CIS192 Python Programming

Introduction

Raymond Yin

University of Pennsylvania

August 31, 2016
Outline

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

2 Python
   • What is Python?
   • The Basics
What’s CIS 192?

- CIS 19X courses
  - 0.5 Credits each
  - Designed to teach practical skills
- CIS 192: Python Programming
  - Powerful scripting language used in academia and industry
  - Simple to read and write in
- Take this class if
  - You have some programming experience
  - You are relatively new to Python
Class: CIS 192 Python Programming
- Listed as CIS 192 201
- Room: Towne 303
- Time: Wednesdays, 12:00 - 1:30pm

Instructor: Raymond Yin
- Undergrad in CS (Not a professor)

TAs
- Harry Smith
- Alex Frias
Class: CIS 19X Shared Lecture
- Room: Towne 100
- Time: Tuesdays, 6:00 - 7:30pm
- General, useful info: command line intro, Git, overview of the Internet

Instructor: Swapneel Sheth
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
  - Individually or in a pair
- **In Class**: 5%
  - Participation/Attendance/Piazza
- **Late Policy**
  - Drop one homework (lowest grade)
Homeworks

- Python 3 (3.42 or latest stable)
- Submit on Canvas!
- Graded for correctness (80%) and style (20%)
- Due Sunday nights at 11:59pm
- HW1 due next Sunday, September 11

Academic Integrity

- The Office of Student Conduct
- Don’t copy-paste code from other people
- Don’t have mid-level discussions
  - High-Level: What are the pros/cons of using Python for X?
  - Low-Level: What is the syntax for decorators?
  - Mid-Level: How did you do HW 1?
Outline

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

2. **Python**
   - What is Python?
   - The Basics
Reading

- No text book!
- Python Official Documentation
- In-class slides and code (available on CIS 192 website)
- Piazza
Programming Environment

- Unix system recommended
  - Ubuntu Virtual Machine / Eniac
- Editor
  - Can use anything you want (Sublime Text, Atom, vim, emacs, etc.)
  - IDEs also available: PyDev for Eclipse, PyCharm
  - *Set your editor to interpret tabs as four spaces*
    - Python is whitespace-sensitive
Outline

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

2 Python
   • What is Python?
   • The Basics
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.
Easy to Use

Skating uphill like this is amazing. Years of gliding downhill and pushing uphill, and now suddenly it's gliding both ways.

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—
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 5 2014, 20:42:22)
[ GCC 4.2.1 (Apple Inc. build 5666) (dot 3) ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> int(3.00)
3
>>> int = 5
>>> int(3.00)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
>>>
History

- Designer: Guido van Rossum
  - Benevolent Dictator For Life (BDFL)
- Multi-Paradigm: Object-Oriented, Functional, Imperative...
- Strongly and Dynamically Typed
- Whitespace delimited blocks
- Garbage Collected
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 marginal efficiency
  - 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 (a.k.a. an interpreter)

Type “Python” at the terminal

```
ryin@Raymonds-MBP:~$ python3
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  5 2014, 20:42:22)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
>>> print('sup everyone')
sup everyone
>>> 5 + 6
11
``` 

Test out language behavior here

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

- All 3 mean the same thing
- Variable naming convention
  - Functions and variables: lower_with_underscore
    - my_num = 5
  - Constants: UPPER_WITH_UNDERSCORE
    - SECONDS_PER_MINUTE = 60
- $x = 1$
- $y = x$
- $x = 'a'$
- $x = 1$
- $y = x$
- $x = 'a'$

![Diagram](image_url)
Binding

- $x = 1$
- $y = x$
- $x = 'a'$
Objects

- Python treats all data as objects
- **Identity**
  - Memory address
  - Does not change
- **Type**
  - Does not change
- **Value**
  - Mutable: value can be changed (e.g. `[1, 2]`)
  - Immutable: value cannot be changed after creation (e.g. `(1, 2)`)
- **Equality**
  - Use `is` for referential equality (do x and y point to the same object?)
  - Use `==` for structural equality (are x and y equal based on object’s `__eq__` method?)

Raymond Yin (University of Pennsylvania)
Every object has a type

Inspect types with \texttt{type(object)}

\texttt{isinstance(object, type)} checks type hierarchy

Types can be compared for equality, but you usually want \texttt{isinstance}

Some types:
- int, float
- str
- tuple, list, dict
- range, bool, None
- function
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**
  - += *= /= &= ...
  - No ++ or --
Booleans

- True and False
- Boolean operators: and or not
- Any object can be tested for truth value for use in conditionals, or as operands of the above Boolean operations.
- "Falsy"
  - None
  - 0
  - 0.0
  - Any empty string/sequence/collection ([], (), etc.)
Strings

- Can use either single or double quotes
- Use single to show double flip-flop "'" → ’ and "" → "
- 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 characters: r\n
Conditionals

- One **if** block
- Zero or more **elif** blocks
- Zero or one **else** block
Sequences

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

- Immutable sequence of numbers
- \texttt{range(stop)}, \texttt{range(start,stop)}
  \texttt{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** (for item in my_list:)
  - Iterate over an object
- **While loops** (while some_condition:)
  - 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
  - They’re objects, too!
  - 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)
**Importing Modules**

- 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 *`