Objectives

- Review algorithms
- Programming in Python
  - Data types
  - Expressions
  - Variables
  - Arithmetic

Review

- What is an algorithm?
- What did we learn from the PB&J demonstration?

Review: Lab

- Learned some UNIX commands
- Created a Web page

What did you learn?

Review: Lab

- Learned some UNIX commands
- Created a Web page

- Lessons learned:
  - Problems are fixable (often just typos!)
  - No "sorry" → you're learning
  - Learn from, adapt examples
  - Find a good solution

Review: UNIX

- UNIX is a bad parent
  - Doesn't tell you when you've done something right
  - Only tells you when you've done something wrong

Terminal:

```
sprenkle@spartacus Desktop$ cp lab00.ppt.pdf lab00.pdf
sprenkle@spartacus Desktop$
```

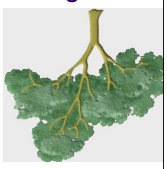
CORRECT! Because didn't get an error message!

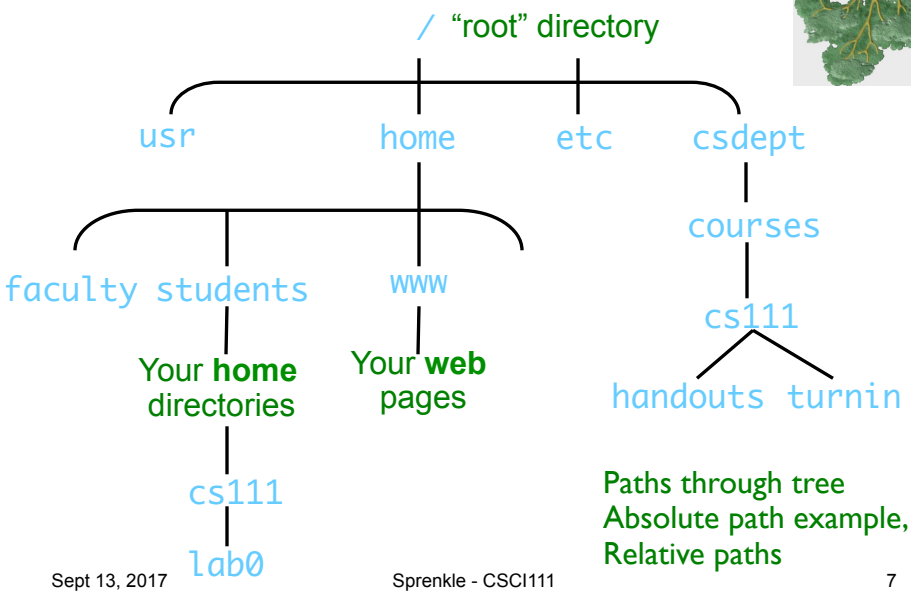
Review: Linux

- How do you …
  - List the files in a directory?
  - Change your current directory?
  - Make a directory?
  - Find out the current directory?
  - Make copies of files?
- What is the shortcut for …
  - The current directory?
  - The parent directory?
  - Your home directory?

