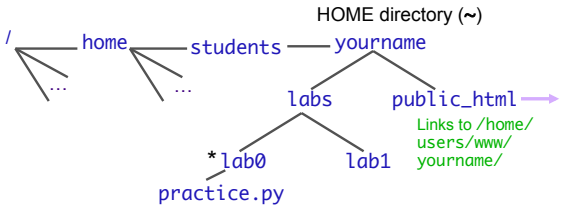
Objectives

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

Review: Linux File System



~ is a shortname for your home directory, i.e., short for `/home/students/yourname`

- 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
- "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

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?

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

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

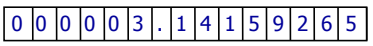
Numeric Primitive Types

Python Data Type	Description	Examples
`int`	Plain integers (32-bit precision)	-214, -2, 0, 2, 100 Range: -2^{31} to $2^{31}-1$
`float`	Real numbers	.001, -1.234, 1000.1, 0.00, 2.45
`long`	Bigger integers (neg or pos, precision limited by computer memory)	2147483648L
`complex`	Imaginary numbers (have real and imaginary part)	1j * 1J → (-1+0j)

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

- We 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!



*In reality, computers represent data in binary, using only 0s and 1s

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."

Can have single quote only inside double quotes*

* Exception later

Booleans: `bool`

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

What is the value's type?

Value	Type
52	
-0.01	
4+6j	
"int"	
4047583648L	
True	
'false'	

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***

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`

Variable Name Conventions

- **Variables** start with lowercase letter
- **Constants** (values that won't change) are in all capitals
  - More on Monday
- Example: Variable for the current year
  - `currentYear`
  - `current_year`
  - `CURRENT_YEAR`
  - ~~`current_year`~~

Naming Variables

- Naming is important
  - 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	`firstName, first_name`
Radius of a circle	`radius`
If someone is employed or not	`isEmployed`

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

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		

Assignment Statements

- Variables can be given any value using the "**=**" sign
  - **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:

```
currentYear = 2008
my_num = 3.4
option = 'q'
```

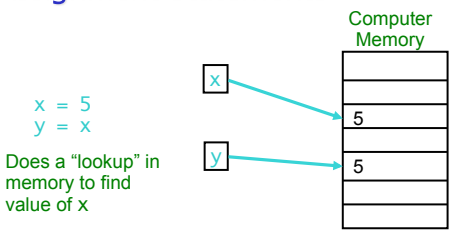
These are **not** equations!
Read "=" as "gets"

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

Assignment Statements



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

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

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"`

Numeric Arithmetic Operations

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

Arithmetic & Assignment

- You can use the assignment operator (=) and arithmetic operators to do calculations
  1. Calculate right hand side
  2. Assign value to variable
- Remember your order of operations! (PEMDAS)
- Examples:

```
x = 4+3*10
y = 3.0/2.0
z = x+y
```

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

Arithmetic & Assignment

- Examples:

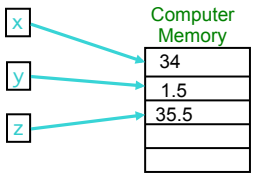
```
x = 4+3*10
y = 3.0/2.0
z = x+y
```

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

Arithmetic & Assignment

- Examples:

```
x = 4+3*10
y = 3.0/2.0
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

What are the values?

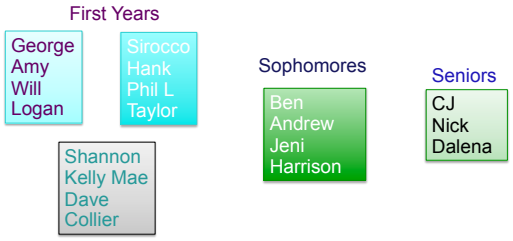
- After executing the following statements, what are the values of each variable?
  - `x = 5`
  - `y = -1 + x`
  - `z = x + y`
  - `y = 2`
  - `x = -7`

What are the values?

- After executing the following statements, what are the values of each variable?
  - `x = 5`
  - `y = -1 + x`
  - `z = x + y`
  - `y = 2`
  - `x = -7`

How can we verify our answers?

Groups for New Programs In CS



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
  - Still some Word characters
    - View your post after you write it
    - Fix as necessary

New Programs in CS

- Did you take a technology/computer/computer science course in high school? What did it teach you?
  - When should students first be exposed to CS or computational thinking?
- What is the difference between "technology education" and "computer science"?
- What is "computational thinking"?
- How could "computational thinking" affect one of your interests (major/hobby/…)

My Notes

- ***Computational thinking*** is "reformulating a seemingly difficult problem into something a person can know how to solve"
- 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
- Later this semester, we'll return to the image of CS

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