Lecture 2

Variables & Assignment
## Announcements for Today

### If Not Done Already
- Enroll in Piazza
- Sign into CMS
  - Fill out the Survey
  - Complete AI Quiz
- Read the textbook
  - Chapter 1 (browse)
  - Chapter 2 (in detail)

### Lab 1
- Please stay in your section
  - If you drop, you are **stuck**
  - E-mail conflicts to Jessica
  - [jd648@cornell.edu](mailto:jd648@cornell.edu)
  - Will review by next week
- Have one week to complete
  - Fill out questions on handout
  - Show to TA before next lab
  - Show in **consulting hours**

8/28/14
Helping You Succeed in this Class

- **Consultants.** ACCEL Lab Green Room
  - Daily office hours (see website) with consultants
  - Very useful when working on assignments

- **AEW Workshops.** Additional discussion course
  - Runs parallel to this class – completely optional
  - See website; talk to advisors in Olin 167.

- **Piazza.** Online forum to ask and answer questions
  - Go here first **before** sending question in e-mail

- **Office Hours.** Talk to the professor!
  - Have decided on MW 3:45-4:45 (starts next week)
Labs vs. Assignments

**Labs**

- Held every week
- Graded on *completeness*
  - Always S/U
  - Try again if not finished
- Indirect affect on grade
  - Can miss up to 2 labs
  - After that, grade reduced
- Similar to language drills
  - Simple, but take time

**Assignments**

- Every two weeks
  - First one due Sep. 18
- Graded on *correctness*
  - Assign points out of 100
- But *first* one is for *mastery*
  - Resubmit until perfect grade
- 40% of your final grade
- Designed to be more fun
  - Graphics, game design
iClickers

• Have you registered your iclicker?
• If not, visit
  ▪  atcsupport.cit.cornell.edu/pollsrvc/
• Instructions on iClickers can be found here:
  ▪  www.it.cornell.edu/services/polling/howto-students.cfm
• Find these links on the course webpage
  ▪  Click “Texts/iClickers”
  ▪  Look under “iClickers”
Warm-Up: Using Python

• How do you plan to use Python?

A. I want to work mainly in the ACCEL lab
B. I want to use my own Windows computer
C. I want to use my own Macintosh computer
D. I want to use my own Linux computer
E. I will use whatever I can get my hands on
Type: Set of values and the operations on them

- **Type int:**
  - **Values:** integers
  - **Ops:** +, −, *, /, %, **

- **Type float:**
  - **Values:** real numbers
  - **Ops:** +, −, *, /, **

- **Type bool:**
  - **Values:** True and False
  - **Ops:** not, and, or

- **Type str:**
  - **Values:** string literals
  - Double quotes: "abc"
  - Single quotes: 'abc'
  - **Ops:** + (concatenation)

Will see more types in a few weeks
Operator Precedence

• What is the difference between the following?
  - 2*(1+3)
  - 2*1 + 3

• Operations are performed in a set order
  - Parentheses make the order explicit
  - What happens when there are no parentheses?

• Operator Precedence: The fixed order Python processes operators in absence of parentheses
Operator Precedence

- What is the difference between the following?
  - 2*(1+3)  \(\text{add, then multiply}\)
  - 2*1 + 3  \(\text{multiply, then add}\)

- Operations are performed in a set order
  - Parentheses make the order explicit
  - What happens when there are no parentheses?

- **Operator Precedence**: The *fixed* order Python processes operators in *absence* of parentheses
Precedence of Python Operators

- Exponentiation: **
- Unary operators: + –
- Binary arithmetic: * / %
- Binary arithmetic: + –
- Comparisons: < > <= >=
- Equality relations: == !=
- Logical not
- Logical and
- Logical or

- Precedence goes downwards
  - Parentheses highest
  - Logical ops lowest
- Same line = same precedence
  - Read “ties” left to right
  - Example: 1/2*3 is (1/2)*3

- Section 2.7 in your text
- See website for more info
- Major portion of Lab 1

8/28/14 Variables & Assignments 10
## Expressions vs Statements

<table>
<thead>
<tr>
<th>Expression</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Represents</strong> something</td>
<td><strong>Does</strong> something</td>
</tr>
<tr>
<td>- Python <em>evaluates it</em></td>
<td>- Python <em>executes it</em></td>
</tr>
<tr>
<td>- End result is a value</td>
<td>- Need not result in a value</td>
</tr>
<tr>
<td><strong>Examples:</strong></td>
<td><strong>Examples:</strong></td>
</tr>
<tr>
<td>- 2.3</td>
<td>- <em>print “Hello”</em></td>
</tr>
<tr>
<td>- ((3+5)/4)</td>
<td>- <em>import sys</em></td>
</tr>
</tbody>
</table>

Will see later this is not a clear cut separation
Variables (Section 2.1)

• A variable
  ▪ is a named memory location (box)
  ▪ contains a value (in the box)
  ▪ can be used in expressions

• Examples:

  Variable names must start with a letter (or _).

