

## Objectives

- Command-line arguments
- Calls to UNIX tools
- Designing Classes, Larger Programs

Nov 12, 2007

Sprenkle - CS111

1

## Summary: Designing Classes

- What does the object/class represent?
- How to model/represent the class's *data*?
  - Instance variable
  - Data type
- What *functionality* should objects of the class have?
  - How will others want to use the class?
  - Put into methods for others to call (API)

Nov 12, 2007

Sprenkle - CS111

2

## Benefits of Classes

- Package/group related data into one object
- Reusing code
  - E.g., Don't need to check if user put in valid time
- Provide interface, can change underlying implementation
  - e.g., Counter's increment -- could implement like in Caesar Ciphers instead

Nov 12, 2007

Sprenkle - CS111

3

## Considerations for using Classes

- Only use class if you're using most of its functionality/information
  - Don't use Counter for validating if a number is within the valid range; not using the wrapping/current value
- Since don't know implementation, may inadvertently duplicate code
  - Redo something done by class
  - Could have efficiency penalties
  - But time saved reusing code is usually worth it

Nov 12, 2007

Sprenkle - CS111

4

## Command-line Arguments

- Using the **sys** module

```
python command_line_args.py <filename>
```

List of arguments, named **sys.argv**

- How to reference (get value) "<filename>"?

Nov 12, 2007

Sprenkle - CS111 `command_line_args.py`

5

## Command-line Arguments

- Using the **sys** module

```
python command_line_args.py <filename>
```

List of arguments, named **sys.argv**

- How to reference (get value of) "<filename>"?
  - `sys.argv[1]`
  - `sys.argv[0]` is the name of the program
- Have to run from command-line (not from IDLE)
  - Can edit program in IDLE though

Nov 12, 2007

Sprenkle - CS111 `command_line_args.py`

6

## Calling UNIX Commands from Python

- **commands** module
  - `getoutput(<command_str>)`
- Example use
  - `output = getoutput("ls *.py")`
  - **output** is what you'd get in terminal from calling that command

Nov 12, 2007

Sprengle - CS111

`python_list.py`

7

## Top-Down Design

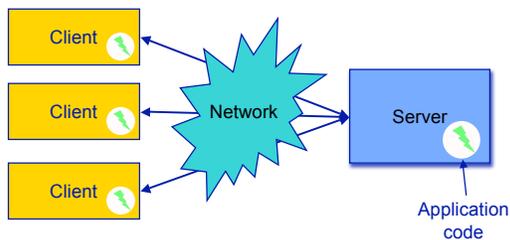
- Break down larger problems into pieces that you can solve
  - Smaller pieces: classes, methods, functions
  - Create stubs, implement smallest pieces and build up
- We've been doing this most of the semester
  - Typically, program was 1) read input, 2) process input, 3) print result
    - Started putting Step 2 into  $\geq 1$  functions
    - Steps 1 and 3 were sometimes a function
  - Now: on larger scale

Nov 12, 2007

Sprengle - CS111

8

## Distributed Programming



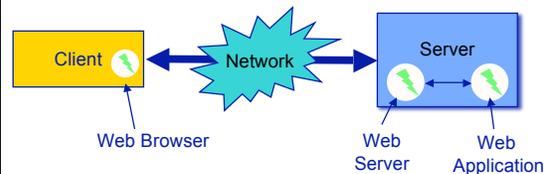
- Server application provides a service
- Client program(s) communicates with server application

Nov 12, 2007

Sprengle - CS111

9

## Web Servers/Applications



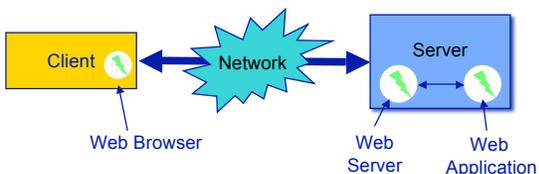
- Specialized **network** programming
- Web browser: makes requests, renders responses; executes JavaScript, client-side code
- Web Server: handles *static* requests
- Web Application: handles *dynamic* requests
  - Response may change based on user input or other state

Nov 12, 2007

Sprengle - CS111

10

## Web Servers/Applications



- May be useful to know who (what clients) are accessing the web application or web server
  - Our next lab

