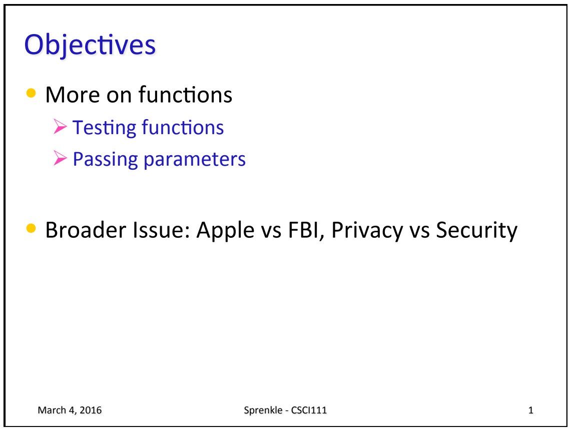
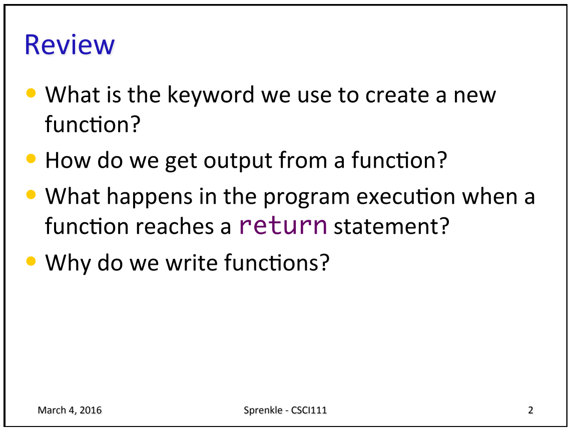


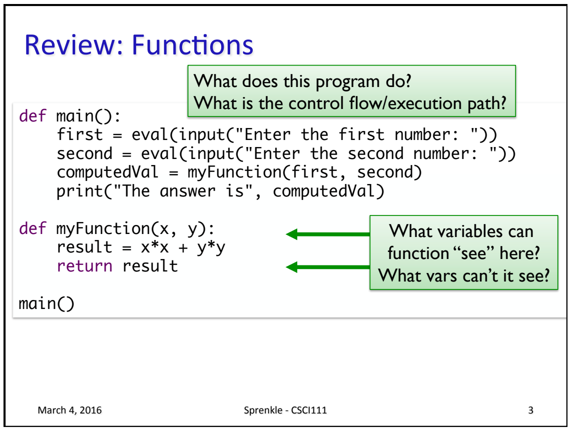
Objectives

- More on functions
  - Testing functions
  - Passing parameters
- Broader Issue: Apple vs FBI, Privacy vs Security



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?

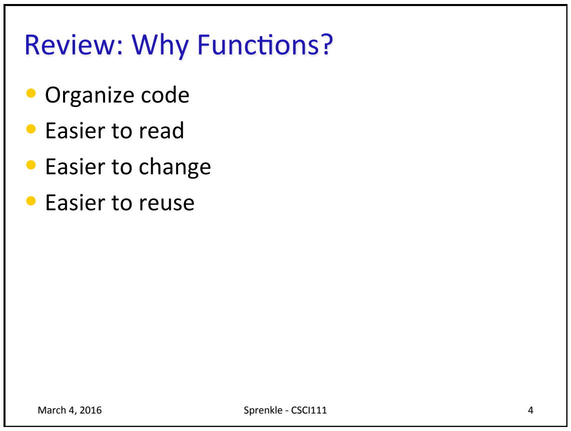


Review: Functions

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

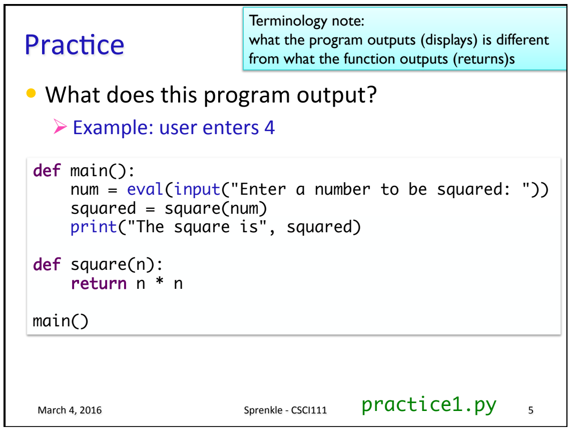
```
def main():
    first = eval(input("Enter the first number: "))
    second = eval(input("Enter the second number: "))
    computedVal = myFunction(first, second)
    print("The answer is", computedVal)

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

main()
```

What variables can function “see” here? What vars can’t it see?



