Homework 8 is due April 11 (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
Outline: 4/10

- Examples from Lab 6 and last week
- Review Recursion
- Classes (all this week and next)
Lab 6 Examples
(selected by me)
Part A: Jess and Yingchuan
Part A: Caitlin and Katherine
Part A: Ryan and Zhu
Part A: Sakaiza and Mai
Part A: Ann and Rachel Carr
Part A: Kirthna and Shayla
Part A: Stella and Allison
Part B: Erika and Chujun
Part B: Shehrbano and Jocelyn Yax
Extensions: Hening
Tree Examples
Stefany and Jessica Keast
and who said money can't grow on trees?
Recap Recursion
Informal quiz: discuss with a partner

1) The goal in this question is to write a recursive function that will determine whether or not a string is a palindrome. What should the return type be?

2) A student is writing this function and tries out the base case shown below. How would you fix it to agree with (1)?

```python
def palindrome(string):
    if len(string) == 0:
        return 0
```

3) Instead of checking whether the string is equal to its reverse, in the recursive case, check whether the first and last characters are equal. If they are not equal, what should be returned? If they are equal, what should happen next?

4) Put (2) and (3) together and write this function in a new file.
Informal quiz: discuss with a partner

1) The goal in this question is to write a recursive function that will determine whether or not a string is a palindrome. What should the return type be?

   boolean (True or False)

2) A student is writing this function and tries out the base case shown below. How would you fix it to agree with (1)?

   ```python
def palindrome(string):
    if len(string) == 0:
        return 0
   ```

3) Instead of checking whether the string is equal to its reverse, in the recursive case, check whether the first and last characters are equal. If they are not equal, what should be returned? If they are equal, what should happen next?

4) Put (2) and (3) together and write this function in a new file.
Informal quiz: discuss with a partner

1) The goal in this question is to write a recursive function that will determine whether or not a string is a palindrome. What should the return type be?
   
   \textbf{boolean (True or False)}

2) A student is writing this function and tries out the base case shown below. How would you fix it to agree with (1)?

   ```python
   def palindrome(string):
       if len(string) == 0:
           return 0
   ```

   ```python
   def palindrome(string):
       if len(string) <= 1:
           return True
   ```

3) Instead of checking whether the string is equal to its reverse, in the recursive case, check whether the first and last characters are equal. If they are not equal, what should be returned? If they are equal, what should happen next?

4) Put (2) and (3) together and write this function in a new file.
Informal quiz: discuss with a partner

1) The goal in this question is to write a recursive function that will determine whether or not a string is a palindrome. What should the return type be?

   boolean (True or False)

2) A student is writing this function and tries out the base case shown below. How would you fix it to agree with (1)?

   ```python
   def palindrome(string):
       if len(string) == 0:
           return 0
   ```

   ```python
   def palindrome(string):
       if len(string) <= 1:
           return True
   ```

3) Instead of checking whether the string is equal to its reverse, in the recursive case, check whether the first and last characters are equal. If they are not equal, what should be returned? If they are equal, what should happen next?

4) Put (2) and (3) together and write this function in a new file.

Live coding
Begin (writing) Classes
What we’ve done with classes so far...

- We have already seen existing classes (Point, Circle, Polygon, etc)
What we’ve done with classes so far...

- We have already seen existing classes (Point, Circle, Polygon, etc).
- Classes allow us to encapsulate common structures and functions so we don’t have to define them over and over again.
What we’ve done with classes so far...

- We have already seen existing classes (Point, Circle, Polygon, etc)
- Classes allow us to encapsulate common structures and functions so we don’t have to define them over and over again
- We can create a new *instance* of a class using the *constructor*

```python
dot = Circle(Point(x,y),r)
```
What we’ve done with classes so far...

- We have already seen existing classes (Point, Circle, Polygon, etc)
- Classes allow us to encapsulate common structures and functions so we don’t have to define them over and over again
- We can create a new *instance* of a class using the *constructor*
  
  \[ \text{dot} = \text{Circle(Point}(x,y),r) \]
- We can access *instance variables* of the class directly or using *methods*
  
  \[
  \begin{align*}
  r &= \text{dot.getRadius()} \\
  r &= \text{dot.radius}
  \end{align*}
  \]
What we’ve done with classes so far…

- We have already seen existing classes (Point, Circle, Polygon, etc)
- Classes allow us to encapsulate common structures and functions so we don’t have to define them over and over again
- We can create a new *instance* of a class using the *constructor*
  
  ```
  dot = Circle(Point(x,y),r)
  ```
- We can access *instance variables* of the class directly or using *methods*
  
  ```
  r = dot.getRadius()
  r = dot.radius
  ```
- We can use/modify class instances using *methods*
  
  ```
  dot.move(dx,dy)
  ```
Goal: write a class called MSDie, which will construct a multi-sided die.

What information should be contained in this class? i.e. what should a die “know” about itself?
Goal: write a class called MSDie, which will construct a multi-sided die.

What information should be contained in this class? i.e. what should a die “know” about itself?

The number of sides it has, current value (top)

Nasco: Foam Polyhedral Dice
Goal: write a class called MSDie, which will construct a multi-sided die.

What information should be contained in this class? i.e. what should a die “know” about itself?

The number of sides it has, current value (top)

What actions should we be able to perform with this die?

Nasco: Foam Polyhedral Dice
Goal: write a class called MSDie, which will construct a multi-sided die.

What information should be contained in this class? i.e. what should a die “know” about itself?

The number of sides it has, current value (top)

What actions should we be able to perform with this die?

Roll the die, check the current value, set value
Dice example

I should be able to do something like this in the shell:

```python
>>> die1 = MSDie(6)
>>> die1.roll()
>>> die1.get_value()
2
>>> die1.set_value(5)
>>> die1.get_value()
5
>>> die2 = MSDie(8)
>>> die2.roll()
>>> die2.get_value()
3
>>> die2.roll()
>>> die2.get_value()
2
```
import random

class MSDie:
    def __init__(self, num_sides):
        self.sides = num_sides
        self.value = 1

    def roll(self):
        self.value = random.randint(1, self.sides)

    def get_value(self):
        return self.value

    def set_value(self, new_value):
        self.value = new_value
Dice example

```python
import random

class MSDie:
    def __init__(self, num_sides):
        self.sides = num_sides
        self.value = 1

    def roll(self):
        self.value = random.randint(1, self.sides)

    def get_value(self):
        return self.value

    def set_value(self, new_value):
        self.value = new_value
```

Constructor always defined using `__init__`. All instance variables assigned.
Dice example

```python
import random

class MSDie:
    def __init__(self, num_sides):
        self.sides = num_sides
        self.value = 1

    def roll(self):
        self.value = random.randint(1, self.sides)

    def get_value(self):
        return self.value

    def set_value(self, new_value):
        self.value = new_value
```

Three methods, all involve modifying or accessing instance variables.
import random

class MSDie:
    def __init__(self, num_sides):
        self.sides = num_sides
        self.value = 1

    def roll(self):
        self.value = random.randint(1, self.sides)

    def get_value(self):
        return self.value

    def set_value(self, new_value):
        self.value = new_value

All methods and constructor must have “self” as a first parameter.
Dice example

```python
import random

class MSDie:
  
def __init__(self, num_sides):
    self.sides = num_sides
    self.value = 1
  
def roll(self):
    self.value = random.randint(1, self.sides)
  
def get_value(self):
    return self.value
  
def set_value(self, new_value):
    self.value = new_value
```
Whenever we access an instance variable, we must use “self.”