Advanced Lists & Dictionaries

Introduction to Programming - Python
Advanced Lists
Multi Dimensional Lists

- All of the lists we have been creating so far have been one dimensional (i.e. linear) in nature
- High level programming languages also have the ability to construct lists that can store multiple values within the same element
Multi Dimensional Lists

• A one dimensional list can be thought of as a line
• A two dimensional list can be thought of as a plane
• A three dimensional list can be thought of as a cube
• … etc!
Creating a two dimensional list

You can create a two dimensional list in Python by simply nesting a list inside of another list. Example:

```python
mylist = [ [ 'a', 'b', 'c' ],
           [ 'd', 'e', 'f' ] ]

print (mylist[0])
print (mylist[0][0])
print (mylist[1][1])

>> [ 'a', 'b', 'c' ]
>> a
>> e
```
Live Example
Programming Challenge: Tic Tac Toe

- Write a program that uses a multi-dimensional list to simulate a tic tac toe board.
- Initially set up your list to contain all “empty” spaces represented by a “.” character.
- Ask the user for a coordinate (i.e. 0, 0).
- If the space they request is still free you can place an X in the slot.
- Have the computer pick a random spot to place an O character.
- If there are no more spots left then you can end the game.
Shuffling / Randomizing a List

• You can shuffle (or randomize) the elements in a list by applying the following algorithm:
  • Create an empty list
  • Enter into a while loop
  • Obtain a random number between 0 and the length of the list you wish to shuffle
  • Place a copy of the data that exists at this random position into our new list
  • Remove the item from original list
  • Test the length of the original list – if it is zero we can end the while loop
  • Your new list now contains a shuffled version of your original list
Game of Sticks

• The “game of sticks” is a game where players take turn removing a stick from a pile

• Players may remove 1, 2 or 3 sticks per turn.

• The player that removes the last stick is the loser.

• Write a program that simulates this game with one human player and one computer player
Game of Sticks

• Next, rewrite the program so that the computer plays against another computer player
Game of Sticks

- Now, have each computer remember their plays throughout the game

- When one of the computer players wins, have that computer remember the winning plays for the next hand
Game of Sticks

- Have your “learning” computer players play 100,000 games against each other

- When they are finished, have the players store their findings in two text files
Game of Sticks

- Finally, incorporate this text file into your first game (human vs computer)

- You will have created a monster! You can’t win!
Dictionaries

first lets list off the data types we are already familiar with in python...
Dictionaries

• A Dictionary in Python is a sequence object like a list

• Unlike a list, a Dictionary doesn’t use integer based index values.

• Instead, Dictionaries use immutable objects (like Strings) to index their content

• In other languages Dictionaries are often referred to as “Associative Arrays” or “Hashmaps”
## Lists vs. Dictionaries

<table>
<thead>
<tr>
<th>List</th>
<th>Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Sequence Structure</td>
<td>• Sequence Structure</td>
</tr>
<tr>
<td>• Mutable (can be changed if you know which index you are modifying)</td>
<td>• Mutable (can be changed if you know which index you are modifying)</td>
</tr>
<tr>
<td>• Items are stored in a particular order based on index values</td>
<td>• Items are not stored in any particular order</td>
</tr>
<tr>
<td>• Items can be indexed using an integer</td>
<td>• Items can be indexed using anything that is immutable (integer, String, etc)</td>
</tr>
</tbody>
</table>
Creating a Dictionary

• You can create a Dictionary using the curly braces – “{“ and “}”, like this:

    my_dictionary = { }  

• This will create an empty Dictionary
Adding items to a Dictionary

• In order to add an item to a Dictionary you need to specify a “key” – this is usually in the form of a String.

• You can then associate some data with that key. For example I could associate the number 3.2 with the key “python” by doing this:

my_dictionary[“python”] = 3.2

• This will place the number 3.2 into the Dictionary at the position marked by the String “python”
Accessing Dictionary items

• You can access all items in a Dictionary by printing it out, like this:

```python
print (my_dictionary)
```

• However, you often just want to access one item – this works the same as with an array, but you will use a key instead of an integer index:

```python
print (my_dictionary["python"])
```
Creating a Dictionary with Values

- Dictionaries store key / value pairs. You can initialize a Dictionary with a known set of key / value pairs by using the following syntax:

  ```python
  my_dictionary = { "python":3.2, "java":1.8 }
  ```

