

Lecture 2

# **Variables & Assignment**

# Announcements for Today

---

## If Not Done Already

---

- Enroll in Piazza
- Sign into CMS
  - Fill out the Survey
  - Complete AI Quiz
- (**Optional**) 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 Lacy
  - [ls192@cornell.edu](mailto:ls192@cornell.edu)
  - Will review by next week
- Have one week to complete
  - Complete in online system
  - Show **at start** of next lab
- But finish Lab 0 **TODAY**

# Official Announcement

<WICC>

## CLS Partner Finding Social

**Tuesday, September 3**  
**Gates 310 and 3rd Floor Lounge**

5-6 pm for 1000-2000 level classes  
6-7pm for 3000+ level classes

# 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!
  - Available in Bailey lower lobby between lectures

# Consulting Hours Drama

---

- Carpenter Hall **fire escape** is under construction
  - Violates fire code to use ACCEL after 4:30
  - Affects **consultant hours** and **Section 211**
- From now until **September 12<sup>th</sup>**
  - Sunday consulting hours are in **Upson 142**
  - Mon-Thurs consulting hours are in **Gates G15**
  - Section 211 meets in **Phillips 318**
- Revert to normal after September 12<sup>th</sup>

# 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. 25
- Graded on **correctness**
  - Assign points out of 100
- But **first** one is for *mastery*
  - Resubmit until perfect grade
- 40% of your final grade
- Partner mixer **TODAY!**
  - 5-6pm, 3<sup>rd</sup> floor of Gates

# Academic Integrity

---

- Every semester we have cases of *plagiarism*
  - Claiming the work of others as your own
  - This is an **Academic Integrity violation**
- This course has a very specific policy
  - Do not listen to (non-staff) upperclassmen
  - Look at the course website for the new details
- Complete **Academic Integrity Quiz** on CMS
  - Must complete successfully to stay in class

# iClickers

---

- Have you registered your iclicker?
- If not, visit (now with no surcharge!)
  - <https://cs1110.cs.cornell.edu/py/clicker>
- See the course web page for more:
  - <http://www.cs.cornell.edu/courses/cs1110/2019fa>
  - Click “Materials/Textbook”
  - 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

# Converting Values Between Types

---

- Basic form: *type(expression)*
  - This is an expression
  - Evaluates to value, converted to new type
  - This is sometimes called **casting**
- **Examples:**
  - `float(2)` evaluates to `2.0` (a **float**)
  - `int(2.6)` evaluates to `2` (an **int**)
  - Note information loss in 2<sup>nd</sup> example

# Converting Values Between Types

---

- Conversion is measured *narrow* to *wide*

**bool** ⇒ **int** ⇒ **float**

- **Widening:** Convert to a wider type
  - Python does automatically
  - **Example:** `1/2.0` evaluates to `0.5`
- **Narrowing:** Convert to a narrower type
  - Python never does automatically
  - **Example:** `float(int(2.6))` evaluates to `2.0`

# Operator Precedence

---

- What is the difference between these two?
  - $2*(1+3)$
  - $2*1 + 3$

# Operator Precedence

---

- What is the difference between these two?
  - $2*(1+3)$       **add, then multiply**
  - $2*1 + 3$       **multiply, then add**
- Operations are performed in a **set order**
  - Parentheses make the order explicit
  - What happens when no parentheses?

# Operator Precedence

---

- What is the difference between these two?
  - $2*(1+3)$       **add, then multiply**
  - $2*1 + 3$       **multiply, then add**

- **O**
  - **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 2.5 in your text
- See website for more info
- Was major portion of Lab 1



# Expressions vs Statements

## Expression

- **Represents** something
  - Python *evaluates it*
  - End result is a value
- Examples:
  - 2.3 
  - (3+5)/4 

## Statement

- **Does** something
  - Python *executes it*
  - Need not result in a value
- Examples:
  - `print('Hello')`
  - `import sys`

Will see later this is not a clear cut separation

# Variables

---

- A **variable**
  - is a **box** (memory location)
  - with a **name**
  - and a **value** in the box

- Examples:

x 

5
---

 Variable **x**, with value 5 (of type **int**)

area 

20.1
------

 Variable **area**, w/ value 20.1 (of type **float**)

# Using Variables

---

- Variables can be used in expressions
  - Evaluate to the value that is in the box
  - **Example:**  $x$  5  $1 + x$  evaluates to **6**
- Variables can change values
  - **Example:**  $x$  ~~5~~ 1.5  $1 + x$  evaluates to **2.5**
  - Can even change the **type** of their value
  - Different from other languages (e.g. Java)

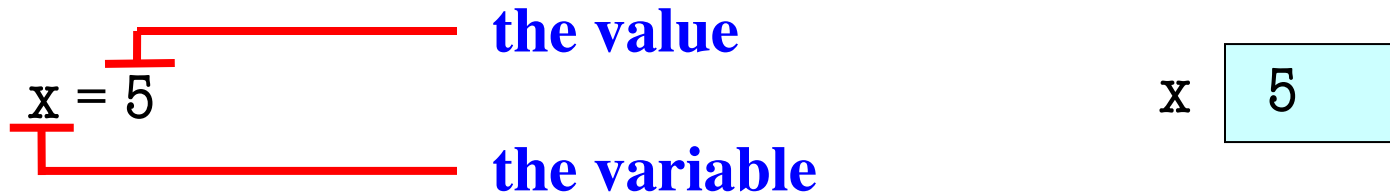
# Naming Variables

---

- Python has strict rules of how to assign names
  - Names must only contain letters, numbers, \_
  - They cannot start with a number
- **Examples**
  - `e1` is a **valid** name
  - `le2` is **not valid** (it is a **float**)
  - `a_b` is a **valid** name
  - `a+b` is **not valid** (it is + on two variables)

