CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson
Fall 2018
Swarthmore College
Outline Dec 10:

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

Notes

- Office hours this week: Thursday 2-4pm
- Final exam: 2-5pm on Saturday 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
```python
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)
```
Fractal Tree example: Bess
Fractal Tree example: Beluchi
Final Exam
Studying for the final

- Exam time: **Saturday Dec 15: 2-5pm**, 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 **Thurs! 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: classes, stack, recursion
class Student:
    def __init__(self, name):
        self.name = name
        self.courses = []

    def num_courses(self):
        return len(self.courses)

    def is_taking(self, course):
        n = self.num_courses()
        for i in range(n):
            if course == self.courses[i]:
                return True
        return False

    def __str__(self):
        return self.name

ayaka = Student(...)
d = ayaka.num_courses()
print(ayaka)

2 instance variables
2 methods
Recursion practice problems

- hello.py
- all_even.py
- count_even.py
- counter.py
- strlen.py
- increment.py
- cross.py
- tower.py

in cs21/practice/ directory
Extra: list comprehensions
List comprehensions: way to apply a function to every element of a list and get another list back

```python
>>> 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

```python
>>> lst = list("abcdefg")
>>> lst
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

```python
>>> 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)]

```python
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

```python
>>> 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