Objectives

- Wrap up functions
 - > Top-down Design
 - Default parameters
- Modules
- Exception Handling

Mar 9, 2016

Sprenkle - CSCI111

Lab Review

http://retrievalpractice.org/

• "... we have excellent evidence that students remember material better when they test themselves and try to retrieve information from their own minds. And yet most students still study by reviewing their notes over and over again — probably the least-effective study strategy they can employ. "

http://chronicle.com/article/Small-Changes-in-Teaching-The/235583

Mar 9, 2016 Sprenkle - CSCI111

Labs

- Main skill you're learning: given a problem, find and apply appropriate solutions
 - Often involves reviewing examples and slides
 - Includes reading/understanding documentation
- Learning good organization, documentation, problem-solving techniques
 - Good to consider why those are best practices
- Make sure you understand what is happening in examples
 - Copy and execute the code
 - Modify the code and execute it to see how your modifications change the execute

Mar 9, 2016

Sprenkle - CSCI111

Lab Review: Design Decisions

- My suggestions for how to design your code (e.g., how to design your functions) are based on some more advanced rules about code organization
 - Reusability
 - > Separation of concerns

Mar 9, 2016

Sprenkle - CSCI111

Testing at Different Levels

Why is a function that automatically tests another function

not sufficient/complete testing?

Mar 9, 2016 Sprenkle - CSCI111

Testing at Different Levels

- Why is a function that automatically tests another function not sufficient/complete testing?
 - The test function does not test the user input/the message that is displayed

Mar 9, 2016 Sprenkle - CSCI111

Review: Refactoring

Converting Functionality into Functions

- 1. Identify functionality that should be put into a function
 - What is the function's input?
 - What is the function's output?
- 2. Define the function
 - Write comments
- 3. Call the function where appropriate
- 4. Create a main function that contains the "driver" for your program
 - Put at top of program
- 5. Call main at bottom of program

Mar 9, 2016

Sprenkle - CSCI111

Practice: Refactoring Palindrome

• Convert the functionality for checking if a phrase is a palindrome into a function

```
phrase=input("What is your phrase? ")
  lowerPhrase=phrase.lower()
  lphrasenospace=lowerPhrase.replace(" ","")
  end=len(lphrasenospace)
  phrase_backwards=""
for x in range(end-1,-1,-1):
    phrase_backwards=phrase_backwards+lphrasenospace[x]
  if phrase_backwards==lphrasenospace:
    print(phrase, "is a palindrome")
  else:
       print(phrase, "is not a palindrome")
                                Sprenkle - CSCI111
Mar 9, 2016
```

TOP-DOWN DESIGN

Mar 9, 2016

Sprenkle - CSCI111

Designing Code

- 1st Approach: Bottom-up
 - Create functions
 - Call functions
- 2nd Approach: Refactoring
 - ➤ Write code
 - Refactor code to have functions
 - Call those functions
- 3rd approach: Top-down Design
 - Write code, calling functions
 - Write "stub" functions
 - Fill-in functions later

Mar 9, 2016 Sprenkle - CSCI111

Top-Down Design:

Alternative Approach to Development

- 1. Create overview, e.g., in main
- 2. Define functions later

def main():
 # get the binary number from the user, as a string
 binNum = input("Please enter a binary number: ")
 isBinary = checkBinary(binNum)
 if not isBinary : # equivalent to isBinary == False
 print(binNum, "is not a binary number.")
 sys.exit() decVal = binaryToDecimal(binNum)
print(binNum, "is", decVal)

- Know what functions you need Know the requirements for your functions

Problem: Create a Summary Report

- Given: a file containing students names and their years (first years, sophomore, junior, or senior) for this class
- **Problem**: create a report (in a file) that says the year and how many students from that year are in this class, on the same line.

writeSumReport.py

Mar 9, 2016 Sprenkle - CSCI111

Problem: Create a Summary Report Given: a file containing students names and their years (first years, sophomore, junior, or senior) for

this class

 Problem: create a report (in a file) that says the year and how many students from that year are in this class, on the same line.

def main():

get name of data file

open output file
for searchTerm in searchTerms:
 numFound = numOccurrences(searchTerm, dataFileName)
 outputFile.write("%s %d\n" % (searchTerm, numFound))

close output file
Example of top-down design:

Can fill in details, e.g., the comments, the function numOccurrences

Mar 9, 2016 Sprenkle - CSC1111 13

Development Advice

- Build up your program in steps
 - Always write small pieces of code
 - ➤ Test function separately from other code, using a test function
 - Test, debug. Repeat
- Development Options:

May use more than one

- > Refactor:
- Write function body as part of main, test
- Then, separate out into its own function
- ➤ Top-down design
- ➤ Bottom-up design

Example: Could still refactor

Mar 9, 2016 Sprenkle - CSCI111

14

PARAMETER DEFAULTS

Mar 9, 2016 Sprenkle - CSCI111

Defaults for Parameters

- · Can assign a default value to a parameter
 - ➤ In general, in function header, default parameter(s) should come *after* all the parameters that *need* to be defined
- Example: range function
 - Didn't have to specify start or increment when calling the function
 - > Default start = 0
 - > Default increment = 1

Mar 9, 2016 Sprenkle - CSCI111 16

Using Default Parameters

 By default, the genWinningNum function could assume that there are 4 numbers

Assigns a value to numNums
ONLY IF not passed a parameter

15

17

def genWinningNum(numNums=4):
 winNum = ""

for i in range(numNums):
 winNum += str(randint(MIN_VALUE,MAX_VALUE))
return winNum

Examples of calling function: genWinningNum(6)

genWinningNum()
genWinningNum(4)

Mar 9, 2016 Sprenkle - CSCI111

CREATING MODULES

Mar 9, 2016 Sprenkle - CSCI111

18

Where are Functions Defined?

- Functions can go inside of program script
 - Defined before use/called (if no main() function)
 - > Or, below the **main**() function (preferred)
- Functions can go inside a separate module



19

21

23

Mar 9, 2016

Sprenkle - CSCI111

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 9, 2016 Sprenkle -

Sprenkle - CSCI111 20

Defining Constants in Modules

- Constant in oldmac.py
 - **≻EIEIO**

Mar 9, 2016

Sprenkle - CSCI111

Creating Modules

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



- Then, to call main function
 - ▶oldmac.main()
- Note the sub-directories now listed in the directory

Mar 9, 2016 Sprenkle - CSCI111 22

Creating Modules

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

Mar 9, 2016

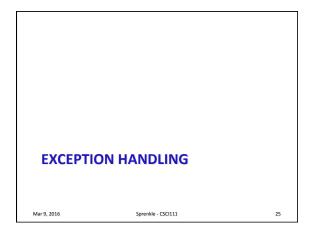
Sprenkle - CSCI111

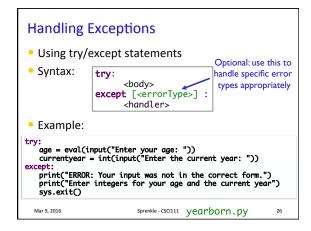
Benefits of Defining Functions in Separate Module

- Reduces code in *primary* driver script
- Easier to reuse by importing from a module
- Maintains the "black box"
 - ➤ Abstraction
- Isolates testing of function
- Write "test driver" scripts to test functions separately from use in script

Mar 9, 2016

Sprenkle - CSCI111





Handling Exceptions Other types of exceptions File exceptions: File doesn't exist Don't have permission to read/write file Mar 9, 2016 Sprenkle-CSCI111 file_handle.py 27

