

Objectives

- Function wrapup
 - Creating modules
- Algorithm review
- Introduction to Files
- Broader Issue: Volunteer Computing

Mar 5, 2010

Sprenkle - CSCI111

1

Review: Functions

CONSTANT = 12 Where does program start "doing stuff"?

```
def main():  
    first = input("Enter the first number: ")  
    second = input("Enter the second number: ")  
    computedVal = myFunction(first, second)  
    print "The answer is", computedVal
```

```
def myFunction(x, y):  
    result = x*x + y*y + CONSTANT  
    return result
```

```
main()
```

What variables
can function
"see" here?
What vars
can't it see?

Mar 5, 2010

Sprenkle - CSCI111

2

Review: Why Functions?

- Organize our code
- Easier to read
- Easier to change
- Easier to reuse

Mar 5, 2010

Sprenkle - CSCI111

3

Swapping Characters

- Had this team:



- Wanted this team (temporarily):



Mar 5, 2010

Sprenkle - CSCI111

4

CREATING MODULES

Mar 5, 2010

Sprenkle - CSCI111

5

Where are Functions Defined?

- Functions can go inside of program script
 - Defined before use/called (if no `main()` function)
- Functions can go inside a separate **module**

Mar 5, 2010

Sprenkle - CSCI111

6

Benefits of Defining Functions in Separate Module

- Reduces code in main script
- Easier to reuse by importing from a module
- Maintains the “black box”
- Isolates testing of function
- Write “test driver” scripts to test functions separately from use in script

Mar 5, 2010

Sprenkle - CSCI111

menu.py

7

Creating Modules

- Modules group together related functions and constants
- Unlike functions, no special keyword to define a module

➤ A module is named by its filename

Just a Python file!

- Example, oldmac.py

➤ In Python shell: `import oldmac`

➤ Explain what happened

Mar 5, 2010

Sprenkle - CSCI111

8

Creating Modules

- So that our program doesn't execute when it is imported in a program, at bottom, add

```
if __name__ == '__main__':  
    main()
```

Not important how this works; just know when to use

- Then, to call `main` function
➤ `oldmac.main()`
- Note the files now listed in the directory

Mar 5, 2010

Sprenkle - CSCI111

9

Creating Modules

- Then, to call `main` function
➤ `oldmac.main()`
- Why would you want to call a module's `main` function?
➤ Use `main` function as driver to test functions in module
- To access one of the defined constants
➤ `oldmac.EIEIO`

Mar 5, 2010

Sprenkle - CSCI111

10

Defining Constants in Modules

- Add constant to `menu.py`
➤ `STOP_OPTION`
- Show use in `menu_withfunctions.py`

Mar 5, 2010

Sprenkle - CSCI111

11

Summary: Program Organization

- Larger programs require **functions** to maintain readability
➤ Use `main()` and other functions to break up program into *smaller, more manageable* chunks
➤ “Abstract away” the details
- As before, can still write smaller scripts without any functions
➤ Can try out functions using smaller scripts
- Need the `main()` function when using other functions to keep “driver” at top
➤ Otherwise, functions need to be defined **before** use

Mar 5, 2010

Sprenkle - CSCI111

12

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

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

Mar 5, 2010

Sprenkle - CSCI111

13

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

Here is where most of the rest of the semester focuses
No longer *primitive*

Mar 5, 2010

Sprenkle - CSCI111

14

Review

- The data type of the loop variable depends on what's after **in**

```
string = "some string"
```

```
for x in xrange(len(string)):  
    # loop body ...  
  
for x in string:  
    # loop body ...
```

What is the data type of the loop variable **x**?

Mar 5, 2010

Sprenkle - CSCI111

15

Review

- The data type of the loop variable depends on what's after **in**

```
string = "some string"
```

```
for x in xrange(len(string)):  
    # loop body ...  
  
for x in string:  
    # loop body ...
```

Integer

String

Mar 5, 2010

Sprenkle - CSCI111

16

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?

