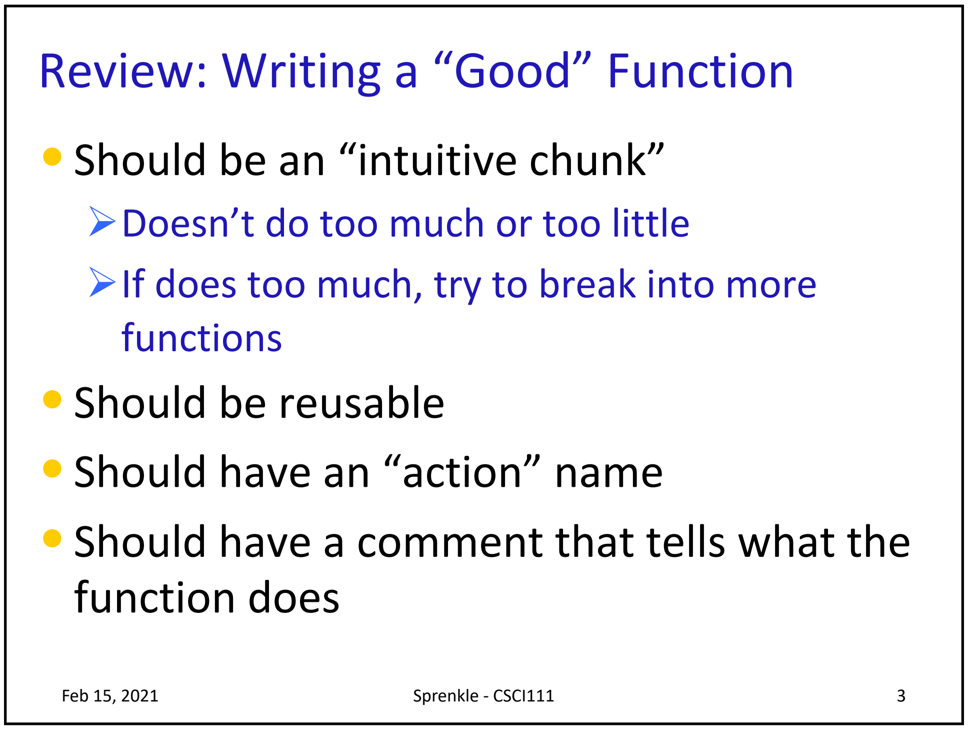
Objectives

- Passing parameters
- Creating Modules
- Alternative development approaches

Review

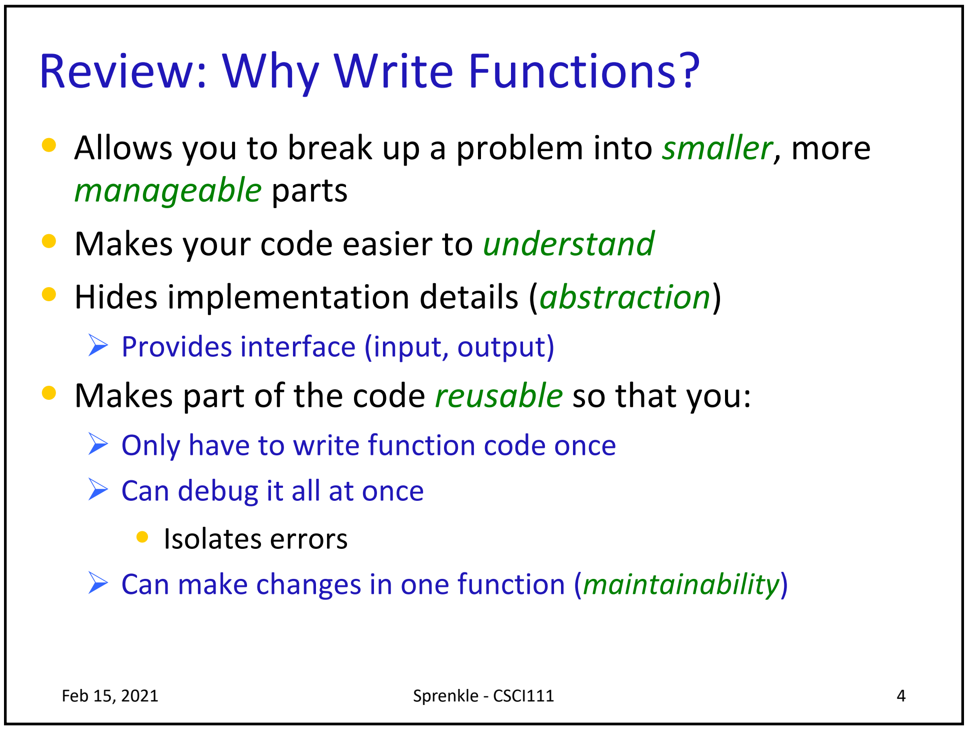
- What makes a “good” function?
- What are benefits of functions?
- How do we organize programs with functions (so far)?
- What new development approach did we discuss?
  - What are its steps?



