

Lecture 2:

Variables & Assignments

(2.1-2.3,2.5, 2.6 or videos (see [Schedule](#)))

CS 1110

Introduction to Computing Using Python



Cornell Bowers C&S
Computer Science

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Lecture 2:

Variables & Assignments

(Sections 2.1-2.3,2.5, 2.6)

Have pencil and paper (or stylus and tablet) ready. We'll do visualization exercises that involve drawing diagrams today.

Recommendations for note taking:

- Print out posted lecture slides and write on them
- Have the slides pdf ready and annotate electronically

Lecture Afterthoughts

There were many questions about what certain operators do (/, //, %). **You do not need to memorize their behavior.** We want you to know about them so that *when the need arises*, you can make use of them.

Similarly, we want you to know that operator precedence *exists* so you can understand how Python works. Instead of memorizing these slides, you can reference the exact ordering when it matters to the code you are writing.

3

Helping you succeed in this class

<http://www.cs.cornell.edu/courses/cs1110/2022sp/staff/>

Ed Discussions. Online forum to ask/answer questions
Consulting/Office Hours.

- See calendar for which are 1-on-1 help (managed by QueueMeIn") versus "public help" (all students there form a single audience)

Prof Office Hours (on same calendar)

- After lecture: public help.
- Bookable 1-on-1 appointments with [Professor Bracy](#)
- Bookable 1-on-1 appointments with [Professor Lee](#)

AEW (ENGRG 1010). "Academic Excellence Workshops"

- *Optional* discussion course that runs parallel to this class. See website for more info

4

Lab 1 Activities

Activity 1: if you aren't passing the "banner" question, get help with your installation! *In the meantime*, you can add the following string to your answer to get the system to accept it:

Python 3.x Anaconda

Activity 2: the password is:

learn.by.testing.hypotheses

(no spaces, all one "word", period separators)

Activity 3:

Q3: add the name of Python's behavior to your answer:

short circuit evaluation

Q8: password:

shortcircuit

(no spaces)

5

Which of the following is **false**?

A **type**...

- is a set of values & operations on these values
- represents something
- can be determined by using `type()` in Python
- can be changed by using `type()` in Python
- determines the meaning of an operation

If there are multiple false answers.

pick one!



6

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: strings
- Double quotes: "abc"
- Single quotes: 'abc'
- Ops: + (concatenation)

7

Converting from one type to another aka "casting"

```
>>> float(2)
2.0
```

converts value 2 to type float

```
>>>int(2.6)
2
```

converts value 2.6 to type int

```
>>>type(2)
<class 'int'>
```

...different from:

```
type(<value>)
```

which tells you the type

8

What does Python do?



```
>>> 1/2.6
```

(A) turn 2.6 into the integer 2, then calculate $1/2 \rightarrow 0.5$

(B) turn 2.6 into the integer 2, then calculate $1//2 \rightarrow 0$

(C) turn 1 into the float 1.0, then calculate $1.0/2.6 \rightarrow 0.3846\dots$

(D) Produce a `TypeError` telling you it cannot do this.

(E) Exit Python

9

Widening Conversion (OK!)

From a **narrower** type to a **wider** type

(e.g., `int` \rightarrow `float`)

Width refers to information capacity. "Wide" \rightarrow more information capacity

Python does automatically if needed:

- Example: `1/2.0` evaluates to a float: `0.5`

From narrow to wide:
`bool` \rightarrow `int` \rightarrow `float`

Note: does not work for **str**

- Example: `2 + "ab"` produces a `TypeError`

10

Narrowing Conversion (is it OK???)

From a **wider** type to a **narrower** type

(e.g., `float` \rightarrow `int`)

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

What about:

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

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

11

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`?

12

Operator Precedence

What is the difference between:

$$2*(1+3)$$

$$2*1 + 3$$

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 process operators when no parentheses

Precedence of Python Operators

- **Exponentiation:** **
 - **Negation:** -
 - **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
 • Part of Lab 1

Operators and Type Conversions

Evaluate this expression:

$$7 + 3.0 / 3$$

- A. 3
- B. 3.0
- C. 3.333333335
- D. 8
- E. 8.0

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

Operators and Type Conversions

Evaluate this expression:

$$7 + \frac{3.0}{3}$$

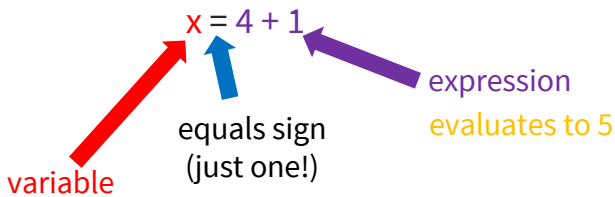
$$7 + 1.0$$

$$\frac{7 + 1.0}{8.0}$$

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

New Tool: Variable Assignment

Example:



