메소드에서 클래스의 이름을 읽어 올 수 있었습니다.

static method에서는 이렇게 할 수가 없습니다.⁴

⁴아니면 어쩌지?

Decorator

Decorator

Syntax

```
@decorator
def function(arg1, arg2, ... argN):
    statements
    return value
-----
def function(arg1, arg2, ... argN):
    statements
    return value
function = decorator(function)
```

Example

```
@trace
def square(x):
    return x*x

enable_tracing = True
if enable_tracing:
    debug_log = open("debug.log", "w")

def trace(func):
    if enable_tracing:
        def callf(*args, **kwargs):
            debug_log.write("Calling %s: %s, %s\n" % (func.__name__, args, kwargs))
            r = func(*args, **kwargs)
            debug_log.write("%s returned %s\n" % (func.__name__, r))
            return r
        return callf
    else:
        return func
```

26 The content in this publication is subject to change, at any time, without obligation to notify.

Proprietary & Confidential 

Decorator는 일종의 [Syntactic sugar](#)로서 코드의 Readability를 높여주는 문법 요소입니다. 슬라이드의 Syntax에서 보이듯이 중간 구분선 위의 코드는 아래의 코드와 동일한 의미를 갖게 됩니다. 그래서 wrapper function을 만든다거나 static method와 class method를 만들 때 사용할 수 있습니다. wrapper function을 만드는 것은 슬라이드의 **Example**을 참고하시면 됩니다. static method와 class method를 만드는 방법은 [Static and Class methods](#static-class-method)에서 보인 예를 가지고 살펴 보겠습니다.

먼저 static method는 아래와 같이 만들었습니다.

```
class AClass(object):
    def astatic( ):
        print("a static method")
    astatic = staticmethod(staticmethod)
```

이 코드를 Decorator를 사용하여 아래와 같이 바꿀 수 있습니다.

```
class AClass(object):
    @staticmethod
    def astatic( ):
        print("a static method")
```

왠지 astatic 함수가 static method라는 사실이 더 분명히 와닿는 것 같습니다.

class method도 같은 방식으로 고쳐 쓸 수 있습니다.

```
class ABase(object):
    @classmethod
    def aclassmet(cls):
        print("a class method for %s" % cls.__name__)

class ADeriv(ABase):
    pass
```