

# CS21: INTRODUCTION TO COMPUTER SCIENCE

---

Prof. Mathieson

Fall 2017

Swarthmore College

# Outline Dec 11:

**RIGHT NOW: continue  
working on fractal trees!**

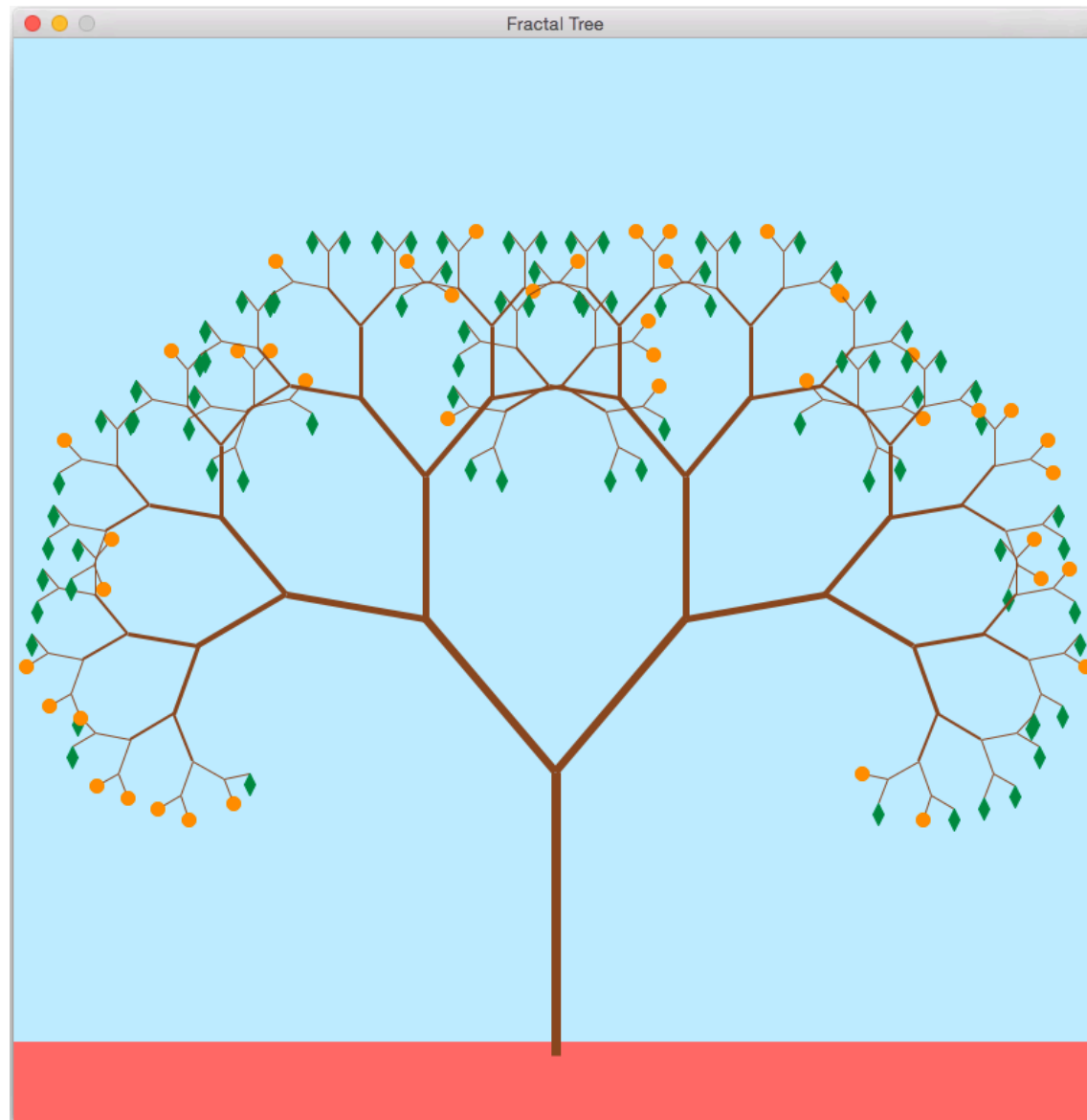
- Finish: fractal trees (recursive graphics)
- Final exam info and Q&A
- Review: stack diagrams, recursion, lists
- Digital Humanities recap
- Extra: list comprehensions
- Survey and feedback about CS 21

## Notes

- Office hours this week: **Wed 2-4pm**
- Let me know if you would like to meet this week
- Final exam: 7-10pm on Friday Dec 15 (Sci Center 101)

