Lecture 2: Variables & Assignments
(Sections 2.1-2.3,2.5)

CS 1110
Introduction to Computing Using Python

http://www.cs.cornell.edu/courses/cs1110/2019sp
Helping you succeed in this class

**Canvas.** You should have received an email on Tuesday. Check your spam folders. This is how we make announcements.

http://www.cs.cornell.edu/courses/cs1110/2019sp/staff/

**Consulting Hours.** ACCEL Lab Green Room
- Big block of time multiple people (see staff calendar)
- Good for assignment help

**Staff Office Hours.**
- Staff: 1 person, 1 hour at a time (see staff calendar)
- Good for conceptual help

**My Office Hours.** Right next door after class (Baker 219) or by appointment (see staff page under “Office Hours”)

**Piazza.** Online forum to ask/answer questions
From last time: **Types**

Type: set of values & operations on them

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

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

Type **bool:**
- Values: true, false
- Ops: not, and, or

Type **str:**
- Values: string literals
  - Double quotes: “abc”
  - Single quotes: ‘abc’
- Ops: + (concatenation)
Converting from one type to another aka “casting”

\[
\text{\texttt{\texttt{float(2)}}}
\]

converts value \texttt{2} to type \texttt{float}

\[
\text{\texttt{int(2.6)}}
\]

converts value \texttt{2.6} to type \texttt{int}

\[
\text{\texttt{type(2)}}
\]

tells you the type

\[
\text{\texttt{<class \textquoteleft int\textquoteright>}}
\]
What should Python do?

`>>> 1/2.6`

(A) turn 2.6 into the integer 2, then calculate 1/2 → 0.5
(B) turn 2.6 into the integer 2, then calculate 1//2 → 0
(C) turn 1 into the float 1.0, then calculate 1.0/2.6 → 0.3846…
(D) Produce a TypeError telling you it cannot do this.
(E) Exit Python
Widening Conversion (OK!)

From a **narrower** type to a **wider** type (e.g., int → float)

Python does automatically if needed:

- Example: `1/2.0` evaluates to a float: `0.5`
- Example: `True + 1` evaluates to an int: `2`
  - True converts to `1`
  - False converts to `0`

Note: does not work for **str**
- Example: `2 + “ab”` produces a TypeError
Narrowing Conversion (OK???)

From a **wider** type to a **narrower** type (e.g., float → int)

- causes information to be lost
- Python **never** does this automatically

What about:

```python
>>> 1/int(2.6)
0.5
```

*Python casts the 2.6 to 2.0 but / is a float division, so Python casts 1 to 1.0 and 2 to 2.0*
Types matter!

You Decide:
• What is the right type for my data?
• When is the right time for conversion (if any)

• Zip Code as an int?
• Grades as an int?
• Lab Grades as a bool?
• Interest level as bool or float?

What are your goals:
Accuracy? Clarity? Fairness?
Operator Precedence

What is the difference between:

$2 \times (1+3)$  
$2 \times 1 + 3$

*add, then multiply*  
*multiply, then add*

Operations performed in a set order

- Parentheses make the order explicit

What if there are no parentheses?

→ **Operator Precedence**: fixed order to processes operators when no 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
    (except for **)
  - Example: 1/2*3 is (1/2)*3

- Section 2.5 in your text
- See website for more info
- Major portion of Lab 1
Operators and Type Conversions

Operator Precedence

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

Evaluate this expression:
\( \text{False} + 1 + 3.0 / 3 \)

A. 3
B. 3.0
C. 1.3333
D. 2
E. 2.0
Operators and Type Conversions

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

Evaluate this expression:
False + 1 + 3.0 / 3
False + 1 + 1.0
1 + 1.0
2.0
New Tool: Variable Assignment

An *assignment statement*:
- takes an *expression*
- evaluates it, and
- stores the *value* in a *variable*

**Example**:  
(read right to left)

\[ x = 5 \]

- variable
- equals sign (just one!)
- expression evaluates to 5
Executing Assignment Statements

>>> x = 5

Press ENTER and…

Hmm, looks like nothing happened…

• But something did happen!
• Python *assigned* the *value* 5 to the *variable* x
• Internally (and invisible to you):

<table>
<thead>
<tr>
<th>memory location</th>
<th>stored value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>5</td>
</tr>
</tbody>
</table>

x = 5
Retrieving Variables

```python
>>> x = 5
>>> x
5
Press ENTER and…
Interactive mode tells me the value of x
```
In More Detail: Variables (Section 2.1)

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

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

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

The type belongs to the **value**, not to the **variable**.
In More Detail: Statements

>>> x = 5

Press ENTER and…

>>> Hm, looks like nothing happened…

- 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)
### 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></td>
<td>▪ Python <em>evaluates it</em></td>
</tr>
<tr>
<td></td>
<td>▪ End result is a value</td>
</tr>
<tr>
<td></td>
<td>▪ Python <em>executes it</em></td>
</tr>
<tr>
<td></td>
<td>▪ Need not result in a value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples:</th>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ 2.3</td>
<td>▪ x = 2 + 1</td>
</tr>
<tr>
<td>▪ (3+5)/4</td>
<td>▪ x = 5</td>
</tr>
<tr>
<td>▪ x == 5</td>
<td></td>
</tr>
</tbody>
</table>

*Look so similar but they are not!*
You can assign more than literals

```python
>>> x = 5
```

```
>>> x = 3.0 ** 2 + 4 - 1
```

```
>>> x = 2 + x
```

```
19
```

"x gets 5"

"x gets the value of this expression"

"x gets 2 plus the current value of x"
Keeping Track of Variables

• Draw boxes on paper:
  
  >>> x = 5

• New variable declared?
  
