Hyperiondev

TASK

Beginner Data Structures — The List

Visit our website

Introduction

WELCOME TO 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.

Get in touch

Connect for support

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/portal** 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!

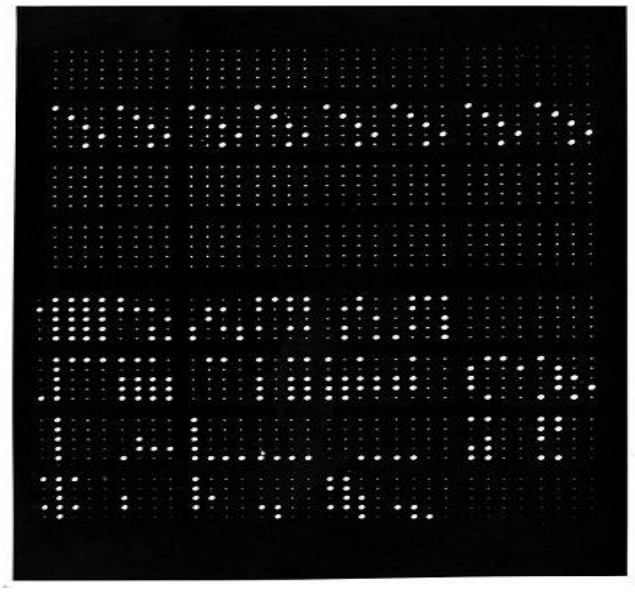
A note from the

HyperionDev Team

The Williams-Kilburn Tube

The Williams-Kilburn Tube (or the Williams Tube) was an early computer storage device that used cathode-ray tubes to store bits (0s 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 0s 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.

WHAT ARE LISTS?

The list is a data type, just like strings, integers and floats.

Lists 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 data types. Compound data types are made up of smaller pieces which, in the case of lists, are values of any data type.

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 an 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, placed in between square brackets.

For example:

```
string_list = ['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 containing 10 elements will have indices from 0 to 9. Alternatively, you can use negative indices to read the items in a list backwards. 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:

```
pet_list = ['cat', 'dog', 'hamster', 'goldfish', 'parrot']
print (pet_list[0])
```

This will print out the string, ‘cat’.

SLICING LISTS

Slicing a list enables you to extract multiple items from that list. 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 first indicate a start and end position for the items you would like to access. You would then 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:

```
num_list = [1, 4, 2, 7, 5, 9]
print (num_list[1:2])
```

This prints everything from the first to the second element of the list, NOT including the second element (we've discussed earlier why the end element of a range isn't included).

CHANGING ELEMENTS IN A LIST

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

For example:

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

Now, 'Chris' will be replaced with 'Tom' in the example above.

ADDING ELEMENTS TO A LIST

You can add an element to the end of a list using the **`append()`** method.

For example:

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

This 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:

```
char_list = ['P', 'y', 't', 'h', 'o', 'n']
del char_list[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? You simply use a for loop to loop over every item in the list. It doesn't matter if the list comprises 100 items, 3 items or no items, the logic is the same and can be done as shown in the following example.

For Example:

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

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

if 'Apples' in grocery_list:
        print ('The item Apples was found in the list grocery_list')
else:
        print ('The item Apples was not found in the list grocery_list')
```

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.

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']
```

Completed the task(s)?

Ask your mentor to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that 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.