Review: Writing a “Good” Function

- Should be an “intuitive chunk”
  - Doesn’t do too much or too little
  - If does too much, try to break into more functions
- Should be reusable
- Should have an “action” name
- Should have a comment that tells what the function does

3



Review: Why Write Functions?

- Allows you to break up a problem into *smaller*, more *manageable* parts
- Makes your code easier to *understand*
- Hides implementation details (*abstraction*)
  - Provides interface (input, output)
- Makes part of the code *reusable* so that you:
  - Only have to write function code once
  - Can debug it all at once
    - Isolates errors
  - Can make changes in one function (*maintainability*)

4

Review: Where are Functions Defined?

- Functions can go inside program script
  - If no `main()` function, defined ***before*** use/called
  - If `main()` function, defined anywhere in script
- Functions can go inside a separate ***module***

Review: Refactoring: Converting Functionality into Functions

1. Identify functionality that should be put into a function
   - What should the function do?
   - What is the function's input?
   - What is the function's output (i.e., what is returned)?
2. Define the function
   - Write comments
3. Test the function programmatically
   - Comment out the other code temporarily
4. Call the function where appropriate
5. Create a `main` function that contains the "driver" for your program
   - Put at top of program
6. Call `main` at bottom of program

Why Refactoring?

- Common practice: write code, then realize it would be better (more readable, reusable, …) if it were in a function
- For us: helpful to separate the code implementation from the function implementation

7

Passing Parameters

- Only ***copies*** of the actual parameters are passed to the function
  - For **immutable** data types (which are what we've talked about so far)
- The *actual* parameters in the calling code do not change
- **Swap example:**
  - Swap two values in script
  - Then, put into a function

$$\begin{tikzcd} \begin{matrix} x = 5 \\ y = 7 \end{matrix} \arrow[r] & \begin{matrix} x = 7 \\ y = 5 \end{matrix} \end{tikzcd}$$

Use Python visualizer

`swap.py`

8

CREATING MODULES

Feb 15, 2021 Sprenkle - CSCI111 9

9

Where are Functions Defined?

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

Feb 15, 2021 Sprenkle - CSCI111 10

10

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

11

Creating Modules

- So that our module 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

12

Creating Modules

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

13

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

Refactoring `circleArea.py` → `shapes.py`

14

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

Development approach:

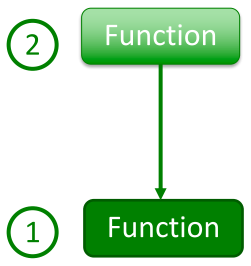
BOTTOM-UP DEVELOPMENT

Bottom-Up Development

- Define a function
  - ➢ Document
  - ➢ Test the function

Bottom-Up Development

- Use the function in context/ call the function
- Define a function
  - ➢ Document
  - ➢ Test the function



Bottom-Up Development Example

1. Define (and document and test) a function that
   - Given a team's wins and losses
   - Returns the team's win percentage

2. Create a program that
   - Prompts for a team's wins and losses
   - Displays the team's win percentage

`winpercent.py`

Broader Issue: Google Search

- Why is Google search a "broader issue"?
- How does Google search work?
  - How is it tested?
- What are some ways you think searches could be improved?
  - How do you measure "improved search"?
- Will you use Google differently, now that you know how it works (kind of)?
- Has Google violated anti-trust laws?

Broader Issue: Google Search

- What power do search engines have?
- Is Google search biased?

Exam Friday

- **Do not panic**
- In-class, on paper
  - Emphasis on critical thinking
- Exam Preparation Document is on course web page
- Similar problems to class and lab
  - Review questions
  - Worksheets
  - Problems
- Content: up through Lab 4
- No broader issue this week

This Week

- Lab 4
  - Practicing ***functions***
  - Due Friday
- Prelab due before lab tomorrow
- Exam Friday
- No broader issue this week