Lab Overview

- Review lab 8
- Prep for lab 9

Lab 8 Feedback

- Define constants for HEADS and TAILS
- Extra step in game module's `flipCoin` function

```
HEADS=0
TAILS=1

def flipCoin():
    flip = random.randint(0,1)
    if flip == HEADS:
        return HEADS
    else:
        return TAILS
```

Equivalent code
(needs comments)

```
HEADS=0
TAILS=1

def flipCoin():
    return random.randint(HEADS,TAILS)
```


Common Issues

- Test multiple of 6 the first time – make sure to exit on first iteration of loop
- Comments on game's functions
  - Describe *interface* → tell others how to use
    - What are parameters? Restrictions?
    - What is returned?
- Demonstrating and testing game's functions
  - Need to show the results—that the code is working
  - Otherwise, need to write test functions.

Multiple Checks with While Loop

- Draw control flow diagram

```
num = eval(input("Enter a number: "))
remainder = num % 6

while remainder != 0:
    second = eval(input("Enter a number: "))
    remainder2 = second % 6
    if remainder == 0 or remainder2 == 0:
        break

print("Done!")
```

Difference btw File *Name* and *Object*

- File name is a `string`
- File object is a `file`
- Need the file **name** to create the file **object**

- Need to remember data types because not explicit in Python
- Use good variable names to help

LAB 9 PREPARATION

Motivating using list's `sort` method with a *key*

- We may not want to sort a list of objects by the "standard" way to sort objects
- Consider sorting strings: How does Python sort strings usually?

Using list's `sort` method with a key

- We may not want to sort a list of objects by the "standard" way to sort objects
- Consider sorting strings: How does Python sort strings usually?
  - Alphabetically, upper-case first
- To alphabetize strings, sorting them by their lowercase value:

```
words.sort(key=str.lower)
```

Method to call to do comparison

`sort_ignore_case.py`

Using list's `sort` method with a key

```
words = ["Washington", "and", "Lee", "computer", "science”]
words.sort()

print("Words in Python str-standard sorted order:”)
for word in words:
    print(word)
print()

print("Words in sorted order, ignoring upper and lower case:")

words.sort(key=str.lower)

for word in words:
    print(word)
```

Method is named as Classname.methodname

sort_ignore_case.py

Using list's `sort` method with a key

```
words = ["Washington", "and", "Lee", "computer", "science"]
words.sort()

print("Words in Python str-standard sorted order:”)
for word in words:
    print(word)
print()

print("Words in sorted order,

words.sort(key=str.lower)

for word in words:
    print(word)
```

```
Words in Python str-standard sorted order:
Lee
Washington
and
computer
science

Words in sorted order, ignoring upper and
lower case:
and
computer
Lee
science
Washington
```

sort_ignore_case.py

Review: Dictionaries

- How do you create a new dictionary?
- How do you find out if there is a mapping for a key in the dictionary? (Two ways)
- How do you access the value for a key?
- How do you add a mapping?
- How can you iterate through a dictionary?

Review the problems we solved using a dictionary

Review: Objects and Classes

- Goal: Package ***data*** and ***functionality*** into one structure
- **Class**: a template for objects that have the same data and functionality
  - Instance variables represent object's data
  - Methods represent object's functionality
- **Objects** are an *instance* of a class
  - Examples: c1 = Card(2, "hearts") and c2 = Card(13, "spades")
    - Each is an instance of the Card class
    - Have the same functionality/methods but different state

Review: Defining our own classes

- Where do we define the data that is needed to represent every object of a class?
  - How do we access that data?
- What are defined methods like?
- Special method name for constructor?
- Special name for method that helps with printing?
- Keyword that must be the first parameter of every defined method?

