http://tw.pycon.org/
2013 Taiwan
PYCON

PyCon Taiwan 2013 Tutorial

林信良（Justin Lin） 
caterpillar@openhome.cc
http://openhome.cc

Course Objectives

- Learning Python ecosystem
  - languages, tools, libraries…
- Understanding core culture of Python communities
  - coding styles, paradigms, documents, communities …
- Making a connection with PyConTW 2013

Instructor ?

Justin Lin

Technology Evangelist

Taiwan | Information Technology and Services

Current	**Technology Evangelist** at **Free lancer**
Past	Consultant at Sun Microsystems
	Technical Writer, Trainer, Consultant at Free Lancer
	Deputed Manager at Zong Chin Technology Corporation
Education	National Taiwan University
Connections	**45** connections
Websites	OpenHome.cc
	eGossip (beta)

Justin Lin's Summary

- Technical writing experience since 1999.
- Java programming experience since 2002.
- Training experience since 2005.
- Research interests include programming languages, web-related open source framework.
- Online documents covers a number of areas in C/C + +, Java, Scala, Ruby / Rails, Python, JavaScript, etc.

LinkedIn http://www.linkedin.com/in/caterpillar

Student ?

- PyCon Taiwan 2013 Tutorial Invitation  收件匣 x

---

" . . .

對我而言，要瞭解語言後的文化與生態系，約莫是三到六個月的時間，若以我至三月中前對 Python 生態系的瞭解過程與心得，配合 PyConTW 的議程，將之濃縮為六個小時的課程，你覺得如何？

. . . "

(Understanding cultures and ecosystem of a language takes me about three to six months. How about wrapping up what I have learned from Python ecosystem before mid-March and considering the agenda of PyConTW to build up a six-hour course?)

Schedule

- The 1st class
  - Preface (currently here)
  - Picking and installing an Interpreter
    - Implementations
    - Preparing course environment
    - Where're my libraries?
    - What's the relationship among distutils, Distribute and Pip?
  - Hello! World!
    - Introduction to Unicode Support
    - Basic Input and Output
  - Integrated Development Environment
  - Reference

- The 2nd class
  - Learning Python language
  - Built-in Types
    - Numerical Types
    - String Type
    - List Type
    - Set type
    - Dict Type
    - Tuple Type
  - `if`, `for`, `while` and `for` Comprehensions
    - `if..else`
    - `for` and `while`
    - `for` Comprehensions
  - Functions, Modules, Classes and Packages
    - Functions
    - Modules
    - Classes
    - Packages
  - References

- The 3rd class
  - The Community
  - Documentation
    - DocStrings
    - Official Documentation
    - PyDoc
    - EpyDoc
  - Data Management Functions
    - Built-in Functions
    - `reduce`
  - Persistence
    - `marshal`, `pickle`, `cPickle`
    - DBM
    - `shelve`
    - DB-API 2.0 (PEP 249)
  - References

- The 4th class:
  - Libraries vs Frameworks
    - Inversion of Control
    - Do We Need a Framework?
  - Getting Started with Django
    - Creating a Project
    - Creating a Database and an App
    - Playing API with the Python shell
  - Writing Your First View
    - Controller? or Views?
    - URLconf
  - References

- The 5th class
  - ToDo…
  - Using the Template System
  - Removing Hardcoded URLs in Templates

Picking and installing an Interpreter

- 2.x vs 3.x
  - Python 3.0 (a.k.a. "Python 3000" or "Py3k") final was released on **December 3rd, 2008**.
  - Python 3.3.0 was released on September 29th, 2012.
  - Python 2.7.3 was released on **April 9, 2012**.
  - **Python 2.7.x is highly recommended** unless you have a strong reason not to.
  - As more and more modules get ported over to Python3, the easier it will be for others to use it.

Implementations

- **CPython** (http://www.python.org)
  - Written in C.
  - Compiles Python code to intermediate bytecode.
  - Provides **the highest level of compatibility** with Python packages and C extension modules.
- PyPy (http://pypy.org)
  - Features a **JIT** (just-in-time) compiler.
  - Aims for maximum compatibility with the reference CPython implementation while **improving performance**.

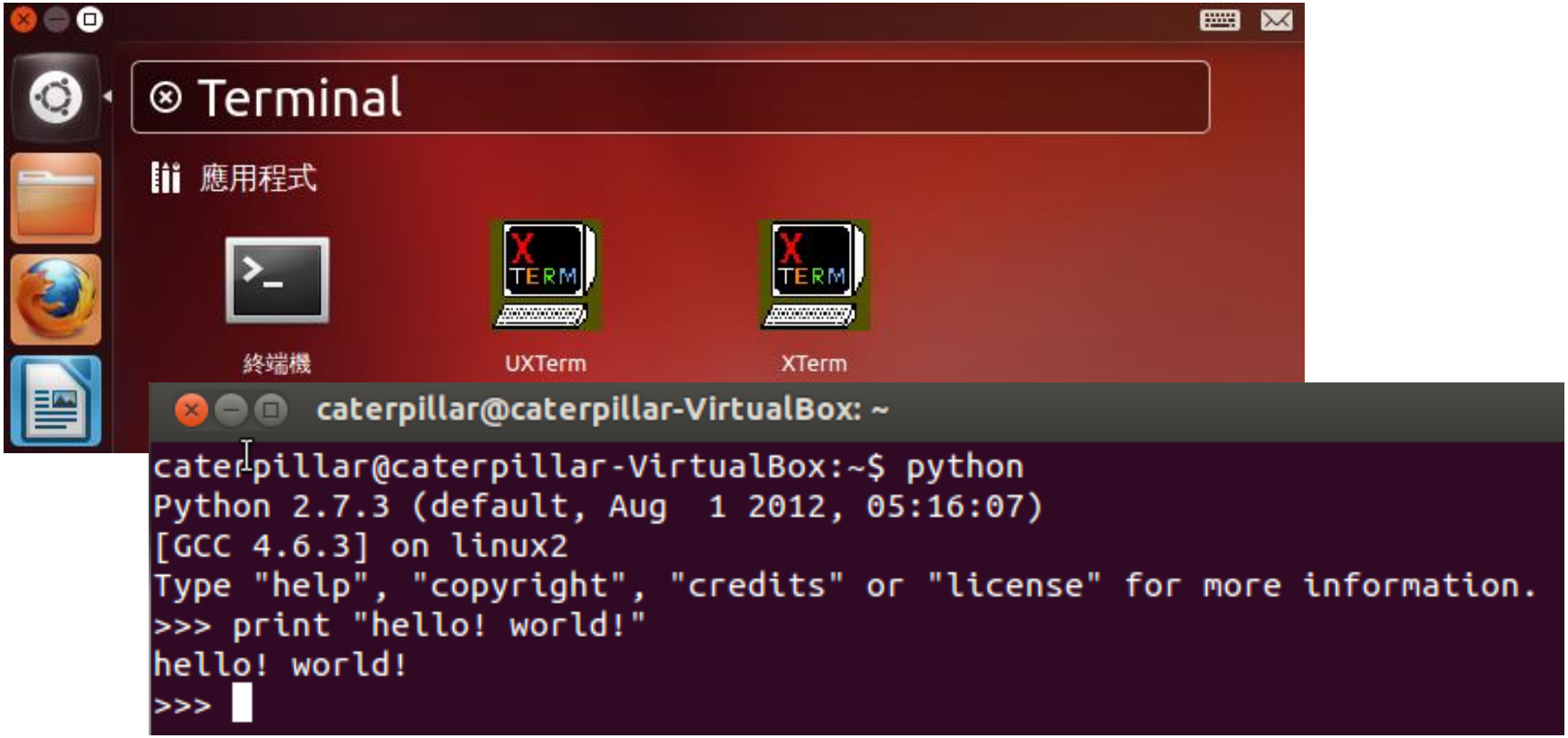
- Jython (http://www.jython.org/)
  - An implementation of Python for the **JVM**.
  - Compiles Python code to **Java byte code.**
  - Can import and use any Java class the same as a Python module.
- IronPython (http://ironpython.net/)
  - An open-source implementation of the Python programming language which is tightly integrated with the **.NET Framework**.
  - Can use the .NET Framework and Python libraries.
  - Other .NET languages can use Python code just as easily.

Preparing Course Environment

- Ubuntu 12.04 LTS
- **Python 2.7.3**
  - Distribute
  - Pip
  - Virtualenv

Exercise 0

- **Installing Python 2.7.3**
- Ubuntu 12.04 comes with Python 2.7.3 out of the box.
- All you have to do is to open a terminal and `python`!



Exercise 1

- **Installing Distribute, Pip and Virtualenv**
- **Distribute** extends the packaging and installation facilities provided by the **distutils** in the standard library.
  - run the python script available below:
    http://python-distribute.org/distribute_setup.py

```
~$ mkdir scripts
~$ cd scripts
~/scripts$ wget http://python-distribute.org/distribute_setup.py
~/scripts$ sudo python distribute_setup.py
```

What You Should See

```
caterpillar@caterpillar-VirtualBox: ~/scripts
caterpillar@caterpillar-VirtualBox:~$ mkdir scripts
caterpillar@caterpillar-VirtualBox:~$ cd scripts
caterpillar@caterpillar-VirtualBox:~/scripts$ wget http://python-distribute.org/
distribute_setup.py
--2013-01-24 11:08:35--  http://python-distribute.org/distribute_setup.py
正在查找主機 python-distribute.org (python-distribute.org)... 88.191.140.69
正在連接 python-distribute.org (python-distribute.org)|88.191.140.69|:80... 連上
了。
已送出 HTTP 要求，正在等候回應... 200 OK
長度： 17319 (17K) [text/x-python]
Saving to: `distribute_setup.py'

100%[======================================>] 17,319      54.7K/s   in 0.3s

2013-01-24 11:08:37 (54.7 KB/s) - `distribute_setup.py' saved [17319/17319]

caterpillar@caterpillar-VirtualBox:~/scripts$ sudo python distribute_setup.py
[sudo] password for caterpillar:
```

~

```
Installed /usr/local/lib/python2.7/dist-packages/distribute-0.6.34-py2.7.egg
Processing dependencies for distribute==0.6.34
Finished processing dependencies for distribute==0.6.34
After install bootstrap.
Creating /usr/local/lib/python2.7/dist-packages/setuptools-0.6c11-py2.7.egg-info
Creating /usr/local/lib/python2.7/dist-packages/setuptools.pth
caterpillar@caterpillar-VirtualBox:~/scripts$
```

- The new``easy_install`` command you have available is considered by many to be deprecated, so we will install its replacement: **pip**.
- The **virtualenv** kit provides the ability to create virtual Python environments that do not interfere with either each other, or the main Python installation.

```
~/scripts$ sudo easy_install pip
~/scripts$ sudo pip install virtualenv
```

What You Should See

```
caterpillar@caterpillar-VirtualBox: ~/scripts
caterpillar@caterpillar-VirtualBox:~/scripts$ sudo easy_install pip
[sudo] password for caterpillar:
Searching for pip
Reading http://pypi.python.org/simple/pip/
Reading http://www.pip-installer.org
```

~

```
Finished processing dependencies for pip
caterpillar@caterpillar-VirtualBox:~/scripts$ sudo pip install virtualenv
Downloading/unpacking virtualenv
  Downloading virtualenv-1.8.4.tar.gz (1.9MB): 1.9MB downloaded
  Running setup.py egg_info for package virtualenv

    warning: no previously-included files matching '*' found under directory 'do
cs/_templates'
    warning: no previously-included files matching '*' found under directory 'do
cs/_build'
Installing collected packages: virtualenv
  Running setup.py install for virtualenv

    warning: no previously-included files matching '*' found under directory 'do
cs/_templates'
    warning: no previously-included files matching '*' found under directory 'do
cs/_build'
    Installing virtualenv script to /usr/local/bin
    Installing virtualenv-2.7 script to /usr/local/bin
Successfully installed virtualenv
Cleaning up...
caterpillar@caterpillar-VirtualBox:~/scripts$
```

Where’re my libraries?

- The **`sys.path`** is a list of strings that specifies the search path for modules.
- Use the environment variable **`PYTHONPATH`** to augment the default search path for module files.

```
caterpillar@caterpillar-VirtualBox: ~
caterpillar@caterpillar-VirtualBox:~$ export PYTHONPATH=~/scripts
caterpillar@caterpillar-VirtualBox:~$ python
Python 2.7.3 (default, Aug  1 2012, 05:16:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.path
['', '/usr/local/lib/python2.7/dist-packages/distribute-0.6.34-py2.7.egg', '/usr
/local/lib/python2.7/dist-packages/pip-1.2.1-py2.7.egg', '/home/caterpillar/scri
pts', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-linux2', '/usr/lib/python2.
7/lib-tk', '/usr/lib/python2.7/lib-old', '/usr/lib/python2.7/lib-dynload', '/usr
/local/lib/python2.7/dist-packages', '/usr/local/lib/python2.7/dist-packages/set
uptools-0.6c11-py2.7.egg-info', '/usr/lib/python2.7/dist-packages', '/usr/lib/py
thon2.7/dist-packages/PIL', '/usr/lib/python2.7/dist-packages/gst-0.10', '/usr/l
ib/python2.7/dist-packages/gtk-2.0', '/usr/lib/python2.7/dist-packages/ubuntu-ss
o-client', '/usr/lib/python2.7/dist-packages/ubuntuone-client', '/usr/lib/python
2.7/dist-packages/ubuntuone-control-panel', '/usr/lib/python2.7/dist-packages/ub
untuone-couch', '/usr/lib/python2.7/dist-packages/ubuntuone-installer', '/usr/li
b/python2.7/dist-packages/ubuntuone-storage-protocol']
>>>
```

What's the relationship among Distutils, Setuptools, Distribute and Pip?

- Distutils
  - The Python standard library for building and installing additional modules.
  - For simple installation scenarios.
  - Basic steps:
    - Untar the downloaded file (e.g. tar xzvf Django-X.Y.tar.gz)
    - Change into the directory. Basically, all you need is **setup.py**.
    - `sudo python setup.py install`
- Setuptools
  - Extends distutils.
  - De facto standard of Python community.
  - Has problems of slow development, messy code…

- Distribute
  - Extends distutils.
  - Intended to **replace Setuptools** as the standard method for working with Python module distributions.
  - Providing **a backward compatible** version to replace Setuptools and make all distributions that depend on Setuptools work as before.
  - So, once setuptools or distribute is installed, `easy_install` is prepared.
  - The `easy_install` command is considered by many to be deprecated due to lack of unstallation command, svn-only support…
- Pip
  - **An `easy_install` replacement.**
  - Allows for uninstallation of packages, and is actively maintained, unlike `easy_install`.
  - Virtualenv is its good partner.
  - Basic commands:
    - `pip install [PACKAGE_NAME]`
    - `pip unstall [PACKAGE_NAME]`

Hello! World!

