Lab 0 Objectives

- Intro to Labs
- Intro to Operating Systems
- Why programming languages?
- Start Lab #0
  - UNIX/Linux intro, worksheet
  - Sakai (Forum for "Broader CS Issues")
  - Create Web page
  - Use jEdit (Text Editor), IDLE
  - Use Python interpreter in interactive mode
  - Write Python programs

A lot of different things but doable!

Intro to Labs

- Introduce Student Assistants
  - Will Richardson '11
  - Camille Cobb '12
- 3 hours to get started on labs
  - Often will need to finish lab after lab period
  - Use this lab (P405), preferably, or P413

Operating Systems

- Manage hardware resources
- Three popular operating system variations:

Mac	PC/Windows	UNIX

  - Compare in terms of cost, popularity, available software, security
- Learn Linux (a UNIX variation) in this class

Note: "PC" for Windows is a misnomer because all of these OSs are for "personal computers".

P405 Machines

- Run both Linux and Windows
  - Linux natively
  - Windows virtually
    - Takes a while to start up
  - If need to switch, restart. By default → Linux
- Computer should be in Linux
  - If not, tell someone or move to another computer
- P413: Linux-only

Pause While You Log In

- Follow handout's instructions
- Open browser
- Navigate to Lab 0, from course's "Schedule" page
  - We're starting on the first objective "Learning to Use the Linux Machines" on paper
  - Return to Web page for rest of lab
- How different is the UI than Windows or Mac?

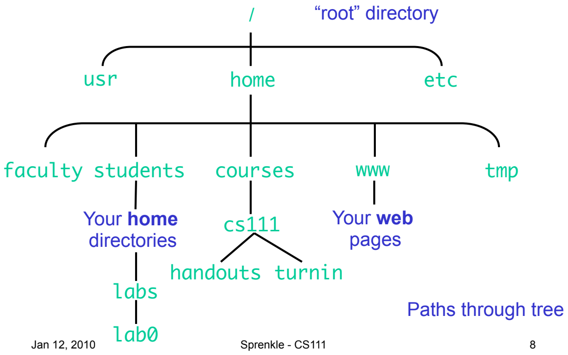
Intro to UNIX Handout

- A lot of words
- Not that difficult
- Introduces terminology and techniques that will be second nature to you in a few weeks

Intro to UNIX

- Can execute operations by typing commands in terminals or using GUIs
  - We will use terminals most of the time
  - Today: learn essential UNIX commands
  - Why?
    - Faster to use keyboard than mouse
    - Easier to automate
- File structure
  - Organize our files
  - Hierarchy of *directories* ("folders" in Windows world)

(Partial) Linux File Structure



/ "root" directory

usr home etc

faculty students courses www tmp

Your **home** directories

cs111

Your **web** pages

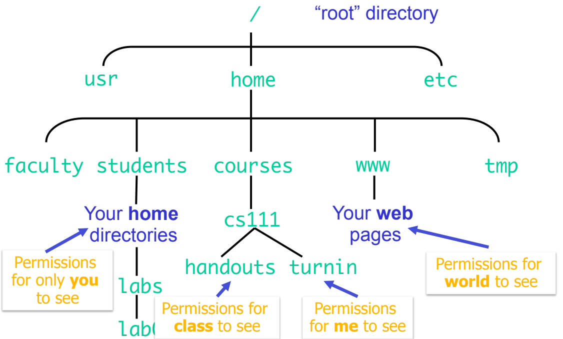
handouts turnin

labs

lab0

Paths through tree

(Partial) Linux File Structure



/ "root" directory

usr home etc

faculty students courses www tmp

Your **home** directories

cs111

Your **web** pages

Permissions for only **you** to see

handouts turnin

Permissions for **world** to see

labs

Permissions for **class** to see

Permissions for **me** to see

lab0

Synchronizing …

- Everyone started Linux worksheet
- Review:
  - True or False: I should shut down the machine when I am done using it.
  - True or False: My CS account is the same as my W&L account.
- Open a new terminal using your shortcut in the top bar