  Variable `x`, with value 5 (of type `int`)
  Variable `area`, with value 20.1 (of type `float`)
Variables (Section 2.1)

• A variable
  ▪ is a **named** memory location (**box**)
  ▪ contains a **value** (in the box)
  ▪ can be used in expressions

• Examples:

<table>
<thead>
<tr>
<th>5</th>
<th>Variable <strong>x</strong>, with value 5 (of type <strong>int</strong>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.1</td>
<td>Variable <strong>area</strong>, w/ value 20.1 (of type <strong>float</strong>)</td>
</tr>
</tbody>
</table>

Variable names must start with a letter (or _).

The type belongs to the value, not to the variable.
Variables (Section 2.1)

- **A variable**
  - is a **named** memory location (**box**)
  - contains a **value** (in the box)
  - can be used in expressions

- **Examples:**
  - Variable \( x \), with value 5 (of type \( \text{int} \))
  - Variable \( \text{area} \), w/ value 20.1 (of type \( \text{float} \))
Variables (Section 2.1)

- A variable
  - is a **named** memory location (**box**)
  - contains a **value** (in the box)
  - can be used in expressions

- **Examples:**
  - Variable `x`, with value 5 (of type `int`)
  - Variable `area`, w/ value 20.1 (of type `float`)

1e2 is a **float**, but e2 is a variable name
Variables and Assignment Statements

• Variables are created by **assignment statements**
  - Create a new variable name and give it a value

  \[ x = 5 \]

• This is a **statement**, not an **expression**
  - Tells the computer to DO something (not give a value)
  - Typing it into `>>>` gets no response (but it is working)

• Assignment statements can have expressions in them
  - These expressions can even have variables in them

  \[ x = x + 2 \]

Two steps to execute an assignment:
1. evaluate the expression on the right
2. store the result in the variable on the left

8/28/14
Variables and Assignment Statements

- Variables are created by assignment statements
  - Create a new variable name and give it a value
    - `x = 5`
  - This is a statement, not an expression
    - Tells the computer to DO something (not give a value)
    - Typing it into `>>>` gets no response (but it is working)
- Assignment statements can have expressions in them
  - These expressions can even have variables in them
    - `x = x + 2`

Two steps to execute an assignment:
1. evaluate the expression on the right
2. store the result in the variable on the left
Variables and Assignment Statements

- Variables are created by **assignment statements**
  - Create a new variable name and give it a value
    - The value
      \[ x = 5 \]
    - The variable
    - \[ x \]
  - This is a **statement**, not an **expression**
    - Tells the computer to DO something (not give a value)
    - Typing it into >>> gets no response (but it is working)
- Assignment statements can have expressions in them
  - These expressions can even have variables in them
    - \[ x = x + 2 \]

Two steps to execute an assignment:
1. evaluate the expression on the right
2. store the result in the variable on the left

8/28/14
Variables and Assignment Statements

- Variables are created by **assignment statements**
  - Create a new variable name and give it a value
    
    \[
    x = 5
    \]

- This is a **statement**, not an **expression**
  - Tells the computer to DO something (not give a value)
  - Typing it into `>>>` gets no response (but it is working)

- Assignment statements can have expressions in them
  - These expressions can even have variables in them
    
    \[
    x = x + 2
    \]

Two steps to execute an assignment:
1. evaluate the expression on the right
2. store the result in the variable on the left
Variables and Assignment Statements

- Variables are created by **assignment statements**
  - Create a new variable name and give it a value
    - \(x = 5\)
    - the value \(x\)
    - the variable \(5\)

- This is a **statement**, not an **expression**
  - Tells the computer to DO something (not give a value)
  - Typing it into >>> gets no response (but it is working)