Review: Defining our own classes

- Where do we define the data that is needed to represent every object of a class?
  - How do we access that data?
  - Answer: In the constructor. Use `self._data` to represent that data. Can access that data in other methods as self._data
- What are defined methods like?
  - Answer: functions
- Special method name for constructor?
  - `__init__`
- Special name for method that helps with printing?
  - `__str__(self)` – returns a string representation of the object
- Keyword that must be the first parameter of every defined method?
  - `self`

Card Class (Incomplete)

Doc String

```
class Card:
    """ A class to represent a standard playing card.
    The ranks are ints: 2-10 for numbered cards, 11=Jack,
12=Queen, 13=King, 14=Ace.
    The suits are strings: 'clubs', 'spades', 'hearts',
'diamonds'."""
    def __init__(self, rank, suit):
        """Constructor for class Card takes int rank and
       string suit."""
        self._rank = rank
        self._suit = suit

    def getRank(self):
        "Returns the card's rank."
        return self._rank

    def getSuit(self):
        "Returns the card's suit."
        return self._suit
```

Methods

Identify the instance variables
- How do we use them in other Card methods?

card.py

Card Class (Incomplete)

Doc String

```
class Card:
    """ A class to represent a standard playing card.
    The ranks are ints: 2-10 for numbered cards, 11=Jack,
12=Queen, 13=King, 14=Ace.
    The suits are strings: 'clubs', 'spades', 'hearts',
'diamonds'."""
    def __init__(self, rank, suit):
        """Constructor for class Card takes int rank and
       string suit."""
        self._rank = rank
        self._suit = suit

    def getRank(self):
        "Returns the card's rank."
        return self._rank

    def getSuit(self):
        "Returns the card's suit."
        return self._suit
```

Methods

Identify the instance variables
- How do we use them in other Card methods?

Convention: instance variables are named beginning with _

card.py

Review: Algorithm for Creating Classes

1. Identify need for a class
2. Identify state or attributes of a class/an object in that class
   - Write the constructor (`__init__`) and `__str__` methods
   - Test those methods
3. Identify methods (i.e., functionality) the class should provide
   - How will a user call those methods (parameters, return values)?
     - Develop API
4. Implement, test one method
   - Repeat until have complete API

Lab 9: Dealing with Real Data

- **Problem**: Determine most common first and last names at W&L
  - 4 data files, containing student names
    - Last names, female first names, male first names, all first names
    - 1 name per line
  - What data structure to use?
- Create your own class to help with data
- Create output file used by another application
  - Common use of programming

Writing To a File

- Review: What data type does file's `write` method take as a parameter?

Writing To a File

- Review: What data type does file's `write` method take as a parameter?
- To write numeric data to a file, you need to convert it to a string
  - Can use `str()` or use string formatting, which makes it easier to print out a line of text

Lab Overview

1. Implement partial solution using a dictionary to map the name to its count
   - handles basic set up of solution, including reading and processing file
2. Implement a class that packages the name (a key) and its count together
   - Data and functionality given
   - Test the class
3. Implement Step 1 with objects of class you created in Step 2
   - Complete solution
4. Graph data generated from Step 3
5. Make web page with graphs

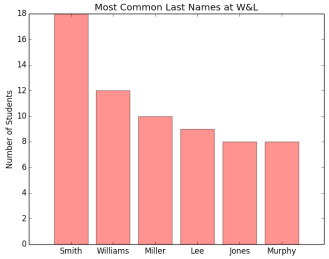
Graphing

- I provide code that will create a bar chart using the `matplotlib` library
  - generateFreqGraphs.py
- You will need to provide the appropriate information to the Python code to generate the graph
  - You can either
    - Use the user interface
    - Write code to directly call the plotFrequencyData function

Graphing: Using the User Interface

```
$ python3 generateFreqGraphs.py
What is the name of your properly-formatted data file?
data/lastnames.dat
How many results do you want to display? 6
What is the title of this graph? Most Common Last Names at
W&L
What is the y-axis label of this graph? Number of Students
['Smith', '18']
['Williams', '12']
['Miller', '10']
['Lee', '9']
['Jones', '8']
['Murphy', '8']
```

Generates Graph:



Can also save generated graph by clicking save icon

Graphing: Using Function Calls

```
from generateFreqGraphs import *

labels, values = processDataFile("data/lastnames.dat", 6)


plot = plotFrequencyData(labels, values, \
     "Most Commonly Occurring Last Names at W&L", \
     "Number of Students")

plot.savefig("data/lastnames.png")
```

We could then put this code into a loop to run it for all the files and updating the title accordingly.

graphing_example.py

Overview

1. Implement partial solution using a dictionary to map the name to its count
   - handles basic set up of solution, including reading and processing file
2. Implement a class that packages the name (a key) and its count together
   - Data and functionality given
   - Test the class
3. Implement Step 1 with objects of class you created in Step 2
   - Complete solution
4. Graph data generated from Step 3
5. Make web page with graphs