

Lab 8

- Feedback on Lab 7
- Review
 - Lists
 - Files
 - Modules

1

LAB 7 FEEDBACK

2

Review Caesar Cipher

- Consider the following solutions

```
for char in message:
    asciiVal = ord(char)
    if asciiVal == 32:
        ...
    else:
        ...
```

```
for char in message:
    if char == " ":
        ...
    else:
        ...
```

March 16, 2021

Sprenkle - CSCI111

3

3


Review Caesar Cipher

- Consider the following solutions

```
for char in message:
    asciiVal = ord(char)
    if asciiVal == 32:
        ...
    else:
        ...
```

I know what " " means.
I don't immediately know
what 32 means.
**Lesson: prefer words
over numbers.**

```
for char in message:
    if char == " ":
        ...
    else:
        ...
```



March 16, 2021

Sprenkle - CSCI111

4

4

Comment Example

```
def encodeLetter(letter, key):  
    """  
    Encodes a single letter by the given key.  
    Parameters:  
    - letter: a single, lowercase character string  
    - key: an integer (between -25 and 25, inclusive)  
    POST: returns the encoded character as a str  
    """
```

- Does not say *who* called function, where parameters came from, or where returned to
 - Any code can call the function and pass in input from anywhere (e.g., hardcoded, from user input, test function, ...)
- Does not say variable name returned

5

Comment Example

```
def encodeLetter(letter, key):  
    """Encodes a single letter.  
    PRE: Input parameters are a single, lowercase  
    character string (char) and an integer key  
    (between -25 and 25, inclusive)  
    POST: returns the encoded character as a str"""
```

- Does not say *who* called function, where parameters came from, or where returned to
 - Any code can call the function and pass in input from anywhere (e.g., hardcoded, from user input, test function, ...)
- Does not say variable name returned
- Format doesn't matter as much as contains necessary content

6

Test Functions

- Designing test function
 - Pick good test cases
 - Automatically (i.e., program) checks results so it's easy to spot problems
 - Report input/test cases that cause the problems
- Benefits:
 - Quickly and automatically test functions
 - Quickly add new test cases
 - Can rerun test cases quickly if function implementation changes
 - If tested well, you can use the function in other programs with confidence

7

Review

- What are things we can do to lists?
- How do we work with files?
 - What are things we can do with files?
- What is a module?
 - What are the benefits of modules?
 - How do we create a module?
 - How do we use functions defined in a module?

8

Review: List Operations

Similar to operations for strings

Concatenation	<code><seq> + <seq></code>
Repetition	<code><seq> * <int-expr></code>
Indexing	<code><seq>[<int-expr>]</code>
Length	<code>len(<seq>)</code>
Slicing	<code><seq>[:]</code>
Iteration	<code>for <var> in <seq>:</code>
Membership	<code><expr> in <seq></code>

March 16, 2021

Sprenkle - CSCI111

9

9

Review: List Methods

Method Name	Functionality
<code><list>.append(<i>x</i>)</code>	Add element <i>x</i> to the end
<code><list>.sort()</code>	Sort the list
<code><list>.reverse()</code>	Reverse the list
<code><list>.index(<i>x</i>)</code>	Returns the index of the first occurrence of <i>x</i> , Error if <i>x</i> is not in the list
<code><list>.insert(<i>i</i>, <i>x</i>)</code>	Insert <i>x</i> into list at index <i>i</i>
<code><list>.count(<i>x</i>)</code>	Returns the number of occurrences of <i>x</i> in list
<code><list>.remove(<i>x</i>)</code>	Deletes the first occurrence of <i>x</i> in list
<code><list>.pop(<i>i</i>)</code>	Deletes the <i>i</i> th element of the list and returns its value

Note: methods do **not return** a **copy** of the list ...

