+ Homework 9 is due April 18 (last homework, start early!)
+ Final project is due May 2
+ Remaining graded labs: Lab 10, Lab 11
+ Labs on last two days of classes: practice final
+ Last day to pre-register for CSC 212!
Outline: 4/14

- Recap classes so far

- Review and practice: dictionaries, while loops, and files

- Next week: continue classes with a focus on biology and physics applications
Recap Classes
Informal quiz: discuss with a partner

1) How many instance variables are contained within this class?

2) Complete the `add_class(..)` method.

3) Write a getter for the first name.

4) Do constructors return something? Why or why not?

class Student:

```python
def __init__(self, name):
    name_lst = name.split()
    self.first = name_lst[0]
    self.last = name_lst[1:]
    self.class_lst = []

def add_class(self, new_class):

def drop_class(self, old_class):
    self.class_lst.remove(old_class)
```
Informal quiz: discuss with a partner

1) How many instance variables are contained within this class? 3

2) Complete the `add_class(..)` method.

3) Write a getter for the first name.

4) Do constructors return something? Why or why not?

```python
class Student:
    def __init__(self, name):
        name_lst = name.split()
        self.first = name_lst[0]
        self.last = name_lst[1:]
        self.class_lst = []

    def add_class(self, new_class):

    def drop_class(self, old_class):
        self.class_lst.remove(old_class)
```
Informal quiz: discuss with a partner

1) How many instance variables are contained within this class? 3

2) Complete the `add_class(..)` method.

3) Write a getter for the first name.

4) Do constructors return something? Why or why not?

```python
class Student:
    def __init__(self, name):
        name_lst = name.split()
        self.first = name_lst[0]
        self.last = name_lst[1:]
        self.class_lst = []

    def add_class(self, new_class):
        self.class_lst.append(new_class)

    def drop_class(self, old_class):
        self.class_lst.remove(old_class)
```
Informal quiz: discuss with a partner

1) How many instance variables are contained within this class? 3

2) Complete the `add_class(..)` method.

3) Write a getter for the first name.

4) Do constructors return something? Why or why not?

```python
class Student:
    def __init__(self, name):
        name_lst = name.split()
        self.first = name_lst[0]
        self.last = name_lst[1:]
        self.class_lst = []

    def add_class(self, new_class):
        self.class_lst.append(new_class)

    def drop_class(self, old_class):
        self.class_lst.remove(old_class)

    def get_first(self):
        return self.first
```
Informal quiz: discuss with a partner

1) How many instance variables are contained within this class? 3

2) Complete the add_class(..) method.

3) Write a getter for the first name.

4) Do constructors return something? Why or why not?

```python
class Student:
    def __init__(self, name):
        name_lst = name.split()
        self.first = name_lst[0]
        self.last = name_lst[1:]
        self.class_lst = []

    def add_class(self, new_class):
        self.class_lst.append(new_class)

    def drop_class(self, old_class):
        self.class_lst.remove(old_class)

    def get_first(self):
        return self.first
```

They return an instance of the class that we can assign to a variable, but we do not use the keyword return.
Review dictionaries
Idea behind dictionaries

- When we index into a list, we have to use an integer.
- What if we could make the index any type we wanted?
Idea behind dictionaries

+ When we index into a list, we have to use an integer.
+ What if we could make the index any type we wanted?
+ Dictionaries give us a way to be flexible about the index, without sacrificing speed.
+ New type (like list, str, bool, int, float), now we have dict (mutable)
Idea behind dictionaries

+ When we index into a list, we have to use an integer.
+ What if we could make the index any type we wanted?
+ Dictionaries give us a way to be flexible about the index, without sacrificing speed.
+ New type (like list, str, bool, int, float), now we have dict (mutable)
+ Example: counts of choosing a random color over and over again

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>orange</th>
<th>yellow</th>
<th>green</th>
<th>blue</th>
<th>purple</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>105</td>
<td>98</td>
<td>83</td>
<td>113</td>
<td>101</td>
<td>92</td>
</tr>
</tbody>
</table>
Review while loops
While loop structures

One common while loop structure: assign a numerical variable before the loop, then update its value within the loop. The condition compares the variable to some fixed value.

```python
var = __ # assign a variable
while var < other_value:
    var = __ # update variable
```
While loop structures

+ One common while loop structure: assign a numerical variable before the loop, then update it’s value within the loop. The condition compares the variable to some fixed value.

```python
var = __  # assign a variable
while var < other_value:
    var = __  # update variable
```

+ Another common while loop structure: assign a boolean variable to True before the loop, then flip to False if some condition is met within the loop.

```python
my_bool = True  # assign a variable
while my_bool:
    if <condition>:
        my_bool = False  # update variable
```
Review files