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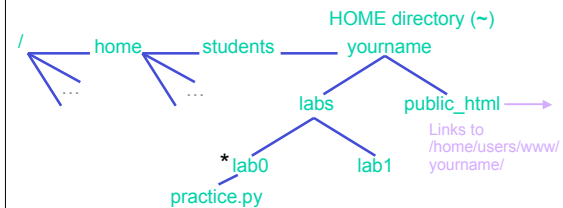
- Review Linux, algorithms
- Programming in Python
  - ➤ Data types
  - ➤ Expressions
  - ➤ Variables
  - ➤ Comments
  - ➤ Arithmetic
- Broader Issues

Review: Linux File System

HOME directory (~)

/ — home — students — yourname
... ... labs   public_html →

*lab0   lab1

Links to /home/users/www/ yourname/

practice.py

- **~** 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?

Computational Problem Solving 101

- 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

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!

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

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 Sara's cat"

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 _
  - Cannot start with a number
  - Cannot be a Python **reserved word**
    - like for, while, def, etc.
- 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

- 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
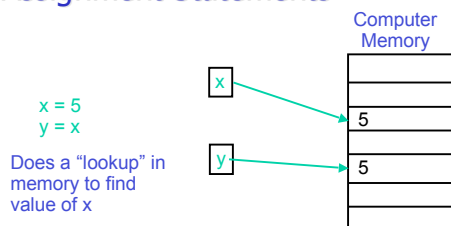
- Variables can be given any value using the "=" sign
  - Syntax: <variable> = <expression>
  - Semantics: <variable> is set to value of <expression>
- After a variable is set to a value, the variable is said to be **initialized**
- These aren't equations! Read "=" as "gets"

  current_year = 2008
  my_num = 3.4
  option = 'q'

Assignment Statements

Computer Memory

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

Variables: The Rules

1. Only the variable(s) to the **left** of the = change
   - In this class, we'll usually only have one variable on the left
2. You should **initialize** a variable before using it on the right-hand side (rhs) of a statement
3. You can only have one variable with any given name in a particular block.
   - Otherwise value changes

Literals

- Pieces of data that are not variables are called **literals**
  - We've been using these a lot already
- Ex:

  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
  - First, calculate right hand side
  - Then, assign value to variable
- Remember your order of operations! (PEMDAS)
- Examples:

  x = 4+3*10

  y = 3.0/2.0

  z = x+y

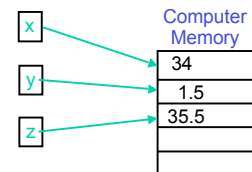 The right-hand sides are **expressions**, just like in math.

Arithmetic & Assignment

- Examples:

  x = 4+3*10

  y = 3.0/2.0

  z = x+y

  Computer Memory: 34, 1.5, 35.5

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

Four Puzzles in Cyberspace

- Context: Book *Code v2* by Lawrence Lessig
- You read Chapter 2
  - Presents the problems, not the author's proposed solutions

Groups:

Alex	Andrew	Clay	Dave
G. Colin	Greg	J. Ty	Joa
Joe	Julie	Lucy	Nay
R. Arturo	Stuart	Vasil	

Broader CS Issues

- Good summaries!
  - Good English, complete sentences

- Mechanics details
  - Post as **comments** to keep the blog a bit more organized
  - Follow instructions on "CS Issues" about what summary should contain
  - Can edit your own posts
  - May want to write in a word processor and then copy over to blog

Four Puzzles from Cyberspace

- How many of you knew about MMOGs before reading this article?
  - How may of you participated in this or something similar?
- How many of you read online?
  - Printed out my PDF version?
  - Printed out the Web page?
- What are the main themes?

Four Puzzles from Cyberspace

- Each group choose: Which is the most important puzzle to solve?
- Every advancement in technology has positive and negative effects
  - What are the positive and negative effects of email? IM?
- What CS information would you need to know to be able to propose solutions?