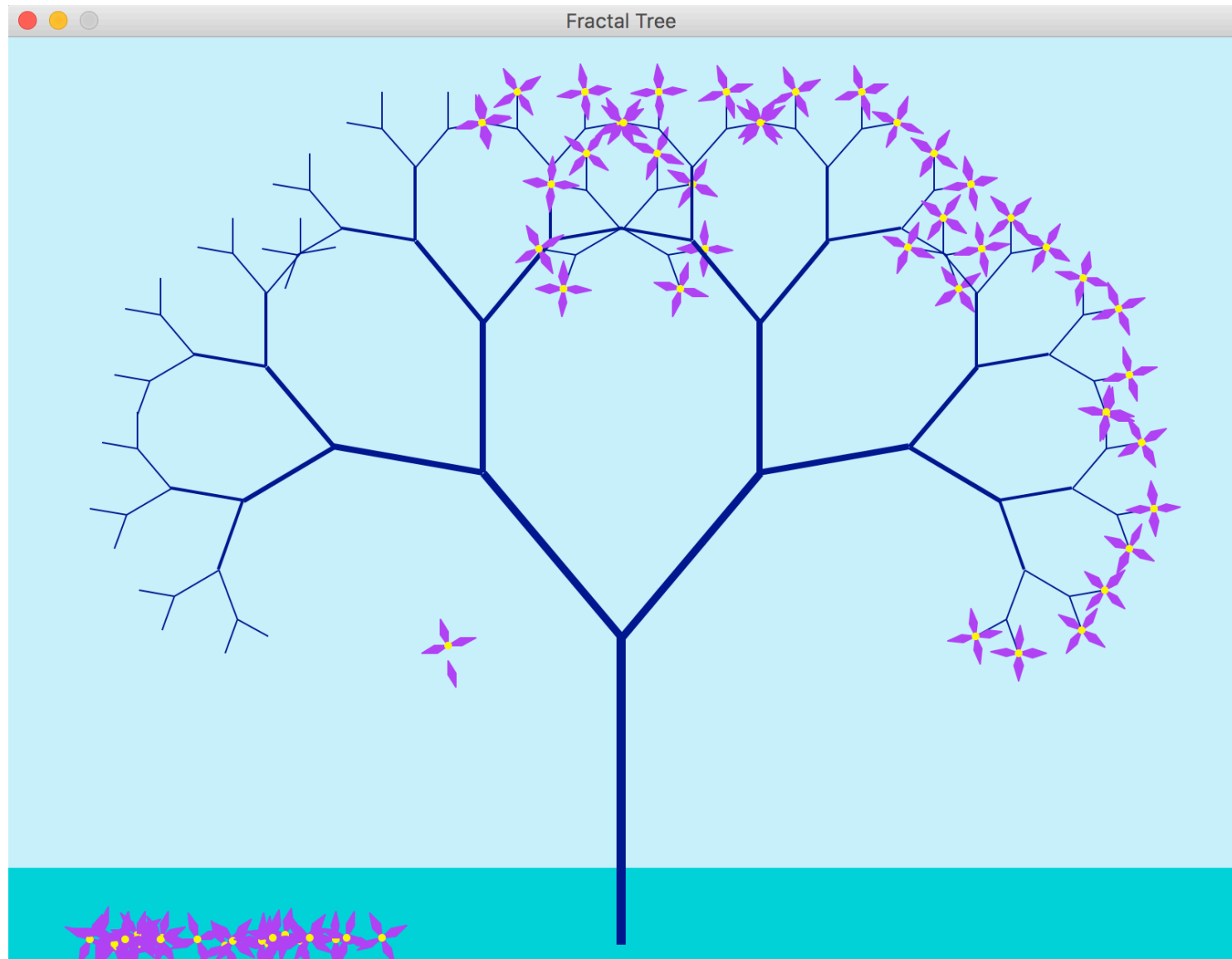
# Recursive Trees

# Example with oranges and leaves at the base case

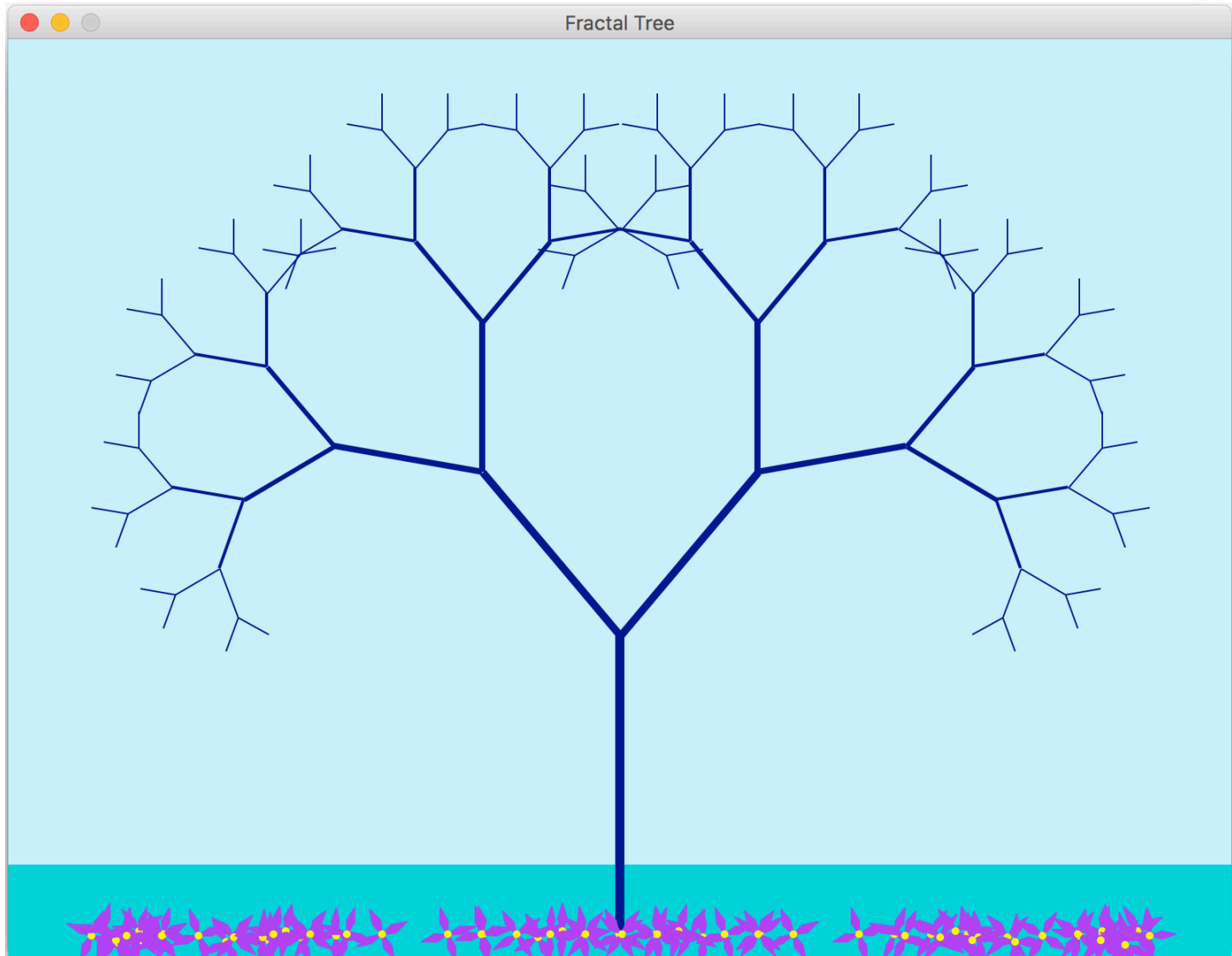


# Example with falling flowers

```
flwr = Flower(x,y)  
flwr.draw(window)
```



# Example with falling flowers



# Fractal Tree code

```
def draw_tree(window, order, x, y, length, angle):
    """Recursive function to draw the fractal tree.
    window: type GraphWin, the window on which to draw the tree
