Hyperion Development

Task: Beginner Data Structures - The List

www.hyperiondev.com

Introduction

Welcome to The Beginner Data Structures - The List Task!

In this Task you will learn about data structures in programming. A data structure is a specialised format for organising and storing data so that it may be used efficiently. General data structure types include arrays, lists, files, tables, trees and so on. Different data structures are suited to different kinds of applications and some are highly specialised to specific tasks. The most common data structure in Python is a list and this is what we will be focusing on.

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.

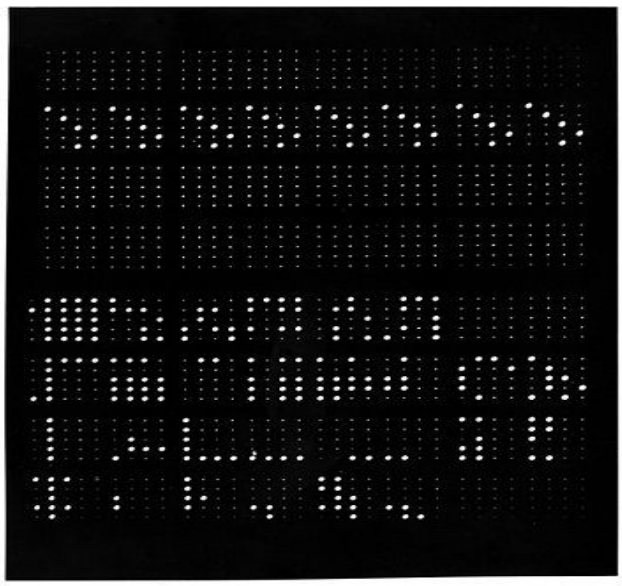
Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

A note from the Hyperion Team...

The Williams-Kilburn Tube

The Williams-Kilburn Tube(or Williams Tube) was an early computer storage device that used cathode-ray tubes to store bits(Os and 1s). This device was the first electronic memory that stored its bits as dots on a screen. On the face of the tube the dots represent the 1s and the spaces represent the Os as seen in the image below. The dots faded after less than a second and needed to be refreshed constantly.

The device was invented by Fred Williams at Manchester University in 1946, and was developed alongside Tom Kilburn. The device was used in the Manchester Mark I Computer and by IBM, only having 2k bits of memory.



To find out more about how this storage device worked, visit http://www.radiomuseum.org/forum/williams_kilburn_williams_kilburn_ram.html.

-The Hyperion Team

What are Lists?

The list is a datatype, just like strings, integers and floats. They are known as sequence types because they behave like an ordered collection of items. Sequence types are different from numeric types, such as integers and floats, because they are compound datatypes. Compound datatypes are made up of smaller pieces which, in the case of lists, are values of any datatype. Lists are written as a list of values (items), separated by commas, between square brackets []. The values or items in a list can be strings, integers, floats or even other lists or a mix of different types.

Creating a List

To create a list you simply need to give it a appropriate name and provide it with values. You enter the name of the list followed by the assignment operator and then the list of comma-separated values in between square brackets.

For Example:

```
stringList = ['John', 'Mary', 'Harry']
```

Indexing Lists

We are able to access all elements in a list using the index operator []. The index starts from 0 for the leftmost item, so a list having 10 elements will have indices from 0 to 9. Alternatively the index can begin with -1 for the index of the rightmost item, so a list having 10 elements will have indices from -10 to -1. This is very similar to how you would access individual characters in a String.

For Example:

```
petList = ['cat', 'dog', 'hamster', 'goldfish', 'parrot']
print petList[0]
```

This will print out the string 'cat'

Slicing Lists

Slicing a list enables you to extract multiple items. We can access a range of items in a list by using the slicing operator (colon). In order to slice a list you need to indicate a start and end position of the items you would like to access. You place these positions between the index operator [] and separate them with the colon. The item in the start position is included in the sliced list, while the item in the end position is not included.

For Example:

```
numList = [1, 4, 2, 7, 5, 9]
print numList[1:2]
```

Prints everything from the 1st to the 2nd element of list, NOT including the 2nd element.

Changing Elements in a Lists

You can use the assignment operator (=) to change single or multiple elements in a list .

For Example:

```
nameList = ['James', 'Molly', 'Chris', 'Peter', 'Kim']
nameList[2] = 'Tom'
```

'Chris' will be lost and replaced with 'Tom'

Adding Elements to a Lists

You can add an element to the end of a List using the 'append()' method.

For Example:

```
newList = [34, 35, 75, 'Coffee', 98.8]
newList.append('Tea')
```

Adds the String 'Tea' to the end of the List

Deleting Elements From a List

You can use the 'del' keyword to delete one or more items from a list or delete the list entirely.

For Example:

```
charList = ['P', 'y', 't', 'h', 'o', 'n']
del charList[3]
```

Deletes the single element 'h'

Looping Over Lists

What if you have a list of items and you want to do something to each item? Python is able to do this very quickly and much more easily than Java. You simply use a for loop to loop over every item in the list. It doesn't matter if the list had 100 items, 3 items or no items, the logic is the same and can be done in one line in Python.

For Example:

```
foodList = ['Pizza', 'Burger', 'Fries', 'Pasta', 'Salad']
for food in foodList:
        print food
```

This loop prints out every item in the list.

Checking if Something is in a List

You can simply use an if statement to check if a certain item is in a list.

For Example:

```
groceryList = ['Bread', 'Milk', 'Butter', 'Cheese', 'Cereal']

if 'Apples' in groceryList:
        print 'The item Apples was found in the list groceryList'
else:
        print 'The item Apples was not found in the list groceryList'
```

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

Follow these steps:

- Write a Python program called "John.py" that takes in a user's input as a String.

- While the String is not "John", add every String entered to a list until "John" is entered. Then print out the list. This program basically stores all incorrectly entered Strings in a list where "John" is the only correct String.

- Example program run (what should show up in the Python Console when you run it):

  Enter your name : <user enters Tim>
  Enter your name : <user enters Mark>
  Enter your name: <user enters John>
  Incorrect names: ['Tim', 'Mark']

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.