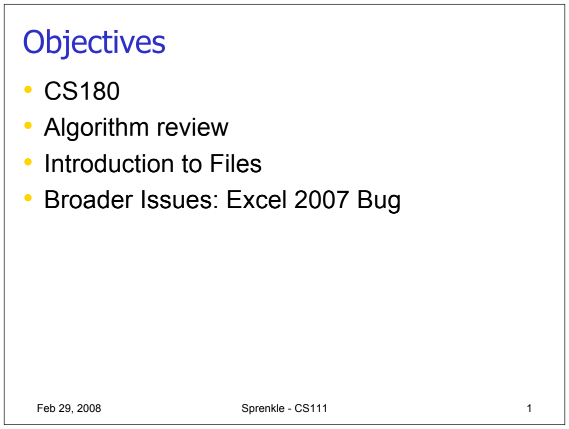
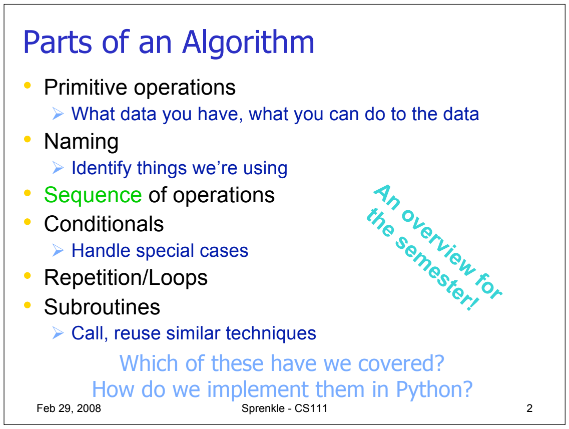


Objectives

- CS180
- Algorithm review
- Introduction to Files
- Broader Issues: Excel 2007 Bug

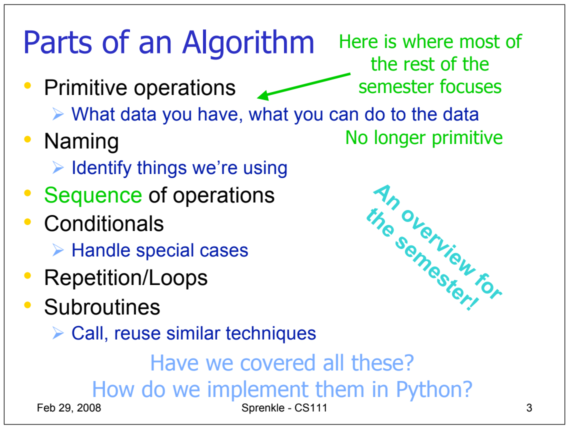


Parts of an Algorithm

- 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

An overview for the semester!

Which of these have we covered?
How do we implement them in Python?



Parts of an Algorithm

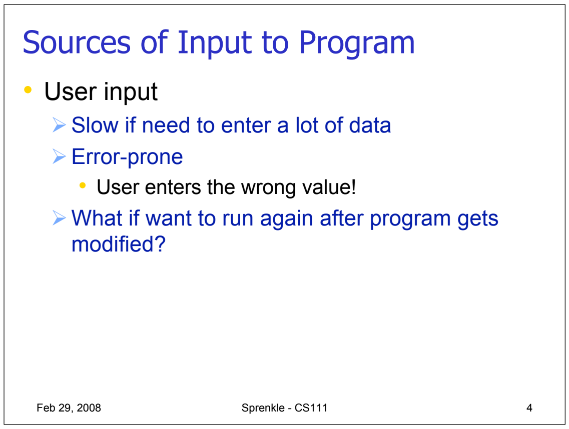
Here is where most of the rest of the semester focuses

- 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

No longer primitive

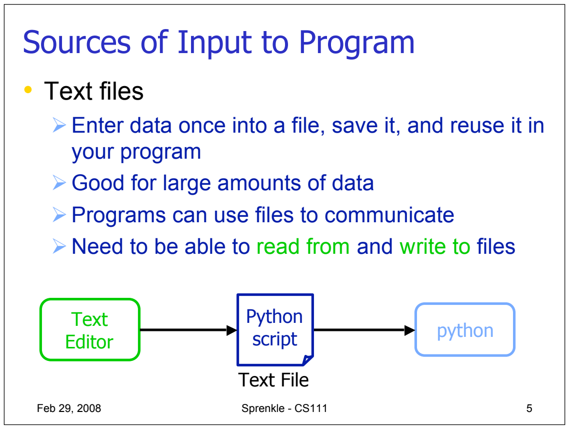
An overview for the semester!