    order: type int, the level of recursion (starts high and decreases to 0)
    x,y: type int, the position of the base of this branch
    length: the length of this branch
    angle: the angle of the direction of this branch
    """

    # compute the coordinates of end of the current branch
    delta_x = length * math.cos(angle)
    delta_y = length * math.sin(angle)
    x_end = x + delta_x
    y_end = y + delta_y

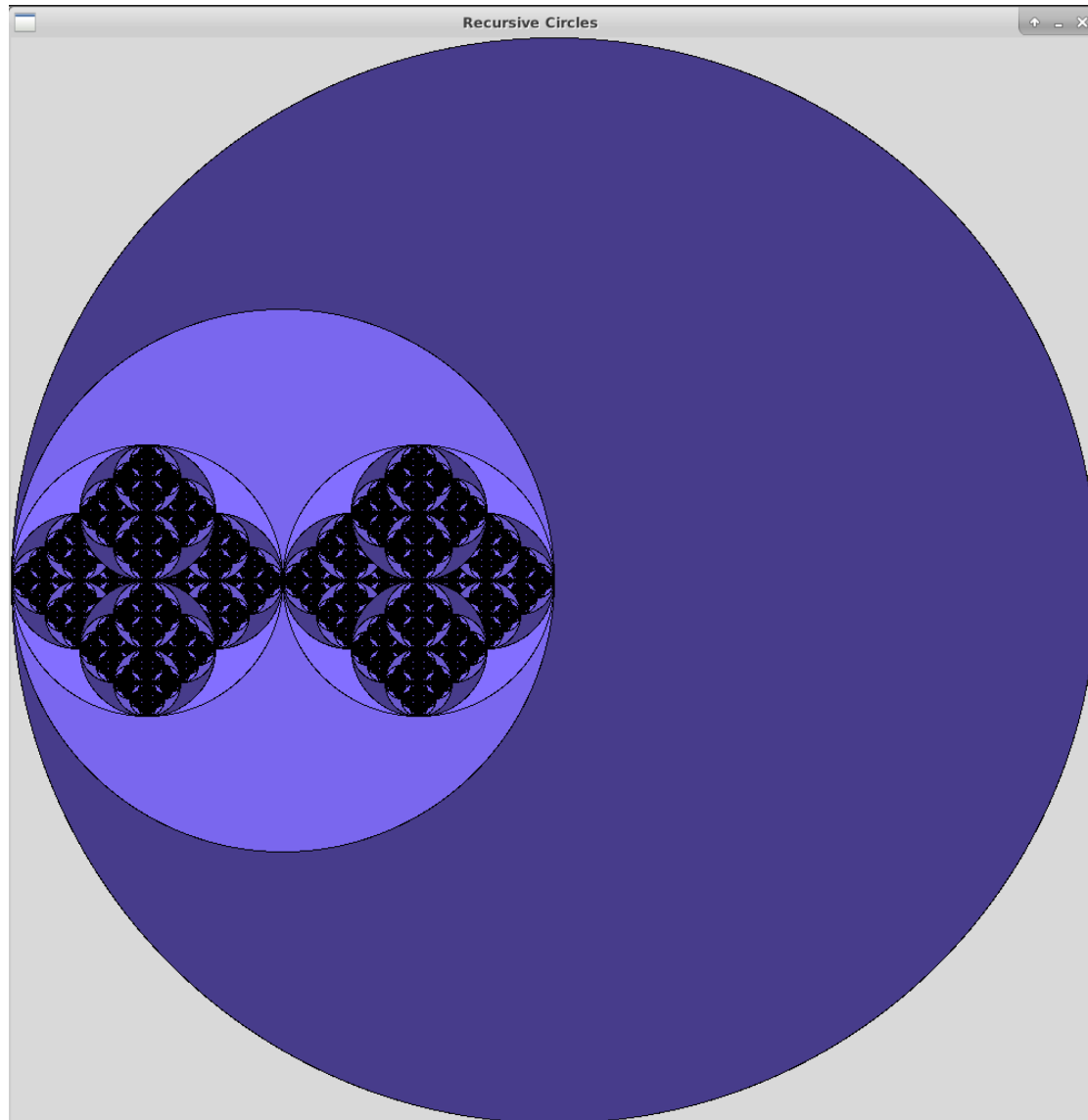
    # draw current branch on the window
    start = Point(x,y)
    end = Point(x_end, y_end)
    branch = Line(start, end)
    branch.draw(window)

    # if we are at the base case (order = 0), draw an orange or leaf
    if order == 0:
        leaf = Leaf(x_end, y_end)
        leaf.draw(window)

    # if we are not at the base case, make two recursive calls
    else:
        # think carefully about each modified argument
        draw_tree(window, order-1, x_end, y_end, length*0.7, angle+theta)
        draw_tree(window, order-1, x_end, y_end, length*0.7, angle-theta)
```

