

CS 3813/718 Fall 2012

# Python Programming

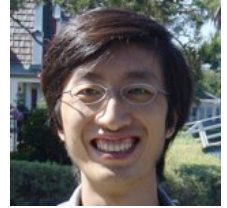
Professor Liang Huang

[huang@cs.qc.cuny.edu](mailto:huang@cs.qc.cuny.edu)

<http://vision.cs.qc.cuny.edu/huang/python-2012f/>

# Logistics

- Lectures: TTh 9:25-10:40 am SB B-141
- Personnel
  - Instructor Prof. Liang Huang [huang@cs](mailto:huang@cs)
  - TA Kai Zhao [z.kaayy@gmail](mailto:z.kaayy@gmail)
  - Admin Xiuyi Huang [xiuyi@cs](mailto:xiuyi@cs)
- Office Hours (subject to change!)
  - LH -- SB A-328 T 4-5 pm, Th 10:50-11:30am
  - KZ -- SB A-207 T 10:50-11:30 am, Th 4-5pm
  - additional office hours available before quizzes/exams



# Logistics - cont'd

- Homepage [vision.cs.qc.cuny.edu/huang/python-2012f/](http://vision.cs.qc.cuny.edu/huang/python-2012f/)
  - schedule, syllabus, homework, handouts, etc.
- Newsgroup [cs3813780-qc@googlegroups.com](mailto:cs3813780-qc@googlegroups.com)
  - announcements, Q/A --> post here first!
- Course Email [cs3813780.qc@gmail.com](mailto:cs3813780.qc@gmail.com)
  - to reach both the instructor and the TA
  - don't email us individually
- Blackboard -- the [2<sup>nd</sup>] worst software I ever used!
  - grades and electronic submissions

# Grades

- Programming Homework:  $5\% + 7\% + 8\% = 20\%$ 
  - electronic submission; work **individually**: only high-level discussions are OK, and you have to declare who you discussed with
  - grading is mostly black-box: you should follow **strict I/O formats!**
  - please work on Linux or Mac; we don't support Windows users
  - late penalty: you can submit **only one** HW late for **32 hours**.
- Quizzes:  $(7+3)\% \times 3 = 30\%$  and Midterm:  $(15+5)\% = 20\%$
- Final Programming Project: 22% (project designed by me)
- Class Participation: 8%
  - asking/answering questions in class and newsgroup
  - catching/fixing bugs in slides/exams/hw & other suggestions

# Academic Integrity

- an automatic F if you're caught on any of these:
  - copying another person's code for HW/Project
  - copying code from online resources for HW/Project
  - discussions with others for take-home quiz/exams
  - cheating during quizzes/exams
  - any other cheating behavior defined by University
- catching cheating is easier than you thought! :-)
- I will report every single case to the University

# Textbooks (for reference)

- Textbooks

*but we will not follow any textbook.*

- *How to Think Like a Computer Scientist: Learning Python*

- by Allen B. Downey, Jeffrey Elkner and Chris Meyers

- strongly recommended!

- NLTK textbook by Steve Bird and Ed Loper (optional)

- Tutorials

- (Official) *Python Tutorial*

- by Guido van Rossum (inventor of Python)

- *A Quick, Painless Tutorial on the Python Language*

- by Norm Matloff

# Course Outline

## 1. Python Basics

- Syntax, Control Flow -- quiz 1
- Basic Data Structures (list, dict, tuple, ...) -- hw 1

## 2. Object-Oriented and Functional Programming

- OOP (Objects, Inheritance, ...) -- quiz 2
- FP (map, filter, reduction, iters, lambdas) -- hw 2

## 3. Python for Algorithmic Problem Solving -- midterm

## 4. Combining Python and C/C++/Java -- hw3

## 5. Basic Text Processing using Python -- quiz 3

# Why Python?

- Because it's easy and great fun!
  - less than 10 years old, yet very popular now
    - a wide-range of applications, esp. in AI and Web
  - extremely easy to learn
    - many schools have shifted their intro-courses to Python
  - fast to write
    - much shorter code compared to C, C++, and Java
  - easy to read and maintain
    - more English-like syntax and a smaller semantic-gap