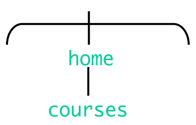
Intro to UNIX: Essential Commands

- Manipulating Files
  - `ls` - list the files, directories in a directory
  - `mkdir` - make a directory
  - `cp` - copy a file/directory
  - `mv` - move a file/directory
  - `rm` - remove (delete) a file/directory
- Navigating Directories
  - `pwd` - "print working directory"
  - `cd` - "change directory"

Intro to UNIX: Shortcuts

- .
  - Current directory
- ..
  - Parent directory

Often used with `cp, mv, cd` commands



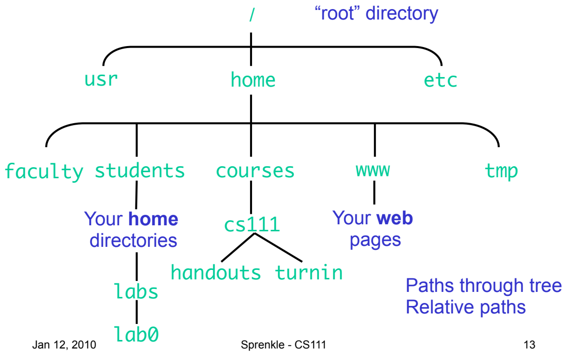
/

home

courses

- `cd` or `cd ~`
  - Change to your HOME directory

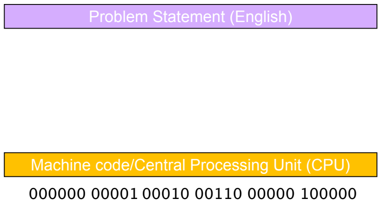
(Partial) Linux File Structure



Paths through tree
Relative paths

Why Do We Need Programming Languages?

- Computers can't understand English
  - Too ambiguous
- Humans can't easily write machine code



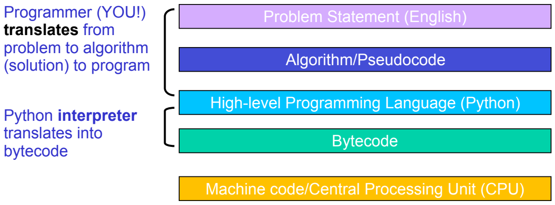
Problem Statement (English)

Machine code/Central Processing Unit (CPU)

000000 00001 00010 00110 00000 100000

Why Do We Need Programming Languages?

- Computers can't understand English
  - Too ambiguous
- Humans can't easily write machine code



Programmer (YOU!) **translates** from problem to algorithm (solution) to program

Python **interpreter** translates into bytecode

Problem Statement (English)

Algorithm/Pseudocode

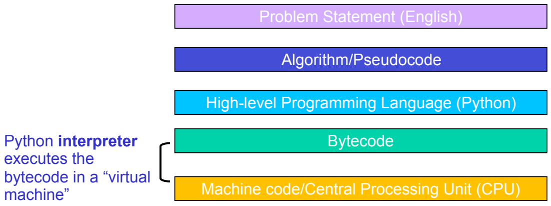
High-level Programming Language (Python)

Bytecode

Machine code/Central Processing Unit (CPU)

Why Do We Need Programming Languages?

- Computers can't understand English
  - Too ambiguous
- Humans can't easily write machine code



Python **interpreter** executes the bytecode in a "virtual machine"

Problem Statement (English)

Algorithm/Pseudocode

High-level Programming Language (Python)

Bytecode

Machine code/Central Processing Unit (CPU)

Python Is …

- A programming language
- An interpreter (which is a program) that executes Python code

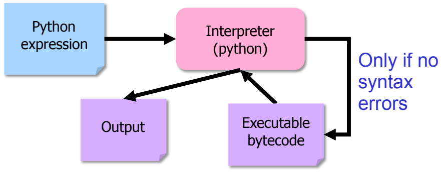
---

Python

- A common *interpreted* programming language
  - Runs on many operating systems
- First released by Guido van Rossum in 1991
- Named after *Monty Python's Flying Circus*
- Minimalist syntax, emphasizes readability
- Flexible, fast, useful language
- Used by scientists, engineers, systems programmers

