

## Objectives

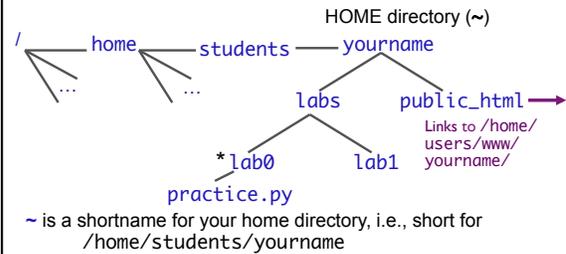
- Review Linux, algorithms
- Programming in Python
  - Data types
  - Expressions
  - Variables
  - Arithmetic
- Broader Issue

Jan 13, 2012

Sprenkle - CSCI111

1

## Review: Linux File System



- What is the *syntax* for the copy command?
- How would you copy `practice.py` to your `public_html` directory if you were in `public_html`? If you were in `labs`?

## Review: Labs

- Won't be as long until later in the semester
  - Definitely easier if you're prepared ahead of time, i.e., review your notes and examples
- "That's it?"
  - Often, students get overwhelmed by the directions, but then it isn't actually that bad
- Worth 38% of your grade
  - Should get in B+/A- range *easily* with help from student assistants and me

Jan 13, 2012

Sprenkle - CSCI111

3

## Review

- What is an algorithm?
- What are the parts of an algorithm?
- Why do we need programming languages?
- What are some properties of programming languages?

Jan 13, 2012

Sprenkle - CSCI111

4

## Parts of an Algorithm

- Input, Output
- ➔ Primitive operations
  - What data you have, what you can do to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques

Jan 13, 2012

Sprenkle - CSCI111

5

## Primitive Data Types

- Primitive data types represent **data**
  - In PB&J example, our data had **types** slice of bread, PB jar, jelly jar, etc.
- Python provides some basic or **primitive data types**
- Broadly, the categories of primitive types are
  - Numeric
  - Boolean
  - Strings

Jan 13, 2012

Sprenkle - CSCI111

6

## Numeric Primitive Types

| Python Data Type     | Description                                      | Examples                         |
|----------------------|--|----------------------------------|
| <code>int</code>     | Plain integers (32-bit precision)                | -214, -2, 0, 2, 100              |
| <code>float</code>   | Real numbers                                     | .001, -1.234, 1000.1, 0.00, 2.45 |
| <code>complex</code> | Imaginary numbers (have real and imaginary part) | $1j * 1j \rightarrow (-1+0j)$    |

Jan 13, 2012

Sprenkle - CSCI111

7

## How big (or small or precise) can we get?

- Computer cannot represent all values
- Problem: Computer has a **finite** capacity
  - The computer only has so much memory that it can devote to one value.
  - Eventually, reach a cutoff
    - Limits size of value
    - Limits precision of value

PI has more decimals, but we're out of space!

0 0 0 0 0 3 . 1 4 1 5 9 2 6 5

Example: in Python interpreter, `.1 + .1 + .1` yields `0.30000000000000004`.  
\* In reality, computers represent data in binary.

## Strings: `str`

- Indicated by double quotes `" "` or single quotes `' '`
- Treat what is in the `" "` or `' '` literally
  - Known as **string literals**
- Examples:
  - `"Hello, world!"`
  - `'c'`
  - `"That is Buddy's dog."`

Single quote must be inside double quotes\*  
\* Exception later

Jan 13, 2012

Sprenkle - CSCI111

9

## Booleans: `bool`

- 2 values
  - `True`
  - `False`
- More on these later...

Jan 13, 2012

Sprenkle - CSCI111

10

## What is the value's type?

| Value      | Type |
|------------|------|
| 52         |      |
| -0.01      |      |
| 4+6j       |      |
| "3.7"      |      |
| 4047583648 |      |
| True       |      |
| 'false'    |      |

Jan 13, 2012

Sprenkle - CSCI111

11

## What is the value's type?

| Value      | Type                 |
|------------|----------------------|
| 52         | <code>int</code>     |
| -0.01      | <code>float</code>   |
| 4+6j       | <code>complex</code> |
| "3.7"      | <code>str</code>     |
| 4047583648 | <code>int</code>     |
| True       | <code>boolean</code> |
| 'false'    | <code>str</code>     |

Jan 13, 2012

Sprenkle - CSCI111

12

## Parts of an Algorithm

- Input, Output
- Primitive operations
  - What data you have, what you can do to the data
- ➔ Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques

Jan 13, 2012

Sprenkle - CSCI111

13

## Introduction to Variables

- Variables save data/information
  - Example: first slice of bread or knife #1
  - Type of data the variable holds can be any of primitive data types as well as other data types we'll learn about later
- Variables have names, called **identifiers**