**On to Python...**

# “Hello, World”

- C

```
#include <stdio.h>

int main(int argc, char ** argv)
{
    printf("Hello, World!\n");
}
```

- Java

```
public class Hello
{
    public static void main(String argv[])
    {
        System.out.println("Hello, World!");
    }
}
```

- now in Python

```
print "Hello, World!"
```

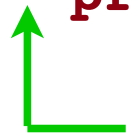
# Printing an Array

```
void print_array(char* a[], int len)
{
    int i;
    for (i = 0; i < len; i++)
    {
        printf("%s\n", a[i]);
    }
}
```

has to specify len,  
and only for one type (char\*)

C

```
for element in list:
    print element
```

 only indentations  
no { ... } blocks!

or even simpler:

```
print list
```

```
for ... in ...:
    ...
```

no C-style for-loops!

```
for (i = 0; i < 10; i++)
```

Python

# Reversing an Array

```
static int[] reverse_array(int a[])
{
    int [] temp = new int[ a.length ];
    for (int i = 0; i < len; i++)
    {
        temp [i] = a [a.length - i - 1];
    }
    return temp;
}
```

Java

```
def rev(a):
    if a == []:
        return []
    else:
        return rev(a[1:]) + [a[0]]
```

def ...(...):  
...

no need to specify  
argument and return types!  
python will figure it out.  
(dynamically typed)

or even simpler:

a without a[0]      singleton list

a.reverse() ← built-in list-processing function

Python

# Quick-sort

```
public void sort(int low, int high)
{
    if (low >= high) return;
    int p = partition(low, high);
    sort(low, p);
    sort(p + 1, high);
}

void swap(int i, int j)
{
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}

int partition(int low, int high)
{
    int pivot = a[low];
    int i = low - 1;
    int j = high + 1;
    while (i < j)
    {
        i++; while (a[i] < pivot) i++;
        j--; while (a[j] > pivot) j--;
        if (i < j) swap(i, j);
    }
    return j;
}
```

Java

```
def sort(a):
    if a == []:
        return []
```

else:

```
    pivot = a[0]
```

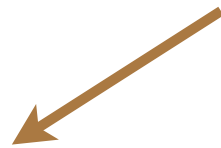
```
    left = [x for x in a if x < pivot]
```

```
    right = [x for x in a[1:] if x >= pivot]
```

```
    return sort(left) + [pivot] + sort(right)
```

how about `return [sort(left)] + [pivot] + [sort(right)]` got error?? 13

$\{x \mid x \in a, x < pivot\}$



smaller semantic-gap!

Python

# Python is...

- a scripting language (strong in text-processing)
  - interpreted, like Perl, but much more elegant
- a **very** high-level language (closer to human semantics)
  - almost like pseudo-code!
- procedural (like C, Pascal, Basic, and many more)
- but also object-oriented (like C++ and Java)
- and even functional! (like ML/OCaml, LISP/Scheme, Haskell, etc.)
- from today, you should use Python for everything
  - not just for scripting, but for serious coding!

**Let's take a closer look...**

# Python Interpreter

- Three ways to run a Python program

1. Interactive `>>> for i in range(5):`  
`... print i,`  
`...`   
`0 1 2 3 4`

- like DrJava

2. (default) save to a file, say, `foo.py`

- in command-line: `python foo.py`

3. add a special line pointing to the default interpreter

- add `#!/usr/bin/env python` to the top of `foo.py`
- make `foo.py` executable (`chmod +x foo.py`)
- in the command-line: `./foo.py`



# The right version of Python

- we will use the latest version 2.7 (e.g. 2.7.3)
- Python 3.x is a very different experimental branch...
- your default machine is “`cs12.cs.qc.cuny.edu`”, where your default “`python`” is already 2.7
- or you can install 2.7 on your own mac/windows
  - TA will help you with installations and versions

```
bash-2.0$ python
Python 2.7.1 (#1, Jan 22 2010, 18:59:00)
[GCC 3.3 20030304 (Apple Computer, Inc. build 1495)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
[<lhuang@Mac OS X:~>] which python
/Library/Frameworks/Python.framework/Versions/2.7/bin/python
```