- The **virtualenv** kit provides the ability to create virtual Python environments that do not interfere with either each other, or the main Python installation.
- Create a virtual Python environment:
  - `virtualenv --distribute venv`
- Activate the environment:
  - `source bin/activate`
- Deactivate the environment:
  - `deactivate`

Exercise 2

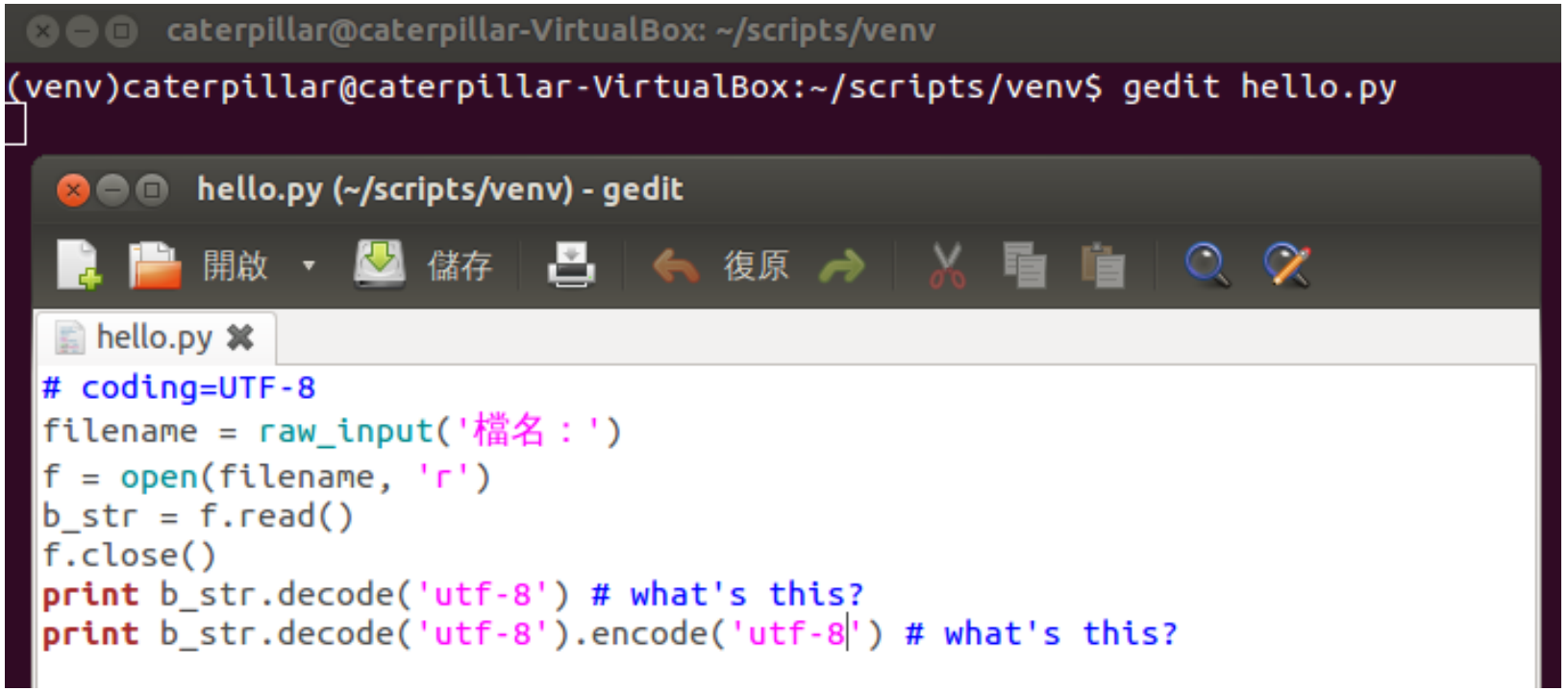
- **Create and activate a virtual Python environment.**
- **Prompt a use to provide a filename, read the file and print the content in the terminal. Consider the character encoding problems.**

```
~/scripts$ virtualenv --distribute venv
~/scripts$ cd venv
~/scripts/venv$ source bin/activate
```

What You Should See

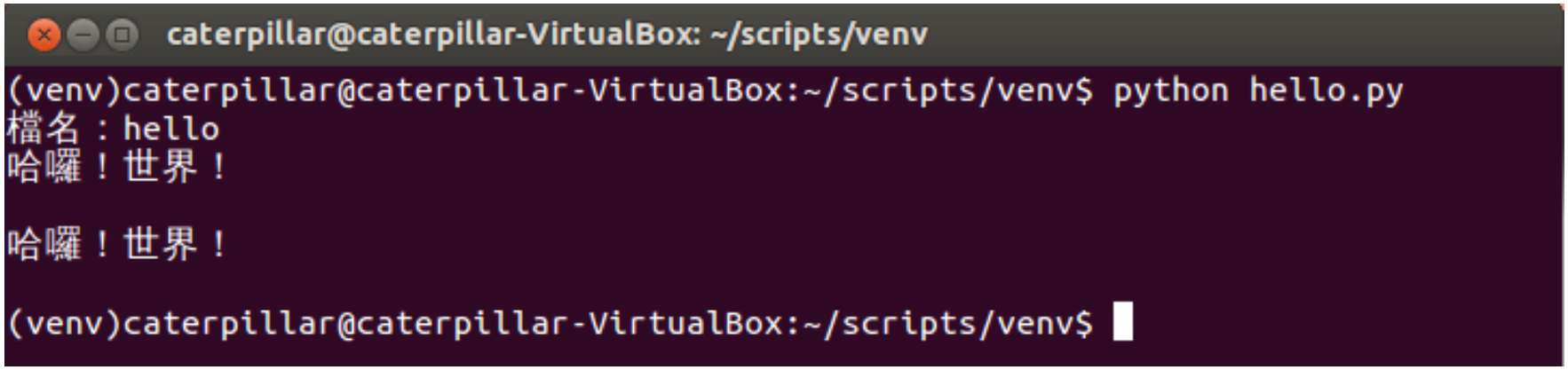
```
caterpillar@caterpillar-VirtualBox:~/scripts$ virtualenv --distribute venv
New python executable in venv/bin/python
Installing distribute.....................................................
.....................................................................
.....................................................................
done.
Installing pip.................done.
caterpillar@caterpillar-VirtualBox:~/scripts$ cd venv
caterpillar@caterpillar-VirtualBox:~/scripts/venv$ source bin/activate
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv$
```

~/scripts/venv$ **gedit hello.py**



~/scripts/venv$ **gedit hello**

What You Should See



Introduction to Unicode Support

- Default encoding of Ubuntu: UTF-8.
- Python 2:
  - Strings are actual byte sequence representing the data.

```
# coding=UTF-8
text = '測試'
print len(text) # print "6"
```

encoding declaration

  - Unicode literals are written as strings prefixed with the `'u'` or `'U'` character

```
# coding=UTF-8
text = u'測試'
print type(text) # print "<type 'unicode'>"
print len(text)  # print "2"
```

- Python 2:
  - `decode` interprets the string using the given encoding and returns a `unicode` instance.
  - `encode` returns an 8-bit string version of the Unicode string.
- Python 3: Unicode by default.
  - `decode` returns a `bytes` instance representing byte sequence.
  - `encode` returns a `str` instance representing the Unicode string.

```
>>> '元'.encode('big5')
b'\xa4\xb8'
>>> '元'.encode('utf-8')
b'\xe5\x85\x83'
>>> '元'.encode('big5').decode('big5')
'元'
>>>
```

Basic Input and Output

- Read a file:

```
import sys
file = open(sys.argv[1], 'r')
content = file.read()
print content
file.close()
```

Import a module

Command line arguments

- Write a file:

```
import sys
file = open(sys.argv[1], 'w')
file.write('test')
file.close()
```

- Three ways for reading all content in a file:

```
import sys
file = open(sys.argv[1], 'r')
while True:
    line = file.readline()
    if not line: break
    print line
file.close()
```

```
import sys
file = open(sys.argv[1], 'r')
for line in file.readlines():
    print line
file.close()
```

```
import sys
for line in open(sys.argv[1], 'r'):
    print line
```

Integrated Development Environment

- Sometimes, it‘s just the problem of flavor.
  - PyCharm / IntelliJ IDEA
    - http://www.jetbrains.com/pycharm/
  - PyDev / Eclipse plugin
    - http://pydev.org/
  - Komodo IDE
    - http://www.activestate.com/komodo-ide
  - Spyder
    - http://code.google.com/p/spyderlib/
  - WingIDE
    - http://wingware.com/
  - NINJA-IDE
    - http://www.ninja-ide.org/
  - Python Tools for Visual Studio
    - http://pytools.codeplex.com/

References

- Implementations
  - http://www.python.org/download/releases/3.0/
  - http://www.python.org/download/releases/2.7.3/
  - http://docs.python-guide.org/en/latest/starting/which-python/
- Preparing course environment
  - http://docs.python-guide.org/en/latest/starting/install/linux/
- Where’re my libraries?
  - http://docs.python.org/2/using/cmdline.html
- What’s the relationship among distutils, Distribute and Pip?
  - http://docs.python.org/2/library/distutils.html
  - http://pypi.python.org/pypi/distribute
  - http://pypi.python.org/pypi/pip
  - http://blog.yangyubo.com/2012/07/27/python-packaging/
  - http://www.openfoundry.org/tw/tech-column/8536-introduction-of-python-extension-management-tools
- Hello! World!
  - http://caterpillar.onlyfun.net/Gossip/Python/IOABC.html
  - http://caterpillar.onlyfun.net/Gossip/Encoding/
  - http://caterpillar.onlyfun.net/Gossip/Encoding/Python.html

Learning Python Language

- What’re the most essential elements of a language?

> ***Algorithms + Data Structures = Programs***
>
> *-- Niklaus E. Wirth -- The chief designer of Pascal*

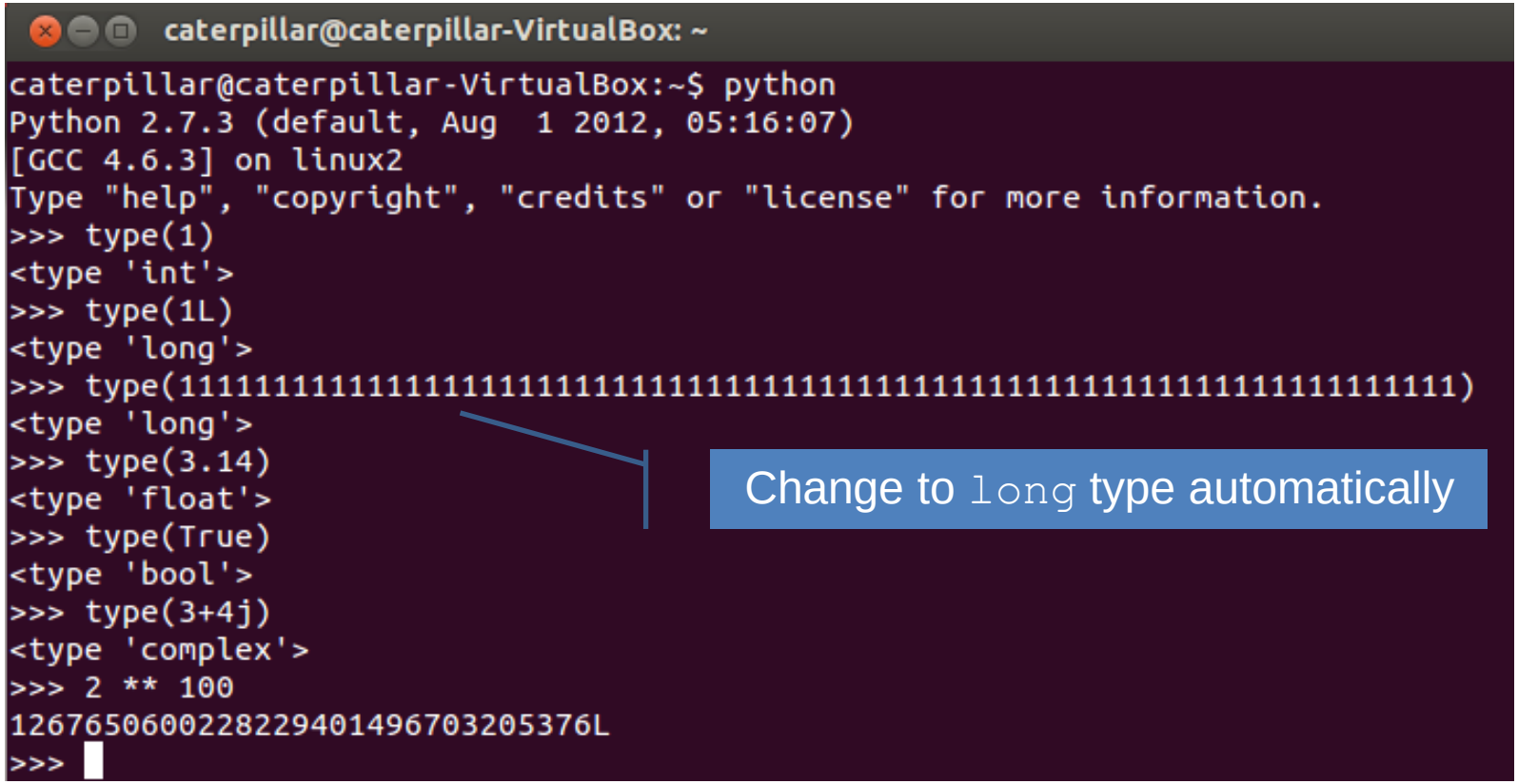
- How to encapsulate your code?
- Focus on the essence of Python, not nuts and bolts.
  - Built-in types, variables and operators
  - Functions, classes and modules

Built-in Types

- **Every thing is an object.**
  - Python, however, does not impose object-oriented programming as the main programming paradigm.
- Numerical types
  - `int, long, float, bool, complex`
- String type
- Container types
  - `list, set, dict, tuple`

Numerical Types

- `int, long, float, bool, complex`
- The `type` function returns the type of any object.



What You Should Know

- Python float division:

Different results in different versions

```
>>> 10 / 3
3
>>> 10 // 3
3
>>> 10 / 3.0
3.3333333333333335
>>> 10 // 3.0
3.0
>>>
```

- Float decision, `repr` and `str`:

```
>>> 1.0 - 0.8
0.19999999999999996
>>> print(1.0 - 0.8)
0.2
>>> repr(1.0 - 0.8)
'0.19999999999999996'
>>> str(1.0 - 0.8)
'0.2'
>>> import decimal
>>> a = decimal.Decimal('1.0')
>>> b = decimal.Decimal('0.8')
>>> a - b
Decimal('0.2')
>>>
```

Call `__repr__` function of an object

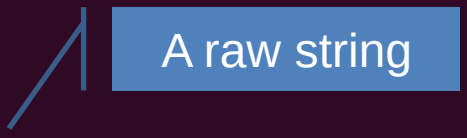
Call `__str__` function of an object

- `__repr__` computes the “official” string representation of an object.
- `__str__` compute the “informal” string representation of an object.
- **`__repr__` is to be unambigous and `__str__` is to be readable.**
- The `decimal` module provides support for decimal floating point arithmetic.