# Screensaver examples

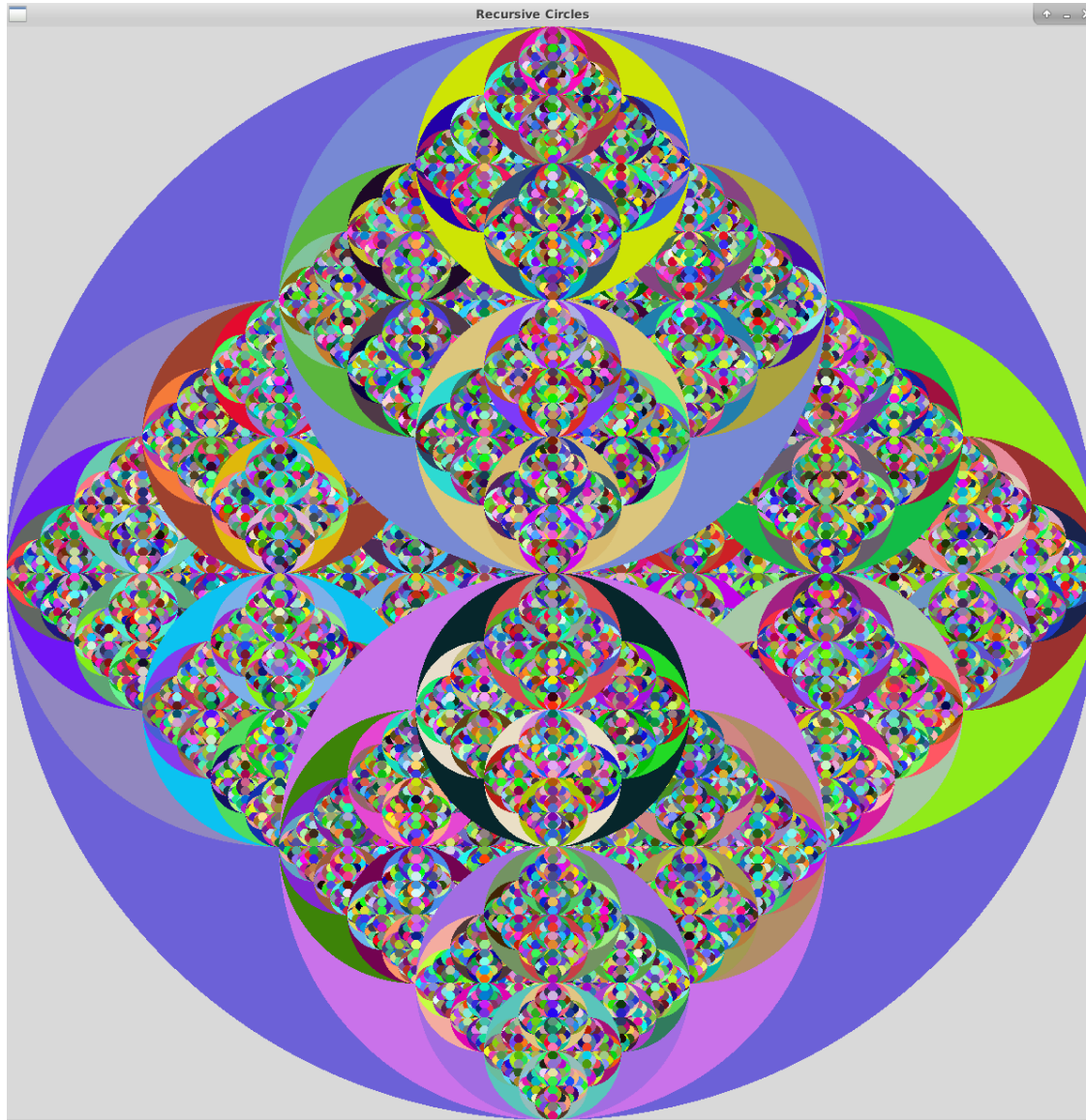
Matt





# Screensaver examples

Anar



# Final Exam

# Email me:

- If you would like a grade estimate before the final
- If you have any cool graphics recursion screenshots
- If there are any practice problem solutions you would like to see
- If you would like to meet before the final

# Studying for the final

- Exam time: **Friday Dec 15: 7-10pm**, Science Center 101
- **Study guide and review problems** are posted
- Go through all notes, code, and practice problems
- Write out as many problems as you can **on paper** (then check in atom)
- Go back over old quizzes and their study guides
- Create a **“cheat-sheet”** for yourself of important concepts and examples (even though you can't use it)
- Come to office hours on **Wed! 2-4pm**

# Questions about the final?

1) How does the length compare to quizzes?

Think long quiz (~4-5x as long), if you just wrote down answers maybe 1 hour

# Questions about the final?

1) How does the length compare to quizzes?

Think long quiz (~4-5x as long), if you just wrote down answers maybe 1 hour

2) How does the difficulty compare to quizzes?

Problems will be larger/more involved than quizzes, but not intended to be more difficult. Caveat: problems will integrate more concepts together

# Questions about the final?

1) How does the length compare to quizzes?

Think long quiz (~4-5x as long), if you just wrote down answers maybe 1 hour

2) How does the difficulty compare to quizzes?

Problems will be larger/more involved than quizzes, but not intended to be more difficult. Caveat: problems will integrate more concepts together

3) Are the study guides enough?

Study guides are necessary, but maybe not sufficient – I would recommend going over problems from class (redo on paper, then check), + extensions we didn't get to

# Questions about the final?

1) How does the length compare to quizzes?

Think long quiz (~4-5x as long), if you just wrote down answers maybe 1 hour

2) How does the difficulty compare to quizzes?

Problems will be larger/more involved than quizzes, but not intended to be more difficult. Caveat: problems will integrate more concepts together

3) Are the study guides enough?

