Welcome to Comp Sci 368-4!  
Introduction to Scripting for CHTC  
Overview, Course Mechanics,  
Introduction to Python  
http://pages.cs.wisc.edu/~cs368-4/
Introductions
Instructor

Tim Cartwright
cat@cs.wisc.edu
or (26)2-4002 but email is best

Background

• B.S., UW–Madison & Ph.D., John Hopkins (Cognitive Science)
• Software developer, educator, consultant
• **Staff** on CHTC team *(not Professor)*
• Open Science Grid (OSG) software and education
Course Objectives

Write basic code in Python
Solve scientific computing problems with scripting
Use Center for High Throughput Computing (CHTC)
Automate common scientific computing workflows
## Syllabus

<table>
<thead>
<tr>
<th>Python</th>
<th>CHTC</th>
<th>Python for CHTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Overview; Intro</td>
<td>9: Intro; Running Jobs</td>
<td>12: Scripted Workflows I</td>
</tr>
<tr>
<td>to Python</td>
<td>10: More Complex Jobs</td>
<td>13: Scripted Workflows II</td>
</tr>
<tr>
<td>2: Basic Syntax</td>
<td>11: Workflows with DAGMan</td>
<td>14: Wrapper Scripts</td>
</tr>
<tr>
<td>3: Collections</td>
<td></td>
<td>15: Science Code in Python</td>
</tr>
<tr>
<td>4: I/O &amp; Exceptions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5: Data, Functions,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>and Classes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6: Modules and the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard Library</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7: Regular Expressions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8: System Interaction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

2012 Fall
Course Philosophy

Learn a new skill

Learn by doing

Learn to fish
My Suggestion:

Write code.
At least a little.
Every day.
Play around!
Course Mechanics
Credit and Homework

• Credit
  – Course offered as credit/no credit
  – All points come from homework (no exam)

• Homework
  – Short coding or CHTC assignment
  – Every day (except last day): 14 total
  – Due by 1:30 p.m. of next class (email tolerated)
  – *No late assignments accepted at all*
  – Each homework given 0, 1, or 2 points
  – Need 18 points (64%) to get credit for the course
## Homework Points

<table>
<thead>
<tr>
<th>Pts</th>
<th>Reason</th>
</tr>
</thead>
</table>
| 2   | • turned in on time, AND  
• code runs, AND  
• solution is correct or nearly so, AND  
• demonstrates real effort |
| 1   | • turned in on time, AND  
• partial solution, may not actually run, AND  
• demonstrates some effort (my discretion) |
| 0   | • late, OR  
• *is plagiarized (= Academic Misconduct)*, OR  
• does not demonstrate any real effort |
Mailing List

compsci368-4-f12-hhh@lists.wisc.edu

• Goes to your @wisc.edu account

• Check spam filters

• Post questions, comments, and discoveries!
  – Except direct homework questions (see rules)
  – All else is fair game
Office Hours

Computer Sciences 4265 (Tim’s office)

Days and times: Doodle poll today!

Other times available by appointment (email)
Python Resources

• Book: *Learning Python* (4th Ed.)
  – Available FREE online via MadCat
  – Not in the UBS textbook area
  – Note Python version info

• Python documentation
  [http://docs.python.org/release/2.4.3/](http://docs.python.org/release/2.4.3/)

• Python 2.4 Quick Reference
  [http://rgruet.free.fr/PQR24/PQR2.4.html](http://rgruet.free.fr/PQR24/PQR2.4.html)
Machines

- Computer Systems Lab (CSL) accounts
  - Old accounts may still be active
  - Otherwise, see login screen on instructional machines
  - Problems? Stop by CompSci 2350 (the CSL), or email lab@cs.wisc.edu

- Personal machine OK for Python, but check version

- Will get CHTC Linux account later
Scripting in Python
Why Scripting?

• Abstracts over low-level details
• Rapid development
• Easy to understand and change
• Pervasive
Why Python?

• Powerful: Has everything you need
• Clear: Modern, clean design
• Highly portable: Runs nearly everywhere
• Widely used in scientific computing
Python vs. C++: Print a File

