Lab 1: Programming with Python (Installation and Getting Start)

Adam Ding [*]

Friday 4th September, 2015

1 Install or update Python

It is not necessary to use the latest version of Python for this course. Student who use Mac OS X or Linux can ignore this section.

1.1 Windows Users

Note that Windows does not contain Python, so students who use Windows should go through this section to have Python correctly installed.

The latest version of `Python` for windows is available at official website:

- Python for Windows: https://www.python.org/downloads/windows/

Because the following experiments require `PyAudio`, which only supports with 32-bits version Python, students should download the `x86` version through the given link and download the `x86` version `Latest Python 2 Release - Python 2.7.x`).

Using the given link, student may find the webpage as Figure 1. Click the link of `Latest Python 2 Release - Python 2.7.x`, and the find the `Windows x86 MSI installer` on the bottom the all the available versions (see Figure 2).

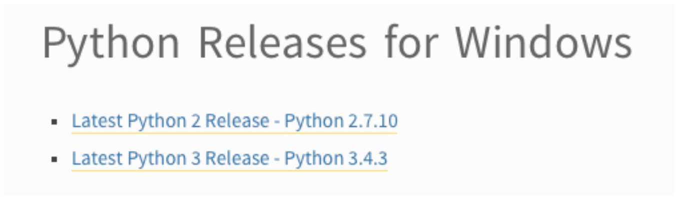
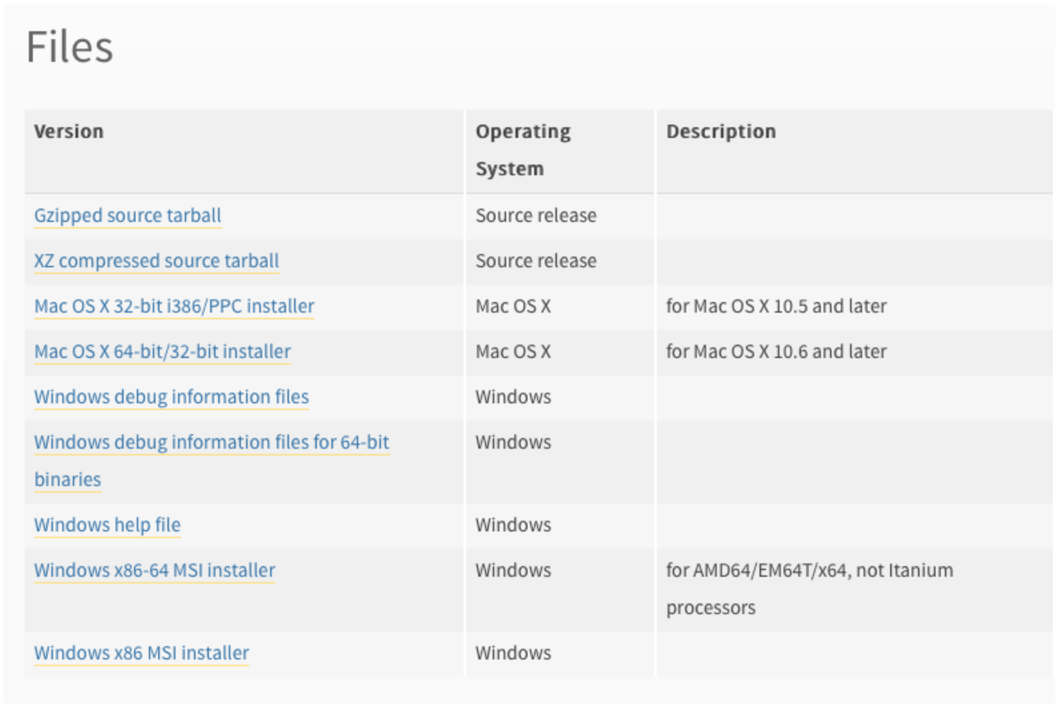

Python Releases for Windows
Latest Python 2 Release - Python 2.7.10
Latest Python 3 Release - Python 3.4.3

Figure 1: Part of the web page giving links of Python3 or Python2.

When finishing the downloading, double click the downloaded file `python-2.7.x.msi` to start the installation. We suggest students install Python at the given default directory (i.e., `C:/Python27/`) during the installation, to avoid some unnecessary modifications when installing `PyAudio` or for future usage.

During the installation, students need to select the option `"Add python.exe to Path"`, which is at the end of the list at the step of `Customize Python 2.7.x` (see Figure 3).