Have we covered all these?
How do we implement them in Python?



Sources of Input to Program

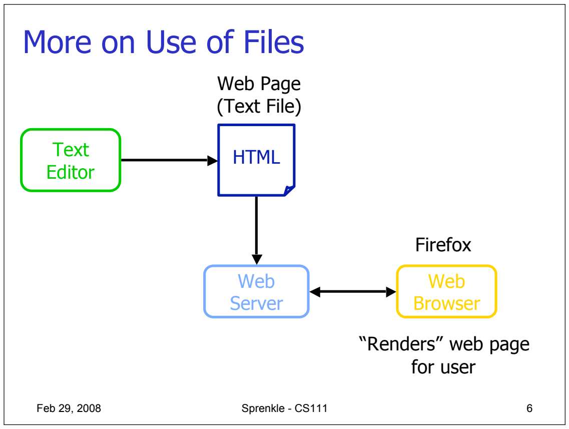
- User input
  - Slow if need to enter a lot of data
  - Error-prone
    - User enters the wrong value!
  - What if want to run again after program gets modified?



Sources of Input to Program

- Text files
  - Enter data once into a file, save it, and reuse it in your program
  - Good for large amounts of data
  - Programs can use files to communicate
  - Need to be able to read from and write to files

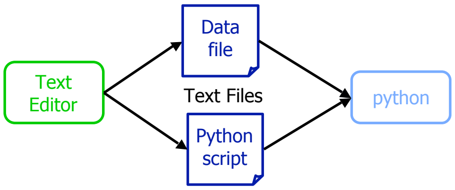
Text Editor
Python script
python
Text File



More on Use of Files

Sources of Input to Program

- Example use of text files as input



Updated Wheel of Fortune

- Uses a file of puzzles
  - Puzzles no longer appear directly in program
  - Can modify puzzle file to get different puzzles

Files

- Conceptually, a file is a **sequence** of data stored in memory
- To use a file in a Python script, create an object of type **file**
  - **file** is a data type
  - <varname> = file(<filename>,<mode>)
    - <filename> : string
    - <mode> : string, either "r" for read or "w" for write
  - Example: dataFile = file("years.dat", "r")

constructor - "constructs" a file object

Common File Methods

Method Name	**Functionality**
read()	Read the entire content from the file, returned as a string object
readline()	Read one line from the file, returned as a string object (which includes the "\n"). If it returns "", then you've reached the end of the file
write(*string*)	Write a string to the file
close()	Close the file. *Must* close the file after done reading from/writing to a file

Reading from a File

- Examples of reading from a file using file methods
  - Show file: data/years.dat
- file_read.py (using read())
  - How is what Python printed different than the file's content?
  - How to fix?
- file_read2.py (using readline())

Typically use .dat or .txt file extension for these types of data/text files

Reading from a File

- Recall that a file is a ***sequence*** of data
- Can use a **`for`** loop to iterate through a file

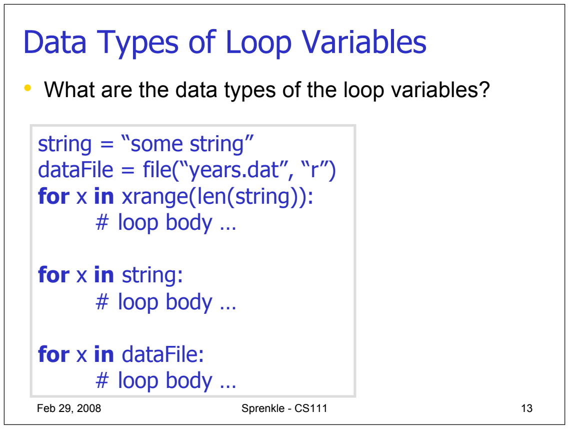
A line (of type **string**) from the file