Study guides are necessary, but maybe not sufficient – I would recommend going over problems from class (redo on paper, then check), + extensions we didn't get to

4) How does the format compare to quizzes?

Very similar to quizzes (some analysis of code, types, running algorithms through examples, some writing your own code)



Review: stack, recursion, lists

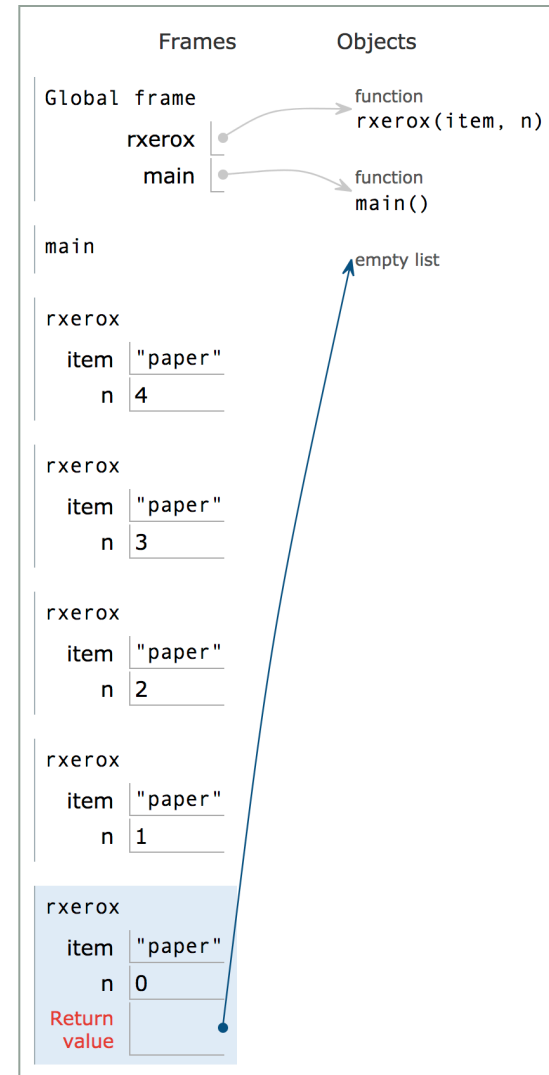
# Example: xerox function from Lab 10

Python 3.6

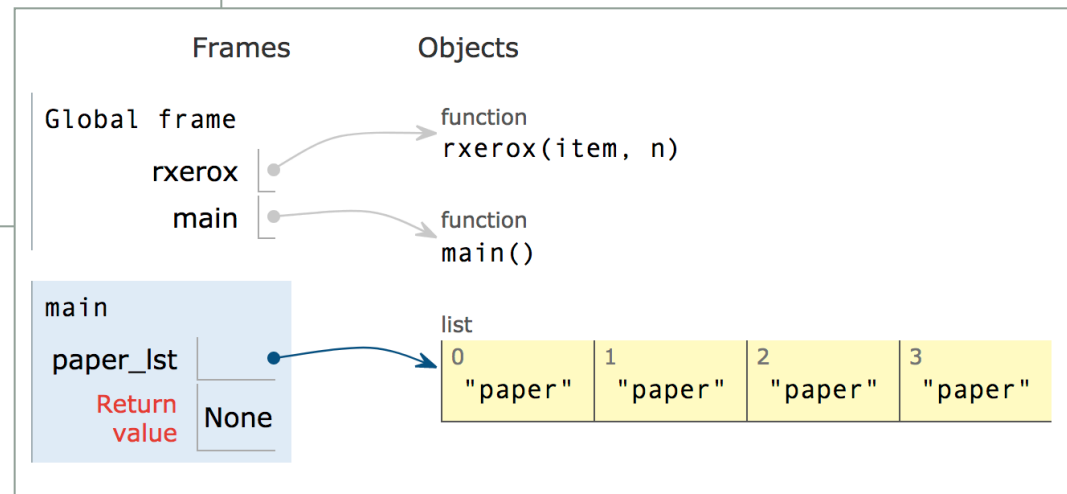
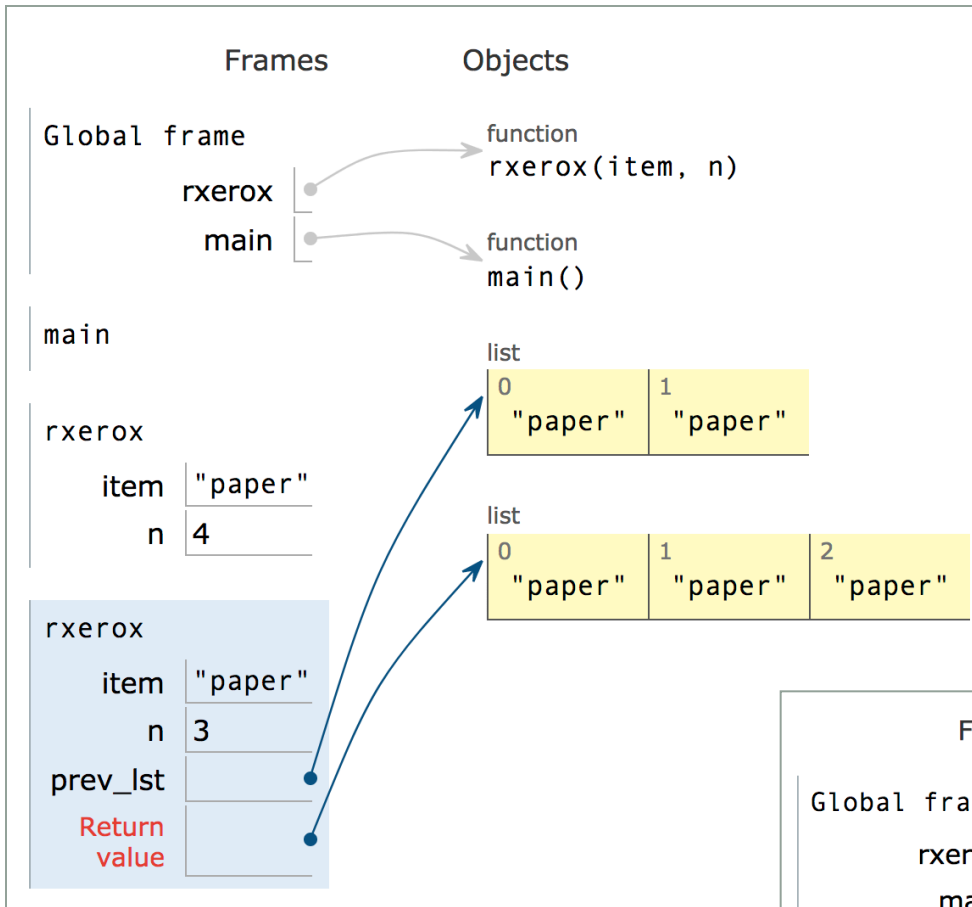
```
1 def rxerox(item,n):  
2     if n == 0:  
3         return []  
4     prev_lst = rxerox(item,n-1)  
5     return [item] + prev_lst  
6  
7 def main():  
8     paper_lst = rxerox("paper", 4)  
9  
10 main()
```

## Notes on PythonTutor:

- 1) Stack should be going up not down
- 2) Values should be on the heap!



# Example: xerox function from Lab 10



# Digital Humanities

# Lab 8 (digital humanities) observations

## **Claudia**

I learned that in most novels the word 'love' occurs more often than the word 'hate.'

## **Skylar**

...Mary Shelley really liked the word "abhorred"... (:

## **Talia**

It was interesting to see how many words with negative connotations were repeated in Frankenstein-- "repugnance" was used three times, for example, and "violence" eight. "Dead" occurs more than twice as many times as "alive."

