CMSC201
Computer Science I for Majors

Lecture 22 – Searching
Welcome Back!
Review: Tuples & Dictionaries

- Create an empty tuple
- Create a dictionary that contains three different (key, value) pairs, similar to “a is for apple”
  - Add one additional (key, value) pair
  - Update one of your (key, value) pairs
  - Remove one of your (key, value) pairs

- Why must dictionary keys be unique?
- Do values need to be unique?
Review: Matching Symbols

• Match the following data types to the symbols needed to create them (may be more than one)

  Dictionary
  List
  String
  Tuple

  {   }
  (   )
  "   "
  [   ]
  ,   ,
Review: Matching Symbols

• Match the following data types to the symbols needed to create them (may be more than one)

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary</td>
<td>{ }</td>
</tr>
<tr>
<td>List</td>
<td>( )</td>
</tr>
<tr>
<td>String</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>Tuple</td>
<td>[ ]</td>
</tr>
</tbody>
</table>
Review: Mutability

Which of the following are mutable data types?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>???</td>
</tr>
<tr>
<td>Dictionary</td>
<td>???</td>
</tr>
<tr>
<td>Float</td>
<td>???</td>
</tr>
<tr>
<td>Integer</td>
<td>???</td>
</tr>
<tr>
<td>List</td>
<td>???</td>
</tr>
<tr>
<td>String</td>
<td>???</td>
</tr>
<tr>
<td>Tuple</td>
<td>???</td>
</tr>
</tbody>
</table>
Review: Mutability

Which of the following are mutable data types?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Immutable</td>
</tr>
<tr>
<td>Dictionary</td>
<td><strong>Mutable</strong></td>
</tr>
<tr>
<td>Float</td>
<td>Immutable</td>
</tr>
<tr>
<td>Integer</td>
<td>Immutable</td>
</tr>
<tr>
<td>List</td>
<td><strong>Mutable</strong></td>
</tr>
<tr>
<td>String</td>
<td>Immutable</td>
</tr>
<tr>
<td>Tuple</td>
<td>Immutable</td>
</tr>
</tbody>
</table>
Review: Implementation

• You are given a dictionary of the NATO phonetic alphabet, in the form:

\[
\text{ALPHA} = \{ "A" : "Alpha", "B" : "Bravo", "C" : "Charlie", \ldots \text{etc.} \}\n\]

• Write a function to convert a string from the user into its phonetic code words
  – You only need to handle letters (case insensitive)
Review: Implementation Example

• Here is an example of how it should work:

Please enter a word: EXAMPLE
The word "EXAMPLE" becomes
"Echo X-ray Alpha Mike Papa Lima Echo"

Please enter a word: dogmeat
The word "dogmeat" becomes
"Delta Oscar Golf Mike Echo Alpha Tango"
Any questions about the material we just reviewed?
Today’s Objectives

• To learn more about searching algorithms
  – Linear search
  – Binary search
Motivations for Searching

• Want to know if something exists
  – Python can do this for us!

• Want to know where something exists
  – Python can actually do this for us too!
  – raceWinners.index(718)

• But how does Python does this?
Exercise: `find()`

• Write a function that takes a list and a variable and returns the index of the variable in the list
  – If it’s not found, return -1
  – You can’t use `index()`!

```python
def find(searchList, var)
```
Exercise: `find()` Solution

```python
def find(searchList, var):
    for i in range(len(searchList)):
        if searchList[i] == var:
            return i

    # outside the loop, means that
    # we didn't find the variable
    return -1
```
Linear Search

• You just programmed up a search function!

• This algorithm is called \textit{linear search}

• It’s a common, fundamental algorithm in CS

• It’s especially useful when our information isn’t in a sorted order
  – But it isn’t very fast
Searching Sorted Information

• Now, imagine we’re looking for information in something sorted, like a phone book

• We know someone’s name (it’s our “variable”), and want to find their number in the book

• What is a good method for locating their phone number?

  – Think about how you would do this.
Algorithm in English

• Open the book midway through.
  – If the person’s name is on the page you opened to
    • You’re done!
  – If the person’s name is after the page you opened to
    • Tear the book in half, throw the first half away and repeat this process on the second half
  – If the person’s name is before the page you opened to
    • Tear the book in half, throw the second half away and repeat this process on the first half
• This is rough on the phone book, but you’ll find the name!
Binary Search
Binary Search

- The algorithm we just demonstrated is better known as *binary search*
  - We talked about it briefly last class, remember?

- Binary search is only usable on **sorted** lists
  - Why?
Solving Binary Search

• Binary search is a problem that can be broken down into
  – Something simple (breaking a list in half)
  – A smaller version of the original problem (searching that half of the list)

• That means we can use ... recursion!
Exercise: Recursive Binary Search

• Write a recursive binary search!

• To make the problem slightly easier, make it “checking to see if something is in a sorted list”
  – If there’s no “middle” of the list, we’ll just look at the lower of the two “middle” indexes
Exercise: Recursive Binary Search

• Write a recursive binary search!
• Remember to ask yourself:
  – What is our base case(s)?
  – What is the recursive step?

```python
def binarySearch(myList, item):
```

• A hint: in order to get the number at the middle of the list, use this line:
  ```python
  myList[len(myList) // 2]
  ```
Time for...

LIVECODING!!!
Announcements

• Final is Thursday, December 15th (3:30 – 5:30)

• Project 2 will come out soon

• The third survey will be announced on Blackboard shortly (0.5% of your grade)
  – Not on Blackboard
  – TA Feedback; anonymous to the TAs