- Assignment statements can have expressions in them
  - These expressions can even have variables in them
    - \(x = x + 2\)
    - the expression \(x\)
    - the variable \(2\)

Two steps to execute an assignment:
1. evaluate the expression on the right
2. store the result in the variable on the left
Variables and Assignment Statements

- Variables are created by **assignment statements**
  
  $x = 5$

- This is a **statement**, not an **expression**
  
  - Tells the computer to DO something (not give a value)
  - Typing it into `>>>` gets no response (but it is working)

- Assignment statements can have expressions in them
  
  - These expressions can even have variables in them

  $x = x + 2$

Two steps to execute an assignment:
1. evaluate the expression on the right
2. store the result in the variable on the left
Execute the statement: \( x = x + 2 \)

- Draw variable \( x \) on piece of paper:

  \[
  \begin{array}{c}
  x \\
  5 
  \end{array}
  \]
Execute the statement: \( x = x + 2 \)

- Draw variable \( x \) on piece of paper:
  \[
  \text{x} \quad 5
  \]

- Step 1: evaluate the expression \( x + 2 \)
  - For \( x \), use the value in variable \( x \)
  - Write the expression somewhere on your paper
Execute the statement: \( x = x + 2 \)

- Draw variable \( x \) on piece of paper:
  
  \[
  \begin{array}{|c|}
  \hline
  x \\
  5 \\
  \hline
  \end{array}
  \]

- Step 1: evaluate the expression \( x + 2 \)
  - For \( x \), use the value in variable \( x \)
  - Write the expression somewhere on your paper

- Step 2: Store the value of the expression in \( x \)
  - Cross off the old value in the box
  - Write the new value in the box for \( x \)
Execute the statement: \( x = x + 2 \)

- Draw variable \( x \) on piece of paper:
  
  \[
  x \quad 5
  \]

- Step 1: evaluate the expression \( x + 2 \)
  - For \( x \), use the value in variable \( x \)
  - Write the expression somewhere on your paper

- Step 2: Store the value of the expression in \( x \)
  - Cross off the old value in the box
  - Write the new value in the box for \( x \)

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Execute the statement: \( x = x + 2 \)

- Draw variable \( x \) on piece of paper:

  \[
  \begin{array}{c}
  x \\
  7
  \end{array}
  \]

- Step 1: evaluate the expression \( x + 2 \)
  - For \( x \), use the value in variable \( x \)
  - Write the expression somewhere on your paper

- Step 2: Store the value of the expression in \( x \)
  - Cross off the old value in the box
  - Write the new value in the box for \( x \)

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Execute the statement: \( x = x + 2 \)

- Draw variable \( x \) on piece of paper:
  
  \[
  \begin{array}{c}
  \text{x} \\
  7
  \end{array}
  \]

- Step 1: evaluate the expression \( x + 2 \)
  - For \( x \), use the value in variable \( x \)
  - Write the expression somewhere on your paper

- Step 2: Store the value of the expression in \( x \)
  - Cross off the old value in the box
  - Write the new value in the box for \( x \)

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

A: I did it correctly!
B: I drew another box named \( x \)
C: I did something else
D: I did nothing—just watched
Execute the statement: \( x = 3.0 \times x + 1.0 \)

- You have this:

\[
x \quad 7
\]
Execute the statement: \( x = 3.0 \times x + 1.0 \)

- You have this:
  \[
  x \quad 7
  \]

- Execute this command:
  - Step 1: **Evaluate** the expression \( 3.0 \times x + 1.0 \)
  - Step 2: **Store** its value in \( x \)
Execute the statement: \( x = 3.0 \times x + 1.0 \)

- You have this:
  
  \[
  x \quad 7
  \]

- Execute this command:
  
  - Step 1: **Evaluate** the expression \( 3.0 \times x + 1.0 \)
  - Step 2: **Store** its value in \( x \)

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Execute the statement: \( x = 3.0 \times x + 1.0 \)

- You have this:
  \[
  x \quad 22.0
  \]

- Execute this command:
  - Step 1: \textbf{Evaluate} the expression \( 3.0 \times x + 1.0 \)
  - Step 2: \textbf{Store} its value in \( x \)

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Execute the statement: \( x = 3.0 \times x + 1.0 \)

- You have this:
  \[
  x = 22.0
  \]

- Execute this command:
  - Step 1: **Evaluate** the expression \( 3.0 \times x + 1.0 \)
  - Step 2: **Store** its value in \( x \)

