

Objectives

- Defining your own functions
 - Variable Scope
 - Documentation
- Broader Issue: ChatGPT

Feb 3, 2023

Sprenkle - CSC1111

1

1

Review

- What are benefits of functions?
- What is the syntax for creating our own functions?
 - How do we indicate that our function requires input?
 - How do we indicate that our function has output?
- What's the difference between output from a function and output from a program?
- How do we call a function we created?
- With respect to functions, what are options for how we organize a program?

Feb 3, 2023

Sprenkle - CSC1111

2

2

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*)

Feb 3, 2023

Sprenkle - CSC1111

3

3

Function Definition Example without Output

```
def moveCircle( circle, newCenter ):
    """
    Moves a Circle object to a new location.
    circle: the Circle to be moved
    newCenter: the center point of where circle
    should be moved
    """
    centerPoint = circle.getCenter()
    diffInX = newCenter.getX() - centerPoint.getX()
    diffInY = newCenter.getY() - centerPoint.getY()
    circle.move(diffInX, diffInY)
```

Feb 3, 2023

Sprenkle - CSC1111

4

4

Function Definition Example with Output

```
def average2(num1, num2):  
    """  
    Parameters: two numbers to be averaged.  
    Returns the average of two numbers  
    """  
    average = (num1 + num2)/2  
    return average
```

Diagram annotations:

- Keyword:** points to `def`
- Function Name:** points to `average2`
- Input Name/Parameter:** points to `num1` and `num2`
- Function header:** points to the entire first line `def average2(num1, num2):`
- Function documentation:** points to the docstring content
- Body (or function definition):** points to the entire function body
- Keyword: How to give output:** points to `return`
- Output:** points to `average`

Feb 3, 2023

Sprenkle - CSC1111

5

5

Review: return vs print

- In general, whenever we want output from a function, we'll use **return**
 - Results in a more flexible, reusable function
 - Let whoever called the function figure out what to display
- Use **print** for
 - Debugging your function (then remove)
 - Otherwise, unintended side effect of calling the function
 - When you have a function that is supposed to display something
 - Sometimes, that is what you want.

Feb 3, 2023

Sprenkle - CSC1111

6

6

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**

Feb 3, 2023

Sprenkle - CSCI111

7

7

Divergence from Text Book: Conventions

Us: main at the top

- See an overview of the code (driver) at the top
- Need to scroll down to see function definitions to understand what the main does
 - Mitigated by having descriptive function names
- Need to call main at the bottom

Book: main at the bottom

- All functions have already been defined before main
- Need to scroll down to the bottom to see the driver/overview of the program
- Need to call main at the bottom


Feb 3, 2023

Sprenkle - CSCI111

8

8

Variable Scope

- Functions can have the same parameter and variable names as other functions
 - Need to look at the variable's **scope** to determine which one you're looking at
 - Use the **stack** to figure out which variable you're using
- Scope levels
 - **Local scope** (also called **function scope**) 
 - Can only be seen within the function
 - **Global scope** (also called **file scope**)
 - Whole program can access
 - More on these later

Feb 3, 2023

Sprenkle - CSC1111

9

9

Tracing through Execution

```
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()
```

When you call `main()`, that means you want to execute this function

Defines functions

Feb 3, 2023

Sprenkle - CSC1111

10

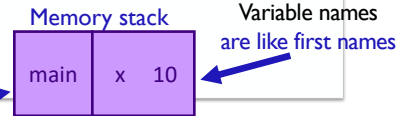
10

Function Variables

```
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()



Function names are like last names
Define the **SCOPE** of the variable

Feb 3, 2023

Sprenkle - CSC1111

11

11

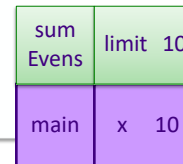
Function Variables

```
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()

Called the function **sumEvens**
Add its parameters to the stack



Feb 3, 2023

Sprenkle - CSC1111

12

12

Function Variables

```
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()
```

sum Evens	total 0 limit 10
main	x 10

Feb 3, 2023

Sprenkle - CSC1111

13

13

Function Variables

```
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()
```

sum Evens	x 0 total 0 limit 10
main	x 10

Feb 3, 2023

Sprenkle - CSC1111

14

14

Function Variables

```
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()
```

sum Evens	x 8 total 20 limit 10
main	x 10

Feb 3, 2023

Sprenkle - CSC1111

15

15

Function Variables

```
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()
```