An *assignment statement*:

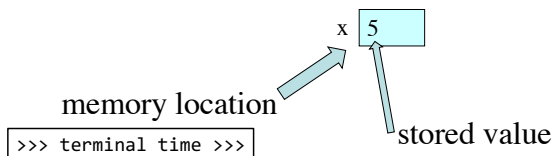
- takes an *expression*
- evaluates it, and
- stores the *value* in a *variable*

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):



Retrieving Variables in Interactive Mode

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

Press ENTER and...

Interactive mode tells me the value of x

>>> terminal time >>>

19

In More Detail: Variables (Section 2.1)

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

• Examples:

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

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

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

The type belongs to the *value*, not to the *variable*.

20

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)

21

Expressions vs. Statements

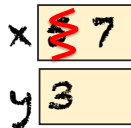
Expression	Statement
<ul style="list-style-type: none"> • Represents something <ul style="list-style-type: none"> ▪ Python <i>evaluates it</i> ▪ End result is a value • Examples: <ul style="list-style-type: none"> ▪ 2.3 Value ▪ (3+5)/4 Complex Expression ▪ x == 5 	<ul style="list-style-type: none"> • Does something <ul style="list-style-type: none"> ▪ Python <i>executes it</i> ▪ Need not result in a value • Examples: <ul style="list-style-type: none"> ▪ x = 2 + 1 ▪ x = 5 <p><i>Look so similar but they are not!</i></p>

22

Keeping Track of Variables

- Draw boxes on paper:

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



- New variable declared?

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

Write a new box.

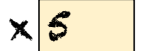
- Variable updated?

```
>>> x = 7
```

Cross out old value. Insert new value.

23

Start with variable **x** having value 5. Draw it on paper:



Task: Execute the Statement: **x = x + 2**

1. Evaluate the RHS expression, **x + 2**
 - For **x**, use the value in variable **x**
 - What value does the RHS expression evaluate to?
2. Store the value of the RHS expression in **x** in variable names on LHS, **x**
 - Cross off the old value in the box
 - Write the new value in the box for **x**

24

Which one is closest to your answer?

A. x <input type="text" value="7"/>	B. x <input type="text" value="5"/> x <input type="text" value="7"/>
C. x <input type="text" value="8"/> x <input type="text" value="7"/>	D.

$x = x + 2$



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

Begin with this:

x

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



Which one is closest to your answer?

A. x <input type="text" value="22.0"/>	B. x <input type="text" value="7"/> x <input type="text" value="22.0"/>
C. x <input type="text" value="8"/> x <input type="text" value="22.0"/>	D.

$x = 3.0*x+1.0$



Executing an Assignment Statement

The command: $x = 3.0*x+1.0$

"Executing the command":

1. **Evaluate** right hand side $3.0*x+1.0$
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

Have variable x already from previous

Create a new variable:

`>>> rate = 4`

x
rate

Execute this assignment:

`>>> rate = x / rate`

Which one is closest to your answer?

A. x <input type="text" value="22.0"/> rate <input type="text" value="5.5"/>	B. x <input type="text" value="22.0"/> rate <input type="text" value="5"/> rate <input type="text" value="5.5"/>
C. x <input type="text" value="22.0"/> rate <input type="text" value="5.5"/>	D. x <input type="text" value="22.0"/> rate <input type="text" value="5"/>

E. $rate = x / rate$



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:

```
>>> x = 1           ← x contains an int value
>>> x = x / 2.0     ← x now contains a float value
```

Alternative: a **statically typed** language

- Examples: Java, C
- Each variable restricted to values of just one type

36

Exercise 2: Understanding Assignment

Begin with:

```
x 22.0
rate 5.5
```

Execute this assignment:

```
>>> rat = x + rate
```

Did you do the same thing as your neighbor? If not, *discuss*.



37

Which one is closest to your answer?

A. ~~x 22.0~~ rate 5.5

B. x 22.0
rate 5.5
rat 27.5

C. x 22.0
~~rate 5.5~~ rate 27.5

D. x 22.0
rate ~~5.5~~
rat 27.5

E. ~~rate 5.5~~ rat = x + rate

39

More Detail: Testing Types

Command: `type(<value>)`

Can test a variable:

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

Can test a type with a Boolean expression:

```
>>> type(2) == int
True
```

41