Day 2: Basic Syntax

Suggested reading: *Learning Python* (3rd Ed.)

Chapter 10: Introducing Python Statements
Chapter 11: Assignment, Expressions, and print
Chapter 12: if Tests
Turn In Homework
Housekeeping

• If you have *not* enrolled:
  – Please consider enrolling or auditing
  – You may attend regardless
  – I cannot provide help (homework, office hours, …)
  – I can add you to the mailing list (email me)
Office Hours

Mondays, 3–4 p.m.
Thursdays, 3–4 p.m.

Computer Sciences 4265

Always best to email first
Write code.
At least a little.
Every day.
Play around!
Basic Python Syntax (cont’d)
From Interactive To Scripts

- Numbers/strings and operations form *expressions*
- Python computes the *value* of an expression

```
'Answer: ' + str(6 * 7)

'Answer: 42'
```

- Interactive Python prints values automatically
- Scripted Python does not
print

**print expression**

- Prints a value (*to standard output*)
- Separate items with comma (prints space between)
- Prints newline at end
- Suppress newline with trailing comma

```python
print 5
print 'Result:', 5.0 / 2.5, 'm/s'
print 'Result of complex calculation:',
print round(532.2 * (4.2 + 1.2), 1)
```
Hello World

#!usr/bin/python

# This line is a comment
print 'Hello, world!'

# Continued line (put nothing after \\)
print '2pi = ' + \
str(2.0 * 3.14159) # 2 * pi
Variables
Variables

```python
variable_name_1 = 42
variable_name_2 = variable_name_1 + 1
```

- Create by assigning a value
- Must create before use
- Subsequent assignment changes value
- Replaced with current value when used

```python
variable = 0
variable = variable + 1
variable = 'Tim'
greeting = 'Hello, ' + variable
```
The Elephant in the Room
Python Object Model

- **All** variables refer to objects
- Objects = Data (in memory) + Operations
- Assignment binds a variable name to an object
- Types live in objects, not variables
- No more references? Python can remove object

```
x = 42
y = x
x = 'hi'
y = x
```

<table>
<thead>
<tr>
<th>x = 42</th>
<th>y = x</th>
<th>x = 'hi'</th>
<th>y = x</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>int</td>
<td>int</td>
<td>str</td>
<td>int</td>
</tr>
<tr>
<td>42</td>
<td>42</td>
<td>hi</td>
<td>42</td>
</tr>
</tbody>
</table>

2011 Fall

Cartwright
Types

`type(object)`

- Python can tell you type of object
- Example of *introspection*

```
type(42)        =>  <type 'int'>
type(3.141)     =>  <type 'float'>
type('hi')      =>  <type 'str'>
```

```
x = 5.0 / 2

x = 5.0 / 2
```

```
type(x)         =>  <type 'float'>
type(type(42))  =>  <type 'type'>
```
None

• Special object which means “no value”
• In other languages: undef, nil, null, …
• In Python, still an object…

```python
>>> x = None
>>> x
None
>>> print x
None
>>> type(x)
<type 'NoneType'>
```
Built-In Help I

```python
>>> dir(str)
[...,'capitalize', 'center', 'count', 'decode',
'encode', 'endswith', 'expandtabs', 'find', 'index',
'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'replace', 'rfind', 'rindex', 'rjust', 'rsplit',
'rstrip', 'split', 'splitlines', 'startswith', 'strip',
'swapcase', 'title', 'translate', 'upper', 'zfill']
```
Built-In Help II

help(something)

- Shows built-in documentation
- Works on objects, types, and their operations

```python
>>> help(str.lower)
...
lower(...)
    S.lower() -> string

Return a copy of the string S converted to lowercase.
```
Back to Variables
Assignment

a = 42
a += 1  # a = 43
a -= 3  # a = 40
a *= 2  # a = 80
a /= 8  # a = 10
...

• a += 1 is slightly more efficient than a = a + 1
• += and *= work on strings, too

a = b = c = 0

• OK but not recommended
Basic Input

```
a = raw_input()
a = raw_input('Enter a number: ')
```

- Gets input from user
- Strips trailing newline automatically
- Result is always a string — convert if needed

```
name = raw_input('Enter your name: ')
print 'Hello, %s!' % name
age = raw_input('Enter your age: ')
print 'Age next year: %d' % (int(age) + 1)
```
Comparisons
Booleans

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
</table>

```python
>>> x = True
>>> x
True
>>> print x
True
>>> print False
False
>>> type(False)
<type 'bool'>
```
## Boolean Operations

<table>
<thead>
<tr>
<th>x</th>
<th>not x</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x or y</th>
<th>x and y</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
Comparison Operators

- same values  ==
- not same values  !=
- same object  is
- not same object  is not
- less than  <
- less than or equal to  <=
- greater than  >
- greater than or equal to  >=

- All comparison operations yield a Boolean value
- Use is/is not with None, True, and False
- Can chain inequalities: 1 < x <= 4
Conditionals

```python
if condition:
    # do when condition is True
elif other-condition:
    # do when condition is False
    # and other-condition is True
else:
    # do when all conditions are False

name = raw_input('Name? ')
if name == 'Tim Cartwright':
    print 'Instructor'
else:
    print 'Student'

student_count += 1
```

0–n times opt.
**Basic Loop**

```python
while condition:
    # do when condition is True
    # then return to top and re-evaluate
if condition-a:
    continue  # return to top now
if condition-b:
    break    # exits loop
# more stuff
```

```python
count = 0
while count < 10:
    print count
    count += 1
```
You Made It!
Other Scripting Languages

- The cellphone metaphor...

- Check for different or additional:
  - **Literals** ("/", true/false, null/nil/undef, 1_234)
  - **Operators** (===, =~)
  - **Conditionals** (elsif **vs.** elseif **vs.** else if; unless)
  - **Loops** (do ... while, unless, foreach)
  - **Block syntax** ({...} **vs.** do...end **vs.** indentation)
  - **Object syntax** (Perl...)
Homework

- Simple number-guessing game
  - You pick the number & the computer guesses
  - Seek a straightforward solution

- BE SURE TO LABEL YOUR PRINTOUT!!!

```bash
#!/usr/bin/python

# Homework for CS 368-004
# Assigned on Day 02, 2011-10-27
# Written by Your Name Here
```