(Partial) Linux File Structure

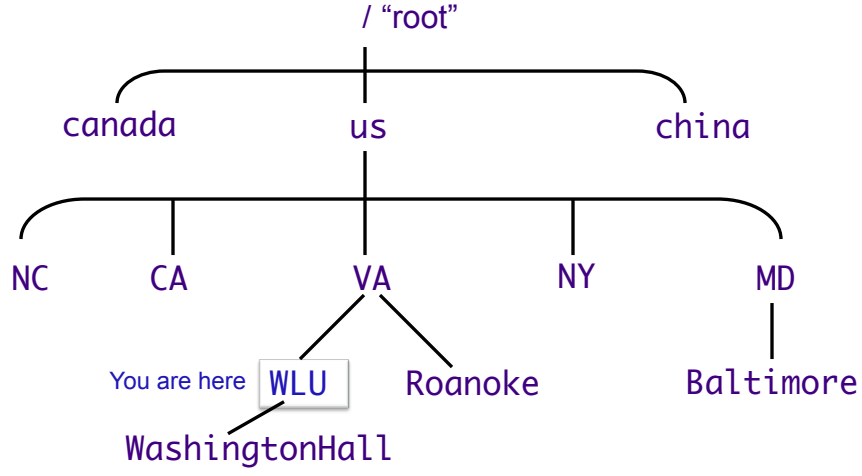
Paths through tree





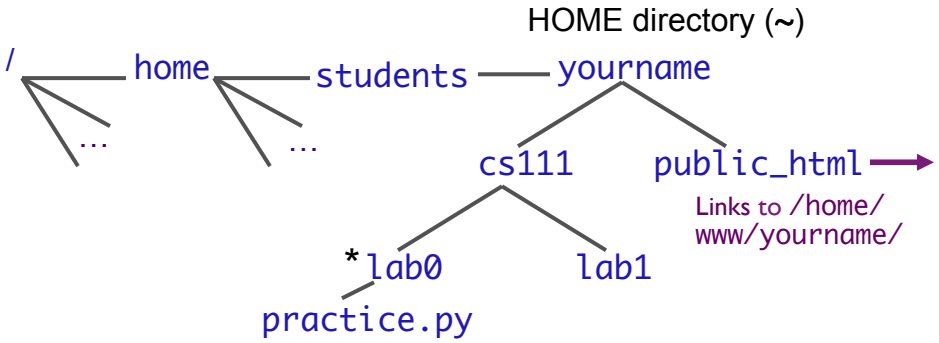
Paths through tree
Absolute path example,
Relative paths

Relative Paths vs Absolute Paths



- Given that you're at WLU, how would you get to Washington Hall? To Roanoke? To Baltimore?

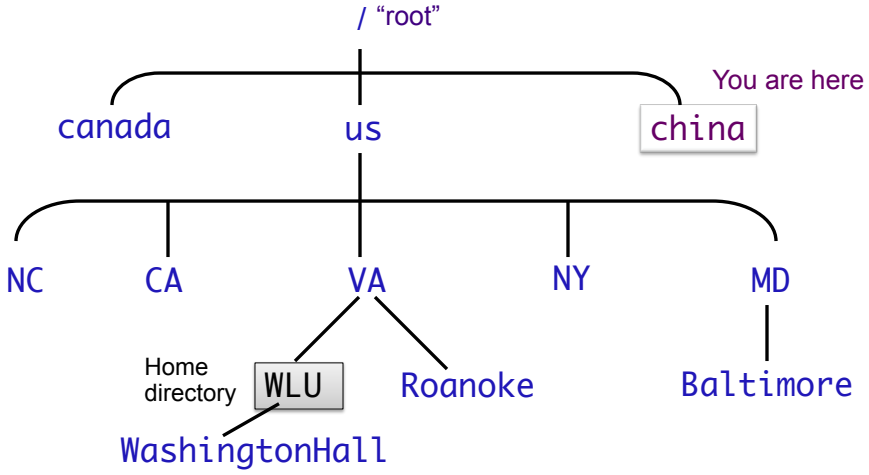
Review: Linux File System



~ 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 `cs111`?

Relative Paths vs Absolute Paths



- Given that you're in China, how would you go to Canada? WLU? Washington Hall?

Review: Labs

- "That's it?"
  - Often, students get overwhelmed by the directions, but then the work isn't that difficult
- Worth 34% of your grade
  - Should get in B+/A- range *easily* with help from student assistants and me

Review: 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

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

Printing Output

- `print` is a special command or a ***function***
  - Displays the result of expression(s) to the terminal
  - Automatically adds a '\n' (carriage return) after it's printed
    - Relevant when have multiple print statements

- `print("Hello, class")`

string literal

Syntax: a set of double quotes
Semantics: represents text

Printing Output

- `print` is a special command
  - Displays the result of expression(s) to the terminal
- `print("Hello, class")`

string literal

`print` automatically adds a `'\n'` (carriage return) after it's printed

- `print("Your answer is", 4*4)`

Syntax: comma
Semantics: print multiple "things" in one line

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

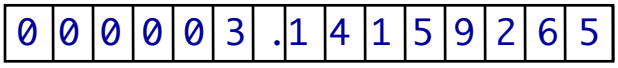
Numeric Primitive Types

Python Data Type	Description	Examples
`int`	Plain integers (32-bit precision)	-214, -2, 0, 2, 100
`float`	Real numbers	.001, -1.234, 1000.1, 0.00, 2.45
`complex`	Imaginary numbers (have real and imaginary part)	1j * 1J → (-1+0j)

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!