Review: Why Functions?

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



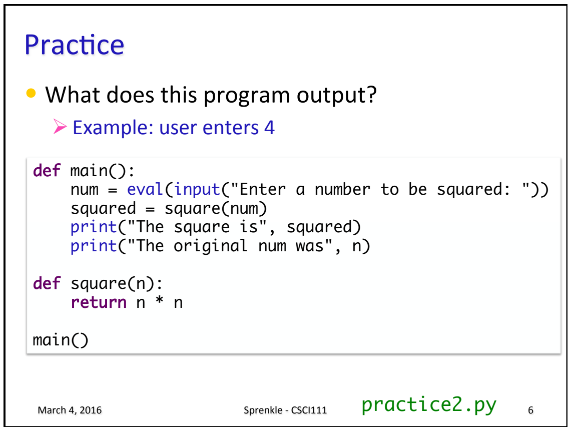
Practice

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

- 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()
```

practice1.py

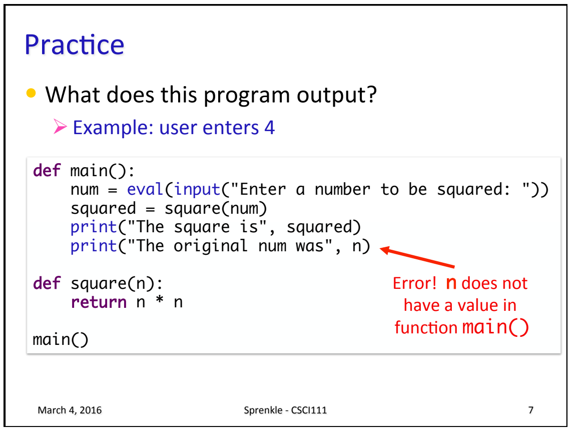


Practice

- 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()
```

practice2.py

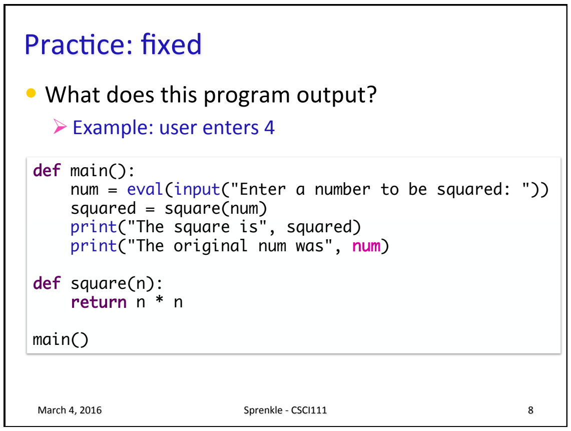


Practice

- 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()
```

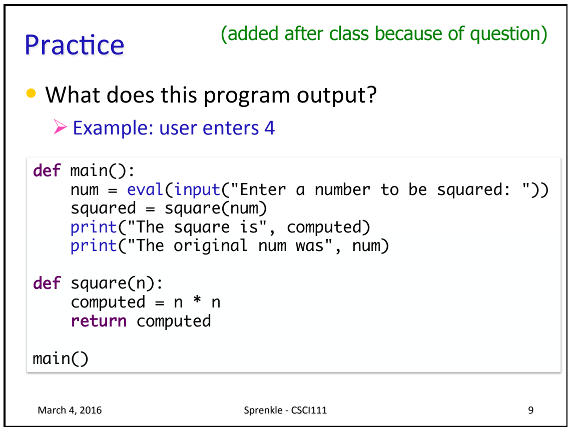
Error! n does not have a value in function main()



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()
```

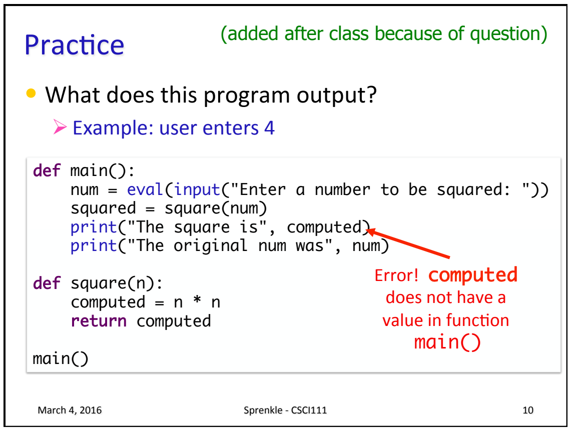


Practice

(added after class because of question)