## **Bayliss**

Mary Shelley LOVES the words "misery" and "sad."

## **Kyle**

"Love" is used often in Mary Shelley's Frankenstein.

# From Prof. Buurma's class

- Used the code to observe patterns in novels they didn't have time to read
- Interesting comparison between **AUSTEN.txt** (all Jane Austen) and **CHAWTON.txt** (~75 novels from other women writers contemporary to Austen)

# AUSTEN.txt vs. CHAWTON.txt

The 10 most prevalent words in ../texts/AUSTEN.txt  
relative to ../texts/CHAWTON.txt

| Score | Word       |
|-------|------------|
| ----- | -----      |
| 144.8 | anyone     |
| 133.9 | anywhere   |
| 122.2 | everybody  |
| 94.1  | talker     |
| 72.4  | dockyard   |
| 65.2  | plaister   |
| 65.2  | despatch   |
| 57.9  | pheasants  |
| 57.9  | anybody    |
| 53.5  | everything |

The 10 most prevalent words in ../texts/CHAWTON.txt  
relative to ../texts/AUSTEN.txt

| Score | Word     |
|-------|----------|
| ----- | -----    |
| 122.9 | countess |
| 94.8  | earl     |
| 83.8  | lordship |
| 68.6  | thou     |
| 42.4  | victim   |
| 42.1  | fled     |
| 39.5  | palace   |
| 39.2  | sword    |
| 38.7  | prisoner |
| 36.5  | n        |