Nov 12, 2007

Sprengle - CS111

11

## Network Addresses

- A computer on a network has an **address**.
  - address is used to uniquely identify the computer (also known as a host) on the network
- The most common address system in use today is the **Internet Protocol (IPv4)** addressing system
  - a 32-bit address, typically written as a "dotted-quad": four numbers, 0 through 255, separated by dots, e.g.,

**137.113.48.2**

Nov 12, 2007

Sprengle - CS111

12

## DNS: Domain Name System

- Translate IP addresses to human-understandable host names and vice versa
  - Example: going from `www.cnn.com` to IP address `64.236.16.20`

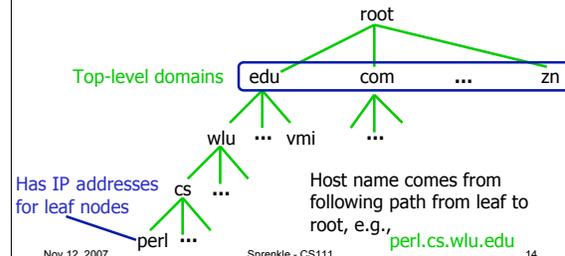
Nov 12, 2007

Sprenkle - CS111

13

## DNS: Domain Name System

- Unique names for computers
- Hierarchical system (tree structure)



## Using the UNIX `host` command

- `host <ipaddress>`
- Examples:
  - `host 64.236.16.20`
    - For `www2.cnn.com`
  - `host 137.113.48.2`
    - `BSC-5000.wlu.edu`
  - `host 209.249.86.17`
    - Host `17.86.249.209.in-addr.arpa` not found: 3(NXDOMAIN)
    - Doesn't have a mapping

Nov 12, 2007

Sprenkle - CS111

15

## Top-Level Domains

- Generic:
  - `edu, com, net, org, gov, mil`
  - Many others now
- Country
  - `us, ca, de, se, ...`

Nov 12, 2007

Sprenkle - CS111

16

## Lab: Parsing Web Access Log

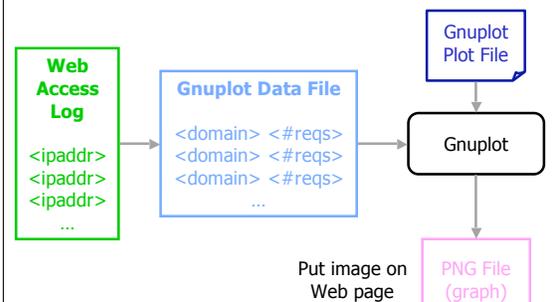
- Problem: Given the IP addresses of requests to a web application, what is the distribution of requests from top-level domains?
  - Show results in graphical way
- Example of real scientific processing
  - Simplified version of my research

Nov 12, 2007

Sprenkle - CS111

17

## Overview: Parsing a Web Access Log



## Pseudocode for Main Function

- Top-down Design

Nov 12, 2007

Sprengle - CS111

19

## Pseudocode for Main Function

- Get input file from user (command-line)
- Create output file name
- Process input file
  - Read each line of input file
    - Convert IP address into host name
    - Compute top-level domain
    - Update mapping of top-level domains to number of requests
  - Print number of lines read
- Write output file
  - Sort domains by number of requests
  - For each top-level domain
    - Print domain id, number of requests

What data structures needed?

Nov 12, 2007

Sprengle - CS111

20

## Pseudocode for Main Function

- Get input file from user (command-line)
- Create output file name
- Process input file
  - Read each line of input file
    - Convert IP address into host name
    - Compute top-level domain
    - Update mapping of top-level domains to number of requests
  - Print number of lines read
- Write output file
  - Sort domains by number of requests
  - For each top-level domain
    - Print domain id, number of requests

Mapping: Tld-name  
→ DomainRequests

Will utilize several built-in modules and our own classes

Nov 12, 2007

Sprengle - CS111

21

## Classes

- WebClientInfo
  - Data:
  - Functionality:
- DomainRequests
  - Data:
  - Functionality:

Nov 12, 2007

Sprengle - CS111

22

## Classes

- WebClientInfo
  - Data: ip address, hostname, top-level domain
  - Functionality: methods to "get" data, constructor, string representation
- DomainRequests
  - Data: name, number of requests
  - Functionality: methods to "get" data, update number of requests, constructor, comparator, string representation

Nov 12, 2007

Sprengle - CS111

23

## Broader Issue

- Facebook's News Feed
- Brought to you by Sue Lister
  - One Laptop Per Child
  - <http://laptop.org>

Nov 12, 2007

Sprengle - CS111

24