String Type

- `''` and `""` are the same in Python and replaceable.
- Use a raw string if you want to represent `'\'` itself.

```
>>> "Just'in"
"Just'in"
>>> 'Just"in'
'Just"in'
>>> 'c:\workspace'
'c:\\workspace'
>>> r'c:\workspace'
'c:\\workspace'
>>> 'c:\todo'
'c:\todo'
>>> r'c:\todo'
'c:\\todo'
>>> print 'c:\todo'
c:      odo
>>> print r'c:\todo'
c:\todo
>>>
```

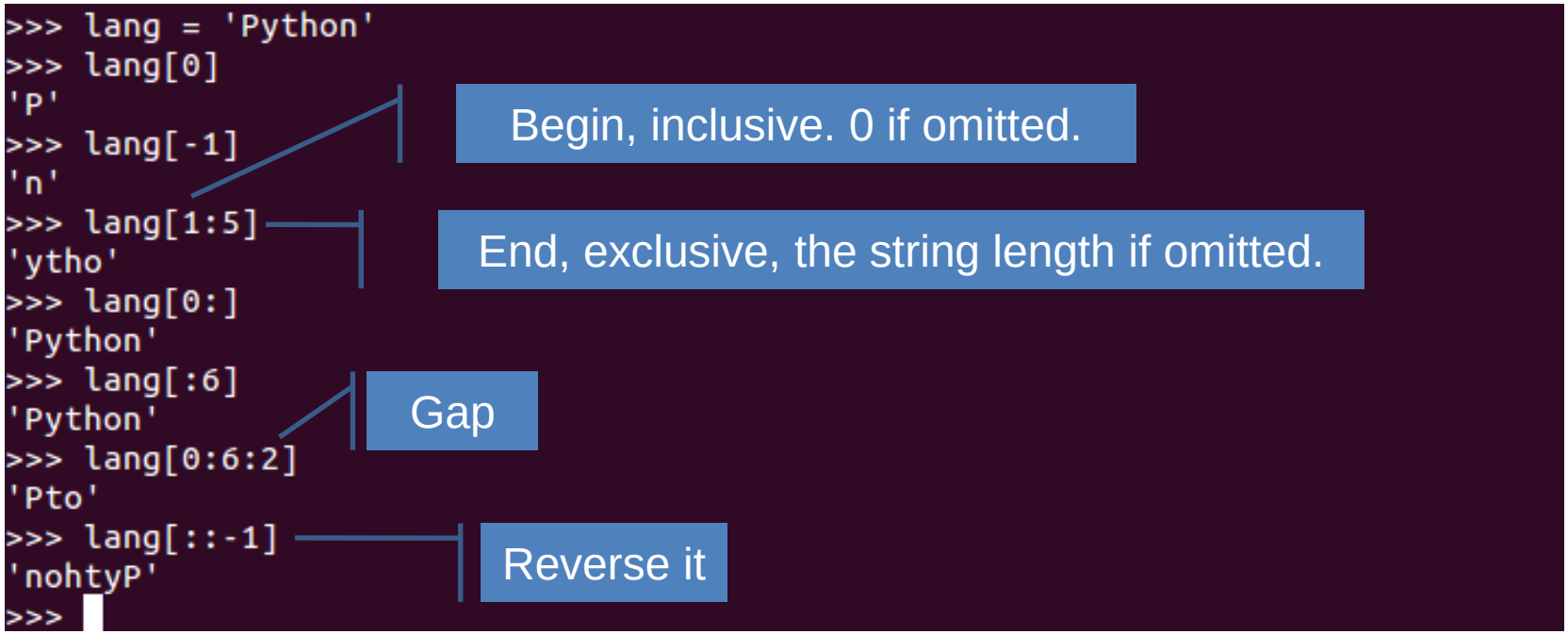


- A string is **immutable**.
- `len` returns the string length. Use `for` to iterate a string. `in` tests if a string contains a substring. `+` is for concatenating two strings. `*` replicates a string.

```
>>> name = 'Justin'
>>> len(name)
6
>>> for c in name:
...     print c
...
J
u
s
t
i
n
>>> 'Just' in name
True
>>> name + name
'JustinJustin'
>>> name * 3
'JustinJustinJustin'
>>>
```

String Slicing

- `[]` can specified an index to get a character from a string. A negative index is counted from the last element.
- **The most useful power of `[]` is slicing.**



String Formatting

- Old String Formatting Operations

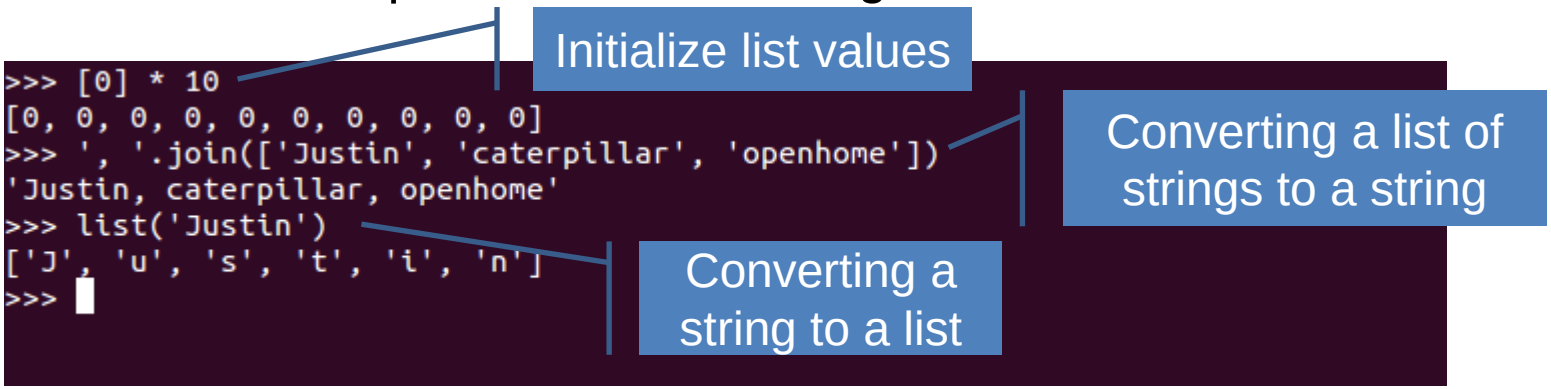
```
>>> '%d %.2f %s' % (1, 99.3, 'Justin')
'1 99.30 Justin'
>>> '%(real)s is %(nick)s!!' % {'real' : 'Justin', 'nick' : 'caterpillar'}
'Justin is caterpillar!!'
>>>
```

- New String Formatting Operations (after Python 2.6)

```
>>> '{0} is {1}!!'.format('Justin', 'caterpillar')
'Justin is caterpillar!!'
>>> '{real} is {nick}!!'.format(nick = 'caterpillar', real = 'Justin')
'Justin is caterpillar!!'
>>> '{0} is {nick}!!'.format('Justin', nick = 'caterpillar')
'Justin is caterpillar!!'
>>> import sys
>>> 'My platform is {pc.platform}'.format(pc = sys)
'My platform is linux2'
>>>
```

List Type

- An ordered and **mutable** collection.
  - `[1, 2, 3]` creates a list with elements `1`, `2`, and `3` in the index `0`, `1` and `2`.
- Shares common operations with strings.
  - `len` returns the list length. Use `for` to iterate a list. `in` tests if a list contains an element. `+` is for concatenating two lists. `*` replicates a list.
  - `[]` can specified an index to get a character from a string. A negative index is counted from the last element.
  - The most useful power of `[]` is slicing.



Set Type

- A unordered collection. Contains no duplicate elements.
- Elements should be **immutable**.

```
>>> admins = {'Justin', 'caterpillar'}
>>> users = {'momor', 'hamini', 'Justin'}
>>> 'Justin' in admins
True
>>> admins & users
set(['Justin'])
>>> admins | users
set(['hamini', 'caterpillar', 'Justin', 'momor'])
>>> admins - users
set(['caterpillar'])
>>> admins ^ users
set(['hamini', 'caterpillar', 'momor'])
>>> admins > users
False
>>> admins < users
False
>>>
```

Exclusive or

∈

Dict Type

- An object that maps keys to values.

```
>>> passwords = {'Justin' : 123456, 'caterpillar' : 933933}
>>> passwords['Justin']
123456
>>> passwords['Hamimi'] = 970221
>>> passwords
{'caterpillar': 933933, 'Hamimi': 970221, 'Justin': 123456}
>>> del passwords['caterpillar']
>>> passwords
{'Hamimi': 970221, 'Justin': 123456}
>>> passwords.items()
[('Hamimi', 970221), ('Justin', 123456)]
>>> passwords.keys()
['Hamimi', 'Justin']
>>> passwords.values()
[970221, 123456]
>>> passwords.get('openhome', '000000')
'000000'
>>> passwords['openhome']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'openhome'
>>>
```

A tuple

```
if 'openhome' in passwords:
    return passwords['openhome']
else:
    return '000000'
```

Tuple Type

- A tuple is like a list, yet it's **immutable**.
- Shares common operations with lists.
  - In fact, sequences in Python (e.g. strings, lists, tuples, etc.) shares several features.
- Mutable or immutable? We'll talk about it soon…
- (In Haskell - a statically-typed language - the types of elements in a tuple composes an unnamed type.)

Exercise 3

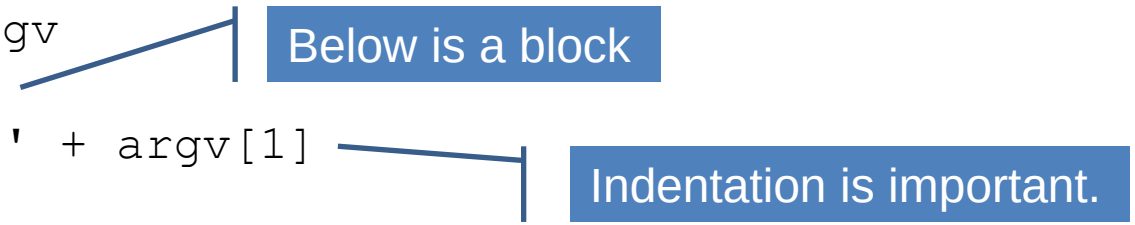
- Open a terminal and type `python`. What will you see in **the interactive shell** if you type the following commands?
  - `1 + 2`
  - `_`
  - `_ + 3`
  - `help()`
  - `len`
  - `keywords`
  - `quit`(or simply `q`)
  - `help(len)`
  - Ctrl + D

- After exiting the interactive shell, what will you see in the terminal if you type the following commands?
  - `python -h`
  - `python -c 'print "Hello! Python!"'`
  - `python -c 'help(len)'`
  - `python -c 'import this'`
- (Try anything you see from the previous slides about built-in types.)

`if`, `for`, `while` and `for` comprehensions

- `if..else` block

```
from sys import argv
if len(argv) != 1:
    print 'Hello, ' + argv[1]
else:
    print 'Hello, Guest'
```



- `if..else` expression, something like the ternary operator `?:` in C or Java.

```
from sys import argv
print 'Hello, ' + (argv[1] if len(argv) != 1 else 'Guest')
```

`for` and `while`

- Use `for in` to iterate a sequence.

```
numbers = [10, 20, 30]
squares = []
for number in numbers:
    squares.append(number ** 2)
print squares
```

- Use `while` for undetermined conditions.

```
print 'Enter two numbers...'
m = int(raw_input('Number 1: '))
n = int(raw_input('Number 2: '))
while n != 0:
   r = m % n
   m = n
   n = r
print 'GCD: {0}'.format(m)
```

`for` comprehensions

- With a list comprehension we can turn this:

```
numbers = [10, 20, 30]
squares = []
for number in numbers:
    squares.append(number ** 2)
print squares
```

- Into this:

```
numbers = [10, 20, 30]
print [number ** 2 for number in numbers]
```

- With a list comprehension we can turn this:

```
numbers = [11, 2, 45, 1, 6, 3, 7, 8, 9]
odd_numbers = []
for number in numbers:
    if number % 2 != 0:
        odd_numbers.append(number)
print odd_numbers
```

- Into this:

```
numbers = [11, 2, 45, 1, 6, 3, 7, 8, 9]
print [number for number in numbers if number % 2 != 0]
```

- Flatten a list of lists.

```
lts = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print [ele for lt in lts for ele in lt]
```

- A set comprehension

```
>>> {name for name in ["caterpillar", "Justin", "caterpillar", "openhome"]}
set(['caterpillar', 'Justin', 'openhome'])
>>>
```

- A dict comprehension

```
>>> names = {'caterpillar', 'Justin', 'openhome'}
>>> passwds = {123456, 987654, 13579}
>>> {name: passwd for name, passwd in zip(names, passwds)}
{'caterpillar': 123456, 'openhome': 13579, 'Justin': 987654}
>>>
```

- (In Haskell, a set comprehension $S = \{ 2 \cdot x \mid x \in \mathbb{N},\ x \leq 10 \}$ in mathematics can be written as [2 * x | x <- N, x <= 10] which looks similar to the set comprehension.)

Exercise 4

- Turn the following code into a single statement.

```
numbers = []
for number in range(20):
    numbers.append(str(number))
print ", ".join(numbers)
```

- (Here's a problem that combines tuple and list comprehensions: which right triangle that has integers for all sides and all sides equal to or smaller than 10 has a perimeter of 24?)

Functions, Modules, Classes and Packages

- In Python, everything is an object.
  - ***Does Python impose object-oriented programming as the main programming paradigm?***
- Points about structuring your program.
  - Encapsulation and separation of abstraction layers.
  - State of an object.
  - Namespace
  - Physical structures of your resources, such as source files, packages, etc.

Functions

xmath.py

```
def max(a, b):
    return a if a > b else b

min = lambda a, b: a if a < b else b

def sum(*numbers):
    total = 0
    for number in numbers:
        total += number
    return total

maximum = max
minimum = min

pi = 3.141592653589793
e = 2.718281828459045
```

λ function
anonymous function

Variable arguments

Functions are first-class values.

Modules

- What's the best way to organize functions in the previous slide?
- Modules are one of the main abstraction layers available and probably the most natural one.
  - A file named modu.py creates a module `modu`.
  - The `import modu` statement will look for *modu.py* in the same. If it isn't found, the Python interpreter will search for modu.py in the `sys.path` recursively; or raise an ImportError exception if it isn't found.

- **A module provides a namespace.** The module's variables, functions, and classes will be available to the caller through the module's namespace
- `import`, `import as`, `from import` are statements.

main.py

```
import xmath
print '# import math'
print xmath.pi
print xmath.max(10, 5)
print xmath.sum(1, 2, 3, 4, 5)

print '# import xmath as math'
import xmath as math
print math.e

print '# from xmath import min'
from xmath import min
print min(10, 5)
```

Create an alias

Copy it into the current module.
`from modu import *` is not recommended.

```
# import math
3.14159265359
10
15
# import xmath as math
2.71828182846
# from xmath import min
5
```

Classes

- Well, where's the playground for classes?
  - When we want to glue together some **state** and some functionality.

bank.py

```
def account(name, number, balance):
    return {'name': name, 'number': number, 'balance': balance}

def deposit(acct, amount):
    if amount <= 0:
        raise ValueError('amount must be positive')
    acct['balance'] += amount

def withdraw(acct, amount):
    if amount > acct['balance']:
        raise RuntimeError('balance not enough')
    acct['balance'] -= amount

def to_str(acct):
    return 'Account' + str(acct)
```

main.py

```
import bank
acct = bank.account('Justin', '123-4567', 1000)
bank.deposit(acct, 500)
bank.withdraw(acct, 200)
print bank.to_str(acct)
```

- **OOP is considering usability more than reusability.**

bank.py

```
class Account:
    def __init__(self, name, number, balance):
        self.name = name
        self.number = number
        self.balance = balance

    def deposit(self, amount):
        if amount <= 0:
            raise ValueError('amount must be positive')
        self.balance += amount

    def withdraw(self, amount):
        if amount > self.balance:
            raise RuntimeError('balance not enough')
        self.balance -= amount

    def __str__(self):
        return 'Account({0}, {1}, {2})'.format(
            self.name, self.number, self.balance)
```

Initializer

Explicit is better than implicit.