Jan 13, 2012

Sprenkle - CSCI111

14

## Variable Names/Identifiers

- A variable name (identifier) can be any one word that:
  - Consists of letters, numbers, or \_
  - Does *not* start with a number
  - Is not a Python reserved word
    - Examples: **for while def**
- Python is case-sensitive:
  - `change` isn't the same as `Change`

Jan 13, 2012

Sprenkle - CSCI111

15

## Variable Name Conventions

- **Variables** start with lowercase letter
- Convention: **Constants** (values that won't change) are all capitals
  - More on Monday
- Example: Variable for the current year
  - `currentYear`
  - `current_year`
  - ~~`CURRENT_YEAR`~~
  - ~~`currentyear`~~ Harder to read
  - ~~`current year`~~ No spaces allowed

Jan 13, 2012

Sprenkle - CSCI111

16

## Importance of Variable Naming

- Helps you *remember* what the variable represents
- Easier for others to *understand* your program
- Examples:

| Info Represented              | Good Variable Name                               |
|-------------------------------|--|
| A person's first name         | <code>firstName</code> , <code>first_name</code> |
| Radius of a circle            | <code>radius</code>                              |
| If someone is employed or not | <code>isEmployed</code>                          |

What are the **types** of each of these variables?

Jan 13, 2012

Sprenkle - CSCI111

17

## Review: Computational Problem Solving

- **Computational Problem:**  
A problem that can be solved by logic
- To solve the problem:
  - ➔ Create a **model** of the problem
  - Design an **algorithm** for solving the problem using the model
  - Write a **program** that *implements* the algorithm

Jan 13, 2012

Sprenkle - CSCI111

18

## Modeling Information

- How would you *model* this information?
- What data type best represents the info?

| Info Represented   | Data Type | Variable Name |
|--------------------|-----------|---------------|
| A person's salary  |           |               |
| Sales tax          |           |               |
| If item is taxable |           |               |
| Course name        |           |               |
| Gender             |           |               |
| Middle initial     |           |               |
| Graduation Year    |           |               |

Jan 13, 2012

Sprenkle - CSCI111

19

## Assignment Statements

- Variables can be given any value using =
  - > **Syntax:** <variable> = <expression>
  - > **Semantics:** <variable> is set to value of <expression>
- After a variable is set to a value, the variable is said to be **initialized**
- Examples:

```
month = 1
impt_num = 4.5
monthName = 'January'
```

These are **not** equations!  
Read "=" as "is set to"

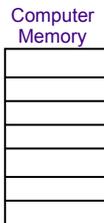
Jan 13, 2012

Sprenkle - CSCI111

20

## Assignment Statements

```
x = 5
y = x
```



- Statements execute in order, from top to bottom
- Value of **x** does not change because of second assignment statement

Jan 13, 2012

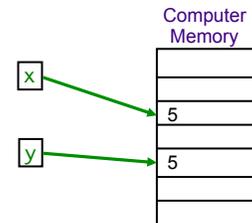
Sprenkle - CSCI111

21

## Assignment Statements

```
x = 5
y = x
```

Does a "lookup" in memory to find value of x



- Statements execute in order, from top to bottom
- Value of **x** does not change because of second assignment statement

Jan 13, 2012

Sprenkle - CSCI111

22

## Variables: The Rules

- Only the variable(s) to **left** of the = change
  - > We'll usually only have one variable on the left
- **Initialize** a variable **before** using it on the right-hand side (rhs) of a statement

Jan 13, 2012

Sprenkle - CSCI111

23

## Literals

- Pieces of data that are not variables are called **literals**
  - > We've been using these a lot already
- Examples:
  - > 4
  - > 3.2
  - > 'q'
  - > "books"

Jan 13, 2012

Sprenkle - CSCI111

24

## Numeric Arithmetic Operations

| Symbol | Meaning                |
|--------|------------------------|
| +      | Addition               |
| -      | Subtraction            |
| *      | Multiplication         |
| /      | Division               |
| %      | Remainder ("mod")      |
| **     | Exponentiation (power) |

Jan 13, 2012

Sprenkle - CSCI111

25

## Arithmetic & Assignment

- You can use the assignment operator (=) and arithmetic operators to do calculations
  - Calculate right hand side
  - Assign value to variable
- Remember your order of operations! (PEMDAS)
- Examples:
  - $x = 4+3*10$
  - $y = 3/2.0$
  - $z = x+y$

The right-hand sides are **expressions**, just like in math.

