

# CSC 111:

# Intro to Computer Science through Programming

Spring 2017  
Prof. Sara Mathieson



# Admin

- + Homework 7 is due April 4 (tomorrow)
- + **Office hours today 3-5pm** (Ford 355, usually move to 345)
- + Thursday office hours moved to 10am-12pm for the rest of the semester
  
- + **Notecard**: please fill out and return to box (anonymous feedback)
  - 1) Something you understand well
  - 2) Concept that needs most work/review (“muddiest point”)
  - 3) Any other feedback?

# Outline: 4/3

- + Examples from last week
- + Midterm Part 5
- + Begin: recursion (Chap 13.1-13.2)
- + Friday: liberal arts module (maps)

Examples from last week



# While loop for binary (Lab 7)

```
# CSC 111 Lab 7
# Elise Snoey and Karen Santamaria
# Part C!

def binary(integer):

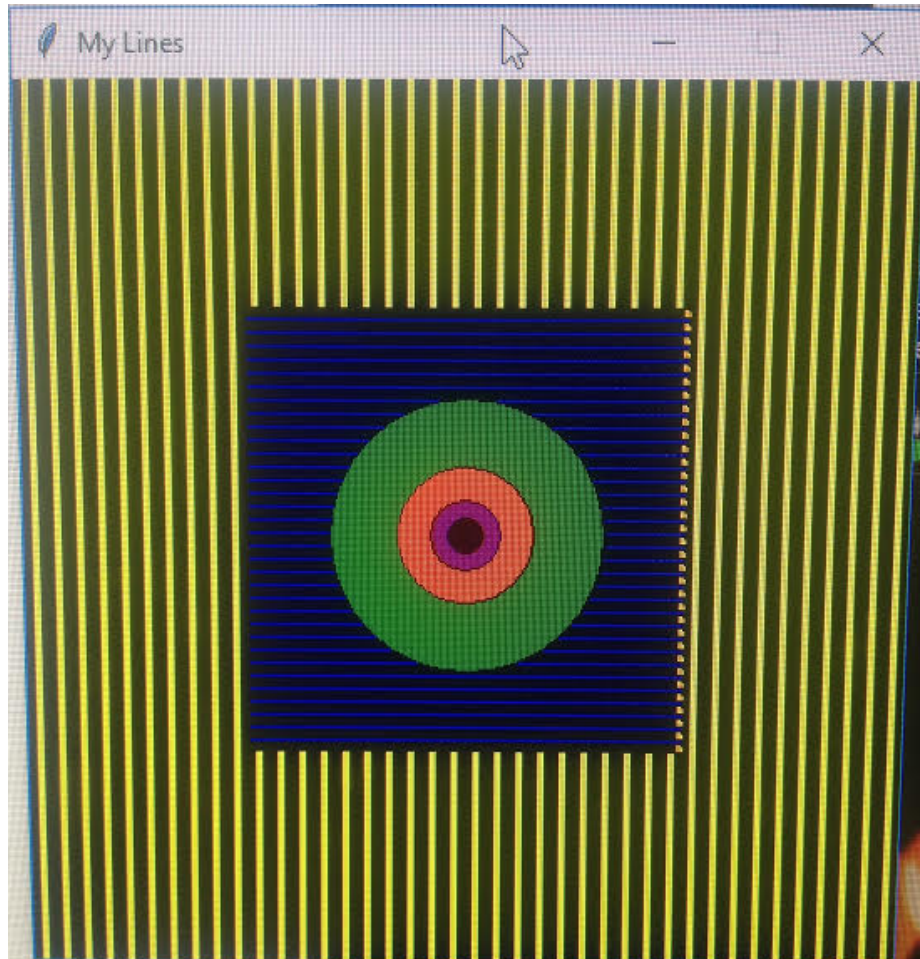
    string = ""
    while integer >= 1:
        remainder = integer % 2
        integer = integer // 2
        string = str(remainder) + string

    return string # changed to return
```

Also implemented by:

- Garcia and Zoraida
- Christa and Megan
- Isabelle and Maggie
- Jess and Yingchuan
- Ruth
- Michelle and Caitlin

# Christa



# Midterm Part 5

# Part 5

In class we have seen how to swap the values of two variables, and in homework we have seen how to swap the values of two elements in a list. In this question, the goal is to swap the values of *three* variables, so that each variable ends up with the value of the “previous” variable. For example, if  $x = 6$ ,  $y = 3$ , and  $z = 1$ , then the end result should be  $x = 1$ ,  $y = 6$ , and  $z = 3$ .

- (a) The following code shows a first attempt at this process. Fill in the table below, showing what will happen *after* each line is executed. The first row has been filled in with the initial values from the example above.

# Part 5

In class we have seen how to swap the values of two variables, and in homework we have seen how to swap the values of two elements in a list. In this question, the goal is to swap the values of *three* variables, so that each variable ends up with the value of the “previous” variable. For example, if  $x = 6$ ,  $y = 3$ , and  $z = 1$ , then the end result should be  $x = 1$ ,  $y = 6$ , and  $z = 3$ .

- (a) The following code shows a first attempt at this process. Fill in the table below, showing what will happen *after* each line is executed. The first row has been filled in with the initial values from the example above.

code	x	y	z	temp1	temp2
	6	3	1	-	-
temp1 = x	6	3	1	6	-
temp2 = y	6	3	1	6	3
x = z	1	3	1	6	3
y = x	1	1	1	6	3
z = y	1	1	1	6	3

# Part 5

- (b) Explain the issue with the code above, and rewrite the code with a few modifications so that it successfully swaps these three variables.