Still remember differences between `__str__` and `__repr__`?

main.py

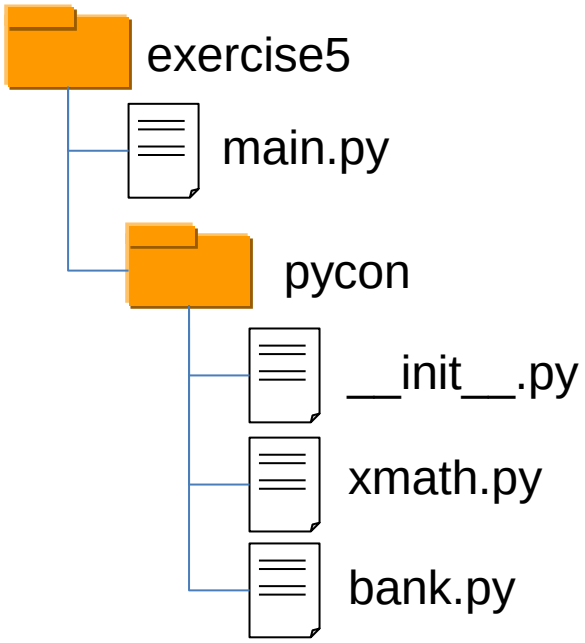
```
import bank
acct = bank.Account('Justin', '123-4567', 1000)
acct.deposit(500)
acct.withdraw(200)
print acct
```

Packages

- Any directory with an __init__.py file - used to gather all package-wide definitions - is considered a package.
- `import pack.modu` will looks for a file **modu.py** in the directory **pack**.
  - This statement will look for an __init__.py file in the directory pack, execute all of its top-level statements.
  - Then it will look for a file pack/modu.py and execute all of its top-level statements.
  - After these operations, any variable, function, or class defined in modu.py is available in the `pack.modu` namespace.

Exercise 5

- There's a quick and dirty **main.py** located in the **/exercises/exercise5** of the lab file. Use modules, classes and packages learned in the previous slides to structure them as follow:



What You Should See?

- Basically, you should have the following main.py and run it correctly.

main.py

```
import pycon.xmath as math
import pycon.bank as bank

print math.max(10, 5)
print math.sum(1, 2, 3, 4, 5)
print math.pi

acct = bank.Account('Justin', '123-4567', 1000)
acct.deposit(500)
acct.withdraw(200)
print acct
```

Python ▾ Tab

```
10
15
3.14159265359
Account(Justin, 123-4567, 1300)
```

References

- String Type
  - http://docs.python.org/2/reference/datamodel.html#object.__repr__
  - http://docs.python.org/py3k/library/stdtypes.html#old-string-formatting
  - http://docs.python.org/py3k/library/string.html#string-formatting
- List, Set, Dict, Tuple Types
  - http://caterpillar.onlyfun.net/Gossip/Python/ListType.html
  - http://caterpillar.onlyfun.net/Gossip/Python/SetType.html
  - http://caterpillar.onlyfun.net/Gossip/Python/DictionaryType.html
  - http://caterpillar.onlyfun.net/Gossip/Python/TupleType.html
- Functions, Modules, Classes and Packages
  - http://caterpillar.onlyfun.net/Gossip/Python/Class.html
  - http://caterpillar.onlyfun.net/Gossip/Python/Class.html
  - http://docs.python-guide.org/en/latest/writing/structure/
- Short Cuts
  - http://maxburstein.com/blog/python-shortcuts-for-the-python-beginner/

The Community

- BDFL
  - **Guido van Rossum** (www.python.org/~guido)
  - The creator of Python, is often referred to as the **Benevolent Dictator For Life**.
- PSF
  - **Python Software Foundation** (www.python.org/psf)
  - Its mission is to promote, protect, and advance the Python programming language, and to support and facilitate the growth of a diverse and international community of Python programmers.
  - A 501(c)(3) non-profit corporation that holds the intellectual property rights behind the Python programming language.

- PEPs
  - **Python Enhancement Proposals**（www.python.org/dev/peps）
  - Describe changes to Python itself, or the standards around it.
  - Notable PEPs
    - PEP 1 -- PEP Purpose and Guidelines.
    - PEP 8 -- Style Guide for Python Code
    - PEP 20 -- The Zen of Python
    - PEP 257 -- Docstring Conventions
- PyCon
  - **Python Conference**（www.pycon.org）
  - PyCon Taiwan（tw.pycon.org）
- PIGgies
  - **Python User Groups**（wiki.python.org/moin/LocalUserGroups）
  - Taiwan Python User Group（wiki.python.org.tw）

Documentation

- What happens if you type `len.__doc__` in the interactive shell?
- Remember `help`? What's the relationship between `help(len)` and `len.__doc__`?
- Where's `len.__doc__` from?

```
>>> len.__doc__
'len(object) -> integer\n\nReturn the number of items of a sequence or mapping.'
>>> help(len)
```

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

len(...)
    len(object) -> integer

    Return the number of items of a sequence or mapping.
(END)
```

Press 'q' to quit

DocStrings

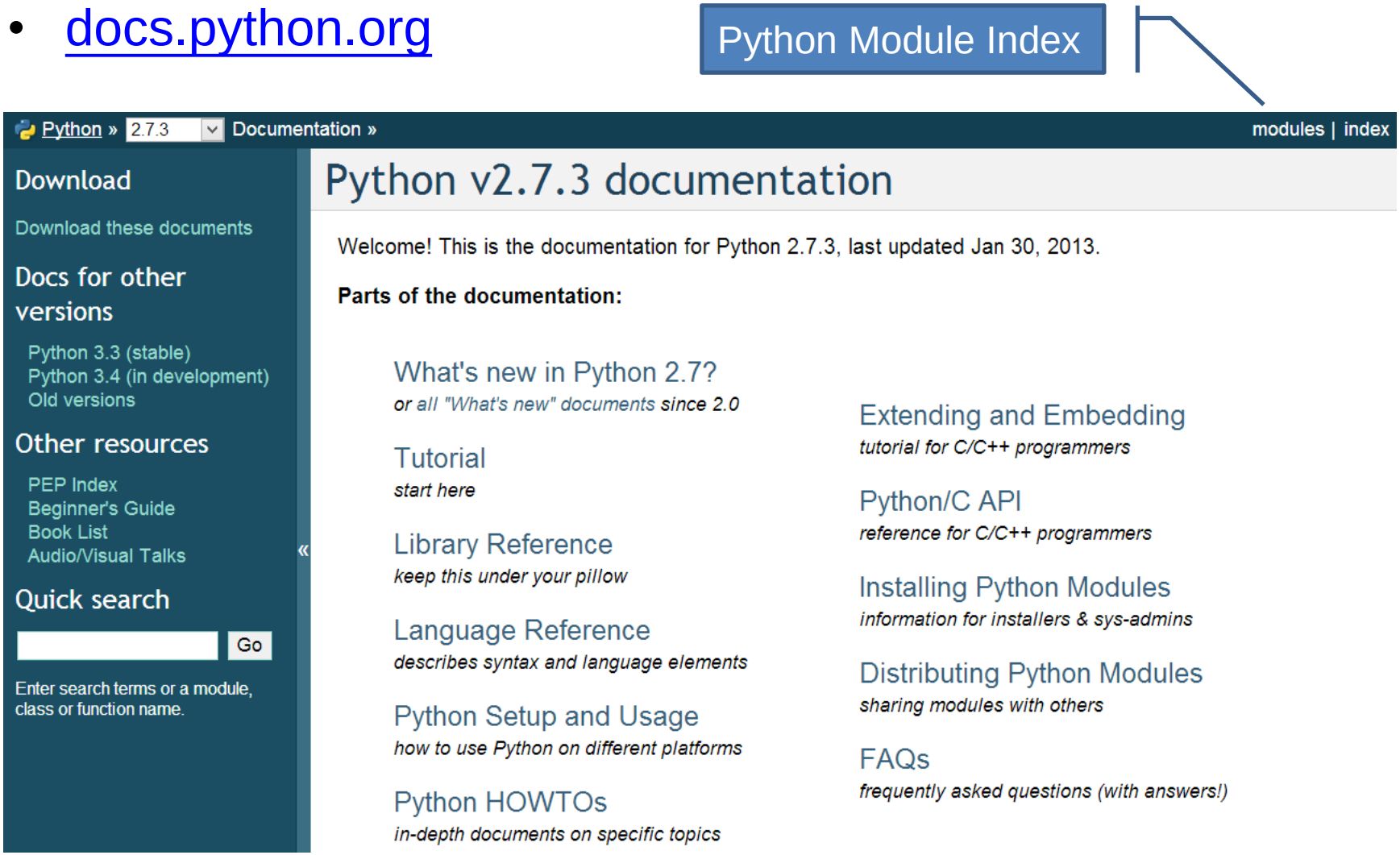
- Type the following code in the interactive shell.

```
def max(a, b):
    '''max(a, b) -> value

       With two arguments, return the largest argument.'''
    return a if a > b else b
```

- Type `max.__doc__` in the interactive shell.
- Type `help(max)` in the interactive shell.
- You'll know what DocStrings are.
- Remember to read **PEP 257** if you want to comply with **DocString Conventions**.

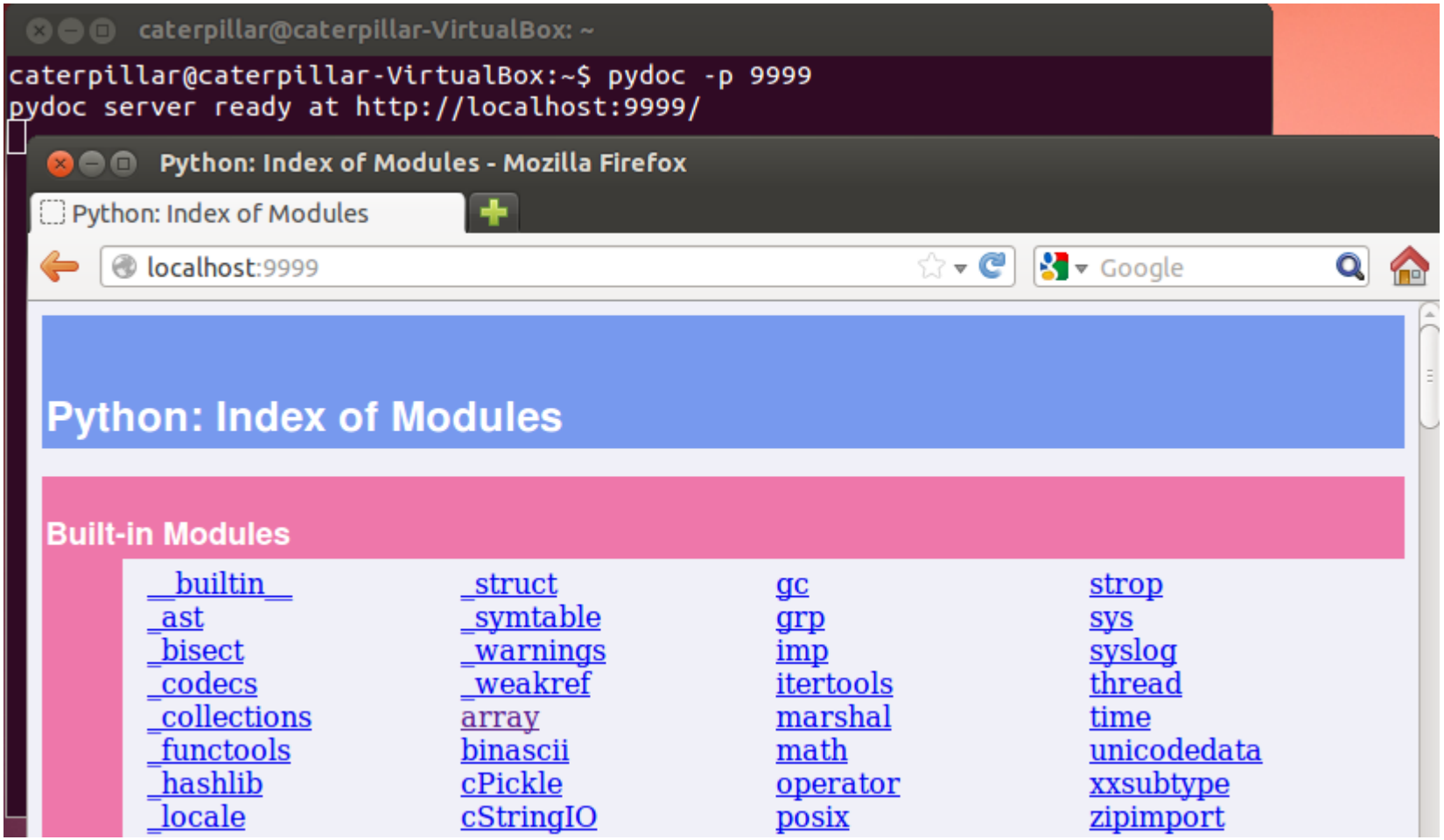
Official Documentation

- docs.python.org



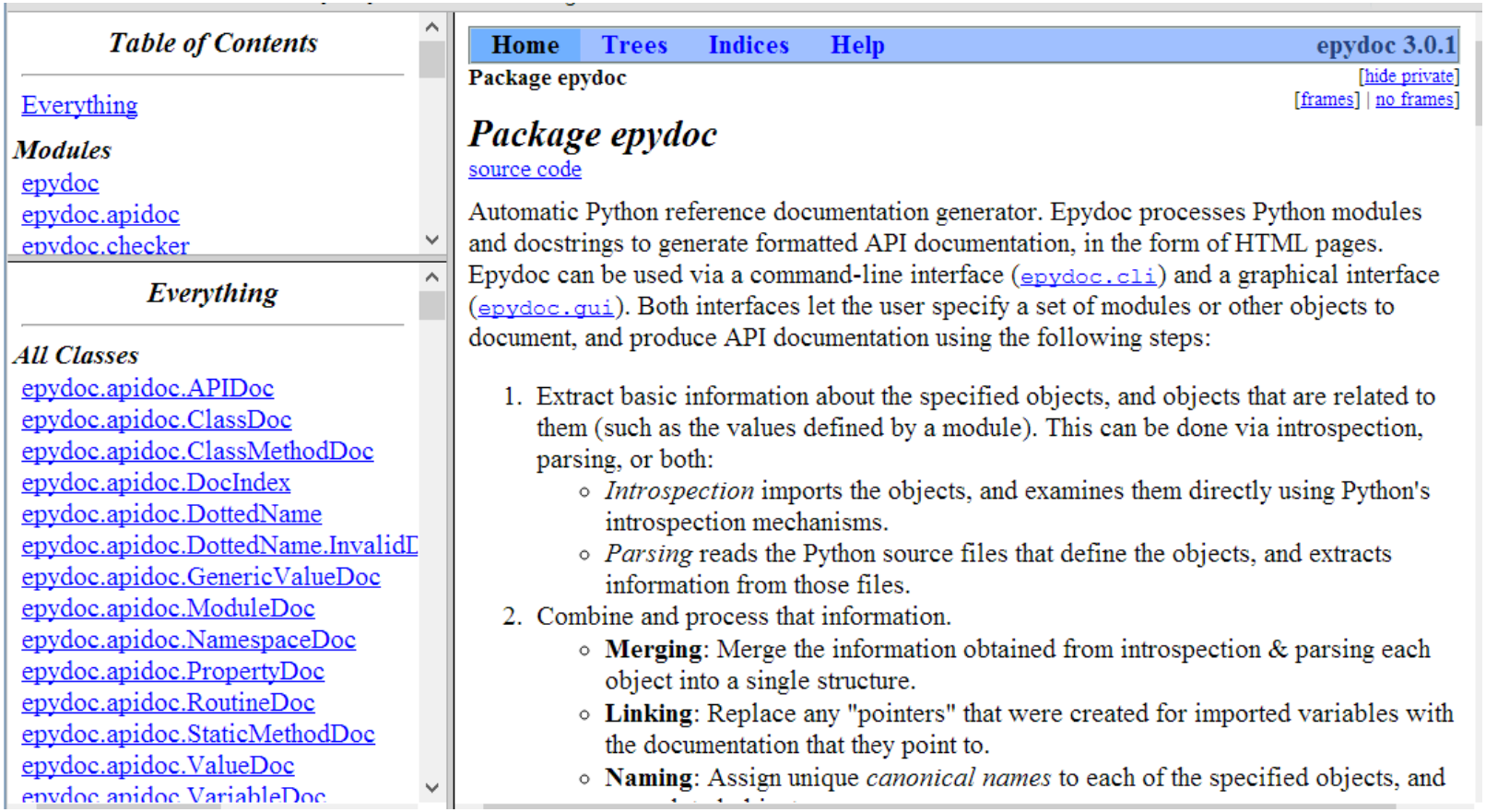
PyDoc

- The pydoc module automatically generates documentation from Python modules.



EpyDoc

- Looks for something like JavaDoc?
- epydoc.sourceforge.net



Data Management Functions

- Built-in Functions (located in the `__builtin__` module)
  - `range(start, stop[, step])`
  - `zip([iterable, ...])`
  - `enumerate(sequence, start=0)`
  - `reduce(function, iterable[, initializer])`

Exercise 6

- How to iterate through a list with an index? For examples, given a list `names = ['Justin', 'caterpillar', 'openhome']`, print the followings.

