

## Objectives

- Files
  - Writing files
  - Handling numeric data
- More on functions
  - Passing parameters
- Top-down design

March 6, 2017

Sprenkle - CSC1111

1

## Review: Files

- How do we create file objects?
- How do we read from files?
- What should we do after opening a file?

March 6, 2017

Sprenkle - CSC1111

2

## Review: Common File Methods

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

March 6, 2017

Sprenkle - CSC1111

3

## Writing to a File

- Create a file object in **write** mode:
  - `myFile = open("output.txt", "w")`
- Example: create a file from user input
  - `file_write.py`

What happens if you execute the program again with different user input?

March 6, 2017

Sprenkle - CSC1111

4

## Handling Numeric Data

- We have been dealing with reading and writing *strings* so far
  - Read from a file: get a string
  - Write to file: use a string
- What do we need to do to **read numbers** from a file?
- How can we **write numbers** to a file?

March 6, 2017

Sprenkle - CSCI111

5

## Handling Numeric Data

- We have been dealing with reading and writing *strings* so far
  - Read from a file: get a string
  - Write to file: use a string
- What do we need to do to **read numbers** from a file?
  - Cast as a numeric type, e.g., `int` or `float`
- How can we **write numbers** to a file?
  - Cast number as a `str`

March 6, 2017

Sprenkle - CSCI111

6

## Problem: Temperature Data

- **Given:** data file that contains the daily high temperatures for last year at one location
  - Data file contains one temperature per line
  - Example: `data/florida.dat`
- **Problem:** What is the average high temperature (to 2 decimal places) for the location?

**Rule of Thumb:** Always look at data file before processing it

March 6, 2017

Sprenkle - CSCI111

`avgData.py`

7

## Review: Functions

- What is the keyword we use to create a new function?
- How do we get output from a function?
- What happens in the program execution when a function reaches a `return` statement?
- Why do we write functions?
- What makes a good function?
- How should you comment your functions?
- What is the name for the process for changing a program to improve readability/organization without changing functionality?

March 6, 2017

Sprenkle - CSCI111

8


## TOP-DOWN DESIGN

March 6, 2017

Sprengle - CSCI111

9

## Designing Code

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

March 6, 2017

Sprengle - CSCI111

10

## Top-Down Design: Alternative Approach to Development

1. Create overview
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)
```

Benefit: Know the requirements for your functions first  
→ What is input, what is output

March 6,

11

## 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.

Too many different ways to approach this problem.  
Find a better problem.

`writeSumReport.py`

March 6, 2017

Sprengle - CSCI111

12

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

May use more than one approach in a program

Could still refactor after using these options

March 6, 2017

Sprenkle - CSC1111

13

## Looking Ahead

- Lab 7 due Friday
- Broader Issue due Friday

March 6, 2017

Sprenkle - CSC1111

14