Objectives

- Continuing lists
- Introduction to Files
- Broader Issue: Cryptocurrency

1

Review

- What is a list?
- What is the syntax for a list?
- How are lists and strings similar?
  - What similar things can we do to lists and strings?
- How are they different?
  - What are the implications of those differences?
- What does None mean/do?
- How do we make a copy of a list?

2

Review: Lists vs. Strings

- Strings are **immutable**
  - Can't be mutated?
  - Err, can't be modified/changed
- Lists are **mutable**
  - Can be changed "in place"
  - 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

3

Review: 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

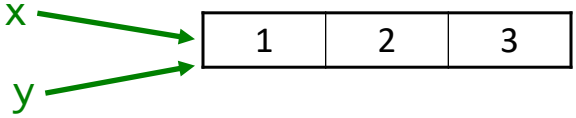
```
list = list.sort()
print(list)
```

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

4

Review: List Identifiers are **Pointers**

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



- 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

5

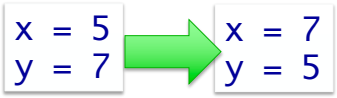
Immutable vs Mutable Parameters

PASSING PARAMETERS

6

Passing Parameters

- Only ***copies*** of the actual parameters are given to the function
  - For **immutable** data types Which are?
- The ***actual*** parameters in the calling code do not change
- **Swap example:**
  - Swap two values in script
  - Then, put into a function



Recall: Immutable Data is Passed by Value

```
def main():
    x = 5
    y = 7

    swap(x, y)

    print("x =", x)
    print("y =", y)

def swap(a, b):
    tmp = a
    a = b
    b = tmp
    print(a, b)

main()
```

This code does not have the desired effect in that x and y are not swapped.

Since integers are passed ***by value***, the values of x and y are not changed by the call to the swap function.

Lists as Parameters to Functions

- Lists are **not** passed-by-value/copied
- Different from immutable types (e.g., numbers, strings)
- Function parameter is actually a ***pointer*** to the list in memory

Impact: If a list that is passed as a parameter into a function is **modified *in* the function**, the list **is modified *outside* the function**

9

Problem: Sort a list of 3 numbers, in descending order

```
# order list such that list3[0] >= list3[1] >= list3[2]
def descendSort3Nums( list3 ):
```

Called as:

```
list = …
descendSort3Nums(list)
print(list)
```

How implemented with list methods?
Can we do this using only 3 comparisons?

10

Descend Sort a List w/ 3 elements

```
def descendSort3Nums(list3):
   if list3[1] > list3[0]:
       # swap 'em
       tmp = list3[0]
       list3[0] = list3[1]
       list3[1] = tmp

   if list3[2] > list3[1]:
       tmp = list3[1]
       list3[1] = list3[2]
       list3[2] = tmp

   if list3[1] > list3[0]:
       tmp = list3[0]
       list3[0] = list3[1]
       list3[1] = tmp
```

```
def main():
    list = [1,2,3]
    descendSort3Nums(list)
    print(list)
```

Function does **not** *return* anything. Simply modifies the `list3` parameter.

11

FILES

12

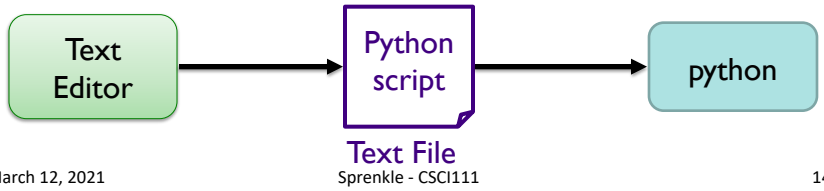
Sources of Input to Program: User Input

- Pros
  - Easy!
  - Intuitive!
- Cons
  - 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?