```
0, Justin
1, caterpillar
2, openhome
```

- Hints:
  - 1. Fill in the blanks with proper codes.

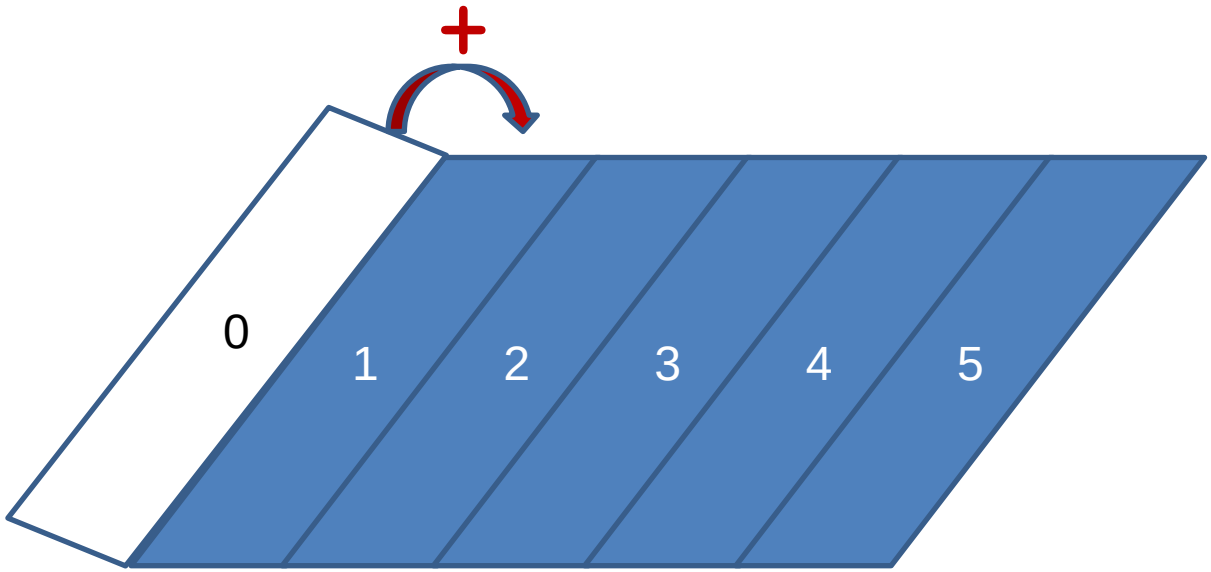
```
names = ['Justin', 'caterpillar', 'openhome']
for ______ in ______:
    print '{0}, {1}'.format(______)
```

  - 2. Look up documentations about `range`, `zip` and `enumerate`.

reduce

- Sometimes, it's called `foldLeft`.

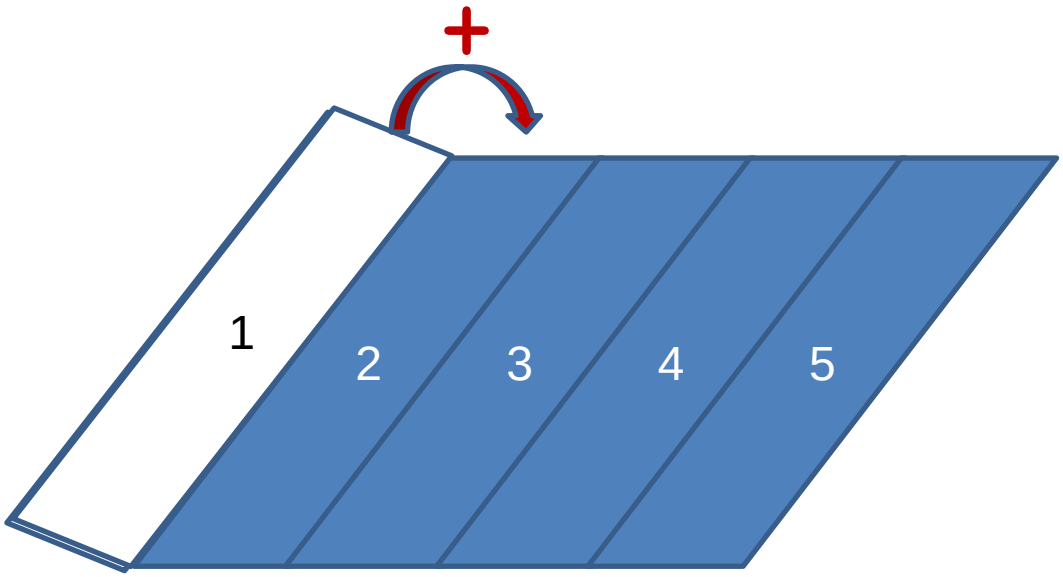
```
reduce(lambda acct, elem: acct + elem, [1, 2, 3, 4, 5], 0)
```



reduce

- Sometimes, it's called `foldLeft`.

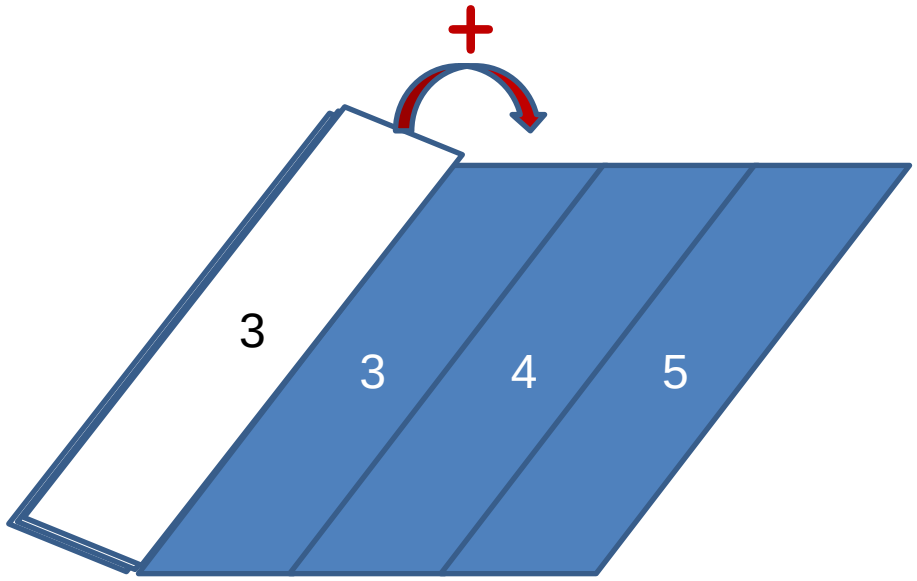
```
reduce(lambda acct, elem: acct + elem, [1, 2, 3, 4, 5], 0)
```



reduce

- Sometimes, it's called `foldLeft`.

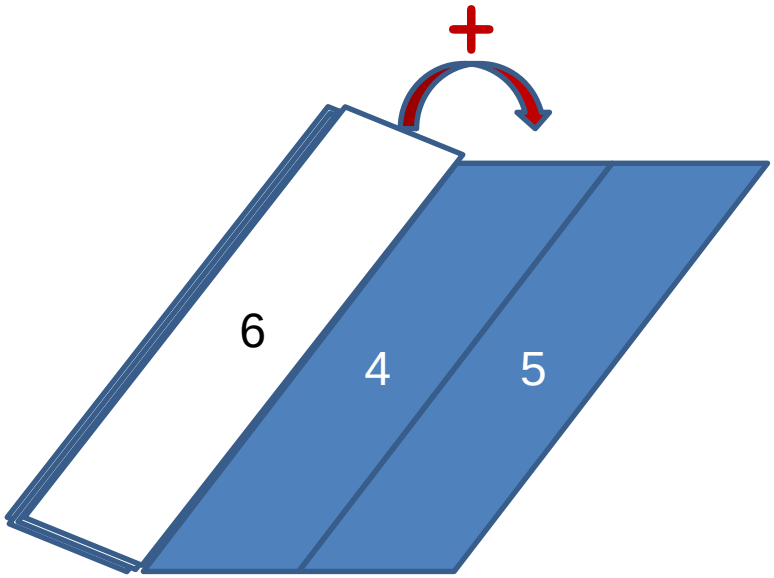
```
reduce(lambda acct, elem: acct + elem, [1, 2, 3, 4, 5], 0)
```



reduce

- Sometimes, it's called `foldLeft`.

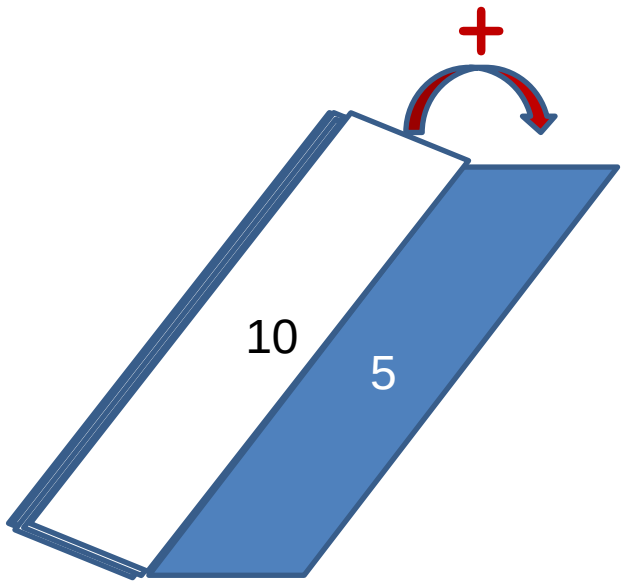
```
reduce(lambda acct, elem: acct + elem, [1, 2, 3, 4, 5], 0)
```



reduce

- Sometimes, it's called `foldLeft`.

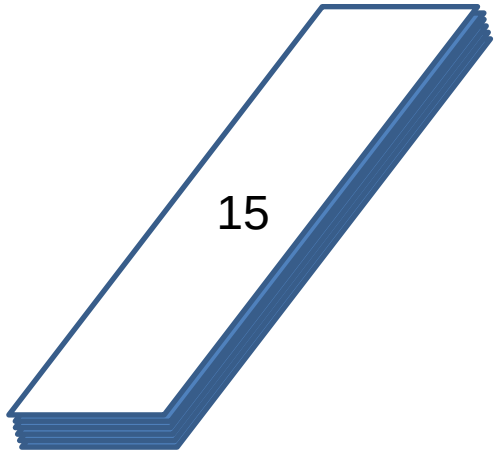
```
reduce(lambda acct, elem: acct + elem, [1, 2, 3, 4, 5], 0)
```



reduce

- Sometimes, it's called `foldLeft`.

```
reduce(lambda acct, elem: acct + elem, [1, 2, 3, 4, 5], 0)
```



`reduce`

- `reduce` is a really versatile function that can be used in millions of different ways.
- Once you want to calculate something from a list, consider using `reduce` instead of a `for` loop.

Exercise 7

- Use **`reduce`** and **list comprehensions** to revise the following code (avaliable in lab/exercises/exercise7/main.py).