**Python**

```python
inStream = open("Data.txt", "r")
for line in inStream:
    print line
```

**C++**

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
    ifstream InStream;
    string line;
    InStream.open("Data.txt");
    getline(InStream, line);
    while (! InStream.eof()) {
        cout << line << endl;
        getline(InStream, line);
    }
}
```
Python Versions

• \( \leq 2.3 \) considered very old, not recommended

• 2.4 – 2.6 still very common
  – Red Hat Linux 5 has 2.4.3 \((CHTC\ submit\ machine)\)
  – Red Hat Linux 6 has 2.6.6 \((instructional\ machines)\)
  – Debian 6 (“squeeze”) has 2.6.6

• 2.7 is current, but end-of-line for 2.x
  – Mac OS X 10.8 (“Mountain Lion”) has 2.7.2

• 3.x is the future — but is not backward compatible

http://wiki.python.org/moin/Python2orPython3
Running Python
Interactive Python

$ python
Python 2.4.3 (#1, Dec 11 2006, 11:39:03)
[GCC 4.1.1 20061130 (Red Hat 4.1.1-43)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 

• Great for trying things out
• Cannot save state
• ∴ Not great for reuse
Running Python Scripts

- **Linux / Unix**
  - `python filename.py`
  - `chmod 0755 filename.py ./filename.py`

- **Mac OS X**
  - Use Terminal (built-in app), same as above

- **Windows**
  - Available from main Python website
  - Not officially supported in the course
Python Bootcamp … Starts Now!
## Numbers

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integers</td>
<td>42, -13, 0, 123456</td>
</tr>
<tr>
<td>Really long integers</td>
<td>12345678901234567890L</td>
</tr>
<tr>
<td>Floating-point numbers</td>
<td>-0.5, 3.141, 2.7182818</td>
</tr>
<tr>
<td>Exponential notation</td>
<td>2.998E8, 6.022e23, 6.626e-34</td>
</tr>
<tr>
<td>Integers in octal (base 8)</td>
<td>0177, 0377</td>
</tr>
<tr>
<td>Integers in hex (base 16)</td>
<td>0x3A, 0Xff, 0x12ab</td>
</tr>
<tr>
<td>Complex numbers</td>
<td>3+4j, -3.5+2.0j, 6J</td>
</tr>
</tbody>
</table>
(Some) Operations on Numbers

- **group**: \( 24 \times (3 + 4) \)
- **calculate**
  - \( \text{abs}(-24.33) \)
  - \( \text{min}(3, 4, 1, \ldots, 8) \)
  - \( \text{max}(3, 4, 1, \ldots, 8) \)
  - \( \text{round}(1234.56) \)
  - \( \text{round}(1234.5678, 2) \)
- **negate**: \(-x\)
- **exponentiate (power of)**
  - \( 2^8 \)
  - \( \text{pow}(2, 8) \)
- **multiply & divide**
  - \( 42 \times 3.141 \)
  - \( 5.0 \div 2 \quad \text{or} \quad 5.0 \div 2.0 \)
  - \( 23 \mod 5 \)
- **add & subtract**
  - \( 12 + 34 \)
  - \( 2011 - 1970 \)
Strings

string (single or double quotes)

's'Hello, "world"!'  "Hello, 'world'!"

really long strings

'"""Really long strings can span multiple lines, etc. Newlines are kept."

escapes

'one line
second line'

'"tab\tseparated\tdata"

"has \"quotes\" inside"

raw string

r'C:\new\test.txt'
(Some) Operations on Strings

concatenate

\['Hello, ' + "world!\n"\]  \['Hello, world!\n'\]

repeat

\['-' * 15\]

\['---------------\]

index

"hello"[1]

\['e'\]

"hello"[-1]

\['o'\]

slice

"hello"[1:4]

\['ell'\]

functions

len('hello')

5

'hello \n'.strip()

'hello'

'CrAzY'.lower()

'crazy'

'Hello'.find('el')

1

'hello'.endswith('lo')

True

'123.0'.isdigit()

False
String Formatting

"... %d ... %f ... %s ..." % (42, 3.1, 'text')

integer  'Count: %d' % (123)  'Count: 123'

float    'Mean: %f' % (6.23 / 17)  'Mean: 0.366471'

string   'Hello, %s!' % ("Tim")  'Hello, Tim!'

% character 'C = %f%%' % (5 / 2.0)  'C = 2.500000%'

multiple 'L: %f%s' % (2.6, 'm')  'L: 2.600000m'

advanced  '%6.2f' % (1.23456789)  '  1.23'

See book or online resources for lots more!
Strings $\neq$ Integers $\neq$ Floats

```python
>>> 1 + '1'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Convert to `integer`

- `int('5')`
- `int(5.1234)`

Convert to `float`

- `float('5')`
- `float(5)`

Convert to `string`

- `str(5)`
- `str(5.1234)`
Wrap Up
Homework

• Part 1
  – Visit course website, find homework #1 in syllabus
  – Run script given there; *print and turn in output*

• Part 2
  – Run interactive Python session
  – Play around with numbers and strings
  – *Print and turn in* interesting discoveries
  – See homework #1 for details
http://pages.cs.wisc.edu/~cs368-4/