Objectives

- Refining our development process
- Passing parameters

Review

- What is the syntax for creating a function?
- What is the special keyword that means "this is the output for the function"?
    - Why do we typically not just print the result within a function?
- What does a variable's "*scope*" mean?
- With respect to functions, what are options for how we organize our program?
- How can we test functions easily?
    - What do we need to test functions?
- What are benefits of functions?

Practice: Trace through the Program's Execution

- 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()
```

practice1.py

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)
    print("The original num was", n)

def square(n):
    return n * n

main()
```

practice2.py

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)
    print("The original num was", n)

def square(n):
    return n * n

main()
```

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

Feb 12, 2021 Sprenkle - CSCI111 5

5

Review: Variable Scope

- Know "lifetime" of variable
  - Only during execution of function
  - Related to idea of "scope"
- Consider: how many functions probably use a variable like x or i? What would the impact be on our programs if all variables had global scope?
  - Example: round(x, n)
- In general, our only *global* variables will be constants because we don't want them to change value
  - e.g., EIEIO

Feb 12, 2021 Sprenkle - CSCI111 6

6

Review: Testing sumEvens

```
import test
…
def testSumEvens():
    actual = sumEvens( 10 )
    expected = 20
    test.testEqual( actual, expected )

def sumEvens(limit):
    total = 0
    for x in range(0, limit, 2):
        total += x
    return total
```

This is the actual result from our function

This is what we expect the result to be

testSumEvens.py

7

WHAT ARE CHARACTERISTICS OF A GOOD FUNCTION?

8

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

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

Comment style: **Docstring**
"documentation string"

```
def printVerse(animal, sound):
    """
    Prints a verse of Old MacDonald using the given
    animal and sound
    animal: a str representing the kind of animal
    sound: a str representing the sound the animal
    """
    print(BEGIN_END + EIEIO)
    print("And on that farm he had a " + animal + EIEIO)
    …
```

Comments from docstrings show up when you use help function

Write the Docstring Comment for sumEvens

```
def main() :
  x=10
  sum = sumEvens( x )
  print("The sum of even #s up to", x, "is", sum)

def sumEvens(limit) :
  """



  """
  total = 0
  for x in range(0, limit, 2):
      total += x
  return total

main()
```

13

13

Another development approach

REFACTORING

14

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 12, 2021 Sprenkle - CSCI111 15

15

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 12, 2021 Sprenkle - CSCI111 16

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

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 facedown on Jelly-side of bread
7. Close bread
8. Clean knife
9. Put away materials

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

Example: PB & J as Functions, 10 x

1. Gather materials (bread, PB, J, knives, plate)
2. Open bread
3. Put 2 pie
4. Spread P
5. Spread Je
6. Place PB
7. Close bread
8. Clean knife
9. Put away materials

```
def main():
    prepare()
    for sandwich in range(10):
        makePBJSandwich()
    cleanUpSupplies()
main()
```

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

From Lab

- Write a program that calculates the area of a circle. Get the radius of the circle as input from the user. Use the most precise value of π available to you, i.e., use the constant pi defined in the math module. Select a “reasonable” number of digits for precision in the result you display to the user.
- Refactor:
  - Function that computes area of circle
  - main function

`circleArea.py`

Exam Next 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 Tuesday’s lab 4
  - Practicing what we learned Wed – Mon
- No broader issue next week