- Function `sumEvens` returned
- no longer have to keep track of its variables on stack
 - lifetime of those variables is over

main	sum 20 x 10
------	----------------

Feb 3, 2023

Sprenkle - CSC1111

16

16

Function Variables

```
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()
```

main	x 10
	sum 20


Feb 3, 2023

Sprenkle - CSC1111

17

17

Variable Scope

- Functions can have the same parameter and variable names as other functions
 - Need to look at the variable's **scope** to determine which one you're looking at
 - Use the **stack** to figure out which variable you're using
- Scope levels
 - **Local scope** (also called **function scope**) 
 - Can only be seen within the function
 - **Global scope** (also called **file scope**)
 - Whole program can access
 - More on these later

Feb 3, 2023

Sprenkle - CSC1111

18

18

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()
```

Feb 3, 2023

Sprenkle - CSC1111

practice1.py

19

19

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()
```

Enter a number to be squared: 4
The square is 16

Feb 3, 2023

Sprenkle - CSC1111

practice1.py

20

20

Practice

- What is the output of this program?

➤ Example: user enters 4

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

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

main()
```

Feb 3, 2023

Sprenkle - CSC1111

practice2.py

21

21

Practice

- What is the output of this program?

➤ Example: user enters 4

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

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

main()
```

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

Feb 3, 2023

Sprenkle - CSC1111

practice2.py

22

22

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()
```

Feb 3, 2023

Sprenkle - CSC1111

practice3.py

23

23

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 3, 2023

Sprenkle - CSC1111

24

24

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 3, 2023

Sprenkle - CSC1111

25

25

WHAT ARE CHARACTERISTICS OF A GOOD FUNCTION?

Feb 3, 2023

Sprenkle - CSC1111

26

26

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 a descriptive, “action” name
- Should have a comment that tells what the function does

Feb 3, 2023

Sprenkle - CSC1111

27

27

Writing Documentation for Functions

- Good style: Each function* **must** have a comment that documents its use
 - `*main()` usually doesn’t have a doc string because it is covered by the program’s description
- 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)
- The exact format matters less than that the content is there
 - I’ll show a few different ways to write the documentation

Feb 3, 2023

Sprenkle - CSC1111

28

28

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; circle must be a Circle object
- **Postcondition:** Things that will be true when function finishes (if precondition is true)
 - E.g., the returned value is the max; circle will be moved to the new point
- Again, the exact format matters less than the content

Feb 3, 2023

Sprenkle - CSC1111

29

29

Example Documentation

- 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"

When you use the `help` function, it shows the docstrings.

Feb 3, 2023

Sprenkle - CSC1111

30

30

Another Example Comment

- Describes at high-level
- Describes parameters

```
def average2(num1, num2):  
    """  
    Parameters: two numbers to be averaged  
    Returns the average of the two numbers  
    """  
    average = (num1 + num2)/2  
    return average
```

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

When you use the `help` function, it shows the docstrings.

Feb 3, 2023

Sprenkle - CSC1111

31

31

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()
```

Feb 3, 2023

Sprenkle - CSC1111

32

32

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) :
    """
    Returns the sum of even numbers from 0 up to but
    not including limit, which is an integer
    """
    total = 0
    for x in range(0, limit, 2):
        total += x
    return total

main()
```

Many other correct doc strings

Feb 3, 2023

Sprenkle - CSCI111

33

33

Exam Next Friday

- **Do not panic**
- In-class, on paper
 - Emphasis on critical thinking
 - Lab was to experiment and cement you're learning. Now you're ready!
- 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

Feb 3, 2023

Sprenkle - CSCI111

34

34

ChatGPT

- Is ChatGPT for assignments an honor code violation?
 - Will it always be? Compare with, say, using a calculator to do math problems
- Big question: Is all AI biased? Does it have to be?
 - The developers of ChatGPT tried to make sure that it couldn't be racist or spread misinformation
 - What did they do to try to prevent it?
 - What else should they have done?
- With functions, I talk about the benefits of black-box. What are the tradeoffs with a black box? What are the implications for our programming with functions vs AI?

Feb 3, 2023

Sprenkle - CSCI111

35

35

Looking Ahead

- Pre-Lab due before lab on Tuesday
- Exam next Friday

Feb 3, 2023

Sprenkle - CSCI111

36

36