- This will create a Dictionary with the keys “python” and “java”
Invalid Indexes

- Note that you cannot access elements in a Dictionary that have not been defined. This would raise an exception if “java” was not a key in the Dictionary:

  \[
  \text{print ( my\_dictionary["java"] )}
  \]

- You can use the “in” operator to test to see if a key is in a dictionary like this:

  \[
  \text{if ( "java" in my\_dictionary") :}
  \]

- Note that this will check for the presence of a key in a dictionary, not for the data that the key is storing!
Deleting a key in a Dictionary

• You can use the `del` command to delete a key in a Dictionary, like this:

```python
del my_dictionary["java"]
```

• Make sure that you know that the key in question has been defined in the Dictionary before you run this command!
Clearing a Dictionary

- You can clear all keys in a Dictionary by doing the following:

```python
my_dictionary.clear()
```
Behind the Scenes:
How Dictionaries work

John Smith
Lisa Smith
Sandra Dee

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>521-8976</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>521-1234</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>13</th>
<th>14</th>
<th>521-9655</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Working with Dictionary Keys and Values
Finding all the keys in a Dictionary

- In a list we can easily visit all "slots" in the list by visiting every index value in the list, like this:

  ```python
  mylist = [ 'a', 'b', 'c', 'd' ]
  for i in range(0, len(mylist)):
      print (mylist[i])
  ```

- However, we can't do this with a Dictionary since the index values for a dictionary are not necessarily going to be integers

- We can use the "keys()" method to ask a Dictionary object to expose all of the keys that are defined within that Dictionary like this:

  ```python
  my_dictionary.keys()
  ```
Iterating over every item in a Dictionary

• To iterate over every time in a Dictionary you need to be able to visit every key value (as opposed to simply knowing the size of a list and iterating over the integer range of the list)

• You can extract all the keys from a Dictionary by doing the following:
  
  for key in my_dictionary.keys():

• The target variable “key” will assume each key value in your Dictionary as the loop progresses.

• You can print out all items with their keys by doing the following:
  
  for key in my_dictionary.keys():
      print (key, " == ", my_dictionary[key])
Iterating over every item in a Dictionary

• There is no guarantee that the keys() method will return the keys of a dictionary in any particular order. Why? Because ugh…

• However, you can ask Python to sort the keys before you iterate over them, like this:

  for key in sorted( my_dictionary.keys() ):

• This will sort the keys in ascending order, which then lets you access the elements in the dictionary in ascending order.
Finding all the values in a Dictionary

• You can also iterate over just the values in a dictionary (not just the keys) using this syntax:

```python
for v in my_dictionary.values():
    print (v)
```

• Note that doing this will only expose the values in a dictionary and not the key – this means that you cannot change the values in the dictionary using this method. This is analogous to iterating over a list like this:

```python
for item in my_list:
    print (item)
```
Iterating over every item in a Dictionary

- You can also iterate over a Dictionary by using the following technique to extract both the key and the value at the same time:

```python
for key, value in my_dictionary.items():
    print (key, value)
```
Programming Challenges
Programming Challenge

• Write a program that asks the user to enter in a series of student names and test scores

• Use a Dictionary to store a new record in the dictionary based on the student name (i.e. the name becomes the key) – store the student score at that position.

• When you are finished print out the names and scores of all students

• Extension: print them out in alphabetical order!
Programming Challenge

• Extension: Allow the user to enter in multiple scores for each student. If the student already has a score simply add it to the total score for the student.

• You will also need to keep track of the number of tests somehow
  • You could create a separate dictionary for # of tests
  • You could store a list in a single dictionary that has two fields – one for points and one for number of tests
Programming Challenge

• Count the frequency of all words in the Declaration of Independence
Programming Challenge

- Write a program that stores a series of price values in a dictionary along with the name of some products.

- Then write a lookup program that lets the user enter in a product. If the product exists you can then display its price.
Programming Challenge

- You are given a data set that contains three fields – a location, a bug species and a count
- The file contains an arbitrary number of records in any order
- Calculate the following:
  - Number of bugs by species
  - Number of bugs by location
  - Number of bugs by species & location
Programming Challenge: Poker

- Write a program that shuffles a deck of cards and selects five cards at random.
- You can obtain a listing of cards through a text file named ‘cards.txt’.
- Shuffle the deck, deal five cards to the user, then ask if they would like to deal a second set of cards. Continue to deal the cards until you run out of cards in the deck.