file object

```
for line in dataFile:
    print line
```

  - Read as: for each line in the file, do something

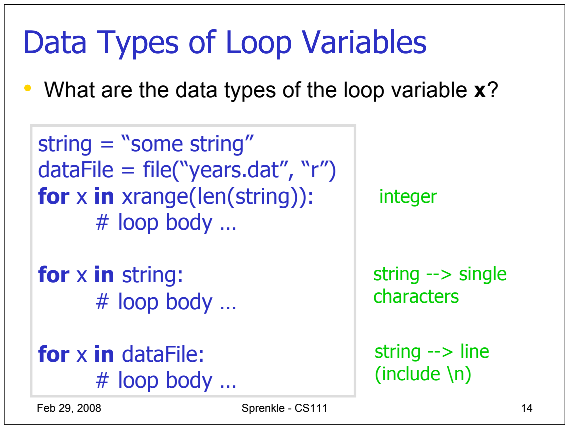
file_read3.py



Data Types of Loop Variables

- What are the data types of the loop variables?

```
string = "some string"
dataFile = file("years.dat", "r")
for x in xrange(len(string)):
        # loop body ...

for x in string:
        # loop body ...

for x in dataFile:
        # loop body ...
```



Data Types of Loop Variables

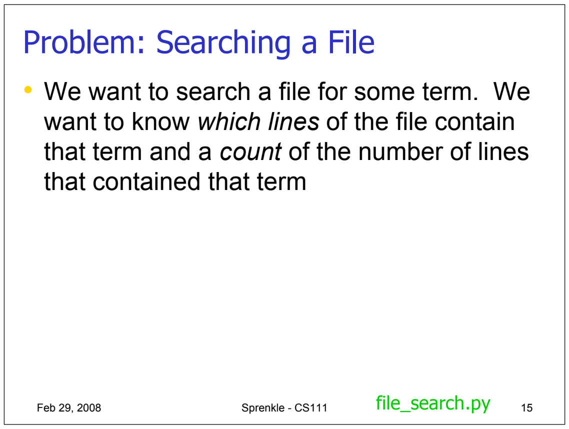
- What are the data types of the loop variable **x**?

```
string = "some string"
dataFile = file("years.dat", "r")
for x in xrange(len(string)):
        # loop body ...

for x in string:
        # loop body ...

for x in dataFile:
        # loop body ...
```

integer

string --> single characters

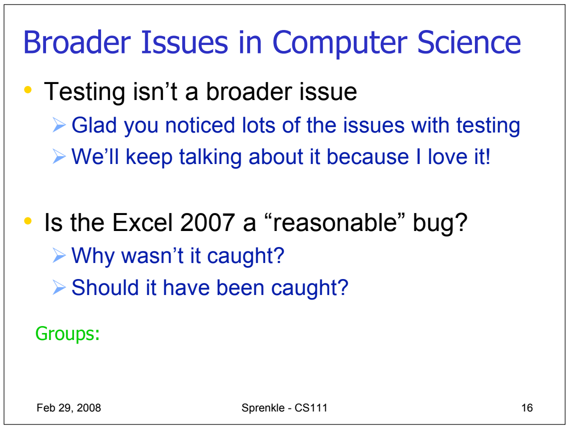
string --> line (include \n)



Problem: Searching a File

- We want to search a file for some term. We want to know *which lines* of the file contain that term and a *count* of the number of lines that contained that term