- 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", computed)
    print("The original num was", num)

def square(n):
    computed = n * n
    return computed

main()
```



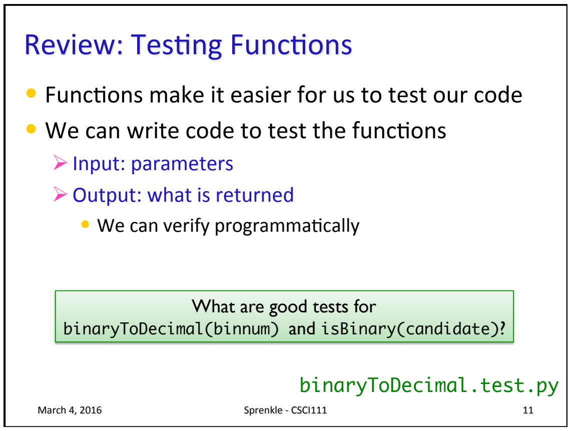
Practice

(added after class because of question)

- 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", computed)
    print("The original num was", num)

def square(n):
    computed = n * n
    return computed

main()
```

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

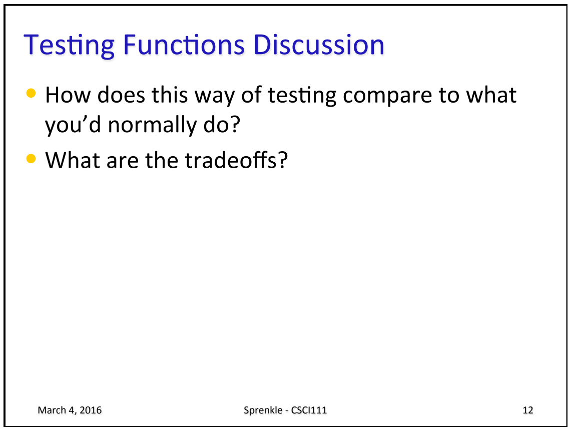


Review: Testing Functions

- Functions make it easier for us to test our code
- We can write code to test the functions
  - Input: parameters
  - Output: what is returned
    - We can verify programmatically

What are good tests for binaryToDecimal(binnum) and isBinary(candidate)?

binaryToDecimal.test.py



Testing Functions Discussion

- How does this way of testing compare to what you'd normally do?
- What are the tradeoffs?

Test Functions

- Designing test function
  - Pick good test cases
  - Automatically (i.e., program) check results so it's easy to spot problems
  - Report input/test cases that cause the problems
- Benefits:
  - Quickly and automatically test functions
  - Quickly add new test cases
  - Can rerun test cases quickly if function implementation changes

WHAT MAKES A GOOD FUNCTION?

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

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

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
```

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>)

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 """
```

REFACTORING

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

Refactoring: Converting Functionality into Functions

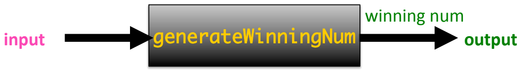
1. Identify functionality that should be put into a function
    - What is the function's input?
    - What is the function's output?
2. Define the function
    - Write comments
3. Call the function where appropriate
4. Create a `main` function that contains the "driver" for your program
    - Put at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

Refactoring Practice

- `pick4num.py`

- Where are places that we can refactor and add functions?

Generate Winning Number

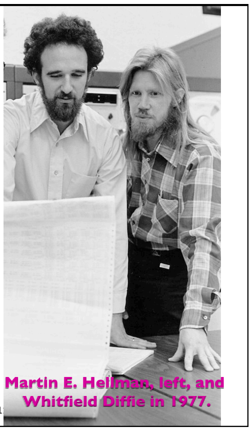


- Input:
  - Options: none; number of digits; range on random numbers
  - Tradeoffs: more general (more parameters), more difficult to use
- Output: winning number

Turing Award Winners

- Turing award = Nobel prize in computer science
- Whitfield Diffie and Martin E. Hellman invented ***public-key cryptography***

- `http://www.nytimes.com/2016/03/02/technology/cryptography-pioneers-to-win-turing-award.html`



Martin E. Hellman, left, and Whitfield Diffie in 1977.

Broader Issue Groups

Allie Andrew Austin J Madhav Stuart	Abdur Corson Gunnar JT Lily	Alice Collin Eric Josie Rachel B	Chris Ethiopia George Max Taylor	Alicia Bennett Honor Viktor
Austin F Margaret Michael Rachel R	Clark Emily Holly Ryan			

Broader Issue Discussion

- What is the difference between privacy and security?
- Whose side are you on?
- What are the most compelling arguments for each side?
  - How do you feel about Apple's argument that their product's appeal is about privacy?
  - Is this case much different from tapping a phone?
- Why do the NSA and other federal agencies hire a lot of W&L students?