```
def ascending(a, b): return a - b
def descending(a, b): return -ascending(a, b)
# selection sort
def sorted(xs, compare = ascending):
    return [] if not xs else __select(xs, compare)

def __select(xs, compare):
    selected = xs[0]
    for elem in xs[1:]:
        if compare(elem, selected) < 0:
            selected = elem

    remain = []
    selected_list = []
    for elem in xs:
        if elem != selected:
            remain.append(elem)
        else:
            selected_list.append(elem)

    return xs if not remain else selected_list + __select(remain, compare)

print sorted([2, 1, 3, 6, 5])
print sorted([2, 1, 3, 6, 5], descending)
```

Persistence

- Object serialization
  - `marshal`, `pickle`, `cPickle`
- DBM (Database Manager)
  - Simple “database” interface. Dbm objects behave like mappings (dictionaries) , except that keys and values are always strings.
- `shelve`
  - A “shelf” is a persistent, dictionary-like object. The values can be essentially arbitrary Python objects.
- DB-API 2.0 (PEP 249)
- Object-Relational Mapping (3rd-party libraries)
  - SQLAlchemy (www.sqlalchemy.org)
  - SQLObject (www.sqlobject.org)

`marshal`, `pickle`, `cPickle`

- A more primitive serialization module is `marshal`. It exists primarily to support Python's .pyc files.
- In general, `pickle` should always be the preferred way to serialize Python objects.
  - It keeps track of the objects it has already serialized, so that later references to the same object won't be serialized again.
  - It can serialize user-defined classes and their instances.
  - Its serialization format is guaranteed to be backwards compatible across Python releases.
- `cPickle` is written in C, so it can be up to 1000 times faster than pickle.

pickle

```
class DVD:
    def __init__(self, title, year=None,
        duration=None, director_id=None):
        self.title = title
        self.year = year
        self.duration = duration
        self.director_id = director_id
        self.filename = self.title.replace(' ', '_') + '.pkl'

    def check_filename(self, filename):
        if filename is not None:
            self.filename = filename
```

```
def save(self, filename=None):
    self.check_filename(filename)
    fh = None
    try:
        data = (self.title, self.year,
                self.duration, self.director_id)
        fh = open(self.filename, 'wb')
        pickle.dump(data, fh)
    except (EnvironmentError, pickle.PicklingError) as err:
        raise SaveError(str(err))
    finally:
        if fh is not None:
            fh.close()

def load(self, filename=None):
    self.check_filename(filename)
    fh = None
    try:
        fh = open(self.filename, 'rb')
        data = pickle.load(fh)
        (self.title, self.year,
         self.duration, self.director_id) = data
    except (EnvironmentError, pickle.PicklingError) as err:
        raise LoadError(str(err))
    finally:
        ...
```

DBM

- The dbm module provides an interface to the Unix “(n)dbm” library.

docs.python.org/2.7/library/anydbm.html#module-anydbm

```
import anydbm

# Open database, creating it if necessary.
db = anydbm.open('cache', 'c')

# Record some values
db['www.python.org'] = 'Python Website'
db['www.cnn.com'] = 'Cable News Network'

# Loop through contents.  Other dictionary methods
# such as .keys(), .values() also work.
for k, v in db.iteritems():
    print k, '\t', v

# Storing a non-string key or value will raise an exception (most
# likely a TypeError).
db['www.yahoo.com'] = 4

# Close when done.
db.close()
```

shelve

- A “shelf” is a persistent, dictionary-like object. The difference with “dbm” databases is that the values (not the keys!) in a shelf can be anything that the pickle module can handle.

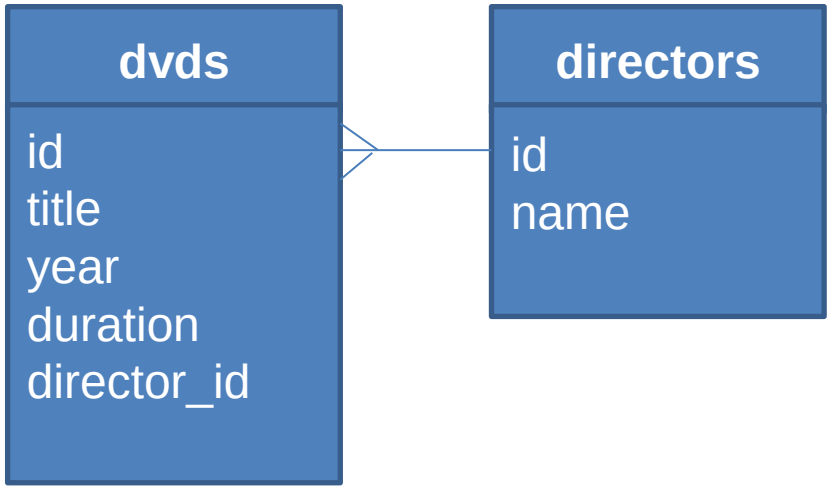
```
class DvdDao:
    def __init__(self, shelve_name):
        self.shelve_name = shelve_name

    def save(self, dvd):
        shelve_db = None
        try:
            shelve_db = shelve.open(self.shelve_name)
            shelve_db[dvd.title] = (dvd.year,
                dvd.duration, dvd.director_id)
            shelve_db.sync()
        finally:
            if shelve_db is not None:
                shelve_db.close()
```

```
    def all(self):
        shelve_db = None
        try:
            shelve_db = shelve.open(self.shelve_name)
            return [DVD(title, *shelve_db[title])
                    for title in sorted(shelve_db, key=str.lower)]
        finally:
            if shelve_db is not None:
                shelve_db.close()
        return []

    def load(self, title):
        shelve_db = None
        try:
            shelve_db = shelve.open(self.shelve_name)
            if title in shelve_db:
                return DVD(title, *shelve_db[title])
        finally:
            if shelve_db is not None:
                shelve_db.close()
        return None
```

```
    def remove(self, title):
        shelve_db = None
        try:
            shelve_db = shelve.open(self.shelve_name)
            del shelve_db[title]
            shelve_db.sync()
        finally:
            if shelve_db is not None:
                shelve_db.close()
```

DB-API 2.0（PEP 249）

- The `sqlite3` module provides a SQL interface compliant with the DB-API 2.0.



```
def connect(name):
    create = not os.path.exists(name)
    conn = sqlite3.connect(name)
    if create:
        cursor = conn.cursor()
        cursor.execute("CREATE TABLE directors ("
            "id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL, "
            "name TEXT UNIQUE NOT NULL)")
        cursor.execute("CREATE TABLE dvds ("
            "id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL, "
            "title TEXT NOT NULL, "
            "year INTEGER NOT NULL, "
            "duration INTEGER NOT NULL, "
            "director_id INTEGER NOT NULL, "
            "FOREIGN KEY (director_id) REFERENCES directors)")
        conn.commit()

    return conn
```

```
def add_dvd(conn, title, year, duration, director):
    director_id = get_and_set_director(conn, director)
    cursor = conn.cursor()
    cursor.execute("INSERT INTO dvds "
                   "(title, year, duration, director_id) "
                   "VALUES (?, ?, ?, ?)",
                   (title, year, duration, director_id))
    conn.commit()

def get_and_set_director(conn, director):
    director_id = get_director_id(conn, director)
    if director_id is not None:
        return director_id
    cursor = conn.cursor()
    cursor.execute("INSERT INTO directors (name) VALUES (?)",
                   (director,))
    conn.commit()
    return get_director_id(conn, director)

def get_director_id(conn, director):
    cursor = conn.cursor()
    cursor.execute("SELECT id FROM directors WHERE name=?",
                   (director,))
    fields = cursor.fetchone()
    return fields[0] if fields is not None else None
```

```
def all_dvds(conn):
    cursor = conn.cursor()
    sql = ("SELECT dvds.title, dvds.year, dvds.duration, "
           "directors.name FROM dvds, directors "
           "WHERE dvds.director_id = directors.id"
           " ORDER BY dvds.title")
    cursor.execute(sql)
    return [(str(fields[0]), fields[1], fields[2], str(fields[3]))
            for fields in cursor]

def all_directors(conn):
    cursor = conn.cursor()
    cursor.execute("SELECT name FROM directors ORDER BY name")
    return [str(fields[0]) for fields in cursor]
```

Exercise 8

- There’re three incomplete source files located in lab/exercises/exercise8. Choose what you are interested in and complete it.
- All code you need were listed in the previous slides.

References

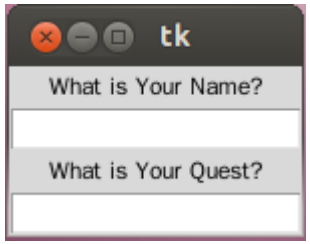
- The Community
  - www.python.org/~guido/
  - http://www.python.org/psf/
  - http://www.python.org/dev/peps/
  - http://www.pycon.org/
  - http://wiki.python.org/moin/LocalUserGroups/
- Documentation
  - http://docs.python.org/2.7/
  - http://docs.python.org/2/library/pydoc.html
- Data Management Functions
  - http://docs.python.org/2.7/library/functions.html
- Persistence
  - http://docs.python.org/2/library/pickle.html
  - http://docs.python.org/2.7/library/dbm.html
  - http://docs.python.org/2/library/shelve.html
  - http://docs.python.org/2.7/library/sqlite3.html

Libraries vs Frameworks

- **What is the difference between a framework and a library?**
- Using libraries, your code is in control: it decides when to ask questions, when to read responses, and when to process those results.

```
name = raw_input('What is your name?')
process_name(name)
quest = raw_input('What is your quest?')
process_quest(quest)
```

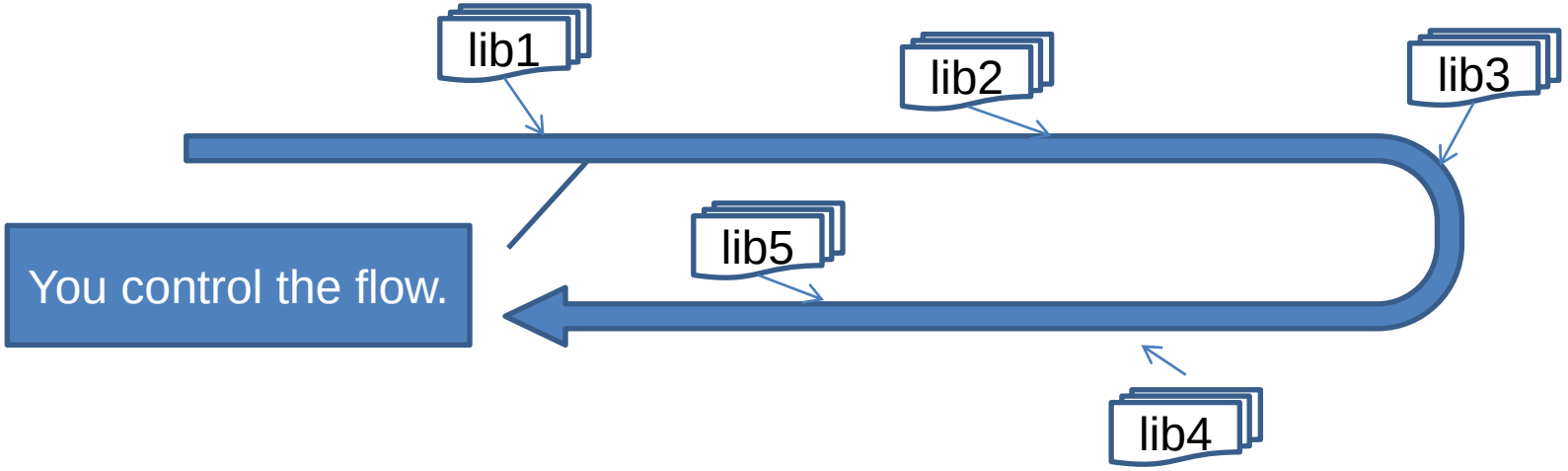
- Using frameworks, it decides when to call your methods, based on the bindings you made when creating the form. **The control is inverted - it calls you rather you calling the framework**.



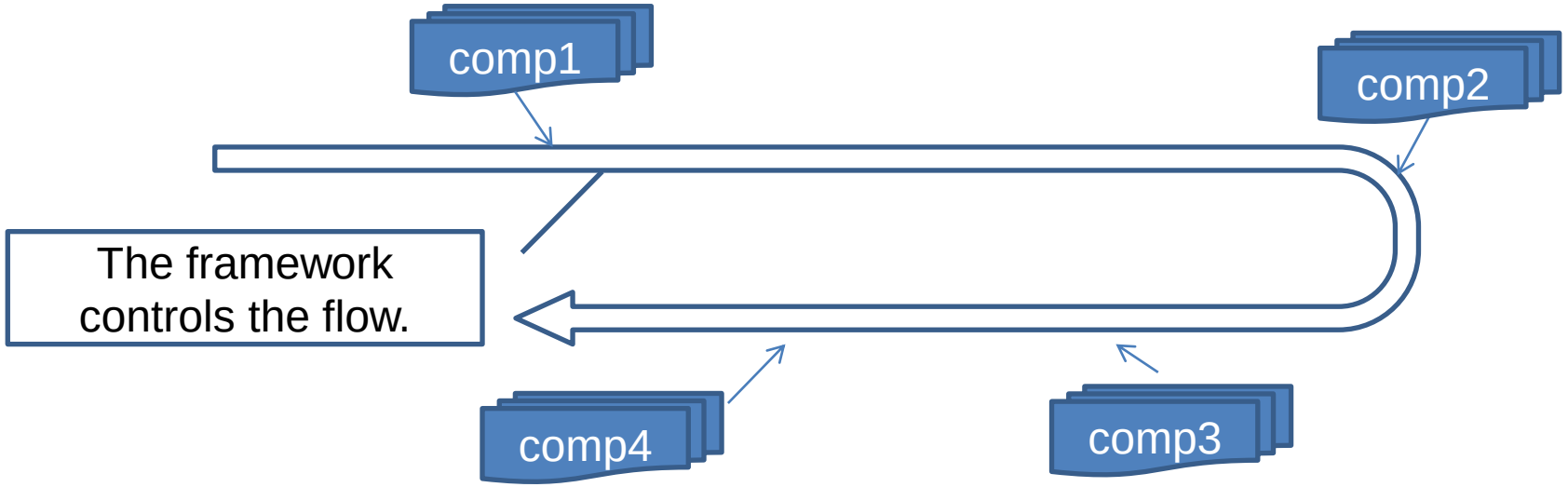

```
import Tkinter

top = Tkinter.Tk()

Tkinter.Label(top, text='What is Your Name?').pack()
name_var = Tkinter.StringVar()
name_entry = Tkinter.Entry(top, textvariable=name_var)
name_entry.pack()
name_entry.bind('<FocusOut>', lambda event: process_name(name_var))

Tkinter.Label(top, text='What is Your Quest?').pack()
quest_var = Tkinter.StringVar()
quest_entry = Tkinter.Entry(top, textvariable=quest_var)
quest_entry.pack()
quest_entry.bind('<FocusOut>', lambda event:
process_name(quest_var))

Tkinter.mainloop()
```

Inversion of Control

- Using libraries

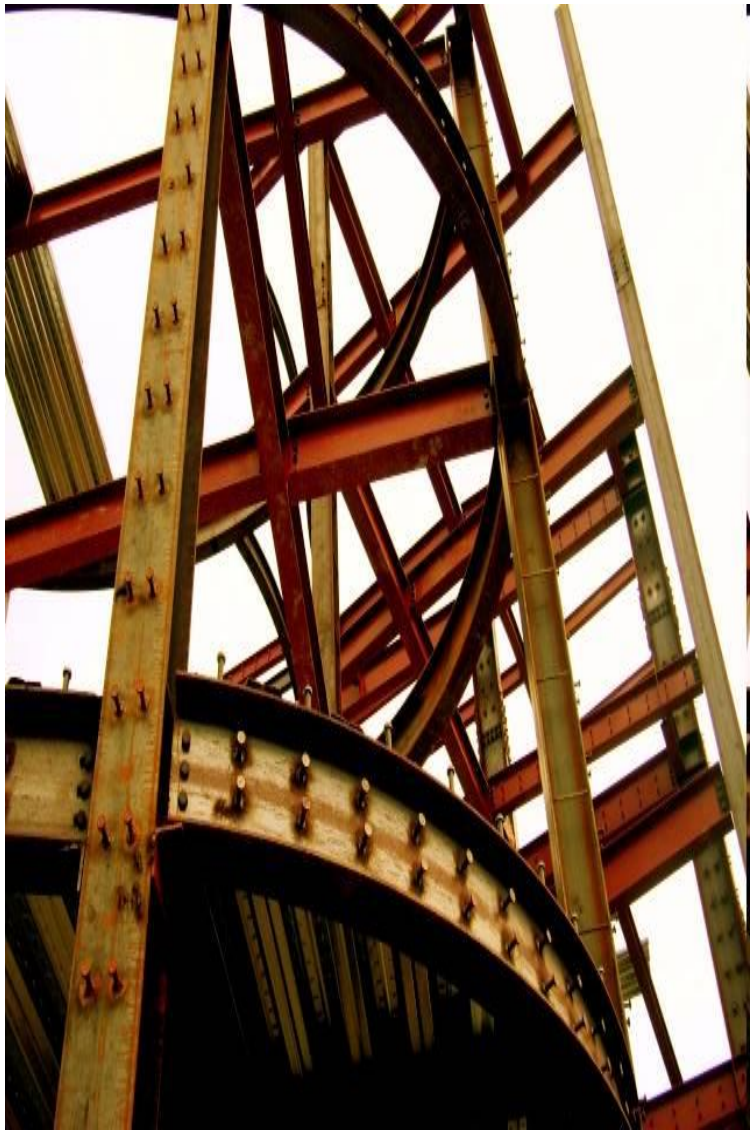


- Using frameworks



Do We Need a Framework?

- **Libraries bring developers freedom.**
- **Frameworks bring developers constraints.**
  - Do we need a framework?
  - Do we want to follow the flow?
  - Do we make decisions according to technical reasons, or business reasons?
- A right framework brings you a heaven; the wrong one brings you a hell.



Getting Started with Django

- Django (www.djangoproject.com) is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.
  - Object-relational mapper
  - Automatic admin interface
  - Elegant URL design
  - Template system
  - Cache system
  - Internationalization

Design Your Models

