

Lab Overview

- Review lab 8
- Prep for lab 9

1

Lab Review

- Descriptions matter: your comments and variable names are an indicator of your level of understanding
- If variables or comments are incorrect, that is a good place to try to correct misunderstandings
- Goal: variable names help cement the abstract model of the problem in your mind

2

Adding New Code

- In general, don't change existing, working, well-tested functions unless told to do so
- Rather than changing `encryptMessage` to also handle `"\n"`, it's better to make `encryptFile` adhere to the preconditions for `encryptMessage`
 - `encryptMessage` would need yet another check on every character to see if it's a `"\n"`
 - `encryptFile` knows that the `"\n"` is at the end of every line.
 - It can remove that easily and then add a `"\n"` back after encoding
- Question: what can best handle this functionality?

March 20, 2023

Sprenkle - CSC1111

3

3

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

March 20, 2023

Sprenkle - CSC1111

4

4

Partial Gymnastics Code

```
def main():
    scores = getScoresFromFile(filename)
    avgDiffScore = scores.pop(0) Returns and deletes first item in list

    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, 2023

Sprenkle - CSCI111

5

5

File Reminders

- When you **open** a file, you should **close** the file

March 20, 2023

Sprenkle - CSCI111

6

6

LAB 9 PREPARATION

March 20, 2023

Sprenkle - CSC111

7

7

Review: Defining our own classes

- How do we define a new class?
- What are defined methods like?
- What is the keyword that **must** be the first parameter of every defined method?
 - What does that parameter represent?
- What is the special method name for the constructor?
- What is the special method that helps with printing?
 - What is the API (input, returned) for that method?
- Where do we define the data that is needed to represent every object of a class? What is the term for that data?
 - How do we access that data?

March 20, 2023

Sprenkle - CSC111

9

9

Review: Defining our own classes

- How do we define a class?
 - `class` keyword
- What are defined methods like?
 - Functions
- What is the keyword that must be the first parameter of every defined method?
 - `self`; represents the object being acted upon
- What is the special method name for constructor?
 - `__init__`
- What is the special method that helps with printing?
 - `__str__(self)` – returns a string representation of the object
- Where do we define the data that is needed to represent every object of a class?
 - In the constructor. Use `self._var` to represent that data. Can access that data in other methods as `self._var`. Called *instance variables*.

March 20, 2023

Sprenkle - CSC1111

10

10

Card Class (Incomplete)

```
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
```

Class Doc String

Method Doc String

Methods

Identify the *instance variables*

- How do we use them in Card methods?

card.py

March 20, 2023

11

11

Card Class (Incomplete)

```
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
```

Class Doc String

Method Doc String

Identify the *instance variables*

- How do we use them in Card methods?

Convention: instance variables are named beginning with `_`

card.py

March 20, 2023

12

12

Algorithm for Creating Classes

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

March 20, 2023

Sprenkle - CSC1111

13

13

Review: 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" )
```

March 20, 2023

Sprenkle - CSC111

14

14

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 counting names
- Create output file used by another application
 - Common use of programming

March 20, 2023

Sprenkle - CSC111

15

15

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/order strings usually?

March 20, 2023

Sprenkle - CSCI111

16

16

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

March 20, 2023

sort_ignore_case.py

Sprenkle - CSCI111

17

17

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

March 20, 2023

Sprenkle - CSCI111

sort_ignore_case.py

18

18

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

```
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, 2023

Sprenkle - CSCI111

sort_ignore_case.py

19

19

Lab Overview

(After warm up dictionary problem)

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 (data) 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


March 20, 2023

Sprenkle - CSC1111

20

20

Graphing

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

March 20, 2023

Sprenkle - CSC1111

21

21

Graphing: Using the User Interface

```
$ python generateFreqGraphs.py
```

What is the name of your properly-formatted data file? data/lastnames_freq.dat

How many results do you want to display? 6

What is the title of this graph? Most Commonly Occurring Last Names at W&L

What is the y-axis label of this graph? Number of Students

```
['Smith', '18']
```

```
['Lee', '11']
```

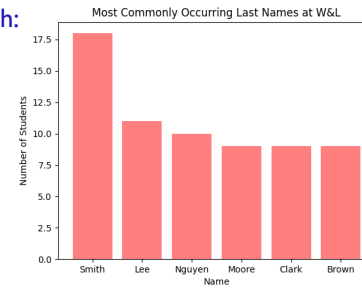
```
['Nguyen', '10']
```

```
['Moore', '9']
```

```
['Clark', '9']
```

```
['Brown', '9']
```

Generates Graph:



Save generated graph by clicking **save** icon

March 20, 2023

Sprenkle - CSCI111

22

Graphing: Using Function Calls

```
from generateFreqGraphs import *  
  
nameLabels, dataToPlot = processDataFile(DATADIR + \  
    "lastnames_freq.dat", 6)  
  
plot = plotFrequencyData(nameLabels, dataToPlot, \  
    "Most Commonly Occurring Last Names at W&L", \  
    "Number of Students")
```

March 20, 2023

Sprenkle - CSCI111

graphing_example.py

23

23

Overview

(After warm up dictionary problem)

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 (data) 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

March 20, 2023

Sprenkle - CSCI111

24

24

FNL

CLEAR MINDS,
FULL HEARTS,
CAN'T LOSE!

- ~~FRIDAY NIGHT LIGHTS~~

COMPUTATIONAL THINKING

March 20, 2023

25

25