CIS192 Python Programming
Introduction

Eric Kutschera
University of Pennsylvania

January 16, 2015
Outline

1. Logistics
   - Rooms and Office Hours
   - Grading
   - Class Materials

2. Python
   - What is Python
   - The Basics
This class is
- 0.5 Credits
- Designed to teach something practical

You should take this class if
- You have done some programming before
- You are relatively new to Python
Class: CIS 192 Python Programming
  - Room: Towne 303
  - Time: Fridays 1:30 - 2:50

Instructor: Eric Kutschera
  - Senior Undergrad in CS (Not a professor)
  - OH: TBD

TAs
  - Guillermo Gutierrez: OH TBD
  - David Xu: OH TBD
Class: CIS 19X Shared Lecture
- Room: Berger (Skirkanich Auditorium)
- Time: Tuesdays 6:00 - 7:30
- Only a few meetings per semester
- Not required to attend

Instructor: Swapneel Sheth
- Ph.D. in CS
Outline

1. Logistics
   - Rooms and Office Hours
   - Grading
   - Class Materials

2. Python
   - What is Python
   - The Basics
Grade Breakdown

- Homework: 70%
  - 1 per week
  - Programming assignments

- Final Project: 25%
  - Anything you want (10 hr of work)
  - Individually or in a pair

- In Class: 5%
  - Participation/Attendance/Quizzes/Piazza

- Late Policy
  - Must notify me by email
  - Drop one homework
Homeworks

- Python 3 (3.42 or latest stable)
- Bitbucket for posting and submission
  - Details posted on Course Website
- Graded for correctness (80%) and style (20%)
  - Automated test cases
  - Automated PEP8 checker
  - Manual style grading
- Due Sunday Nights at 23:59
- HW1 due next Sunday January 25
- Academic Integrity
  - The Office of Student Conduct
1. Logistics
   - Rooms and Office Hours
   - Grading
   - Class Materials

2. Python
   - What is Python
   - The Basics
No text book
Python Official Documentation
In Class Slides and Code
  Available on Course Website
Piazza
Linux is Recommended
- Homework will be tested on a Linux machine
- Ubuntu Virtual Machine or Eniac

Editor
- Can use anything you want
- Instructions for a good Emacs Python config is on the website
Outline

1 Logistics
   - Rooms and Office Hours
   - Grading
   - Class Materials

2 Python
   - What is Python
   - The Basics
Python is Easy

I learned it last night! Everything is so simple! Hello world is just print "Hello, world!"

I dunno... dynamic typing? Whitespace? Come join us! Programming is fun again! It's a whole new world up here! But how are you flying?

I just typed import antigravity. That's it? ... I also sampled everything in the medicine cabinet for comparison. But I think this is the python.
Skating uphill like this is amazing. Years of gliding downhill and pushing uphill, and now suddenly it's gliding both ways.

It's like going from C to Python. You don't realize how much time you were spending on the boring parts until you don't have to do them anymore.

But coding C or assembly makes you a better programmer. Maybe the boring parts build character.

Yeah... but it depends how you want to spend your life. See, my philosophy is...
History

- Designer: Guido van Rossum
  - Benevolent Dictator For Life (BDFL)
- Named after Monty Python’s Flying Circus
- Multi-Paradigm: Object-Oriented, Functional, Imperative ...
- Strongly and Dynamically Typed
- Whitespace delimited blocks
- Garbage Collected
- Main Implementation: CPython
  - Compiles to bytecode
  - Interprets bytecode
Philosophy

The Zen of Python
- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability Counts

Other ideas
- There should be one obvious way to do it
- Clarity over speed
- We’re all consenting adults here
Outline

1 Logistics
   - Rooms and Office Hours
   - Grading
   - Class Materials

2 Python
   - What is Python
   - The Basics
Read Evaluate Print Loop

Type “Python” at the terminal

```
$ python
Python 3.4.2 (default, Oct 8 2014, 13:44:52)
[GCC 4.9.1 20140903 (prerelease)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World!')
Hello World!
>>> 
```

Test out language behavior here

Get information with `dir()`, `help()`, `type()`
Identifiers, Names, Variables

- All 3 mean the same thing
- [A-Za-z0-9_] First character cannot be a number
- Variable naming convention
  - Functions and variables: lower_with_underscore
  - Constants: UPPER_WITH_UNDERSCORE
- $x = 1$
- $y = x$
- $x = 'a'$
- $x = 1$
- $y = x$
- $x = 'a'$
• $x = 1$
• $y = x$
• $x = 'a'$

Diagram:
- Node $X$ is connected to node $1$.
- Node $Y$ is connected to node $a$.
Python treats all data as objects

- **Identity**
  - Memory address
  - Does not change

- **Type**
  - Does not change

- **Value**
  - Mutable $\rightarrow [1,2]$
  - Immutable $\rightarrow (1,2)$
Math

- **Literals**
  - Integers: 1, 2
  - Floats: 1.0, 2e10
  - Complex: 1j, 2e10j
  - Binary: 0b1001, Hex: 0xFF, Octal: 0o72

- **Operations**
  - Arithmetic: +, -, *, /
  - Power: **
  - Integer division: //
  - Modulus: %
  - Bitwise: «, », &, |, ^
  - Comparison: <, >, <=, >=, ==, !=

- **Assignment Operators**
  - +=, *=, /=, &=, ...
Strings

• Can use either single or double quotes
• Use single to show double flip-flop "" \(\rightarrow\) ‘ and "" \(\rightarrow\) "
• Triplequote for multiline string
• Can concat strings by separating string literals with whitespace
• All strings are unicode
• Prefixing with r means raw. No need to escape: r\n’
Conditionals

- One if block
- Zero or more elif blocks
- Zero or one else block
- Booleans: True False
Sequences

- **Immutable**
  - Strings, Tuples, Bytes
- **Mutable**
  - Lists, Byte Arrays
- **Operations**
  - `len()`
  - Indexing
  - Slicing
  - `in`
  - `not in`
Range

- Immutable sequence of numbers
  - `range(stop)`, `range(start, stop)`
  - `range(start, stop, step)`
- start defaults to 0
- step defaults to 1
- All numbers in [start,stop) by incrementing start by step
- Negative steps are valid
- Memory efficient: Calculates values as you iterate over them
Loops

- For each loops
  - Iterate over an object
- While loops
  - Continues as long as condition holds
- Both
  - else: executes after loop finishes
  - break: stops the loop and skips the else clause
  - starts the next iteration of the loop
Functions

- Functions are first class
  - Can pass them as arguments
  - Can assign them to variables
- Define functions with a `def`
- `return` keyword to return a value
- If a function reaches the end of the block without returning
  It will return `None` (null)
Types

- Every object has a type
- Inspect types with `type(object)`
- `isinstance(object, type)` checks type hierarchy
- Types can be compared for equality but usually want `isinstance()`
- Some types:
  - `int`, `float`, `complex`
  - `str`, `bytes`, `tuple`
  - `list`, `bytearray`
  - `range`, `bool`, `None`
  - `function`
Imports

- Allow use of other python files and libraries
- imports: `import math`
- Named imports: `import math as m`
- Specific imports: `from math import pow`
- Import all: `from math import *`