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

A: I did it correctly!
B: I drew another box named \( x \)
C: I did something else
D: I did nothing – just watched
Execute the statement: \( x = 3.0 \times x + 1.0 \)

- You now have this:

\[
x \quad 22.0
\]

- The command:

  - Step 1: **Evaluate** the expression \( 3.0 \times x + 1.0 \)
  - Step 2: **Store** its value in \( x \)

- This is how you execute an assignment statement

  - Performing it is called **executing the command**
  - Command requires both **evaluate** AND **store** to be correct
  - Important **mental model** for understanding Python
Exercise: Understanding Assignment

• Add another variable, interestRate, to get this:

\[ x \times \times 22.0 \quad \text{interestRate} \quad 4 \]

• Execute this assignment:

\[ \text{interestRate} = \frac{x}{\text{interestRate}} \]

• Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Exercise: Understanding Assignment

• Add another variable, interestRate, to get this:
  \[ x \times 22.0 \text{ interestRate } \times 5.5 \]

• Execute this assignment:
  \[ \text{interestRate} = x / \text{interestRate} \]

• Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Exercise: Understanding Assignment

- Add another variable, `interestRate`, to get this:
  
  \begin{align*}
  x & \quad \times \quad 22.0 \\
  \text{interestRate} & \quad \times \quad 5.5
  \end{align*}

- Execute this assignment:

  \[
  \text{interestRate} = x / \text{interestRate}
  \]

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

A: I did it correctly!
B: I drew another box called “interestRate”
C: I stored the value in the box for `x`
D: I thought it would use `int` division
E: I did something else (or nothing)

8/28/14
Exercise: Understanding Assignment

• You now have this:

\[ x \neq 22.0 \quad \text{interestRate} \neq 5.5 \]

• Execute this assignment:

\[ \text{interestRate} = x + \text{interestRate} \]

• Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Exercise: Understanding Assignment

- You now have this:
  \[ x \times 22.0 \quad \text{interestRate} \times 5.5 \quad \text{intrestRate} \times 27.5 \]

- Execute this assignment:
  \[ \text{intrestRate} = x + \text{interestRate} \]

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.
Exercise: Understanding Assignment

• You now have this:
  
  x 22.0  interestRate 5.5  intrestRate 27.5

• Execute this assignment:

  \[\text{intrestRate} = x + \text{interestRate}\]

• Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

  A: I did it correctly!
  B: I stored the value in “interestRate”
  C: I stored the value in x
  D: I did something else (or nothing)
Exercise: Understanding Assignment

• You now have this:

   x × 22.0  interestRate × 5.5  intrestRate 27.5

• Execute this assignment:

   intrestRate = x + interestRate

• Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

   A: I did it correctly!
   B: I stored the value in “interestRate”
   C: I stored the value in x
   D: I did something else (or nothing)

Spelling mistakes in Python are bad!!

8/28/14
Dynamic Typing

- Python is a **dynamically typed language**
  - Variables can hold values of any type
  - Variables can hold different types at different times
  - Use `type(x)` to find out the type of the value in `x`
  - Use names of types for conversion, comparison
- The following is acceptable in Python:
  ```python
  >>> x = 1
  >>> x = x / 2.0
  >>> x = float(x)
  >>> type(x) == float
  ```
- Alternative is a **statically typed language** (e.g. Java)
  - Each variable restricted to values of just one type
Dynamic Typing

• Python is a **dynamically typed language**
  - Variables can hold values of any type
  - Variables can hold different types at different times
  - Use `type(x)` to find out the type of the value in `x`
  - Use names of types for conversion, comparison

• The following is acceptable in Python:
  
  ```python
  >>> x = 1
  x contains an `int` value
  >>> x = x / 2.0
  x now contains a `float` value
  ```

• Alternative is a **statically typed language** (e.g. Java)
  - Each variable restricted to values of just one type

```python
type(x) == int
x = float(x)
type(x) == float
```
Dynamic Typing

• Often want to track the type in a variable
  ▪ What is the result of evaluating x / y?
  ▪ Depends on whether x, y are int or float values

• Use expression type(<expression>) to get type
  ▪ type(2) evaluates to <type 'int'>
  ▪ type(x) evaluates to type of contents of x

• Can use in a boolean expression to test type
  ▪ type('abc') == str evaluates to True