```python
class Reporter(models.Model):
    full_name = models.CharField(max_length=70)

    def __unicode__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter)

    def __unicode__(self):
        return self.headline
```

```
# No reporters are in the system yet.
>>> Reporter.objects.all()
[]

# Create a new Reporter.
>>> r = Reporter(full_name='John Smith')

# Save the object into the database. You have to call save() explicitly.
>>> r.save()

# Now it has an ID.
>>> r.id
1
```

Design Your URLs

```
from django.conf.urls import patterns

urlpatterns = patterns('',
    (r'^articles/(\d{4})/$', 'news.views.year_archive'),
    (r'^articles/(\d{4})/(\d{2})/$', 'news.views.month_archive'),
    (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'news.views.article_detail'),
)
```

Write Your Views and Templates

```
def year_archive(request, year):
    a_list = Article.objects.filter(pub_date__year=year)
    return render_to_response('news/year_archive.html', {'year': year, 'article_list': a_list})
```

```
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <img src="sitelogo.png" alt="Logo" />
    {% block content %}{% endblock %}
</body>
</html>
```

```
{% extends "base.html" %}

{% block title %}Articles for {{ year }}{% endblock %}

{% block content %}
<h1>Articles for {{ year }}</h1>

{% for article in article_list %}
    <p>{{ article.headline }}</p>
    <p>By {{ article.reporter.full_name }}</p>
    <p>Published {{ article.pub_date|date:"F j, Y" }}</p>
{% endfor %}
{% endblock %}
```

Exercise 9

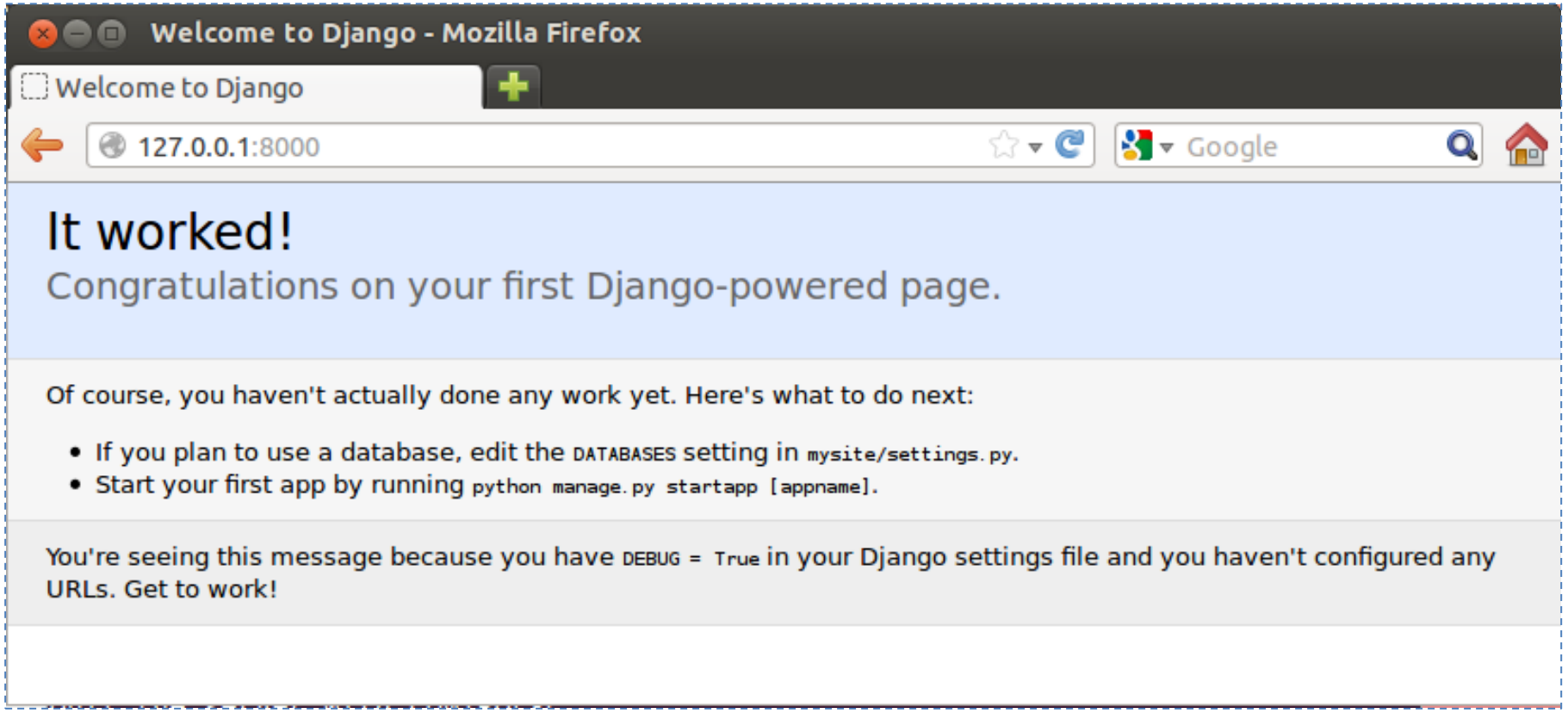
- We'd like to install an offical realse of **Django 1.5 rc1** with **pip** under a virtual Python environment provided by **virtualenv**. And Then, create our first django project.

```
~/scripts$ virtualenv --distribute venv
~/scripts$ cd venv
~/scripts/venv$ source bin/activate
~/scripts/venv$ pip install https://www.djangoproject.com/download/1.5c1/tarball/
~/scripts/venv$ python -c 'import django; print django.get_version()‘
~/scripts/venv$ django-admin.py startproject mysite
~/scripts/venv$ ls -al mysite
~/scripts/venv$ cd mysite
~/scripts/venv$ python manage.py runserver
```

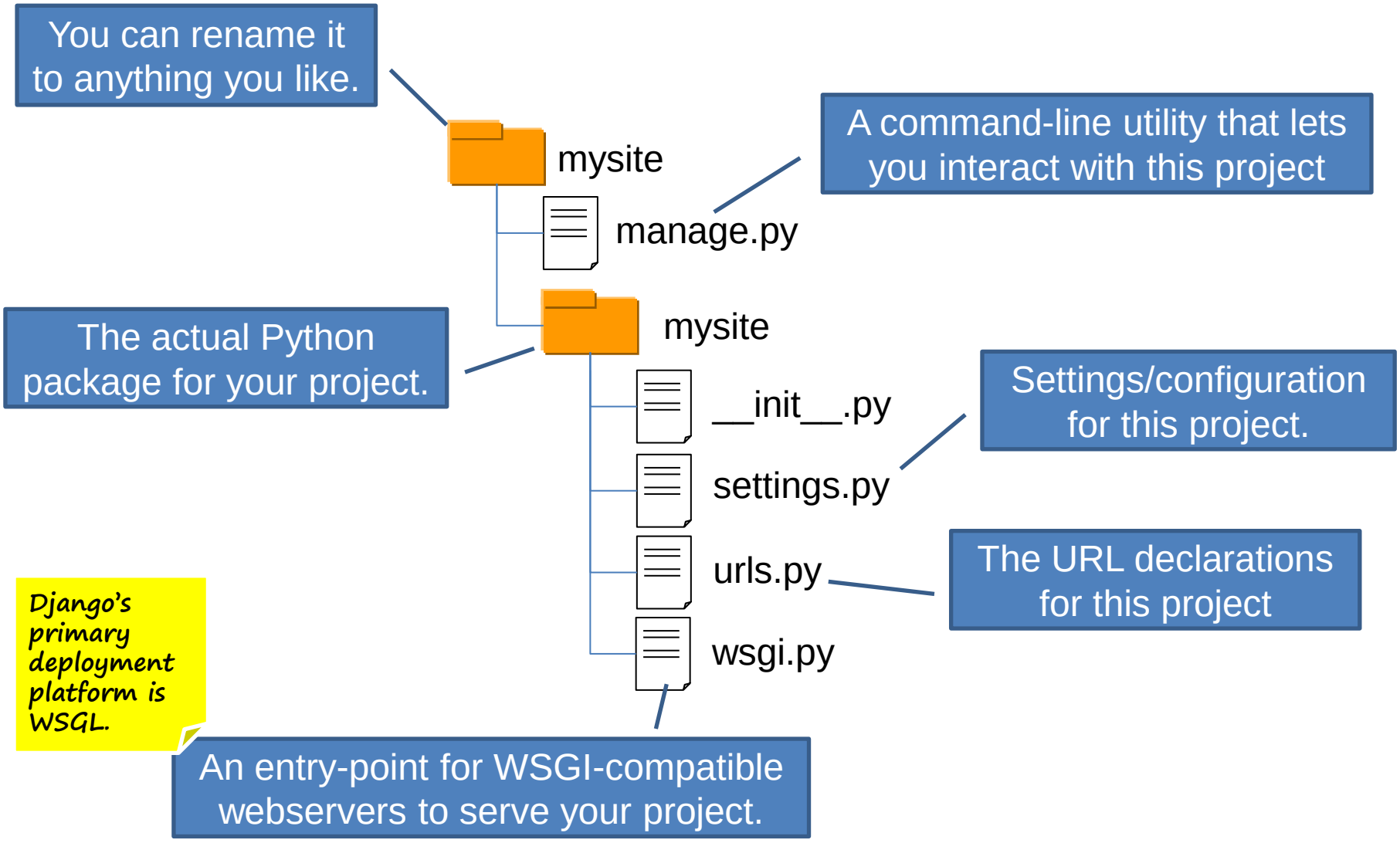
What You Should See

```
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv$ python -c 'import djang
o; print django.get_version()'
1.5c1
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv$ django-admin.py startpr
oject mysite
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv$ ls -al mysite
總計 16
drwxrwxr-x 3 caterpillar caterpillar 4096  2月   5 14:54 .
drwxrwxr-x 7 caterpillar caterpillar 4096  2月   5 14:54 ..
-rw-rw-r-- 1 caterpillar caterpillar  249  2月   5 14:54 manage.py
drwxrwxr-x 2 caterpillar caterpillar 4096  2月   5 14:54 mysite
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv$ cd mysite
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv/mysite$ python manage.py
 runserver
Validating models...

0 errors found
February 05, 2013 - 00:57:50
Django version 1.5c1, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

What You Should See



What `startproject` Created



Exercise 10

- Edit **mysite/settings.py**. Change the following keys in the `DATABASES 'default'` item to match your database connection settings.

```
DATABASES = {
    'default': {
        # Add 'postgresql_psycopg2', 'mysql', 'sqlite3' or 'oracle'.
        'ENGINE': 'django.db.backends.sqlite3',
        # Or path to database file if using sqlite3.
        'NAME': '/home/caterpillar/scripts/venv/mysite/db.sqlite3',
        # The following settings are not used with sqlite3:
        'USER': '',     # Your database username (not used for SQLite).
        'PASSWORD': '', # Your database password (not used for SQLite).
        # Empty for localhost through domain sockets or '127.0.0.1' for localhost through TCP.
        'HOST': '',
        'PORT': '',     # Set to empty string for default.
    }
}
```

- `python manage.py syncdb`

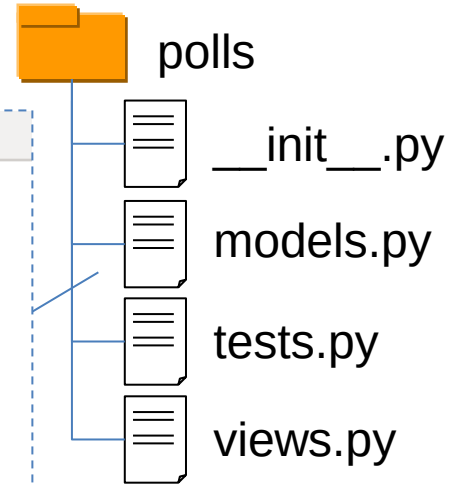
What You Should See

```
Creating tables ...
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_groups
Creating table auth_user_user_permissions
Creating table auth_user
Creating table django_content_type
Creating table django_session
Creating table django_site

You just installed Django's auth system, which means you don't have any superuse
rs defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'caterpillar'):
Email address: caterpillar@openhome.cc
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv/mysite$
```

- Type the following command to create a simple poll app.
  - `python manage.py startapp polls`
- Edit the **polls/models.py** so it looks like this:

models.py

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)

    def __unicode__(self):
        return self.question

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField()

    def __unicode__(self):
        return self.choice_text
```



- Edit the **settings.py** again, and change the `INSTALLED_APPS` setting to include the string `'polls'`.

settings.py

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'polls'
)
```

- Type the following command to create tables for the polls app.
  - `python manage.py sql polls`
  - `python manage.py syncdb`

What You Should See

```
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv/mysite$ python manage.py
 sql polls
BEGIN;
CREATE TABLE "polls_poll" (
    "id" integer NOT NULL PRIMARY KEY,
    "question" varchar(200) NOT NULL,
    "pub_date" datetime NOT NULL
)
;
CREATE TABLE "polls_choice" (
    "id" integer NOT NULL PRIMARY KEY,
    "poll_id" integer NOT NULL REFERENCES "polls_poll" ("id"),
    "choice_text" varchar(200) NOT NULL,
    "votes" integer NOT NULL
)
;

COMMIT;
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv/mysite$ python manage.py
 syncdb
Creating tables ...
Creating table polls_poll
Creating table polls_choice
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
(venv)caterpillar@caterpillar-VirtualBox:~/scripts/venv/mysite$
```

Playing API with the Python shell

- Type the following command to set the `DJANGO_SETTINGS_MODULE` environment variable, which gives Django the Python import path to your settings.py file.
  - `python manage.py shell`