March 16, 2021

Sprenkle - CSCI111

10

10

Review: str Method Flashback

- `string.split([sep])`
 - Returns a *list* of the words in the string `string`, using `sep` as the delimiter string
 - If `sep` is not specified or is `None`, any *whitespace* (space, new line, tab, etc.) is a separator
 - Example:

```
phrase = "Hello, Computational Thinkers!"  
x = phrase.split()
```

What is x? What is its data type? What does x contain?

Review: str Method Flashback

- `string.join(iterable)`
 - Return a string which is the concatenation of the *strings* in the **iterable**/sequence. The separator between elements is `string`.

➤ Example:

```
x = ["1", "2", "3"]  
phrase = " ".join(x)
```

What is x's data type?
What is phrase's data type?
What does phrase contain?

Review: Iterating through a List

- Read as

- For every element in the list ...

An item in the list

list object

```
for item in list:  
    print(item)
```

Iterates through
items in list

- Output equivalent to

```
for x in range(len(list)):  
    print(list[x])
```

Iterates through
positions in list

March 16, 2021

Sprenkle - CSCI111

daysOfWeek.py

13

13

Review: Files

- Conceptually, a file is a **sequence** of data stored in memory

- To use a file in a Python script, create an object of type **file**

- **file** is a *data type*

Built-in function
"constructs" a file object

- `<varname> = open(<filename>, <mode>)`

- `<filename>`: string

- `<mode>`: string, "r" for read, "w" for write, "a" for append (and others)

- Ex: `dataFile = open("years.dat", "r")`

March 16, 2021

Sprenkle - CSCI111

14

14

Review: Writing to a File

- Create a file object in **write** mode:
 - `myFile = open("demo.txt", "w")`
- Call write method on file object:
 - `myFile.write("Write string to file")`
 - `myFile.write("Also this string")`
- Close the file:
 - `myFile.close()`

What will the output file look like?

Review: Modules

- Modules group together related functions and constants
- Unlike functions, no special keyword to define a module
 - A module is named by its filename
- You've used modules in the past
 - `graphics.py`
 - `test.py`
 - `game.py`

Just a
Python file!

Problem: Temperature Data

- **Given:** data file that contains the daily high temperatures for last year at one location
 - Data file contains one temperature per line
 - Example: `data/florida.dat`
- **Problem:** What is the average high temperature for the location?

```
def calculateAvgTemp( datafileName ):
```

Rule of Thumb: Always look at data file before processing it

Problem: Temperature Data

- Implement the algorithm

Problem: Report of Avg Temperature

- **Given:** data files that contains the daily high temperatures for last year at various locations
 - Data file contains one temperature per line
 - Example: `data/florida.dat`
- **Problem:** Write a report of the locations and the average temperature in the form
 - Average temperature should be displayed to two decimal places

```
<location1> <avgtemp1>  
<location2> <avgtemp2>  
...
```

March 16, 2021

Sprenkle - CSCI111 [reportAvgData.py](#) 19

19

Problem: Report of Avg Temperature

- **Algorithm:**
 - Open the file for writing
 - Write out the data to the file
 - Use format
 - Include the `\n`
 - Close the file

March 16, 2021

Sprenkle - CSCI111

20

20

Recursive Copy

- Many Unix commands have command-line options
 - Example: `ls -l`
 - `-l`: long form
 - Command run during turnin so you can see the dates and other information on your submitted files.
- `cp` has the `-r` option, which means to *recursively* copy
 - Meaning to copy the directory and all of its contents (including subdirectories)
 - Example: to copy the lab8 directory and all of its contents into your cs111 directory
 - `cp -r /csdept/courses/cs111/handouts/lab8 ~/cs111`

March 16, 2021

Sprenkle - CSCI111

21

21

Lab 8 Overview

- Modules
- Reading Files
- Writing Files
- Functions/Lists

Combining aaaaall that we've learned in the last 9 weeks. Remember (or review) all that you can do.

March 16, 2021

Sprenkle - CSCI111

22

22