Extra: list comprehensions



# List comprehensions: way to apply a function to every element of a list and get another list back

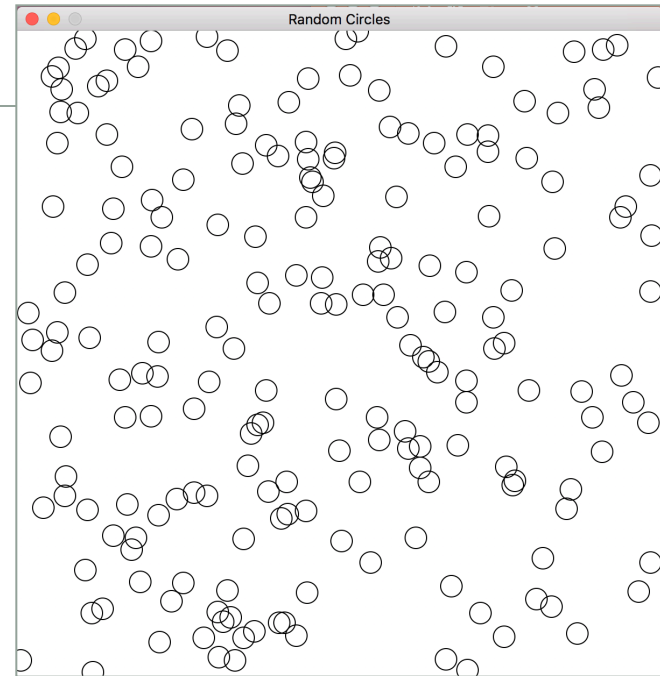
```
>>> lst = list("abcdefg")
>>> lst
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>
>>> triple = [s*3 for s in lst] # do something to every element of a list, creating a new list
>>> triple
['aaa', 'bbb', 'ccc', 'ddd', 'eee', 'fff', 'ggg']
```

# List comprehensions: way to apply a function to every element of a list and get another list back

```
>>> lst = list("abcdefg")
>>> lst
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>
>>> triple = [s*3 for s in lst] # do something to every element of a list, creating a new list
>>> triple
['aaa', 'bbb', 'ccc', 'ddd', 'eee', 'fff', 'ggg']
```

```
circle_lst = [Circle(Point(randrange(0,width), randrange(0,height)), 10) for i in range(200)]

for c in circle_lst:
    c.draw(win)
```



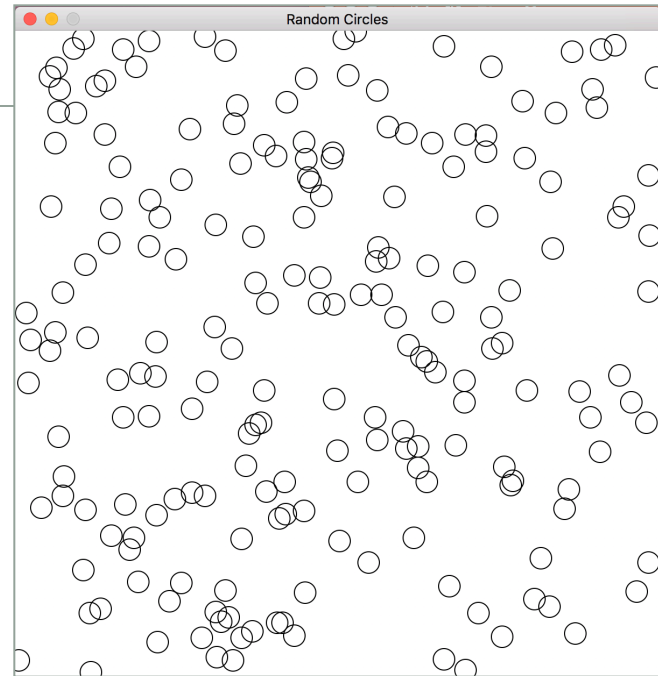
# List comprehensions: way to apply a function to every element of a list and get another list back

```
>>> lst = list("abcdefg")
>>> lst
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>
>>> triple = [s*3 for s in lst] # do something to every element of a list, creating a new list
>>> triple
['aaa', 'bbb', 'ccc', 'ddd', 'eee', 'fff', 'ggg']
```

```
circle_lst = [Circle(Point(randrange(0,width), randrange(0,height)), 10) for i in range(200)]

for c in circle_lst:
    c.draw(win)
```

Don't actually ever do this!



# Preparing for CS 31 or CS 35

- CS 31 (language: C)

<https://www.cprogramming.com/tutorial/c-tutorial.html>

- CS 35 (language: C++)

<http://www.cplusplus.com/doc/tutorial/>

# Survey and 21 Feedback

- Email with survey link from Lauri
- Your feedback will help make 21 better for students in the future
- Please take some time to complete the survey thoughtfully