  >>> y = 3

Write a new box.

• Variable updated?
  
  >>> x = 7

Cross out old value. Insert new value.
Execute the Statement: $x = x + 2$

Draw variable $x$ on piece of paper:

1. Evaluate the expression $x + 2$
   - For $x$, use the value in variable $x$
   - Write the expression somewhere on your paper

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 $x$

Did you do the same thing as your neighbor? If not, discuss.
Which one is closest to your answer?

A. \[ x \times 7 \]

B. \[ x \times 5 \times 7 \]

C. \[ x \times 7 \times 7 \]

D. \[ \_\_\_(ツ)_\_/\_ \]

\[ x = x + 2 \]
And The Correct Answer Is…

A.  
\[ x \times 7 \]

B.  
\[ x 	imes 5 \\
\times 7 \]

C.  
\[ x \times 7 \]

D.  
\[ x = x + 2 \]
Execute the Statement: $x = 3.0 \times x + 1.0$

Begin with this:

1. Evaluate the expression $3.0 \times x + 1.0$
2. Store its value in $x$

Did you do the same thing as your neighbor? If not, discuss.
Which one is closest to your answer?

A. $x \times \approx 22.0$

B. $x \times 7$
   $x \times 22.0$

C. $x \times \approx 22.0$
   $x \times 22.0$

D. $\_\_\_\_(ツ)_/\_/

$x = 3.0 \times x + 1.0$
And The Correct Answer Is...

A. \[
\times \quad \boxed{22.0}
\]

B. \[
\times \quad 7
\times \quad \boxed{22.0}
\]

C. \[
\times \quad \boxed{22.0}
\times \quad \boxed{22.0}
\]

D. \[
\_\_\_(ツ)_/君
\]

\[x = 3.0 \times x + 1.0\]
Executing an Assignment Statement

The command: \( x = 3.0 \times x + 1.0 \)

“Executing the command”:
1. Evaluate right hand side \( 3.0 \times x + 1.0 \)
2. Store the value in the variable \( x \)’s box

• Requires both evaluate AND store steps
• Critical mental model for learning Python
Exercise 1: Understanding Assignment

Begin with:

Declare a new variable:

```python
>>> rate = 4
```

Execute this assignment:

```python
>>> rate = x / rate
```

Did you do the same thing as your neighbor? If not, discuss.
Which one is closest to your answer?

A. \[ \frac{22.0}{5.5} \]

B. \[ \frac{22.0}{5.5} \]

C. \[ \frac{22.0}{5.5} \]

D. \[ \frac{22.0}{5.5} \]

E. \[ \frac{22.0}{5.5} \]

\[ rate = \frac{x}{rate} \]
And The Correct Answer Is...

<table>
<thead>
<tr>
<th>A.</th>
<th>x (22.0 \div 5.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\hat{\Rightarrow} 5.5)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B.</th>
<th>x (22.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate (\hat{\Rightarrow})</td>
<td></td>
</tr>
<tr>
<td>rate (\hat{\Rightarrow} 5.5)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C.</th>
<th>x (22.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate (\hat{\Rightarrow})</td>
<td></td>
</tr>
<tr>
<td>rate (\hat{\Rightarrow} 5.5)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D.</th>
<th>x (22.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate (\hat{\Rightarrow})</td>
<td></td>
</tr>
<tr>
<td>rate (\hat{\Rightarrow} 5)</td>
<td></td>
</tr>
</tbody>
</table>

\[\text{rate} = \frac{x}{\text{rate}}\]
Dynamic Typing

Python is a **dynamically typed** language

- Variables can hold values of any type
- Variables can hold different types at different times

The following is acceptable in Python:

```python
>>> x = 1
>>> x = x / 2.0
```

Alternative: a **statically typed** language

- Examples: Java, C
- Each variable restricted to values of just one type
More Detail: Testing Types

Command: `type(<value>)`

Can test a variable:

```python
>>> x = 5
>>> type(x)
<class 'int'>
```

Can test a type with a Boolean expression:

```python
>>> type(2) == int
True
```
Exercise 2: Understanding Assignment

Begin with:

\[
\begin{array}{c}
\times \\
\text{rate}
\end{array}
\begin{array}{c}
22.0 \\
5.5
\end{array}
\]

Execute this assignment:

\[\text{rat} = x + \text{rate}\]

Did you do the same thing as your neighbor? If not, discuss.
Which one is closest to your answer?

<table>
<thead>
<tr>
<th>Option</th>
<th>Expression</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>$x \cdot \frac{27.5}{5.5}$</td>
<td>$\surd$</td>
</tr>
<tr>
<td>B.</td>
<td>$x \cdot 22.0$</td>
<td>$\surd$</td>
</tr>
<tr>
<td>C.</td>
<td>$x \cdot \frac{22.0}{27.5}$</td>
<td>$\surd$</td>
</tr>
<tr>
<td>D.</td>
<td>$x \cdot 22.0$</td>
<td>$\surd$</td>
</tr>
<tr>
<td>E.</td>
<td>$\________________$</td>
<td>$\text{rat} = x + \text{rate}$</td>
</tr>
</tbody>
</table>
And The Correct Answer Is…

A. \( \times 22.0 \quad 27.5 \)  
   rate \( 5.5 \)

B. \( \times 22.0 \)  
   rate \( 5.5 \)  
   rat \( 27.5 \)

C. \( \times 22.0 \)  
   rate \( 27.5 \)  
   rate \( 27.5 \)

D. \( \times 22.0 \)  
   rate \( 5.5 \)  
   rat \( 27.5 \)

Spelling Matters!

\( \text{rat} = x + \text{rate} \)