

## Objectives

- More on functions
  - Passing parameters
  - Refactoring
  - Testing functions
- Broader Issue: Smart Houses

March 3, 2017

Sprenkle - CSC1111

1

## Review

- 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?

March 3, 2017

Sprenkle - CSC1111

2

## Review: Functions

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

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

```
main()
```

What does this program do?  
What is the control flow/execution path?

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

March 3, 2017

Sprenkle - CSC1111

3

## Review: Why Functions?

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

March 3, 2017

Sprenkle - CSC1111

4

## Practice

Terminology note:  
what the program outputs (displays) is different  
from what the function outputs (returns)

- What does this program output?

➤ 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()
```

March 3, 2017

Sprenkle - CSC1111

practice1.py

5

## Practice

- What does this program output?

➤ 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()
```

March 3, 2017

Sprenkle - CSC1111

practice2.py

6

## Practice

- What does this program output?

➤ 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()**

March 3, 2017

Sprenkle - CSC1111

7

## Practice: fixed

- What does this program output?

➤ 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()
```

March 3, 2017

Sprenkle - CSC1111

8

## WHAT MAKES A GOOD FUNCTION?

March 3, 2017

Sprengle - CSCI111

9

## 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

March 3, 2017

Sprengle - CSCI111

10

## 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)

March 3, 2017

Sprengle - CSCI111

11

## 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

March 3, 2017

Sprengle - CSCI111

12

## 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)           Comment style: Docstring  
    print("And on that farm he had a " + animal + EIEIO)  "documentation string"  
    ...
```

Comments from docstrings show up when you use help function

March 3, 2017

Sprengle - CSC1111

13

## Pre/Post Conditions

```
def binaryToDecimal( binary_string ):  
    """  
    pre: binary_string is a string that contains  
    only 0s and 1s  
    post: returns the decimal value for the binary  
    string  
    """  
    dec_value = 0  
    for pos in range( len( binNum ) ):  
        exp = len(binNum) - pos - 1  
        bit = int(binNum[pos])  
  
        # compute the decimal value of this bit  
        val = bit * 2 ** exp  
  
        # add it to the decimal value  
        decVal += val  
  
    return dec_value
```

March 3, 2017

Sprengle - CSC1111

14

## Getting Documentation

- **dir**: function that returns a list of methods and attributes in an object
  - `dir(<type>)`
- **help**: get documentation
  - In the Python shell
    - `help(<type>)`
    - `import <modulename>`
    - `help(<modulename>)`

March 3, 2017

Sprengle - CSC1111

15

## Where is Documentation Coming From?

- Comes from the code itself in **"doc strings"**
  - i.e., "documentation strings"
- Doc strings are simply strings *after* the function header
  - Typically use triple-quoted strings because documentation goes across several lines

```
def printVerse(animal, sound):  
    """prints a verse of Old MacDonald,  
    filling in the strings for animal and  
    sound """
```

March 3, 2017

Sprengle - CSC1111

16

# REFACTORING

March 3, 2017

Sprenkle - CSCI111

17

## 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

March 3, 2017

Sprenkle - CSCI111

18

## 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

March 3, 2017

Sprenkle - CSCI111

19

## Refactoring Practice

- `pick4num.py`
- Where are places that we can refactor and add functions?

March 3, 2017

Sprenkle - CSCI111

20

## Generate Winning Number



- **Input:**
  - Options: none; number of digits; range on random numbers
  - Tradeoffs: more general (more parameters), more difficult to use
- **Output:** winning number

March 3, 2017

Sprenkle - CSCI111

21

## Broader Issue Groups

Ashley Burke Charlotte Lexi Tony	Austin Buddy Jae John Mike	Alex Collin Josette Mira Molly	Leslie Robert Win Zander	Anna Kate George Sarah Victor
--	--	--	-----------------------------------	--

March 3, 2017

Sprenkle - CSCI111

22

## Broader Issue Discussion

- What are promising home activities to automate?
  - What are the challenges in automatically regulating a home?
  - What are difficult tasks to automate?
- What are the privacy concerns briefly mentioned in the articles?
  - Do they require deeper discussion?
- Why does the UVA group focus on smarter *people* rather than smarter thermostats?
  - Is that the right focus?

March 3, 2017

Sprenkle - CSCI111

23