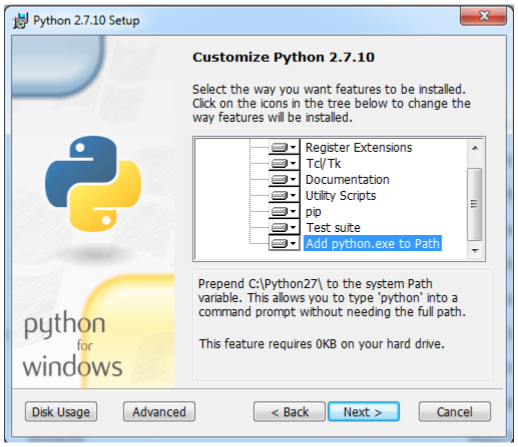
[*]Last Edit: Friday 4th September, 2015 20:39



Files

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later
Windows debug information files	Windows	
Windows debug information files for 64-bit binaries	Windows	
Windows help file	Windows	
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64, not Itanium processors
Windows x86 MSI installer	Windows	

Figure 2: All the downloadable installers of Python2.



Python 2.7.10 Setup

Customize Python 2.7.10

Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.

- Register Extensions
- Tcl/Tk
- Documentation
- Utility Scripts
- pip
- Test suite
- Add python.exe to Path

Prepend C:\Python27\ to the system Path variable. This allows you to type 'python' into a command prompt without needing the full path.

This feature requires 0KB on your hard drive.

Figure 3: Installation window of Python

After finishing the installation, to test Python is correctly installed, student can press `<WIN> + <R>`, which leads to the program `Run` of Windows. In the `Run` window, students may need to type '`cmd`', as shown in Figure 4. Then the `Prompt` (Terminal in Windows) will open. Type '`python`' in `Prompt`, then the current version of Python on you computer will show as Figure 5. Make sure it shows `Python 2.7.x` with 32-bit.

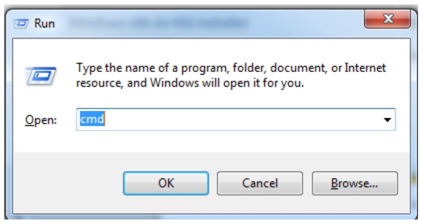
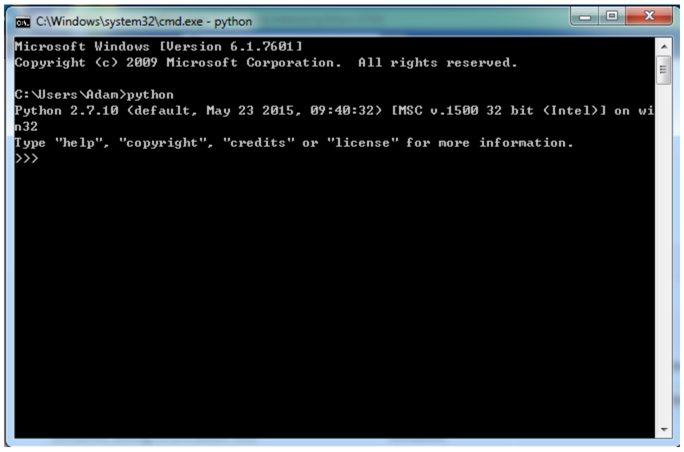

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd
OK
Cancel
Browse...

Figure 4: `Run` of Windows



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Adam>python
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on wi
n32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 5: current Python version.

1.2 Mac OS X and Linux Users

Most of the popular versions of Mac OS X and Linux (e.g. Ubuntu and Fedora) contain Python already, there is no requirements for students using Mac OS X or Linux to re-install Python. Users of either the above platforms can type '`python`' in the Terminal to see the current version of Python in the system.

Students who use Mac can also consider to get the latest version of Python from official website:

- Python for Mac OS X: https://www.python.org/downloads/mac-osx/

Students who use Linux can also consider to download the source release of the latest version Python through `Gzipped source tarball` in Figure 2, and then compile and install it.

Again, it is not necessary to use the latest version of Python for this course. Student who use Mac or Linux can ignore this section.

Assignments 1:

1. Make sure your computer has Python 2.7.x installed (32-bit for Windows Users).

2 Learning Python

In this course, we expect students can have some experience of Python before starting audio-based assignments. Students who is weak on Python programming may spend some time to practice themselves. This course will not start with programming with Python literally.

To whom are not confident with Python programming skill, we suggest to go through this section within the first a few week of the semester. This section will give some suggestion of quickly starting (or rehabbing) programming with Python, including some necessary readings and online materials.

2.1 Suggested readings

2.1.1 *Beginning Python from Novice to Professional*

For beginners, we suggest the book *Beginning Python from Novice to Professional* by Magnus Lie Hetland. For NYU students, the electronic version of the book is available at SpringerLink database from library website:

- 2005 version `http://ezproxy.library.nyu.edu:2178/book/10.1007/978-1-4302-0072-7`
- 2007 version `http://ezproxy.library.nyu.edu:2178/book/10.1007/978-1-4302-0634-7`

In this book, there is a very detailed tutorial of installing Python on different platforms in Chapter 1. This chapter also gives some basics about variables, functions, modules, and how to compile and run your Python script.