13

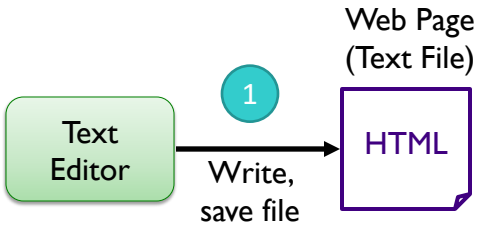
Sources of Input to Program: Text Files

- Pros
  - 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
- Cons
  - Not as intuitive in programming
  - Requires creating a file



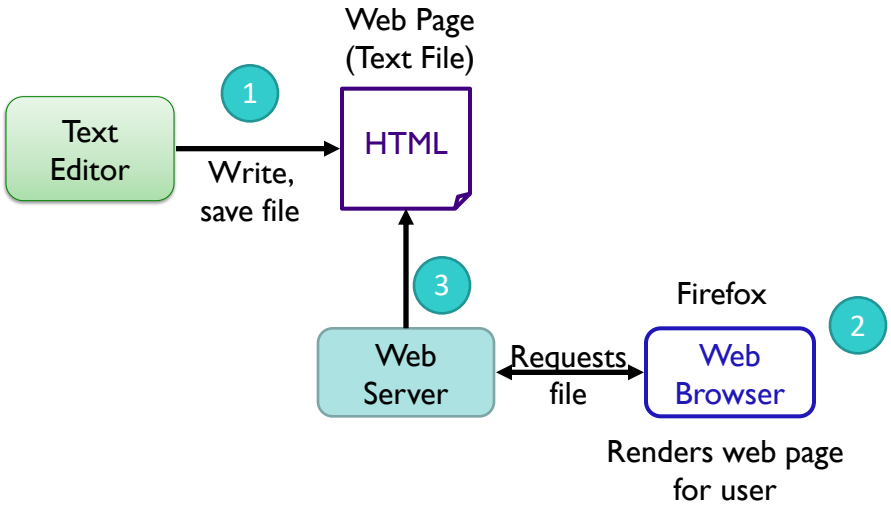
14

Example Use of Files: on the Web



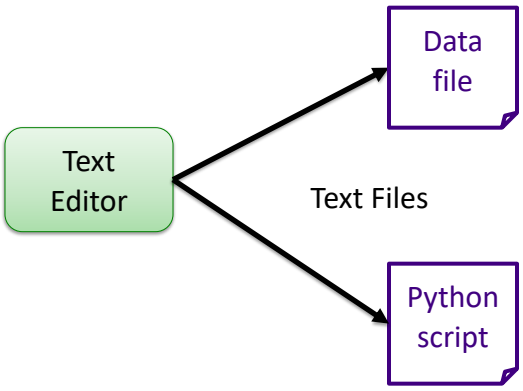
15

Example Use of Files: on the Web



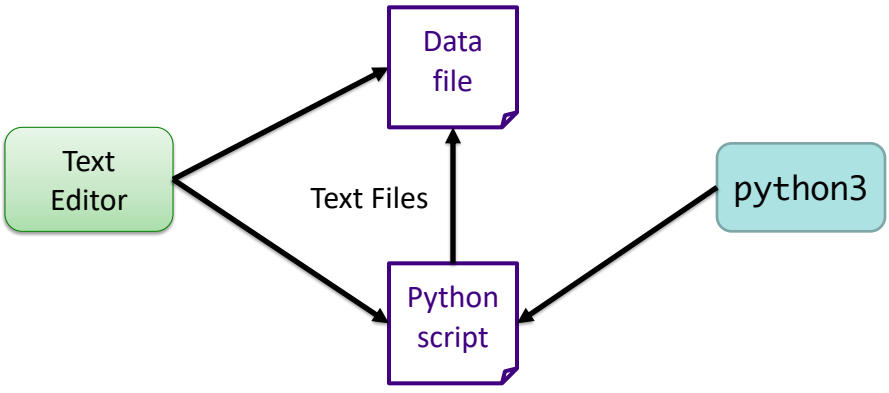
16

Example Use of Text File as Input: Data!



17

Example Use of Text File as Input: Data!



