



Mini-Lecture 3

Variables

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 (Section 2.1)

- A **variable** is
 - a **named** memory location (**box**),
 - 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**)

- Variable names must start with a letter
 - So **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

 **the value**
 $x = 3$


 **the variable**

- 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

 **the expression**
 $x = x + 2$

 **the variable**

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

Which One is Closest to Your Answer?

A:

x

B:

x

x

C:

x

x

D:

— _ (ツ) _ /

Which One is Closest to Your Answer?

A:

x 7



B:

x

x

C:

x

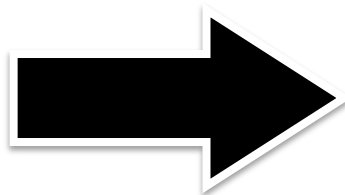
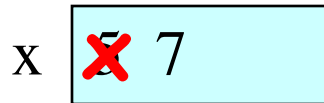
x

$$x = x + 2$$

Assignment Statements

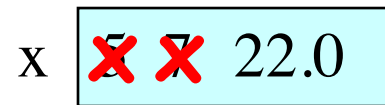
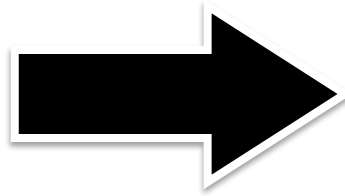
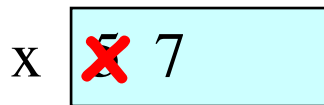
- Make new variables:

`interestRate = 4`



- Change existing variables:

`x = 3.0 * x + 1.0`

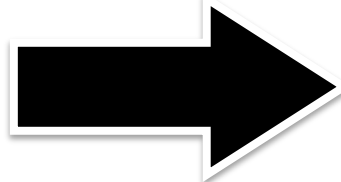


But Beware of Misspellings

`intrestRate = x + interestRate`

x

interestRate



x

interestRate

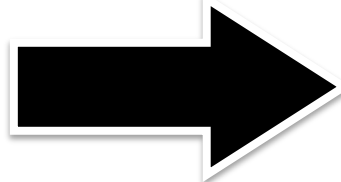
intrestRate

But Beware of Misspellings

`intrestRate = x + interestRate`

x ~~22.0~~ ~~22.0~~ 22.0

interestRate 4



x ~~22.0~~ ~~22.0~~ 22.0

interestRate 4

intrestRate 26.0

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:

```
>>> 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

```
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**