# Part 5

- (b) Explain the issue with the code above, and rewrite the code with a few modifications so that it successfully swaps these three variables.

code	x	y	z	temp1	temp2
temp1 = x	6	3	1	6	-
temp2 = y	6	3	1	6	3
x = z	1	3	1	6	3
{ y = temp1	1	6	1	6	3
{ z = temp2	1	6	3	6	3

# Part 5

- (c) Can you perform this three-variable swap with only *one* temporary variable? If no, explain the issue with such an approach. If yes, provide the code and a table like the one above.



# Part 5

(c) Can you perform this three-variable swap with only *one* temporary variable? If no, explain the issue with such an approach. If yes, provide the code and a table like the one above.

	x	y	z	temp
temp = x	6	3	1	6
x = z	1	3	1	6
z = y	1	3	3	6
y = temp	1	6	3	6

# Part 5

- (d) Would it be possible to write a *function* to swap three variables? i.e. would it be possible to write a function that produces the output below? Explain your answer and reasoning, using the concept of mutable vs. immutable types.

```
>>> x = 6
>>> y = 3
>>> z = 1
>>> three_way_swap(x,y,z)
>>> x
1
>>> y
6
>>> z
3
```

# Part 5

- (d) Would it be possible to write a *function* to swap three variables? i.e. would it be possible to write a function that produces the output below? Explain your answer and reasoning, using the concept of mutable vs. immutable types.

```
>>> x = 6
>>> y = 3
>>> z = 1
>>> three_way_swap(x,y,z)
>>> x
1
>>> y
6
>>> z
3
```

**Not possible! Ints are immutable.**

## Part 5

- (e) Write a function that will return a new string with the  $i^{\text{th}}$  character and  $j^{\text{th}}$  character swapped. Example: `string_swap("spring",1,5)` should return "sgrinp". You may assume that  $i$  is less than  $j$ . Why must this function return a *new* string?

# Part 5

- (e) Write a function that will return a new string with the  $i^{\text{th}}$  character and  $j^{\text{th}}$  character swapped. Example: `string_swap("spring",1,5)` should return "sgrinp". You may assume that  $i$  is less than  $j$ . Why must this function return a *new* string?

```
def string_swap(string, i, j):  
  
    # Note that this will NOT work because strings are not mutable.  
    '''temp = string[i]  
    string[i] = string[j]  
    string[j] = temp'''  
  
    before = string[:i]      # part before the ith character  
    middle = string[i+1:j]  # part between the ith and jth characters  
    after = string[j+1:]    # part after the jth character  
  
    # swap the ith and jth characters  
    new_string = before + string[j] + middle + string[i] + after  
    return new_string # return!
```

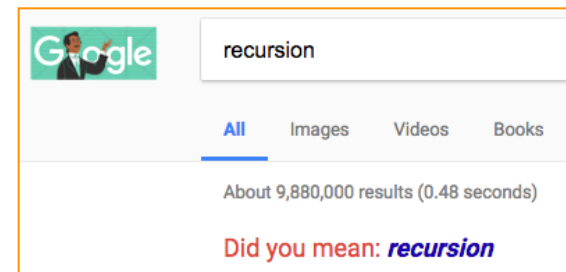
Begin: recursive functions

# What is recursion?

- A recursive function calls **itself**.
- Usually two key components to a recursive function:
  - 1) A **base case** or way to stop the recursion
  - 2) A **recursive call** using a modified argument

# What is recursion?

- A recursive function calls **itself**.
- Usually two key components to a recursive function:
  - 1) A **base case** or way to stop the recursion
  - 2) A **recursive call** using a modified argument



*"To understand recursion, you must first understand recursion"*

Image: "Matryoshka Nesting Dolls: Meaning of Russian Wooden Stacking Doll" - Legomenon



# Factorial function

n	recursion	n!
0	base case	1
1		
2		
3		
4		
5		
6		

$$n! = n*(n-1)*(n-2)...3*2*1$$

or

$$n! = n*(n-1)!$$

with

base case:  $0! = 1$

# Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2		
3		
4		
5		
6		

$$n! = n*(n-1)*(n-2)...3*2*1$$

or

$$n! = n*(n-1)!$$

with

base case:  $0! = 1$

# Factorial function

n	recursion	n!
0	base case	1
1	1*1 ←	1
2	2*1 ←	2
3		
4		
5		
6		

$$n! = n*(n-1)*(n-2)...3*2*1$$

or

$$n! = n*(n-1)!$$

with

base case:  $0! = 1$

# Factorial function

n	recursion	n!
0	base case	1
1	1*1 ←	1
2	2*1 ←	2
3	3*2 ←	6
4		
5		
6		

$$n! = n*(n-1)*(n-2)...3*2*1$$

or

$$n! = n*(n-1)!$$

with

base case:  $0! = 1$

# Factorial function

n	recursion	n!
0	base case	1
1	1*1 ←	1
2	2*1 ←	2
3	3*2 ←	6
4	4*6 ←	24
5		
6		

$$n! = n*(n-1)*(n-2)...3*2*1$$

or

$$n! = n*(n-1)!$$

with

base case:  $0! = 1$

# Factorial function

n	recursion	n!
0	base case	1
1	1*1 ←	1
2	2*1 ←	2
3	3*2 ←	6
4	4*6 ←	24
5	5*24 ←	120
6		

$$n! = n*(n-1)*(n-2)...3*2*1$$

or

$$n! = n*(n-1)!$$

with

base case:  $0! = 1$

# Factorial function

n	recursion	n!
0	base case	1
1	1*1 ←	1
2	2*1 ←	2
3	3*2 ←	6
4	4*6 ←	24
5	5*24 ←	120
6	6*120 ←	720

$$n! = n*(n-1)*(n-2)...3*2*1$$

or

$$n! = n*(n-1)!$$

with

base case:  $0! = 1$