To start the first a few programming assignments, we suggest that the students should go through Chapter 1-2, 5-6, where to learn the necessary basics of Python programming such as:

- definitions of different types of variables, such as `int`, `bool`, etc.
- operation of `list`
- use `def` to define a function
- object and method in Python
- conditionals and loops such as `if`, `elif`, `else`, `for` and `while`

Students should try to ACTUALLY typing out some of the examples in the book and find the answers raised during programming. For instance, try to find the results from the following lines:

```python
print 1/2
print 1.0/2
print float(1)/2
```

and try to figure out that: are the results from the above lines are the same, why or why not?

What is the difference of the following two lines?

```python
y = x
y = x[:]
```

2.1.2 `www.python.org`

Never forget that the most accurate explanations are from the Python Software Foundation. A good material for learning Python online is the official document from `www.python.org`, which includes a very detailed tutorial and a library reference.

- The Python Tutorial: `https://docs.python.org/2/tutorial/index.html`
- The Python Standard Library: `https://docs.python.org/2/library/index.html`

2.1.3 *Learning Python* by Mark Lurtzt⋆ (optional)

For understanding Python further, we suggest *Learning Python* by Mark Lurtz as an advanced book. If this course rises you some interest of using Python, and motivates you to further the learning so that programming with Python might become one of your professional skills, this book will do a lot of help.

This book has about 1600 page. Just in case someone really wants a paper version ☺.

2.2 Supplements of Learning Python

- Python Guide for begginers: `https://wiki.python.org/moin/BeginnersGuide`

Besides understanding basics of Python, such as the contents covered in the first few chapters of *Beginning Python from Novice to Professional*, we list the following webpage as a reference:

- Audio in Python: `https://wiki.python.org/moin/Audio/`

which gives the links of some popular materials about using Python for audio applications.

Furthermore, the following module may be often used in this course:

- `wave` `https://docs.python.org/2/library/wave.html`

For writing Python in a professional standard, we suggest the rules by *Google Python Style Guide*:

- `https://google-styleguide.googlecode.com/svn/trunk/pyguide.html?showone=Naming#Python_Language_Rules`

It is the standard used in Google Inc., including the rules of naming variables, classes, methods, and rules of spacing, etc..

Assignments 2:

1. Write a Python `module` named `my_sort_methods` containing at least 2 different sorting methods. Each method in the module should not use `sort`, `sorted` or any other build-in function in any other `module` for sorting. Each method should at least capable to sort a `list` of non-negative integers.
2. Write a Python script satisfying the following requirements:
   (1) Generate a `list` of 10 random non-negative integers.
   (2) Import your module and use the sorting methods inside to sort the `list` into an ascending one.
   (3) Print out the sorted results by the methods.
   (Deliver your source code: main script and module.)

3. Which sorting algorithms you used for your methods in the module? (At least 2)
   Which one is better (maybe evenly good) and why?

3 Handle `*.wav` Files

As a preparation of next lab, students should get familiar with `wave module` and using it to handle wav files.

3.1 Path of `wav` file

The following lines

```
str_wavfile_name = "pyaudio_16k.wav"
str_folder_path = sys.path[0]
str_wavfile_path = str_folder_path + "/" + str_wavfile_name
```

is contained in a Python script giving the directory of the file `pyaudio_16k.wav` located in a same folder with the script.

In this course, we will keep wav files in the same folder with Python scripts. Note that we save the wav file in a same folder of the scripts just for easy packaging of all necessary files. Students should not be limited by this file structure in practice, but may consider to use it when delivering source code of projects.

Here, we adopted `path` from `sys` module. The command `sys.path[0]` gives the current directory as a string. It works very similar to `pwd`(print working directory) in some systems. The details of `sys` module can be found at:

- `sys` module: `https://docs.python.org/2/library/sys.html`.

3.2 Information of `wav` file

In this example, we show how to read some important information from the header of a wav file using `wave` module.

After `open` the `wav` file by

```
wf = wave.open( str_wavfile_path, 'rb')
```

we use the following codes:

```
# read the properties of the wav file
num_channel = wf.getnchannels()          # number of channels
fs = wf.getframerate()                   # sampling rate
length_signal = wf.getnframes()          # signal length
width = wf.getsampwidth()                # byte per frame
```

to read:

1. number of channels,
2. sampling rate (frames per second),
3. signal length and width (how many bytes per sample),

from the file header. More details of `wave` module can be found at:

- `wave` module: `https://docs.python.org/2/library/wave.html`.

Assignments 3:

1. Record a wav file of your own voice with one channel (mono), 16 kHz sampling rate and 16-bit integer format for a few seconds. (Students may consider to use `Audacity` or some equivalent audio editing softwares.)

2. Write a Python script using `print` command to print out the priori information (as `string`) about your wav file, and then to print out the information read from the wav file, and see if they are identical. (Deliver your source code with your recorded file.)

3. What is the read `width` of your 16-bit wav file?
   Record wav files with identical settings but using 8-bit and 32-bit formats.
   What are the read `width`'s of these files?

4. Give your comments about the relation of the `width` value to digital samples in a wav file.