Jan 13, 2012

Sprenkle - CSCI111

26

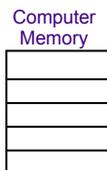
## Arithmetic & Assignment

- Examples:

$$x = 4+3*10$$

$$y = 3/2.0$$

$$z = x+y$$



- For 3rd statement, need to "lookup" values of x and y

Jan 13, 2012

Sprenkle - CSCI111

27

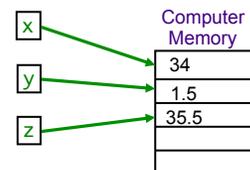
## Arithmetic & Assignment

- Examples:

$$x = 4+3*10$$

$$y = 3/2.0$$

$$z = x+y$$



- For 3rd statement, need to "lookup" values of x and y
  - Note that x and y do not change because of z's assignment statement

Jan 13, 2012

Sprenkle - CSCI111

28

## What are the values?

- After executing the following statements, what are the values of each variable?

$$\triangleright x = 5$$

$$\triangleright y = -1 + x$$

$$\triangleright z = x + y$$

$$\triangleright y = 2$$

$$\triangleright x = -7$$

How can we verify our answers?

Jan 13, 2012

Sprenkle - CSCI111

29

## What are the values?

- After executing the following statements, what are the values of each variable?

$$\triangleright a = 5$$

$$\triangleright y = a + -1 * a$$

$$\triangleright z = a + y / 2$$

$$\triangleright a = a + 3$$

$$\triangleright y = (7+x)*z$$

$$\triangleright x = z*2$$

Jan 13, 2012

Sprenkle - CSCI111

30

## What are the values?

- After executing the following statements, what are the values of each variable?

```
> a = 5
> y = a + -1 * a
> z = a + y / 2
> a = a + 3
> y = (7+x)*z
> x = z*2
```

### Runtime error:

- x doesn't have a value yet!
- We say "x was not initialized"
- Can't use a variable on RHS until seen on LHS!\*

Jan 13, 2012

Sprenkle - CSCI111

31

## Programming Building Blocks

- Each type of statement is a building block
  - > Initialization/Assignment
    - So far: Arithmetic
  - > Print
- We can combine them to create more complex programs
  - > Solutions to problems

Assign.

print

Assign.

print

Assign.

Assign.

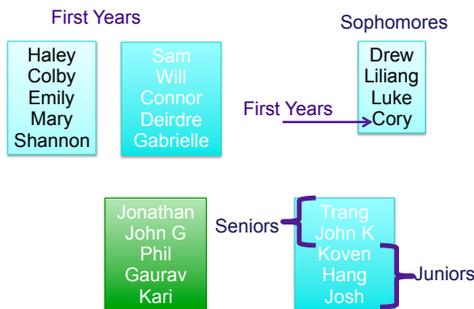
print

Jan 13, 2012

Sprenkle - CSCI111

32

## Groups for New Programs In CS



Jan 13, 2012

Sprenkle - CSCI111

33

## Broader CS Issues

- Good summaries!
  - > Good English, complete sentences
- Good, thoughtful questions
- Mechanics details
  - > Follow instructions on "CS Issues" about what summary should contain
  - > Should be able to edit your own posts
  - > Characters from Word
    - View your post after you write it
    - Fix as necessary
  - > Don't attach Word documents

Jan 13, 2012

Sprenkle - CSCI111

34

## New Programs in CS

- What is "computational thinking"?
- When should students first be exposed to CS or computational thinking?
- How could "computational thinking" affect one of your interests (major/hobby/...)?
- Does the geek stereotype exist?
- Are there any potential negative impacts to computing in education, society, ...?

Jan 13, 2012

Sprenkle - CSCI111

35

## New Programs in CS

- What is "computational thinking"?
- What is the difference between "technology education" and "computer science"?
- When should students first be exposed to CS or computational thinking?
- How could "computational thinking" affect one of your interests (major/hobby/...)?
- Does the geek stereotype exist?

Jan 13, 2012

Sprenkle - CSCI111

36

## "Really" with Professor Sprenkle



We've got the guys with pocket protectors working to upgrade your web experience as we speak. We apologize for the inconvenience. Come back later and continue on the road to saving.

## The Last Word

- **Computational thinking** is "reformulating a seemingly difficult problem into something a person can know how to solve"
- From 2005 to 2009, the number of HS's offering AP CS courses declined by 35%
- Article emphasizes my philosophy: "The course is designed to give [students] a sense of computational thinking no matter what they do after this."
  - You will be better, more logical thinkers
    - Better problem solvers
    - Toward efficiency experts

## Extra Credit Opportunities

- Read an article that relates to CS
- Summarize it on the forum under "Extra Credit"
  - 5 pts extra credit on lab grade