

Objectives

- Wrap up lists
- Introduction to Files

Feb 27, 2017

Sprenkle - CSC1111

1

Midway Check: Parts of an Algorithm

- Primitive operations
 - What data you have, what you can do to the data
- Naming
 - Identify things we're using
- Sequence of operations
- Conditionals
 - Handle special cases
- Repetition/Loops
- Subroutines
 - Call, reuse similar techniques

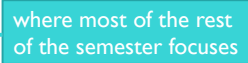


- Which of these have we covered?
- How do we implement them in Python?

Feb 27, 2017

Sprenkle - CSC1111

2

Midway Check: Parts of an Algorithm

- Primitive operations 
 - What **data** you have, what you **can do** to the data
- Naming
 - Identify things we're using
- Sequence of operations
- Conditionals
 - Handle special cases
- Repetition/Loops 
- Subroutines 
 - Call, reuse similar techniques

where most of the rest of the semester focuses

No longer primitive

One more loop structure

Defining our own

Feb 27, 2017

Sprenkle - CSC1111

3

Review

- What is a list?
- What is the syntax to describe a list?
- How are lists and strings similar?
- How are they different?
 - What are the implications of those differences?

Feb 27, 2017

Sprenkle - CSC1111

4

Lists: A Sequence of Data Elements

element daysInWeek

"Sun"	"Mon"	"Tue"	"Wed"	"Thu"	"Fri"	"Sat"
0	1	2	3	4	5	6

Position/
index
in the list

len(daysInWeek) is 7

- Elements in lists can be *any* data type
- Operations similar to strings

Feb 27, 2017

Sprengle - CSC1111

5

Making Lists of Integers Quickly

- If you want to make a list of integers that are evenly spaced, you can use the **range** generator
- Example: to make a list of the even numbers from 0 to 99:

➤ `evenNumList = list(range(0, 99, 2))`

Converts the generated
numbers into a list

Feb 27, 2017

Sprengle - CSC1111

6

Review: Lists vs. Strings

- Strings are **immutable**
 - Can't be mutated?
 - Err, can't be modified/changed
- Lists are **mutable**
 - Can be changed
 - Changes how we call/use methods

```
groceryList=["milk", "eggs", "bread", "Doritos", "OJ", \ "sugar"]
```

```
groceryList[0] = "skim milk"  
groceryList[3] = "popcorn"
```

```
groceryList is now ["skim milk", "eggs", "bread", \ "popcorn", "OJ", "sugar"]
```

One effect: list methods modify the list on which the method was called
→ Don't return a copy of the object, modified

Feb 27, 2017

Sprengle - CSC1111

7

Special Value: **None**

- Special value we can use
 - E.g., Return value from function/method when there is an error
 - Or if function/method does not return anything

(Similar to **null** in Java)

- If you execute

```
list = list.sort()  
print(list)
```

 - Prints **None** because `list.sort()` does **not** return anything

Feb 27, 2017

Sprengle - CSC1111

8

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? Its data type? What does x contain?

Feb 27, 2017

Sprenkle - CSC1111

9

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?

Feb 27, 2017

Sprenkle - CSC1111

10

Copies of Lists

- What does the following code output?

```
x = [1, 2, 3]  
y = x  
y[0] = -1  
print(y)  
print(x)
```

- Run in Python interpreter
- View in Python visualizer

Feb 27, 2017

Sprenkle - CSC1111

11

List Identifiers are Pointers



```
x = [1, 2, 3]  
y = x  
x → [1 | 2 | 3]  
y → [1 | 2 | 3]
```

- `y` is **not** a copy of `x`
 - `y` points to what `x` points to
- How to make a copy of `x`?

```
y = x + [] OR y = []  
           ↑ y.extend(x)  
Empty list
```

Feb 27, 2017

Sprenkle - CSC1111

12

Sources of Input to Program

- User input
 - Slow if need to enter a lot of data
 - Error-prone
 - User enters the wrong value!
 - What if want to run again after program gets modified?

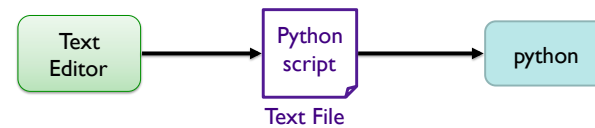
Feb 27, 2017

Sprengle - CSC1111

13

Sources of Input to Program

- Text files
 - Enter data once into a file, save it, and reuse it
 - Good for large amounts of data
 - Programs can use files to *communicate*
 - Need to be able to *read from* and *write to* files

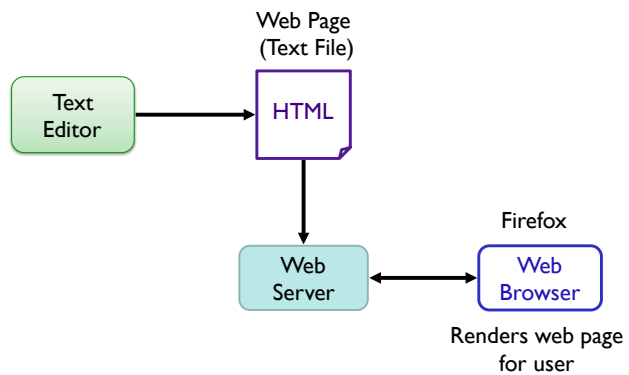


Feb 27, 2017

Sprengle - CSC1111

14

Example Use of Files

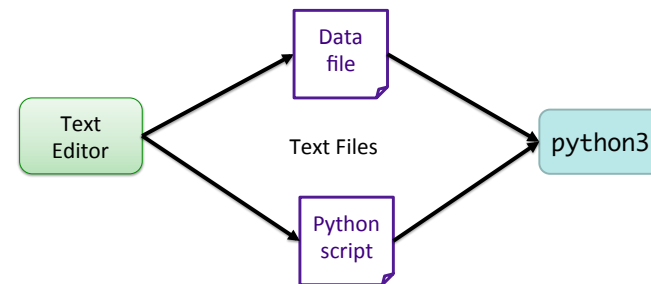


Feb 27, 2017

Sprengle - CSC1111

15

Example Use of Text File as Input



Feb 27, 2017

Sprengle - CSC1111

16

Wheel of Fortune

- Uses a file of puzzles
 - Can modify puzzle file to get different puzzles

Feb 27, 2017

Sprengle - CSC1111

17

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*
 - **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")`

Feb 27, 2017

Sprengle - CSC1111

18

Common File Methods

Method Name	Functionality
<code>read()</code>	Read the entire content from the file, returned as a string object
<code>readline()</code>	Read one line from file, returned as a string object (which includes the "\n"). If it returns "", then you've reached the end of the file
<code>write(string)</code>	Write a string to the file
<code>close()</code>	Close the file. Must close the file after done reading from/writing to a file

Feb 27, 2017

Sprengle - CSC1111

19

Reading from a File

- Examples of reading from a file using file methods
 - Show file: `data/years.dat` Typically use .dat or .txt file extension to name files containing data or text
- `file_read.py` (using `read()`)
 - How is what Python printed different than the file's content?
 - How to fix?
- Using `readline()`
 - Hold off on using to read a whole file

Feb 27, 2017

Sprengle - CSC1111

20

Reading from a File

- Recall that a file is a *sequence* of data
- Can use a **for** loop to iterate through a file

A *line* (of type **str**) from
the file (includes `\n`)

file object

```
for line in dataFile:  
    print(line)
```

- Read as: for each line in the file, do something

This Week

- Tuesday: Lab 6
 - Encodings – Caesar Cipher
 - Lists
 - Files