Basic ORM

```
>>> from polls.models import Poll, Choice
>>> from django.utils import timezone
>>> p = Poll(question="What's new?", pub_date=timezone.now())
>>> p.save()
>>> p.id
1
>>> p.question
"What's new?"
>>> p.pub_date
datetime.datetime(2013, 2, 6, 3, 8, 40, 994702, tzinfo=<UTC>)
>>> p.question = "What's up?"
>>> p.save()
>>> Poll.objects.all()
[<Poll: What's up?>]
>>> Poll.objects.filter(id=1)
[<Poll: What's up?>]
>>> Poll.objects.filter(question__startswith='What')
[<Poll: What's up?>]
>>> Poll.objects.get(pub_date__year=timezone.now().year)
<Poll: What's up?>
>>> Poll.objects.get(id=2)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "/home/caterpillar/scripts/venv/local/lib/python2.7/site-packages/django/
db/models/manager.py", line 143, in get
```

One-to-One Relationship

```
>>> p = Poll.objects.get(pk=1)
>>> p.choice_set.create(choice_text='Not much', votes=0)
<Choice: Not much>
>>> p.choice_set.create(choice_text='The sky', votes=0)
<Choice: The sky>
>>> c = p.choice_set.create(choice_text='Just hacking again', votes=0)
>>> c.poll
<Poll: What's up?>
>>> p.choice_set.all()
[<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]
>>> p.choice_set.count()
3
>>> Choice.objects.filter(poll__pub_date__year=timezone.now().year)
[<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]
>>> c = p.choice_set.filter(choice_text__startswith='Just hacking')
>>> c.delete()
>>>
```

Exercise 11

- Let's write your first view. Open the file **polls/views.py** and put the following Python code in it:

views.py

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, world. You're at the poll index.")

def detail(request, poll_id):
    return HttpResponse("You're looking at poll {id}.".format(id = poll_id))

def results(request, poll_id):
    return HttpResponse("You're looking at the results of poll {id}.".format(id = poll_id))

def vote(request, poll_id):
    return HttpResponse("You're voting on poll {id}.".format(id = poll_id))
```

- Create a file called **urls.py** in the **polls** directory. Include the following code:

urls.py

```
from django.conf.urls import patterns, url

from polls import views

urlpatterns = patterns('',
    # ex: /polls/
    url(r'^$', views.index, name='index'),
    # ex: /polls/5/
    url(r'^(?P<poll_id>\d+)/$', views.detail, name='detail'),
    # ex: /polls/5/results/
    url(r'^(?P<poll_id>\d+)/results/$', views.results, name='results'),
    # ex: /polls/5/vote/
    url(r'^(?P<poll_id>\d+)/vote/$', views.vote, name='vote'),
)
```

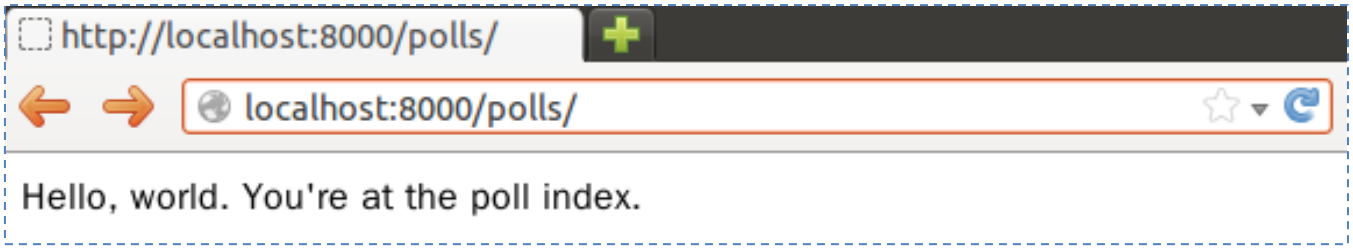
- Open **urls.py** in the **mysite** directory. Include the following code:

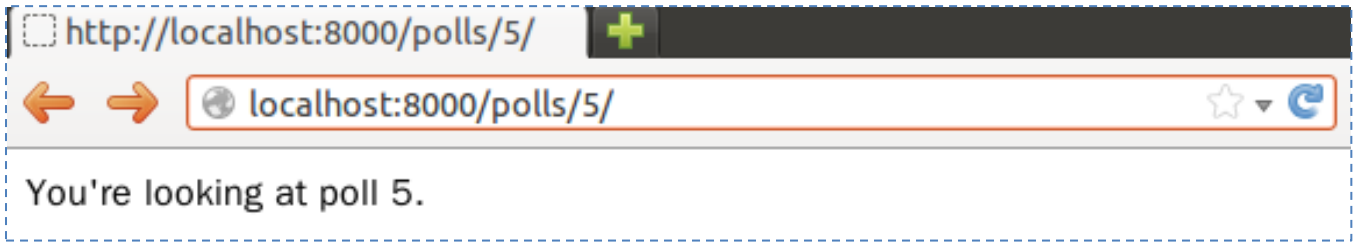
urls.py

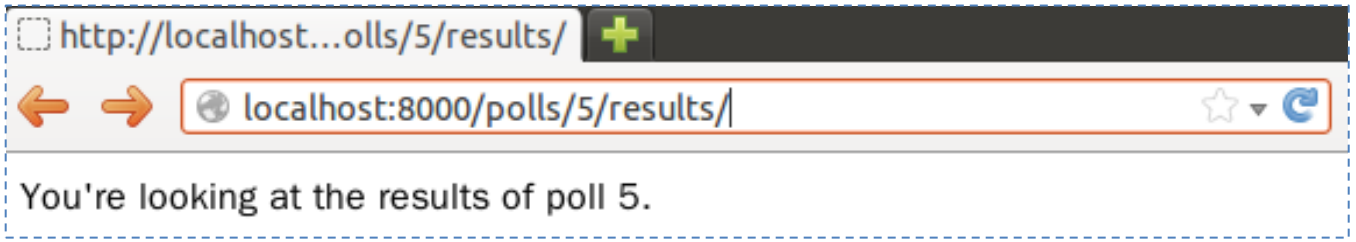
```
urlpatterns = patterns('',
    url(r'^polls/', include('polls.urls'))
    # Examples:
    # url(r'^$', 'mysite.views.home', name='home'),
    # url(r'^mysite/', include('mysite.foo.urls')),
```

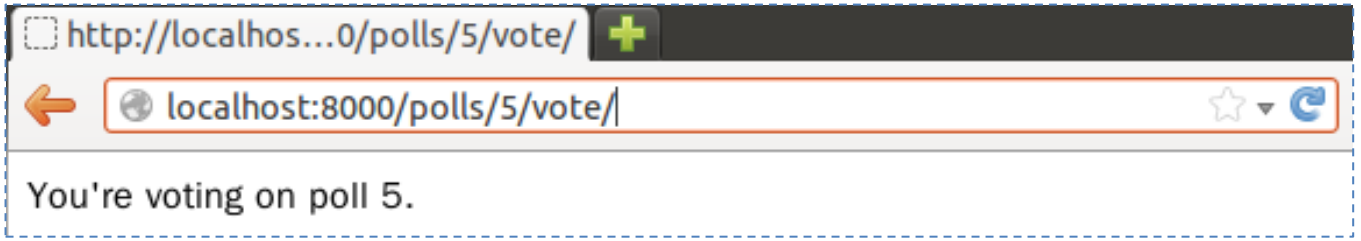
- Type the following command to start the Django development server.
  - `python manage.py runserver`
- Visit the following urls with your browser.
  - http://localhost:8000/polls/
  - http://localhost:8000/polls/5/
  - http://localhost:8000/polls/5/results/
  - http://localhost:8000/polls/5/vote/

What You Should See









Controllers or Views?

- We are using Django **MVC** framework. Are functions `index`, `details`, `results` and `vote` belong to controllers or views?
  - Well, the standard names are debatable.
  - In Django's case, a "view" is the Python callback function for a particular URL.
  - Where does the "controller" fit in, then? In Django's case, it's probably the framework itself.
  - As you'll see soon, you might say that Django is a **MTV** framework – that is, "**Model**", "**Template**", and "**View**".

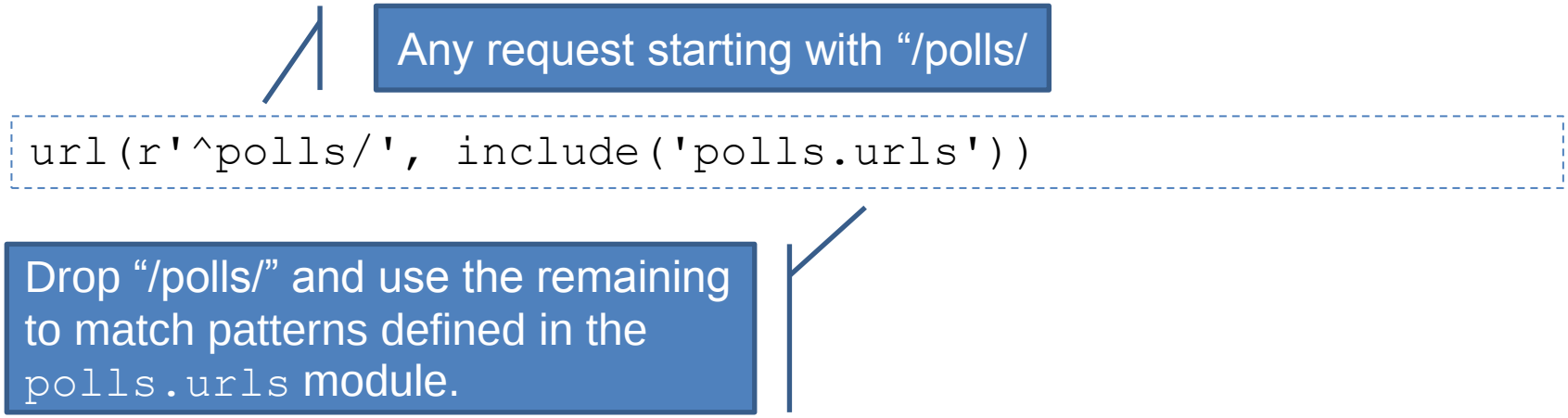
- (Is there `before_filter` in Django as in Rails?
  - No. `before_`, `around_` and `after_ filter` concepts aren't present in Django.
  - It’s not hard to hard-code what you need. Or, you can use a generic decorator, such as those provided by the Django authentication system.)

URLconf

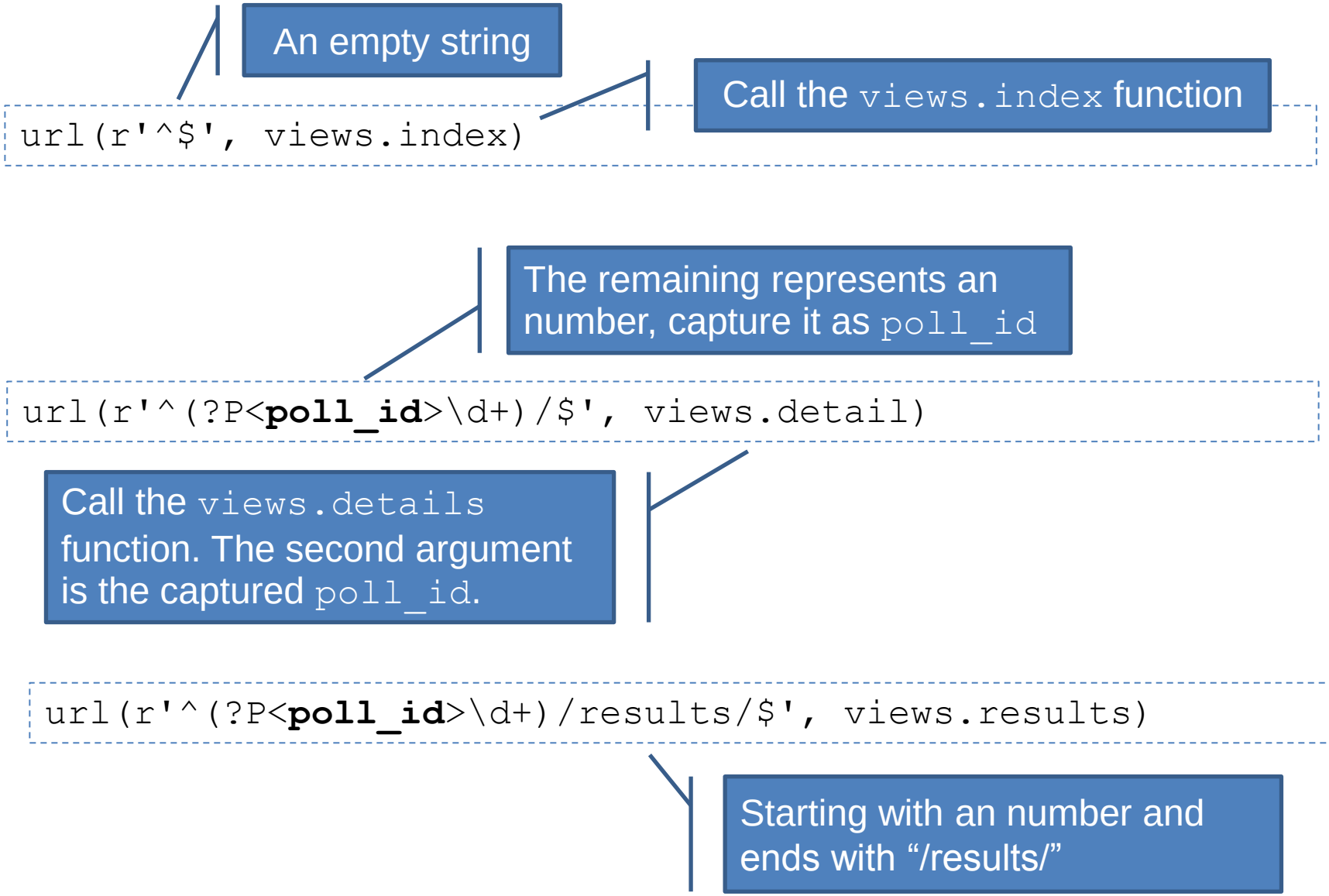
- Determining which view is called is done by Python modules informally titled '**URLconfs**'.
  - These modules are pure Python code and are a simple mapping between URL patterns to Python callback functions (your views).
- The `url()` function needs two required arguments and one suggested argument.
  - `regex`: URL patterns are simple regular expressions.
  - `view`: When Django finds a regular expression match, Django calls the specified view function, with an `HttpRequest` object as the first argument and any "captured" values from the regular expression as other arguments.
  - `name`: Naming your URL lets you refer to it unambiguously from elsewhere in Django especially templates.

Simple URL Patterns

- For `urlpatterns` in **mysite/urls.py**.



- For `urlpatterns` in **polls/urls.py**.



References

- Libraries vs Frameworks
  - http://martinfowler.com/bliki/InversionOfControl.html
- Getting Started with Django
  - https://docs.djangoproject.com/en/1.5/intro/overview/
  - https://docs.djangoproject.com/en/1.5/
  - https://docs.djangoproject.com/en/1.5/intro/install/
  - http://stackoverflow.com/questions/12339608/installing-django-1-5development-version-in-virtualenv
  - https://docs.djangoproject.com/en/1.5/intro/tutorial01/
- Writing Your First View
  - https://docs.djangoproject.com/en/1.5/intro/tutorial03/
  - https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names
  - https://docs.djangoproject.com/en/1.5/topics/auth/default/

ToDo..