18

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*
  - `<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" )`

Built-in function
"constructs" a `file` object

19

Common File Methods

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

20

Reading from a File

- Examples of reading from a file using file methods
  - Show file: `data/famous_pairs.txt`

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()`

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

`for_file_read.py`

Data Types of Loop Variables

What are the data types of the loop variable **x**?

```
myString = "some string"
dataFile = open("datafile.dat", "r")

for x in range(len(myString)):
     # loop body …

for x in myString:
     # loop body …

for x in dataFile:
     # loop body …
```


23

Data Types of Loop Variables

What are the data types of the loop variable **x**?

```
myString = "some string"
dataFile = open("datafile.dat", "r")

for x in range(len(myString)):
     # loop body …

for x in myString:
     # loop body …

for x in dataFile:
     # loop body …
```

integer

string → single characters

string → line (include \n)

24

Wheel of Fortune

- (OK, more like hangman)
- Uses a file of puzzles
  - Can modify puzzle file – add lots more puzzles!

Handling Numeric Data

- We have been dealing with reading and writing *strings* so far
  - Read from a file: get a string
  - Write to file: use a string
- What do we need to do to **read numbers** from a file?
- How can we **write numbers** to a file?

Handling Numeric Data

- We have been dealing with reading and writing *strings* so far
  - Read from a file: get a string
  - Write to file: use a string
- What do we need to do to **read numbers** from a file?
  - Cast as a numeric type, e.g., `int` or `float`
- How can we **write numbers** to a file?
  - Cast number as a `str`

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 (to 2 decimal places) for the location?

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

Broader Issue: Cryptocurrency

- Background: Common challenge in computer science is to use terminology to make people feel excluded
- My goals:
  - Make some terminology less foreign, less initimidating because it's related to topics we've discussed in class
- My approach: Choose a recent article that introduces topic but isn't too hypey/technical/ buzzwordy

29

Mining Cryptocurrency

- From article: "… solv[ing] a mathematical puzzle that helps verify a group of transactions—referred to as a block—then adds them to the blockchain leger. The first computer to do so successfully is rewarded with a small amount of cryptocurrency for its efforts."

30

BitCoin: Proof of Work

- ***Hash*** functions map from one "thing" to a string of fixed length
  - Output may look random but it's not
  - Given the same input, always results in the same output
  - Unlikely that other inputs will result in same output
- Example: URL → TinyURL
  - https://mybig.url/thatisreallylong/but/helpful → https://tinyurl.com/4f3c38wp
  - Recreating the URL will result in the same URL

BitCoin: Mathematical Puzzle

- Problem: given a *challenge string*, find *x* such that the result of hashing the challenge string and x gives a result with certain properties
  - x is called the *proof string*

BitCoin: Mathematical Puzzle

- Problem: given a *challenge string*, find *x* such that the result of hashing the challenge string and x gives a result with certain properties
  - x is called the *proof string*
- Example property: resulting hash must start with at least 8 zeros
  - This resulting hash does not meet criteria

33

BitCoin: Mathematical Puzzle

- Difficult to find the x that results in the desired property
  - Have to try lots of different x's
- But, easy to validate!
  - If someone says *x* results in a hash with desired properties, we can run the hash function and verify if the resulting hash has the desired properties

34

BitCoin: Mathematical Puzzle

- As computational power increases, it takes less time to find an x.
- How can we [easily] make the problem harder?

35

BitCoin: Proof of Work

- How does this relate to *proof of work*?
  - We know, on average/probabilistically, how many tries it would take to get a hash that satisfies the requirements

36

Broader Issue: Cryptocurrencies

- What are cryptocurrencies?
  - What are they good for?
  - What are their limitations?
- With your newly attained computer science knowledge, what ideas make more sense?
  - What is still confusing?
- What questions about cryptocurrencies do you still have?

37

Broader Issue: Cryptocurrencies

- With your newly attained computer science knowledge, what ideas make more sense?
  - What is still confusing?

38

Upcoming Talk

- Join Prof. Aliaa Bassiouny, Lawrence Term Associate Professor of Finance, for a talk with Kevin Batteh '95 '98L titled “Bitcoin, Blockchain and Beyond: How Digital Assets are Regulated and What it all Means.”
- This event on Monday, March 15th at 7 PM is open to the entire W&L community.

Looking Ahead

- Last problems of Lab 7 – due Monday
- Pre Lab 8
  - Lists – skipping some sections
  - Files