CIS 122

A Class of One's Own
Suppose I was writing a grading program

I might want a student class
  ○ Keep track of students scores
  ○ Calculate grades

What properties should a student have?
Representing a Student

- Student Class

- Properties
  - Name
  - Grades

- Methods
  - Add grade
  - Calculate average grade
  - Get letter grade
Let's start at the beginning

Define a student class
  - With a student constructor

What information do we need to make a student?

What information do we want our student to store?
Representing a Student

class Student:

    def __init__(self, studentName):
        self.name = studentName
        self.grades = []
Representing a Student

- Now let's print out our student
  - What should a student look like?

```python
def __repr__(self):
    return self.name
```
Representing a Student

- Now we can make students and display students
- Let's add some functionality
  - addGrade
  - averageGrade
  - letterGrade
class Student:

    def __init__(self, studentName):
        self.name = studentName
        self.grades = []

    def __repr__(self):
        return self.name

    def addGrade(self, grade):
        self.grades.append(grade)
Let's add an averageGrade function
   ○ Reads through student's list of grades
   ○ Returns average grade

```python
def averageGrade(self):
```
Let's add an averageGrade function

- Reads through student's list of grades
- Returns average grade

```python
def averageGrade(self):
    count = 0.0
    total = 0.0
    for grade in self.grades:
        count += 1
        total += grade
    return total / count
```
Let's add a `letterGrade` function

- Determines letter grade based on average grade

```python
def letterGrade(self):
```
• Let's add a letterGrade function
  ○ Determines letter grade based on average grade

```python
def letterGrade(self):
    average = self.averageGrade()
    if average > 90:
        return 'A'
    elif average > 80:
        return 'B'
    elif average > 70:
        return 'C'
    else:
        return 'D'
```
What's so special about classes?

- Why are classes useful?

- Our student objects are just collections of smaller objects
  - String
  - List of floats

- Could have just used lists instead
  - `s1 = [ 'Alice', [ 90, 80, 70 ] ]`
  - `s2 = [ 'Bob', [ 60, 70, 75 ] ]`

- Could write functions designed for this representation
  
  ```python
  def displayStudent(student):
      print student[0]
  ```
What's so special about classes?

- Classes don't make our code any more powerful
  ○ Unlike conditionals, recursion, iteration, ...

- Anything we can represent as a class...
  ○ We could also represent as a list

- Methods are just fancy functions

- So what's the point?
What's so special about classes

- Classes make code more clear

- Suppose we want to print out a student

- If we store student as a fancy list...
  ```python
def displayStudent(student):
  print student[0]
  ```

- If we store student as a class (with named properties)
  ```python
def __repr__(self):
  print student.name
  ```
What's so special about classes

- Classes abstract away implementation
- Outsiders don't need to worry about how a class is written
- If I want a student's grade, I call student.letterGrade()
  - Don't care what data is stored
  - Don't care what computation is involved
- Similar to calling turtle functions
  - What really happens when you call turtle.forward(10)?
  - It doesn't matter to us
  - We just see the end result
What's so special about classes

- Classes package similar code together
- All Student methods are located in my Student class
  - No choice involved
- Other class methods are located in their respective classes
- Keeps code organized
  - Easy to find things
  - Easy to connect things
- Similar motivation for modules