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

Booleans: `bool`

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

What is the value's type?

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

What is the value's type?

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

Literals

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

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

Introduction to Variables

- Variables save data/information
  - Example: first slice of bread or knife A
  - 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
- 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

Naming doesn't matter to computer. Matters to humans

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

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

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		
Graduation Year		

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	int or float	`salary`
Sales tax	float	`salesTax`
If item is taxable	boolean	`isTaxable`
Course name	str	`course_name`
Graduation Year	int	`gradYear`

Variable names are just suggestions, Many other possible variable names

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"

Variables: The Rules

- Only the variable(s) to **left** of the =
  in the current statement 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

Assignment Statements

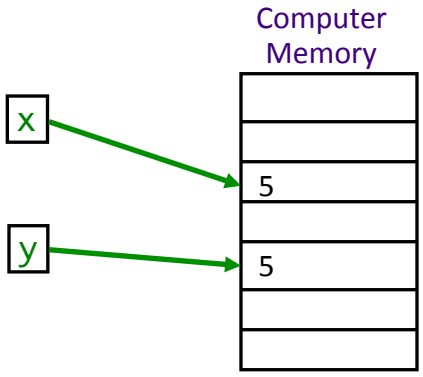
```
x = 5
y = x
```

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

Assignment Statements

```
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

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:

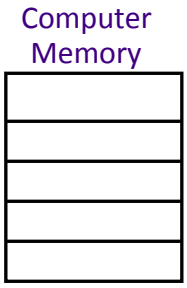
```
x = 4+3*10
y = 3/2.0
z = x+y
```

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

Arithmetic & Assignment

- Examples:

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

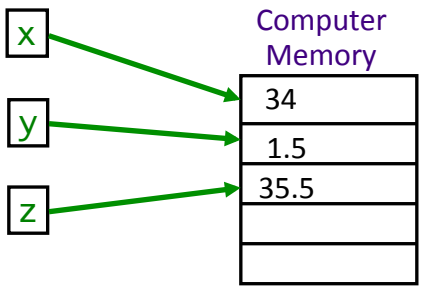


- For last statement
  - need to "lookup" values of x and y
  - computer remembers the result of the expression, not the expression itself

Arithmetic & Assignment

- Examples:

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



- For last statement
  - need to "lookup" values of x and y
  - computer remembers the result of the expression, not the expression itself

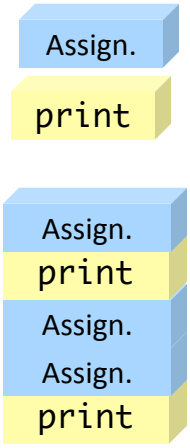
What are the values?

- After executing the following statements, what are the values of each variable?
  - `r = 5`
  - `s = -1 + r`
  - `t = r + s`
  - `s = 2`
  - `r = -7`

How can we verify our answers?

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



Bringing It All Together: A simple program

```
# Demonstrates arithmetic operations and
# assignment statements
# by Sara Sprenkle

x = 3
y = 5

print("x =", x)
print("y =", y)

print("x * y =", x*y)

# alternatively:
# result = x * y
# print("x*y =", result)
```

What does this program output?

arith_and_assign.py

Bringing It All Together: A simple program

```
# Demonstrates arithmetic operations and
# assignment statements
# by Sara Sprenkle

x = 3
y = 5

print("x =", x)
print("y =", y)

print("x * y =", x*y)

# alternatively:
# result = x * y
# print("x*y =", result)
```

Comments: human-readable descriptions. Computer does not execute.

arith_and_assign.py

Looking Ahead

- Complete “Introduction” assignment by Friday
- Read both articles and write one summary about both, following the guidelines