Welcome to CS 368-004!
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>1: Overview; Intro to Python</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2: Basic Syntax</td>
</tr>
<tr>
<td></td>
<td>3: Collections</td>
</tr>
<tr>
<td></td>
<td>4: I/O &amp; Exceptions</td>
</tr>
<tr>
<td></td>
<td>5: Class, Methods, and Modules</td>
</tr>
<tr>
<td></td>
<td>6: Regular Expressions</td>
</tr>
<tr>
<td></td>
<td>7: System Interaction</td>
</tr>
<tr>
<td></td>
<td>8: Standard Library</td>
</tr>
<tr>
<td>CHTC</td>
<td>9: Intro; Running Jobs</td>
</tr>
<tr>
<td></td>
<td>10: More Complex Jobs</td>
</tr>
<tr>
<td></td>
<td>11: Workflows with DAGMan</td>
</tr>
<tr>
<td>Scripting for CHTC</td>
<td>12: Scripted Workflows I</td>
</tr>
<tr>
<td></td>
<td>13: Scripted Workflows II</td>
</tr>
<tr>
<td></td>
<td>14: Wrapper Scripts</td>
</tr>
<tr>
<td></td>
<td>15: Scientific Computing</td>
</tr>
<tr>
<td>Bonus Day?</td>
<td>16: Development Process</td>
</tr>
</tbody>
</table>
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 bonus day and last day): 14 total
  – Due by 1:30 p.m. of next class (email OK)
  – 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 at least some effort |
| 0   | • late, OR  
     | • is plagiarized, OR  
     | • does not demonstrate any real effort |
Mailing List

compsci368-4-f11-hhh@lists.wisc.edu

• Goes to your @wisc.edu account
• Check spam filters
• Post interesting questions, comments, and findings!
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* (3rd Ed.)
  – Available FREE online via MadCat
  – Not in the UBS textbook area

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

• Python 2.4 Quick Reference *(down today?)*
  [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

• Own machine OK for Python, but check version

• Will get CHTC account later
Scripting in Python
Why Scripting?

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

• Has everything you need
• Powerful and clear
• Highly portable
• Widely used in scientific computing
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  (*instructional machines*)
  – Red Hat Linux 6 has 2.6.6
  – Debian 6 (“squeeze”) has 2.6.6

• 2.7 is current, but end-of-line for 2.x
  – Mac OS X 10.7 (“Lion”) has 2.7.1

• 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 appropriate for reuse
Running Python Scripts

• Linux / Unix
  – python filename.py
  – chmod 0755 filename.py
  . /filename.py

• Mac OS X
  – Use Terminal, same as above

• Windows
  – download ActiveState Perl
  – not officially supported in the course
Introduction to Python
# Numbers

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

- **Group**: 24 * (3 + 4)
- **Calculate**:
  - abs(-24.33)
  - min(3, 4, 1, ..., 8)
  - max(3, 4, 1, ..., 8)
  - round(1234.56)
  - round(1234.5678, 2)
- **Negate**: -x
- **Exponentiate (power of)**: 2 ** 8
  - pow(2, 8)
- **Multiply & Divide**:
  - 42 * 3.141
  - 5.0 / 2  or  5.0 // 2.0
  - 23 % 5
- **Add & Subtract**:
  - 12 + 34
  - 2011 - 1970
Strings

- **String**
  - (single or double quotes)
  - 'Hello, "world"!'
  - "Hello, 'world'!"

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

- **Escapes**
  - 'one line\nsecond 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

'%.2f' % (1.23456789)

' 1.23'

See book or online resources for lots more!
Strings ≠ Integers ≠ Floats

>>> 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; \emph{print and turn in output}

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