Python Interpreter

1. Validates Python programming language expression(s)
   - Enforces Python syntax rules
   - Reports syntax errors ← Have a lot of these early on!
2. Executes expression(s)

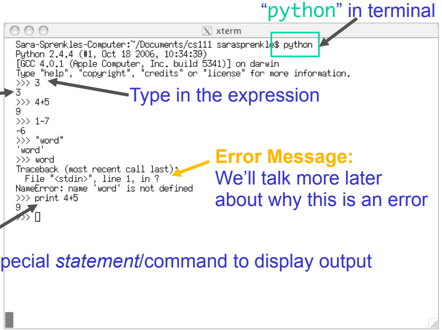

Python expression
Interpreter (python)
Only if no syntax errors
Output
Executable bytecode

Two Modes to Execute Python Code

- **Interactive**: using the interpreter
  - Try out Python expressions
- **Batch**: execute *scripts* (i.e., files containing Python code)
  - What we'll write usually

Interactive Mode

Run by typing "`python`" in terminal



```
Sara-Sprenkles-Computer:~/Documents/cs111 sarasprenkle$ python
Python 2.4.4 (#1, Oct 18 2006, 10:34:39)
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 3
3
>>> 4+5
9
>>> 1-7
-6
>>> "word"
'word'
>>> word
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'word' is not defined
>>> print 4+5
9
>>> 
```

Python displays the result

Type in the expression

Error Message: We'll talk more later about why this is an error

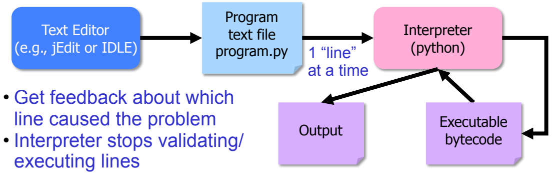
`print`: Special *statement*/command to display output

Your Turn in Interactive Mode…

- Run the Python interpreter in the terminal
  - `python`
- Enter the following expressions and see what Python displays:
  - `3`
  - `4 * -2`
  - `-1+5`
  - `2 +`
  - `print "Hello!"`
- To quit the interpreter, use Control-D

Batch Mode

1. Programmer types a program/script into a **text editor** (jEdit or IDLE).
2. An interpreter turns each expression into bytecode and then executes each expression


Text Editor (e.g., jEdit or IDLE)
Program text file program.py
1 "line" at a time
Interpreter (python)
Output
Executable bytecode

- Get feedback about which line caused the problem
- Interpreter stops validating/ executing lines

Example Python Script

- What does this program do?

Text file named: `hello.py`

```
# Program that prints out "Hello, world!"
# by Sara Sprenkle
# Last modified: 01/06/2009

print "Hello, world!"
```

Print statement

- Validate your guess by executing the program
  - Go into `labs/lab0` directory
  - `python hello.py`

Example Python Script

```
# Program that prints out "Hello, world!"
# by Sara Sprenkle
# Last modified: 01/06/2009

print "Hello, world!"
```

Documentation
-- good *style*

- Only `Hello, world!` is printed out
- Python ignores everything after the “#”
  - Known as “**comments**” or, collectively, as **documentation**
- Your program should *always* start with a high-level description of what the program does, your name, and the date the program was written

---

Review: Executing Python

- Interactive Mode
  - Try out expressions
  - `python`

- Batch Mode
  - Execute Python scripts
  - `python <pythonscript>`

Practice in Interactive, Batch Modes

- Open the IDLE development environment
  - Command: `idle &`
    - `&` Runs command in “background” so you can continue to use the terminal
  - Knowing our programming language is named after Monty Python, any ideas about what the development environment is named after?

Demonstrate Using IDLE

Honors System: Rules of Thumb

- Discussion of problems/programs - OK
  - Clarification questions
  - Algorithm discussion (on paper, board)
- Debugging help
  - Programmer always “owns” keyboard, mouse
  - Helper can read other’s program/debug/help, up to 5 minutes
    - Ask or email me for problems that require more time

Lab 0 Checklist

- Linux Worksheet
- Sakai access
- Web Page
- Python practice, programs
- Print lab