# Variables and Assignment Statements

- Variables are created by **assignment statements**



- This is a **statement**, not an **expression**

- **Expression**: Something Python turns into a value
- **Statement**: Command for Python to do something
- Difference is that has no value itself

- Example:**

```
>>> x = 5  
(NOTHING)
```

But can now use x  
as an expression

# Variables Do Not Exist Until Made

---

- Example:

```
>>> y
```

```
Error!
```

```
>>> y = 3
```

```
>>> y
```

```
3
```

- Changes our model of Python
  - Before we just typed in one line at a time
  - Now program is a **sequence** of lines

# Assignments May Contain Expressions

- **Example:**  $x = 1 + 2$

- Left of equals must always be variable:  ~~$1 + 2 = x$~~
- Read assignment statements right-to-left!
- Evaluate the expression on the right
- Store the result in the variable on the left

- We can include variables in this expression

- **Example:**  $x = y + 2$

x 5

- **Example:**  $x = x + 2$

y 2

This is not circular!  
Read right-to-left.

# Execute the Statement: $x = x + 2$

---

- Draw variable  $x$  on piece of paper:

$x$  5



# Execute the Statement: $x = x + 2$

---

- Draw variable  $x$  on piece of paper:

$x$  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:

$x$ 

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  $x$

# Execute the Statement: $x = x + 2$

---

- Draw variable  $x$  on piece of paper:

$x$  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.

# Which One is Closest to Your Answer?

A:

x  7

B:

x

x

C:

x

x

D:

(  )

# Which One is Closest to Your Answer?

A:

x  7



B:

x

x

C:

x

x

$$x = x + 2$$

# Execute the Statement: $x = 3.0 * x + 1.0$

---

- You have this:

x

# Execute the Statement: $x = 3.0 * x + 1.0$

---

- You have this:

x ~~5~~ 7

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

# Execute the Statement: $x = 3.0 * x + 1.0$

---

- You have this:

x ~~5~~ 7

- Execute this command:
  - Step 1: **Evaluate** the expression  $3.0 * x + 1.0$
  - 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.



# Which One is Closest to Your Answer?

A:

x

B:

x

x

C:

x

x

D:

# Which One is Closest to Your Answer?

A:

x



B:

x

x

C:

x

x

$$x = 3.0 * x + 1.0$$

# Execute the Statement: $x = 3.0 * x + 1.0$

---

- You now have this:

x ~~5~~ ~~7~~ 22.0

- The command:
  - Step 1: **Evaluate** the expression  $3.0 * x + 1.0$
  - 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 ~~5~~ ~~7~~ 22.0    `interestRate` 4

- Execute this assignment:

```
interestRate = x / interestRate
```

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

# Which One is Closest to Your Answer?

A:

x

interestRate

B:

x

interestRate

interestRate

C:

x

interestRate

D:

x

interestRate

# Which One is Closest to Your Answer?

A:

x

interestRate

B:

x

interestRate

E:

~\\_ (ツ) \\_/

C:

x

interestRate

interestRate

# Which One is Closest to Your Answer?

interestRate = x/interestRate

B:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~

interestRate 5.5

C:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ 5.5



D:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ 5

# Exercise: Understanding Assignment

---

- You now have this:

x ~~5~~ ~~7~~ 22.0    interestRate ~~4~~5.5

- 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.



# Which One is Closest to Your Answer?

A:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ ~~5~~ 27.5

B:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ 5.5

intrestRate 27.5

C:

x ~~5~~ ~~7~~ ~~22~~ 0 27.5

interestRate ~~4~~ 5.5

D:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ ~~5~~ 5

intrestRate 27.5

# Which One is Closest to Your Answer?

A:

x

interestRate

B:

x

estRate

E:

〜 (ツ) 〃

C:

x

interestRate

interestRate

intrestRate

# Which One is Closest to Your Answer?

A:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ ~~5~~ 27.5

B:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ 5.5

intrestRate 27.5



intrestRate = x + interestRate

^  
e

# Which One is Closest to Your Answer?

A:

x ~~5~~ ~~7~~ 22.0

interestRate ~~4~~ ~~5~~ 27.5

B:

x ~~5~~ ~~7~~ 22.0



interestRate ~~4~~ 5.5

intrestRate 27.5

intrestRate = x + interestRate

^  
e

Spelling mistakes in  
Python are bad!!

# 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:
  - >>> `x = 1`      ← x contains an **int** value
  - >>> `x = x / 2.0`   ← x now contains a **float** value
- Alternative is a **statically typed language**
  - Each variable restricted to values of just one type
  - This is true in Java , C, C++, etc.

# 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**