Hyperiondev

Beginner Data Structures — The List

Visit our website

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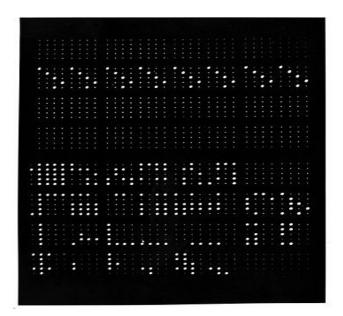
Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

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 continuously refreshed.

Fred Williams invented the device at Manchester University in 1946. It 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 used to store collections of other data types. Lists can store a collection 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, give it an appropriate name and provide it with values. 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:

```python
string_list = ['John', 'Mary', 'Harry']
```

A list can have any number of items and they may even be of different types. It can even have another list as an item.

Indexing Lists

We can access all elements in a list using the index operator ([]). Indices must be integers. The index starts from 0 for the leftmost item. Therefore, 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 -1 refers to the rightmost item in a list. Therefore, 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:

```python
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 (:). To slice a list, you need first to indicate a start and end position for the items you would like to access. 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:

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

Prints everything from the 1st to the 2nd element of the list, NOT including the 2nd element, so '[4]' will be printed.

Changing Elements in a List

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

For Example:

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

'Chris' will be replaced with 'Tom'

Adding Elements to a List

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

For Example:

```python
new_list = [34, 35, 75, 'Coffee', 98.8]
new_list.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:

```python
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? Use a for loop to iterate over every item in the list. It doesn't matter if the list comprised 100 items, 3 items or no items, the logic is the same and can be done as shown in the following example.

For Example:

```python
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:

```python
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')
```

Using the *range()* Function to Populate a List

The *range()* function is a special Python function that will automatically generate a list of integers within a specified range. You will have used to *range()* function when creating *for loops*. The *range()* function needs two integer values, a start number and a stop number. For the function *range(start index: end index)*, the **start index is included** and the **end index is not included**.

An example of how the *range()* function is used to add values to a list is shown below:

```
num_till_10 = range(0,10)
```

This will create a list of integers =[0,1,2,3,4,5,6,7,9] and store it in the variable 'num_till_10'.

Instructions

First read **example.py**. Open it using 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 1

Follow these steps:
- Create a new Python file in this folder called **list_types.py**.
- Imagine you want to store the names of three of your friends in a list of strings. Create a list variable called *friends_names*, and write the syntax to store the full names of three of your friends.
- Now, write statements to print out the name of the first friend, then the name of the last friend, and finally the length of the list.
- Now, define a list called *friends_ages* that stores the age of each of your friends in a corresponding manner, i.e., the first entry of this list is the age of the friend named first in the other list.

Compulsory Task 2

Follow these steps:
- Create a new Python file in this folder called **loop1000.py**.
- You are asked to print out all the numbers from 1 to 1000. Write 2 lines of code in your file to print out all numbers from 1 to 1000.
- Once you've got this to work, add logic inside your loop to only print out the numbers from 1 to 1000 that are even (i.e. divisible by 2). Remember the modulo command — i.e., %. 10%5 will give you the remainder of 10 divided by 5. 10 divided by 5 equals 2 with a remainder of 0. Hence, this

statement returns 0. Any even number is a number that can be divided by 2 with no remainder.

Compulsory Task 3

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']

Optional Bonus Task

Follow these steps:

- Create a new file in this folder called **loop_lists.py**.
- Inside it, define a list of strings of your 5 favourite movies.
- Now, loop over the list. For each item in the list, print out "Movie: " plus the movies name.
- Can you figure out how to print out Movie 1: <Movie 1's name>. Movie 2: ... etc?
- HINT: YOU WILL NEED TO LOOK UP the *enumerate* command in Python using a Google search. This command allows you to loop over a list retaining both the item at every position and its index (i.e. position in the list).

Rate us
Share 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.