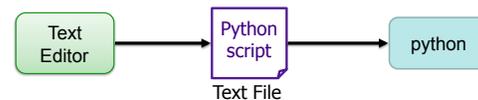
Mar 5, 2010

Sprenkle - CSCI111

17

Sources of Input to Program

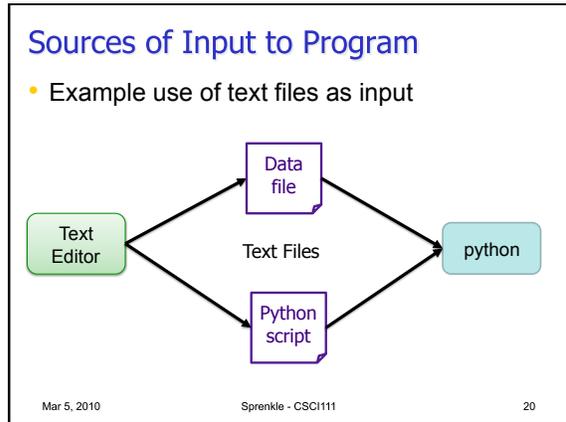
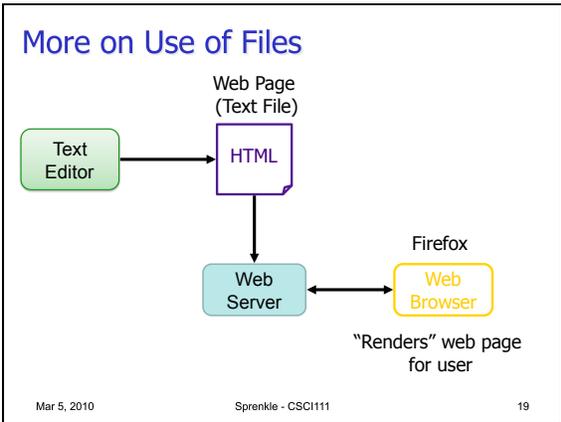
- Text files
 - Enter data once into a file, save it, and reuse it
 - Good for large amounts of data
 - Programs can use files to *communicate*
 - Need to be able to *read from* and *write to* files



Mar 5, 2010

Sprenkle - CSCI111

18



Updated Wheel of Fortune

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

```
def displayPuzzle(puzzle):
    displayedPuzzle = ""
    for char in puzzle:
        if char.isalpha():
            displayedPuzzle += "_"
        else:
            displayedPuzzle += char
    return displayedPuzzle
```

Mar 5, 2010 Sprenkle - CSCI111 21

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*

constructor - "constructs" a file object

```
> <varname> = file(<filename>, <mode>)
```

- <filename> : string
- <mode>: string, either "r" for read or "w" for write

➢ Ex: dataFile = file("years.dat", "r")

Mar 5, 2010 Sprenkle - CSCI111 22

Common File Methods

Method Name	Functionality
read()	Read the entire content from the file, <i>returned</i> as a string object
readline()	Read one line from file, <i>returned</i> 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. <i>Must</i> close the file after done reading from/writing to a file

Mar 5, 2010 Sprenkle - CSCI111 23

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 files containing data/text

Mar 5, 2010 Sprenkle - CSCI111 24

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 **str**)
from the file

file object

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

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

Mar 5, 2010

Sprenkle - CSCI111 file_read3.py 25

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 ...
```

Mar 5, 2010

Sprenkle - CSCI111

26

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 ...
```

integer

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

string → single characters

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

string → line (include \n)

Mar 5, 2010

Sprenkle - CSCI111

27

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

Mar 5, 2010

Sprenkle - CSCI111

file_search.py 28

Next Week

- Monday
 - Files, Introduction to lists
- Tuesday
 - Lab
- Wednesday
 - Lists, Exam Review
- Friday
 - Exam!

Mar 5, 2010

Sprenkle - CSCI111

29

Broader Issue: Volunteer Computing

Amy
Luke
Hank
Ben
CJ

Dave
George
Dalena
Kelly Mae

Collier
Phil
Shannon
Nick

Will
Sirocco
James
Taylor

Logan
Jeni
Andrew
Harrison

Mar 5, 2010

Sprenkle - CSCI111

30

Broader Issue: Volunteer Computing

- What is the goal of the project/problem they are solving?
- Why are computer scientists involved with this problem/its solution?
- What is their solution to the problem?
 - What was their insight to the solution?
- What are some of the results of their solution?
- What are some issues they have had to solve?
- What are other problems that are being solved in similar ways?
- What other problems should we use volunteer computing to solve?
- How does involving the public in science change people's perception of science, if at all?
- How does this article relate to this class?

Mar 5, 2010

Sprenkle - CSCI111

31

Discussion

- Problem: huge computational problems, huge data sets; limited computing resources
 - Supercomputers are expensive
- Insight: lots of computers that are often idle
 - Leverage these cheap resources to create a distributed super computer
- Can break up a huge problem into small pieces that can be solved separately
 - Merge solved pieces back together

Mar 5, 2010

Sprenkle - CSCI111

32

Problems to Solve

- How to break up the problem, how to merge
 - Need *correct, efficient* solutions
- How do we distribute the problems?
- Lots of different OSs, types of machines
 - Process in platform-independent way
- How do we know we're getting the **right** answer?
- What if a volunteer gives unreliable results?
- How can we identify malicious behavior?
- How do we store all the results?

Computer science problems motivated by other domains!

Mar 5, 2010

Sprenkle - CSCI111

33

Kismet: Folding@HOME

Why I chose this article...

- <http://folding.stanford.edu>
- Accurately simulate folding of proteins
- Results help understand diseases and fundamental biology

Washington & Lee University

W&L has a team!

Report generated on	14:48:24 March 03, 2010
Date of last work unit	2010-03-03 04:02:02
Active CPUs within 50 days	2
Team Id	41737
Grand Score	1341158 (certificate)
Work Unit Count	4269 (certificate)
Team Ranking (incl. aggregate)	2059 of 176308
Home Page	http://www.wlu.edu

4