Hyperion Development

Task: Lists and Hashing Application

www.hyperiondev.com

Introduction

Welcome to the Lists and Hashing Application!
Overview:

This Task is aimed to ensure that you have a concrete understanding of Strings, List manipulation, and Hash Map manipulation, as these will be needed for upcoming more advanced tasks. In example.py, you will see examples that deal with operations that can be applied to elements in lists as well as hash maps. The Task also re-introduces Functions and how they can be used to compute certain values on list elements as well as dealing with hash maps (otherwise known as dictionaries).

-The Hyperion Team

You should now be familiar with the basics of lists and dictionaries. It should be noted that there are some list operations that may be very useful to you in upcoming tasks.

You should immediately recognising standard list indexing in the segment below. However a different technique is demonstrated in the segment below that. The first segment demonstrates indexing from the front of the list, whilst the second indexes from the back of the list.

```
>>> some_numbers = ['zero', 'one', 'two', 'three', 'four', 'five']
>>> some_numbers[0]
'zero'
>>> some_numbers[4]
'four'
>>> some_numbers[5]
'five'
```

```
>>> some_numbers[len(some_numbers) - 1]
'five'
>>> some_numbers[len(some_numbers) - 2]
'four'
>>> some_numbers[-1]
'five'
>>> some_numbers[-2]
'four'
>>> some_numbers[-6]
'zero'
```

Another useful way to get into parts of lists is using slicing. Here is another example to give you an idea what they can be used for:

```
>>> things = [0, 'Fred', 2, 'S.P.A.M.', 'Stocking', 42, "Jack", "Jill"]
>>> things[0]
0
>>> things[7]
'Jill'
>>> things[0:8]
[0, 'Fred', 2, 'S.P.A.M.', 'Stocking', 42, 'Jack', 'Jill']
>>> things[2:4]
[2, 'S.P.A.M.']
>>> things[4:7]
['Stocking', 42, 'Jack']
>>> things[1:5]
['Fred', 2, 'S.P.A.M.', 'Stocking']
```

Slicing is used to return part of a list. The slicing operator is in the form **things[first_index:last_index]**. Slicing cuts the list before the *first_index* and before the *last_index* and returns the parts in between. You can use both types of indexing:

```
>>> things[-4:-2]
['Stocking', 42]
>>> things[-4]
'Stocking'
>>> things[-4:6]
['Stocking', 42]
```

Another trick with slicing is the unspecified index. If the first index is not specified the beginning of the list is assumed. If the last index is not specified the whole rest of the list is assumed. Here is an example:

```
>>> things[:2]
[0, 'Fred']
>>> things[-2:]
['Jack', 'Jill']
>>> things[:3]
[0, 'Fred', 2]
>>> things[:-5]
[0, 'Fred', 2]
```

There are several ways to make a copy of a list. The simplest that works most of the time is the slice operator since it always makes a new list even if it is a slice of a whole list:

```
>>> a = [1, 2, 3]
>>> b = a[:]
>>> b[1] = 10
>>> print a
[1, 2, 3]
>>> print b
[1, 10, 3]
```

Taking the slice **[:]** creates a new copy of the list. However it only copies the outer list. Any sublist inside is still a references to the sublist in the original list. Therefore, when the list contains lists, the inner lists have to be copied as well. You could do that manually but Python already contains a module to do it. You use the *deepcopy* function of the *copy* module:

```
>>> import copy
>>> a = [[1, 2, 3], [4, 5, 6]]
>>> b = a[:]
>>> c = copy.deepcopy(a)
>>> b[0][1] = 10
>>> c[1][1] = 12
>>> print a
[[1, 10, 3], [4, 5, 6]]
>>> print b
[[1, 10, 3], [4, 5, 6]]
>>> print c
[[1, 2, 3], [4, 12, 6]]
```

You may now go through the example.py file for more information as well as tips for your next task. You should also go through the example programs of list and dictionary applications in the example programs folder.

Instructions

First read **example.py**, open it using Notepad++ (Right click the file and select 'Edit with Notepad++').

- example.py should help you understand some simple Python. Every task will have example code to help you get started. Make sure you read all of example.py and try your best to understand.
- You may run example.py to see the output. The instructions on how to do this are inside the file. Feel free to write and run your own example code before doing this Task to become more comfortable with Python.
- You are not required to read the entirety of Additional Reading.pdf, it is purely for extra reference.

Compulsory Task

Follow these steps:

- Create a new Python file in this folder called **cafe.py**.
- Create a list called menu, which should contain at least 4 items in the cafe.
- Next, create a hash map called stock, which should contain the stock value for each item on your menu.
- Create another hash map called price, which should contain the prices for each item on your menu.
- Next, create a function which will calculate the total stock worth in the cafe. You will need to remember to loop through the appropriate maps and lists to do this.
- Finally, print out the result of your function.

Still need help?

Just write your queries in your comments.txt file and your tutor will respond.

Task Statistics

Last update to task: 23/12/2015.
Authors: Riaz Moola and Jared Ping.
Main trainer: Umar Randeree.
Task Feedback link: Hyperion Development Feedback.