MIT AITI
Python Software Development

Lab 3: Data Structures

We have demonstrated in lecture how lists, tuples, and dictionaries each provide a valuable way to store information. Through the activities in this lab, you will have the opportunity to explore the capabilities and limitations of these data structures. At the conclusion of this lab, you will know how to choose the right data structure for your software development needs!

1. Messi and Xavi, the stars of Lecture 4, have just finished up a long afternoon of football and decide to head to the grocery store. They have a list of groceries that they would like to pick up at (Shoprite). Their list, `groceries`, is initialized as follows:

   ```
   groceries = ['bananas','strawberries','apples','bread']
   ```

   a. They want to celebrate their victory and add champagne to the end of their original grocery list. Write the code to modify `groceries` accordingly.

   b. Messi decides he doesn't need bread. Write the code to remove this from `groceries`.

   c. The store has 26 aisles, labeled 'a', 'b', 'c', … 'z' from the left side of the store to the 'right' (apples are found in the 'a' aisle, strawberries in the 's' aisle, etc.). What operation could Messi perform on the list to make it easier for him to find the items he needs in the store? Write the code below.

2. The store wants to design a catalog of all items in stock and their prices.
   a. What data structure would you choose to store this information and why?

b. Prices at the store are shown in the table below; write code to store this information in the data structure you chose in part (a).

Item	Price
Apples	100
Bananas	200
Bread	250
Carrots	300
Champagne	1000
Strawberries	1500

c. The price of strawberries goes up in the winter to 1500 ; how would you modify the price in your data structure?

d. Soccer players insisted on more protein options for their diets, so the store decided to sell chicken at a rate of 1800. Write the code to add this information to the data structure from part (c).

3. Shoprite changes some of its items over time, but it *always* carries those in the list above. The CEO of Shoprite Company wants a list of items that their stores *always* carry so they can ensure that these items are available for customers to buy at all times.

    a. Describe the data structure that would best fit this data.

    b. Given the data structure chosen in part a, create the collection of items that will be sent to the CEO.

```
always_in_stock = #your code here
```

The CEO is puzzled because one of his stores sent him two lists with different items. It turns out that the store sent in one list, but forgot some of the items, so they sent in a second list. The CEO thinks it would be much easier to have all these items on *ONE* list, but he is unsure how he can link two *tuples*.

    c. Given your knowledge of tuples, suggest a way that the CEO can combine the two lists.

4. Messi and Xavi are outraged at the prices in this store; they want to check around at a few other stores. For example, apples cost SOME_PRICE at the current store and SOME_LOWER_PRICE at another store.

    a. What data structure could they use to store different market prices associated with all the items on their grocery list?

    b. Given a sorted list of prices (e.g., $0.50, $1.25, $1.50), design a function that will insert another price into the list. Maintain the price order without re-sorting the entire list (hint: use binary search).

```
defbinary_insert(new_float,some_list_of_floats):
     #modifies the input list to include the new_float

     return
```

c. Write a function that returns the minimum amount of money that Messi and Xavi will have to spend on their grocery list.

```
def min_cost(grocery_list,item_to_price_list_dict):
#grocery_list is a list of strings (item names)

#item_to_price_list_dict is a dictionary with key-value
     # pairs as follows: the item name (strings) is the key
     # andthe list of prices (floats) at different grocery is
     # the value

return
```

5. **Challenge Problem: Lists and Queues**

Messi and Xavi lost their 2011 Champions League trophy while they were out celebrating last night. Now, they have to search through the streets of Barcelona to get it back. Starting from Bus Stop A (Messi's home) they want to check all the bus stops throughout the city.

An efficient search strategy will maintain one data structure for nodes that have already been visited (`seen`) and another for the nodes to be visited (`to_visit`). You are given a dictionary that maps each bus stop name (strings) to the list of strings representing adjacent bus stops. For example, the bus network in **Figure 1** would be represented with the following adjacency dictionary:

```
adjacency_dict =
{'A':['B','C'],'B':['A','D','E'],'C':['A','F','E'],'D':['B'],'E':['B
'],'F':['C']}
```

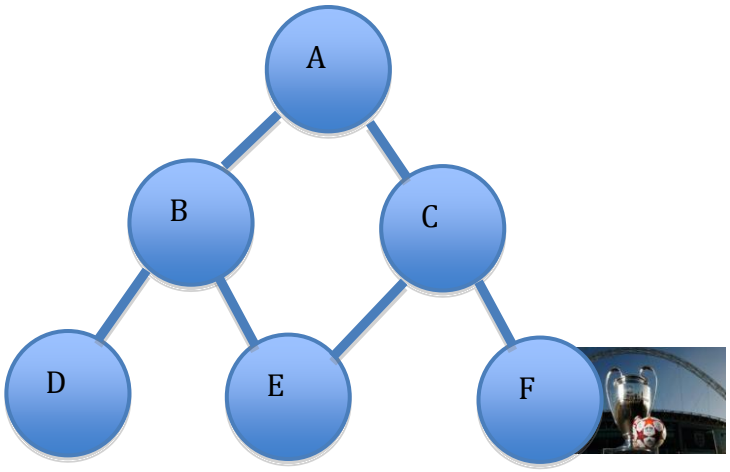

Figure 1: Sample layout of bus stops and connections with trophy at Bus Stop F

Messiand Xavi propose slightly different variations on the following strategy:

At the current bus stop, check for the trophy. If it is there, the quest ends (`return`)! Otherwise,remove the current bus stop from `to_visit`. Get the list of adjacent bus stops from `adjacency_dict`. Add each adjacent bus stop that is not in `seen` to the end of`to_visit`. At this point:

- Messi proposes that they proceed to the first item in `to_visit`.
- Xavi proposes that they proceed to the last item in `to_visit`.

a. What data structure should Messi and Xavi use to store whether or not a node has been `seen` (already visited in the search)? How does this data structure minimize lookup time?

b. Suppose Messi and Xavi decide to use lists to maintain the set of nodes `to_visit`. Whose algorithm will find the trophy fastest, and why?

c. Examine the python documentation on queues. Whose strategy would benefit most from using a queue in their `to_visit` data structure, and why?

Notes:

See documentation on queues at: http://docs.python.org/tutorial/datastructures.html#using-lists-as-queues)

Image credit for trophy picture in Figure 1:
Getty Images, http://www.uefa.com/uefachampionsleague/news/newsid=1633420.html