

## Lab Overview

- Review lab 8
- Prep for lab 9

## Lab 8: Pair Programming

Findley	Joseph
Jordan	Margaret
Ryan	Anna
Lizzie	Alison
Olivia	Chase
Lindsey	Parker
Kalady	Davis
Mary-Frances	Andrew
Chas	Robert
Rachel	Ian

Joseph	Findley
Margaret	Jordan
Chas	Anna
Alison	Lizzie
Chase	Olivia
Parker	Lindsey
Davis	Kalady
Andrew	Mary-Frances
Robert	Chas
Ian	Rachel

## Writing Encodings to the File

- With functions, writing to the file became simpler
  - Called a function that returned a string
  - Could write that string to a file
- Additional code was essentially
  - 1) open file for writing
  - 2) write the encoding to the file
  - 3) close the file

## 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

## Partial Gymnastics Code

```
def main():
    scores = getScoresFromFile(filename)
    avgDiffScore = scores.pop(0)
    avgExecScore = calculateAverageExecScore(scores)
    ...

def calculateAverageExecScore(listOfScores):
    listOfScores.sort()
    totalExecScore = 0
    for pos in range(1, len(listOfScores)-1):
        totalExecScore += listOfScores[pos]
    average = totalExecScore/(len(listOfScores)-2)
    return average
```

...  
For space, no comments,  
partial solution

March 20, 2018

Sprenkle - CSC111

5

## LAB 9 PREPARATION

March 20, 2018

Sprenkle - CSC111

6

## Review: Dictionaries

- What is a dictionary?
- What are some things we can do with dictionaries?

Review the problems we solved using a dictionary

## 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 a class to help with data
- Create output file used by another application
  - Common use of programming

## 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

## 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

March 20, 2018

Sprenkle - CSC111

11

## 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

March 20, 2018

Sprenkle - CSC111

sort\_ignore\_case.py

12

## 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?
- Keyword that must be the first parameter of every defined method?
- What are defined methods like?
- Special method name for constructor?
- Special name for method that helps with printing?

## 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`
- Keyword that must be the first parameter of every defined method?
  - `self`
- 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



## 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?

March 20, 2018

Sprenkle - CSC111

card.py

17

## 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 \_

March 20, 2018

Sprenkle - CSC111

card.py

18

## 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

March 20, 2018

Sprenkle - CSC111

19

## Testing our methods

- Can test similarly to how we tested functions

```
# test the str method
test.testEqual( str(c1), "Ace of spades")
test.testEqual( str(c2), "King of hearts")
test.testEqual( str(c3), "2 of diamonds")

# test get rummy value
test.testEqual( c1.getRummyValue(), 15 )
test.testEqual( c2.getRummyValue(), 10 )
test.testEqual( c3.getRummyValue(), 5 )

# test the card color
test.testEqual( c1.getCardColor(), "black" )
test.testEqual( c2.getCardColor(), "red" )
test.testEqual( c3.getCardColor(), "red" )
```

Mar 21, 2017

Sprenkle - CSC111

20

## 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

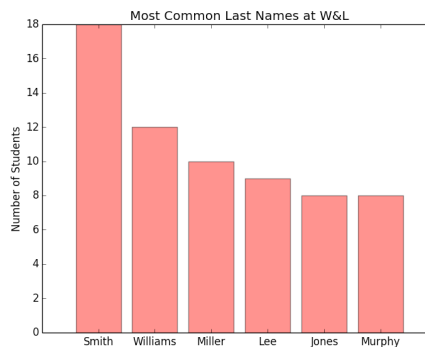
- I provide code that will create a bar chart using the `matplotlib` library
  - `generateFreqGraphs.py`
- You will need to provide the appropriate information to the Python code to generate the graph
  - You can either
    - Use the user interface
    - Write code to directly call the `plotFrequencyData`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

Save generated graph  
by clicking **save** icon

Generates Graph:



March 20, 2018

## 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")
```

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

graphing\_example.py

March 20, 2018

Sprenkle - CSCI111

26

## 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

CLEAR MINDS,  
FULL HEARTS,  
CAN'T LOSE!

- ~~FRIDAY NIGHT LIGHTS~~

COMPUTATIONAL THINKING