## **Helping You Succeed in this Class**

- Consultants. Phillips 318 (after hours)
  - 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.
- Ed Discussions. Forum to ask and answer questions
  - Go here first **before** sending question in e-mail
- Office Hours. Talk to the professor!
  - Couches in Statler Balcony between classes

#### **iClickers**

- Have you registered your iClicker?
- If not, visit (free service; no surcharge!)
  - https://cs1110.cs.cornell.edu/py/clicker
- See the course web page for more:
  - http://www.cs.cornell.edu/courses/cs1110/2024fa
  - Click "Materials/Textbook"
  - Look under "iClickers"

# **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* 

 $bool \Rightarrow int \Rightarrow float$ 

- 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

3

1

### **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: The *fixed* order Python processes operators in *absence* of parentheses

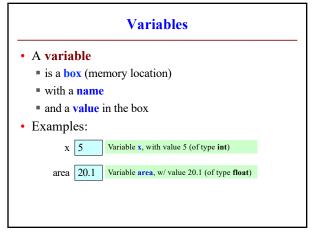
**Precedence of Python Operators** 

- 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

5

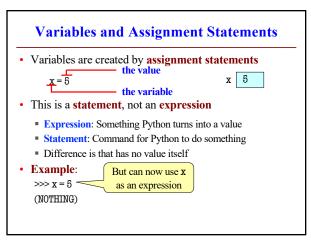
1





- 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 x** 1.5 **1 + x** evaluates to **2.5**
  - Can even change the **type** of their value
  - Different from other languages (e.g. Java)

7



**Assignments May Contain Expressions** 

- **Example**: x = 1 + 2
  - Left of equals must always be variable:
- - 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

у 2

■ Example: x = x+2/

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

10

## **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 = x / 2.0  $\leftarrow$  x now contains a **float** value

← x contains an **int** 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

11 12