

Objectives

- Programming with Functions
- Refining our development process
- Modules and classes

Feb 5, 2018

Sprenkle - CSCI111

1

Review

- What are benefits of functions?
- How can we test functions easily?
 - What do we need to test functions?

Feb 5, 2018

Sprenkle - CSCI111

2

Practice

- What is the output of this program?

➤ Example: user enters 4

```
def main():  
    num = eval(input("Enter a number to be squared: "))  
    squared = square(num)  
    print("The square is", squared)  
  
def square(n):  
    return n * n  
  
main()
```

Feb 5, 2018

Sprenkle - CSCI111

practice1.py

3

Practice - Scope

- What is the output of this program?

➤ Example: user enters 4

```
def main():  
    num = eval(input("Enter a number to be squared: "))  
    squared = square(num)  
    print("The square is", squared)  
    print("The original num was", num)  
  
def square(n):  
    return n * n  
  
main()
```

Feb 5, 2018

Sprenkle - CSCI111

practice2.py

4

Practice - Scope

- What is the output of this program?

➤ Example: user enters 4

```
def main():  
    num = eval(input("Enter a number to be squared: "))  
    squared = square(num)  
    print("The square is", squared)  
    print("The original num was", n)  
  
def square(n):  
    return n * n  
  
main()
```

Error! **n** does not have a value in function **main()**

Design Patterns

- Former general design pattern:
 1. Optionally, get user input
 2. Do some computation
 3. Display results
- Now general design pattern:
 1. Optionally, get user input
 2. Do some computation in **functions**, get results
 3. Display results

WHAT MAKES A FUNCTION GOOD?

Feb 5, 2018

Sprenkle - CSCI111

7

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
- Always have comment that tells what the function does

Feb 5, 2018

Sprenkle - CSCI111

8

Writing Comments for Functions

- Good style: Each function **must** have a comment
 - Describes functionality at a high-level
 - Include the *precondition*, *postcondition*
 - Describe the parameters (their types) and the result of calling the function (precondition and postcondition may cover this)

Writing Comments for Functions

- Include the function's pre- and post- conditions
- **Precondition**: Things that must be true for function to work correctly
 - E.g., num must be even
- **Postcondition**: Things that will be true when function finishes (if precondition is true)
 - E.g., the returned value is the max

Example Comment

- Describes at high-level
- Describes parameters

```
def printVerse(animal, sound):  
    """  
    Prints a verse of Old MacDonald, plugging in the  
    animal and sound parameters (which are strings),  
    as appropriate.  
    """  
    print(BEGIN_END + EIEIO)  
    print("And on that farm he had a ", animal, EIEIO)  
    ...
```

Comment style: **Docstring**
“documentation string”

Comments from docstrings show up when you use help function

Feb 5, 2018

Sprenkle - CSCI111

11

Development approach:

BOTTOM-UP DEVELOPMENT

Feb 5, 2018

Sprenkle - CSCI111

12

Bottom-Up Development

- Define a function
 - Documentation
- Call the function

Feb 5, 2018

Sprenkle - CSCI111

13

Bottom-Up Development Example

- Define a function that
 - Given a team's wins and losses
 - Returns the team's win percentage
- Create a program that
 - Prompts for a team's wins and losses
 - Displays the team's win percentage

`winpercent.py`

Feb 5, 2018

Sprenkle - CSCI111

14

Another development approach

REFACTORING

Feb 5, 2018

Sprenkle - CSCI111

15

Refactoring

- After you've written some code and it passes all your test cases, the code is probably still not perfect
- **Refactoring** is the process of improving your code *without* changing its functionality
 - Organization
 - Abstraction
 - Example: Easier to read, change
 - Easier to test
- Part of iterative design/development process
- Where to refactor with functions
 - Duplicated code
 - "Code smell"
 - Reusable code
 - Multiple lines of code for one purpose

Feb 5, 2018

Sprenkle - CSCI111

16

Example: PB & J

1. Gather materials (bread, PB, J, knives, plate)
2. Open bread
3. Put 2 pieces of bread on plate
4. Spread PB on one side of one slice
5. Spread Jelly on one side of other slice
6. Place PB-side facedown on Jelly-side of bread
7. Close bread
8. Clean knife
9. Put away materials

- Which of these are the “core” part of making a PB & J sandwich?
- How would you describe the rest of the parts?

Feb 5, 2018

Sprenkle - CSCI111

17

Example: PB & J

1. Gather materials (bread, PB, J, knives, plate)
2. Open bread
3. Put 2 pieces of bread on plate
4. Spread PB on one side of one slice
5. Spread Jelly on one side of other slice
6. Place PB-side facedown on Jelly-side of bread
7. Close bread
8. Clean knife
9. Put away materials

Feb 5, 2018

Sprenkle - CSCI111

18

Example: PB & J as Functions

1. Gather materials (bread, PB, J, knives, plate)
2. Open bread
3. Put 2 pieces of bread on plate
4. Spread PB on one side of one slice
5. Spread Jelly on one side of other slice
6. Place PB-side facedo
7. Close bread
8. Clean knife
9. Put away materials

```
def main():  
    prepare()  
    makePBJSandwich()  
    cleanUpSupplies()  
main()
```

Feb 5, 2018

Sprenkle - CSCI111

19

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. 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

Feb 5, 2018

Sprenkle - CSCI111

20

Our First Problem

- Create three variables (i, j, and result) to calculate and display $\text{result} = i^2 + 3j - 5$ for the case where $i=7$ and $j=2$. Your code will not look exactly like this formula. Display the result and verify that it is correct. Consider if you were the user of the program and make the program display appropriate output.

`arithmetic.py`

Feb 5, 2018

Sprenkle - CSCI111

21

Development approach:

TOP-DOWN DEVELOPMENT

Feb 5, 2018

Sprenkle - CSCI111

22

Top-Down Development

- I have a big problem
- But, that problem can be broken into smaller problems
- Example:
 - I need to draw thick, green lines, whose points are chosen by the user, five times

What piece of this could we break into a function?

- What would the function do?
- What would the input to the function be?
- What would the output of the function be?

Feb 5, 2018

Sprenkle - CSCI111

23

Top-Down Development

- I have a big problem
- But, that problem can be broken into smaller problems
- Example:
 - I need to draw thick, green lines, whose points are chosen by the user, five times
- Break it down:
 - Let user choose two points
 - Draw a thick green line based on those points
 - Repeat five times

Put the rest in a main function

Feb 5, 2018

Sprenkle - CSCI111

24

Our Solution Sketch

- Main program
 - Repeat five times
 - Get user's two points
 - Draw a thick green line based on those points
- Draw a wide green line
 - Input: the two points
 - Process
 - Constructs a line using the two points
 - Makes the line green, thick
 - Draws the line
 - Output: none

During iterative development process, will find issues and need to revise.

Feb 5, 2018

Sprenkle - CSCI111

25

Our Solution Sketch

- Main program
 - Repeat five times
 - Get user's two points
 - Draw a thick green line based on those points
- Draw a wide green line
 - Input: the two points, **the GraphWin to draw the line**
 - Process
 - Constructs a line using the two points
 - Makes the line green, thick
 - Draws the line in the window
 - Output: none

Feb 5, 2018

Sprenkle - CSCI111

26

This Week

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