Lecture 0: Introduction to Python

IPEC Winter School 2015
B-IT

Dr. Tiansi Dong & Dr. Joachim Köhler
Why Python?

- Software quality: Readability, coherence, reuse, OOP
  - Coherence: State changes trigger events called reactions, that in turn change other states. A coherent execution order is one in which each reaction executes before any others that are affected by its changes.

- Developer productivity: Python code is normally 1/3 ~ 1/5 the size of C++ or Java

- Program portability

- Support libraries: a large collection of library

- Component integration: easily communicate with other applications
Python compared with other languages

- is more powerful than Tcl
- has a cleaner syntax and simpler design than Perl
- is simpler and easier to use than Java and C++
- is both more powerful and more cross-platform than Visual Basic
- is more readable and general purpose than PHP
- is more mature and has a more readable syntax than Ruby
- has dynamic flavor as SmallTalk and Lisp
What can we do with Python?

- System programming
- GUIs
- Internet Scripting
- Component Integration
- Database Programming
- Rapid Prototyping
- Numeric and Scientific Programming
- Gaming, Images, Serial Ports, XML, Robots
- Speech Technology
- ...

Python applications are everywhere

- **Google** extensively uses Python in its web search system
- **YouTube**'s video sharing service is largely written in Python
- **BitTorrent** peer-to-peer file sharing system
- **Intel, Cisco, HP, Seagate, and IBM** use Python for hardware testing
- **NASA, Los Alamos, and JPL** use Python for scientific programming
- **NSA** (National Security Agency) uses Python for cryptography and intelligence analysis
- **Fraunhofer IAIS** uses Python for Speech Technology
The execution speed of Python may not always be as fast as that of C and C++. The standard Python compiles source codes into an intermediate format (byte code) and then interpret the byte code into binary machine code. Python is not compiled all the way down to the binary machine code (e.g. instructions for an Intel chip).

Many Python applications and libraries run at C speed: file processing, GUI construction, NumPy for numeric processing.

Python's speed-of-development gain outweighs the speed-of-execution loss.
Python: an object-oriented scripting language

- Meaning of “script” and “program”
  - Script: a simpler top-level file, such as, shell tool, control language
  - Program: a more complicated application, with many files

- Python is both “script” and “program”
Design philosophy

- “fit your brain”
  - A small set of core concepts, from which all features of the language follows
- Using the minimalist approach
  - Simple is better than complex
  - Explicit is better than implicit
- Using modules and OOP
  - Promote code re-usability and readability
- Python increases developers' productivity
Elegant Features of Python

- Object-oriented, free of charge, easy to use, easy to learn
- Portable: From PDA to supercomputer; Python is available on Linux, Unix, MS, Mac, OS/2, Cell phones...
- Powerful
  - Dynamic typing, automatic memory management, support large systems with modules, classes, exceptions, built-in object types, built-in tools, large libraries, third-party utilities
- Mixable
  - Python programs can easily mixed with programs written in other languages
Install and configure Python

- Download Python from http://www.python.org
- Windows
  - Python can be installed through a simple double-click on a self-installer MSI program
- Linux
  - Download Python RPM file, and install it as a super-user
- Unix
  - Download its full C source code, config, make, and make install (as super-user)
Install and configure Python

• Python Environment Variables on Window Systems
  - PATH
    • System shell search path(for finding “python”)
  - PYTHONPATH
    • Python module search path(for imports)
    • Platform-dependent, separated by “:” on Unix, and “;” on Windows
  - PYTHONSTARTUP
    • Path to Python interactive start-up file
    • Set the path of a file of Python code, which is executed whenever Python interpreter is started.
  - TCL_LIBRARY, TK_LIBRARY
    • GUI extension variable(tkinter)
How to set Python Environment Variables

- Suppose you have a Python file `dic.py` at the `/usr/local/pycode/utilities` directory on your Unix/Linux platform or `c:\pycode\utilities` on your Windows platform.

- Unix/Linux
  - Under csh shell
    ```
    setenv PATHONPATH /usr/local/pycode/utilities
    ```
  - Under ksh
    ```
    setenv="PATHONPATH /usr/local/pycode/utilities"
    ```

- Windows
  - Set `PYTHONPATH=c:\pycode\utilities`
Run Python Programs

- Python Interpreter

  - Linux/Unix

    % python

    Python 3.0.0 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit (Intel)]...

    Type “help”, “copyright”, “credits” or “license” for more information

    >>>

  - Windows

    C:\> c:\python30\pathon

    Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit (Intel)]...

    Type “help”, “copyright”, “credits” or “license” for more information

    >>>
%python

>>> print('Hello world!')
Hello world!

>>> print(2 ** 8)
256

>>> X = 'okey'

>>> X * 3
'okeyokeyokey'

>>> import os

>>> os.getcwd()
'C:\python30'
Running code interactively

Try:

```python
>>> os.getcwd('dir')

>>> os.getcwd('dir')
```

• Do not indent at the interactive prompt!

Try:

```python
>>> for x in 'okey':
...     print(x)   # press Enter twice
...
```
Running code interactively

Try:

```plaintext
>>> import sys
>>> print(sys.platform)
>>> print(2 ** 100)
>>> quit()
```
# A Python file script0.py

```python
import sys
print(sys.platform)
print('hello world!')
```

```bash
$ python script0.py
win32
hello world!
```
Import python files

- If you have a python file `script0.py`, and want to use functions defined in this file at a Python interpreter:
  ```
  >>> import script0
  ```

- If you change some definitions in `script0.py`, you will need to reload the file.
  - Python 2.6
    ```
    >>> reload(script0)
    ```
  - Python 3.0
    ```
    >>> from imp import reload
    >>> reload(script0)
    ```
List names available in an imported file

```python
>>> import os

>>> dir(os)
```
Python Types

- **Numbers** 1234, 3.1324, 1+8j, 1e3
- **Strings** 'spam', “hello world”, b'a\x01c'
- **Lists** [1, [2, 3], [3, 'three']]
- **Dictionaries** {'name': 'Bush', 'firstname': 'George W.'}
- **Tuples** (1, 'X')
- **Files** mfile = open('script0.py', 'r')
- **Sets** x = set('abc'), x={'a', 'b', 'c'}
- **Other core types** Booleans, types, None
- **Program unit types** Functions, modules, classes
- **Implementation-related types** Compiled code, stack traceback
• Modules are the largest program structure in Python programs

• Python programs are composed of module files, linked together by import statement.

• Each module file is a self-contained package of variables – a namespace

• Difference between import and from

• reload is not transitive: reloading a module A only reloads that module A. Module B, if imported by module A, will not be reloaded.

• exec(open('module.py').read()) will launch a module without importing, and overwrite variables with the same name
Exercise (Learn by doing)

- Install Python on your laptop
- Type all the examples above
- Type and read sample codes which I give you, and run it
- Test and more sample codes in

http://www.rmi.net/~lutz/about-pp4e.html