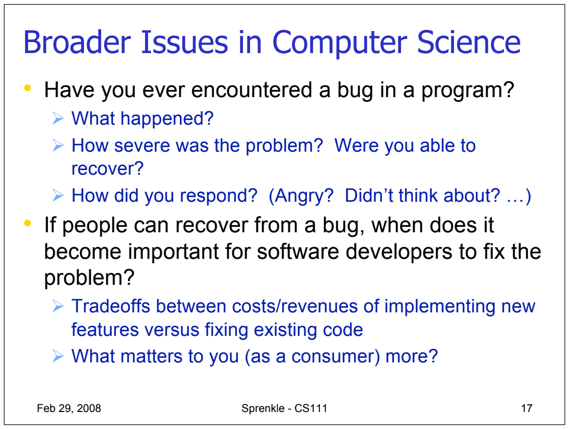
file_search.py



Broader Issues in Computer Science

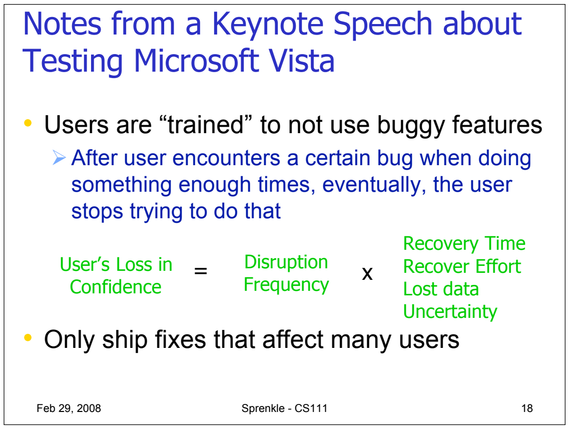
- Testing isn't a broader issue
  - Glad you noticed lots of the issues with testing
  - We'll keep talking about it because I love it!
- Is the Excel 2007 a "reasonable" bug?
  - Why wasn't it caught?
  - Should it have been caught?

Groups:



Broader Issues in Computer Science

- Have you ever encountered a bug in a program?
  - What happened?
  - How severe was the problem? Were you able to recover?
  - How did you respond? (Angry? Didn't think about? …)
- If people can recover from a bug, when does it become important for software developers to fix the problem?
  - Tradeoffs between costs/revenues of implementing new features versus fixing existing code
  - What matters to you (as a consumer) more?

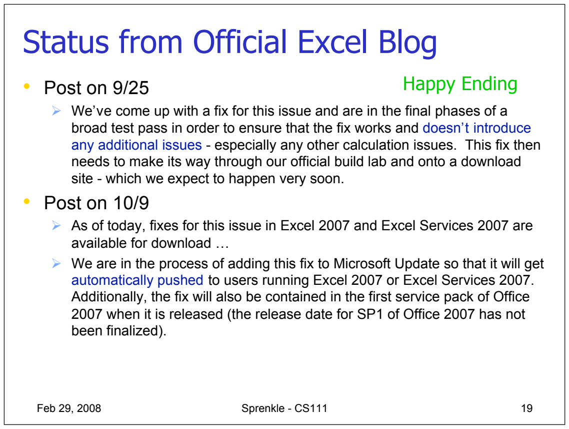


Notes from a Keynote Speech about Testing Microsoft Vista

- Users are "trained" to not use buggy features
  - After user encounters a certain bug when doing something enough times, eventually, the user stops trying to do that

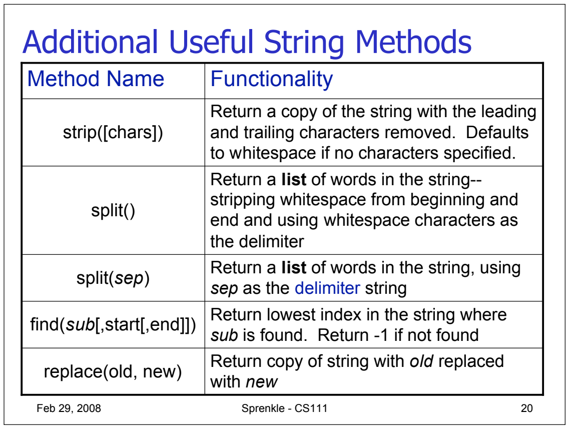
User's Loss in Confidence = Disruption Frequency x Recovery Time
Recover Effort
Lost data
Uncertainty

- Only ship fixes that affect many users



Status from Official Excel Blog

- Post on 9/25 Happy Ending
  - We've come up with a fix for this issue and are in the final phases of a broad test pass in order to ensure that the fix works and doesn't introduce any additional issues - especially any other calculation issues. This fix then needs to make its way through our official build lab and onto a download site - which we expect to happen very soon.
- Post on 10/9
  - As of today, fixes for this issue in Excel 2007 and Excel Services 2007 are available for download …
  - We are in the process of adding this fix to Microsoft Update so that it will get automatically pushed to users running Excel 2007 or Excel Services 2007. Additionally, the fix will also be contained in the first service pack of Office 2007 when it is released (the release date for SP1 of Office 2007 has not been finalized).



Additional Useful String Methods

Method Name	Functionality
strip([chars])	Return a copy of the string with the leading and trailing characters removed. Defaults to whitespace if no characters specified.
split()	Return a **list** of words in the string--stripping whitespace from beginning and end and using whitespace characters as the delimiter
split(*sep*)	Return a **list** of words in the string, using *sep* as the delimiter string
find(*sub*[,start[,end]])	Return lowest index in the string where *sub* is found. Return -1 if not found
replace(old, new)	Return copy of string with *old* replaced with *new*