Hyperion Development

Task: Beginner Data Structures - Lists and Dictionaries

www.hyperiondev.com

Introduction

Welcome to The Beginner Data Structures - Lists and Dictionaries Task!

This Task is aimed to ensure that you have a concrete understanding of Strings and list manipulation, and also give you a little introduction to dictionaries. In example.py, you will see examples that deal with operations that can be applied to elements in lists as well as dictionaries. This Task also touches on functions and how they can be used to compute certain values on list elements as well as dictionaries (otherwise known as hash maps).

Connect with your mentor

CONNECT

Remember that with our courses - you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to www.hyperiondev.com/support to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

List Indexing

You should now be familiar with the basics of lists. You should immediately recognise standard list indexing in the segment below. 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'
```

Slicing Lists

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 then the beginning of the list is assumed. If the last index is not specified then 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]
```

Copying Lists

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 reference 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.

Python List Methods

There are many useful built-in List methods available for you to use. We have already looked at the 'append()' method.
Some other List methods can be found below:

- extend() - Adds all elements of a list to the another list
- insert() - Inserts an item at the defined index
- remove() - Removes an item from the list
- pop() - Removes and returns an element at the given index
- index() - Returns the index of the first matched item
- count() - Returns the count of number of items passed as an argument
- sort() - Sorts items in a list in ascending order
- reverse() - Reverses the order of items in the list

List Comprehension

List comprehension can be used to construct lists in an elegant and concise way. It is a powerful tool that will apply some operation to every element in a list, and then put the element into a new list. List comprehension consists of an expression followed by a for statement inside square brackets.

For Example:

```
numList = ['1', '5', '8', '14', '25', '31']
newNumListInts = [int(element) for element in numList]
```

For each element in numList, we are casting it to an Integer and putting it into a new list, newNumListInts.

Dictionaries

Dictionaries are used to store data and are very similar to Lists. However, Lists are ordered sets of elements, whereas dictionaries are unordered sets. Also, elements in dictionaries are accessed via keys and not via their positions like how lists are. When the key is known, you can use it to retrieve the value associated with it.

Creating a Dictionary

To create a dictionary simply place the items inside curly braces and separate them by commas. An item has a key and a value, which is expressed as a pair (key: value). Items in a dictionary can have a value of any datatype, however the key must be either a String or number and must be unique.

For Example:

```
int_key_dict = {1: 'apple',
                2: 'banana',
                3: 'orange'
                }
```

Accessing Elements from a Dictionary

While you might use indexing to access elements in a list, dictionaries use keys. Keys can be used to access values either by placing them inside square brackets [], such as with indices in lists, or with the get() method. However, if you use the get() method, it will return 'None' instead of 'KeyError', if the key is not found.

For Example:

```
profile_dict = {'name': 'Chris',
                'surname': 'Smith',
                'age': 28,
                'cell': '083 233 3242'
               }
print profile_dict['surname']    # prints out 'Smith'
print profile_dict.get('cell')   # prints out '083 233 3242'
```

Changing Elements in a Dictionary

We can add new items or change items using the assignment operator (=). If there is already a key present, the value get updated, else if there is no key a new key: value pair is added.

Dictionary Membership Test

You can test if a key is in a dictionary by using the keyword 'in'. You simply enter the key you want to test for membership, followed by the 'in' keyword and lastly the name of the dictionary. This will return either True or False, depending on whether the dictionary contains the key or not. The membership test is for keys only, not for values.

Instructions

Before you get started we strongly suggest you start using Notepad++ or IDLE to open all text files (.txt) and python files (.py). Do not use the normal Windows notepad as it will be much harder to read.

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

- 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. Feel free to write and run your own example code before doing the 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 1

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 dictionary called stock, which should contain the stock value for each item on your menu.
- Create another dictionary 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 dictionaries and lists to do this.
- Finally, print out the result of your function.

Compulsory Task 2

Follow these steps:

- Create a new Python file in this folder called hash.py
- Create a dictionary called countryMap, where the the KEYS are the name of a country (i.e. a String), and the VALUE for each key is the name of that country's capital city.
    - For Example:

```
countryMap = {
   'UnitedKingdom': 'London',
   'Sweden': 'Stockholm',
   'Canada': 'Ottawa',
}
```

- What does print countryMap['Sweden'] return?

Things to look out for:

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **Python or Notepad++** may not be installed correctly.
2. If you are not using Windows, please ask your mentor for alternative instructions.

Give your thoughts..

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes. Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.