**CS469 Data Structures and Algorithms**

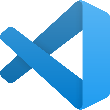
**HOS00D – Data Structures**

03/29/2023 Reviewed by Christopher Sharp

01/04/2024 Reviewed by Anh Nguyen

09/26/2024 Reviewed by Shahid Khan

School of Technology and Computing (STC) @City University of Seattle (CityU)

 A picture containing clock

Description automatically generated

**Before You Start**

* The Python HOS assignments (HOS00C and HOS00D) are OPTIONAL. If you already learned Python in other courses, feel free to skim the document.
* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.**If you are not sure what to do:

1. Consult the resources listed below.
2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Understand data structure in Python
* Understand the List Data Type and Tuples in Python
* Understand how to access, modify Dictionary and Set

**Data Structures**

1. Open VS code and then open the course’s folder

A screenshot of a cell phone

Description automatically generated

**Lists**

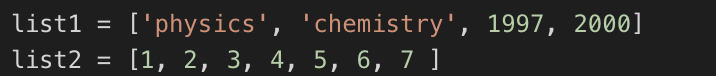
The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. The important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

|  |
| --- |
| list1 = ['physics', 'chemistry', 1997, 2000]  list2 = [1, 2, 3, 4, 5 ]  list3 = ["a", "b", "c", "d"] |

Python considers the first item in a list to be at position 0, not position 1.

1. Create a file called **List\_basic.py** and type the following code



**Accessing values**

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index.

1. In the same file**List\_basic.py** add the below code

A close up of a sign

Description automatically generated

1. Type the following in the terminal to check the output of the above code

>>> **python3 List\_basic.py**

A picture containing drawing

Description automatically generated

**Updating Lists**

1. In the same file**List\_basic.py** add the below code

A black sign with white text

Description automatically generated

1. Type the following in the terminal to check the output of the above code

**>>> python3 List\_basic.py**

A close up of a black background

Description automatically generated

**Adding element to a list- Appending**

The simplest way to add a new element to a list is to *append* the item to the list. When you append an item to a list, the new element is added to the end of the list. The append() method makes it easy to build lists dynamically.

1. In the same file**List\_basic.py** add the below code

A close up of text on a black background

Description automatically generated

1. Type the following in the terminal to check the output of the above code

**>>> python3 List\_basic.py**

A picture containing drawing

Description automatically generated

**Adding element to a list- Inserting elements**

You can add a new element at any position in your list by using the insert() method. You do this by specifying the index of the new element and the value of the new item.

1. In the same file**List\_basic.py** add the below code

A close up of text on a black background

Description automatically generated

1. Type the following in the terminal to check the output of the above code

**>>> python3 List\_basic.py**

A close up of a logo

Description automatically generated

**Removing element**

There are 3 ways to remove a element from a list.

1. If you know the position of the item you want to remove from a list, you can use the del statement.
2. The pop() method removes the last item in a list, but it lets you work with that item after removing it. The term *pop* comes from thinking of a list as a stack of items and popping one item off the top of the stack. In this analogy, the top of a stack corresponds to the end of a list. You can use pop() to remove an item from any position in a list by including the index of the item you want to remove in parentheses.
3. If you only know the value of the item you want to remove, you can use the remove() method.

1. Create a file **List\_remove.py** and type the following code.

A picture containing drawing

Description automatically generated

1. Type the following in the terminal to check the output of the above code

>>> python3 List\_remove.py



The del used at index 1 deletes the value Yamaha. You can no longer access the value that was removed from the list after the del statement is used.

1. Add the below code in the same file

A picture containing drawing

Description automatically generated

1. Type the following in the terminal to check the output of the above code

   >>>python3 List\_remove.py

A picture containing drawing

Description automatically generated

At pop(), a value from the list and store that value in the variable popped\_motorcycle. We print the list to show that a value has been removed from the list. Then we print the popped value to prove that we still have access to the value that was removed.

The output shows that the value ‘suzuki’ was removed from the end of the list and is now assigned to the variable popped\_motorcycle.

1. Add the below code in the same file

A picture containing food, drawing

Description automatically generated

1. Type the following in the terminal to check the output of the above code

>>>**python3 List\_remove.py**



The code removes “ducati” from the list. You can use the remove() method to work with a value that's being removed from a list.

**Using Loop in lists**

1. Create a file **replaceNegative.py** and enter the following code. The program replaces any negative numbers in the list with positive ones.

A black sign with white text

Description automatically generated

1. Type the following in the terminal to check the output of the above code

>>> **python3 replaceNegative.py**



The range() function causes Python to start counting at 0 and it stops when it reaches the length of the list. The len() function calculates the length of the list. The i consist of the range index and it is used for navigating throughout the list. If the value in the list is less than 0, that is negative value then that value is converted to absolute value and returned back to list.

**Tuples**

A tuple is a sequence of immutable Python objects. The tuple data type is similar to list, except tuples are typed with parentheses and cannot be modified, appended, or removed. The following is how the tuples look like.

tup1 = ('physics', 'chemistry', 1997, 2000);

1. Create a**tuple.py** file and define a tuple with multiple data types and try to modify the grade by assigning 4.0 value to item at index 1 which is 2.0

A black sign with white text

Description automatically generated

Note that you will get error message like because Tuples can’t be modified unlike List

1. Type the following in the terminal to check the output of the above code

>>> **python3 tuple.py**

A picture containing food

Description automatically generated

1. Create a **sortTuple.py** with the following code to sort tuples by its first value.

A screenshot of a cell phone

Description automatically generated

1. Type the following in the terminal to check the output of the above code

>>> **python3 sortTuple.py**

Output will be sorted.



“**def**” is the function and it will be explained in the later chapters. the sort function takes in a keyword argument called key. What key does is it provides a way to specify a function that returns what you would like your items sorted by. The function gets an "invisible" argument passed to it that represents an item in the list and returns a value that you would like to be the item's "key" for sorting.

**Dictionary**

A dictionary is a collection of many values. Unlike indices in list, indices for dictionaries can use many different data types, it’s called keys, and a key with its associated value is called a key-value pair. Each key is separated from its value by a colon (:), the items are separated by commas and the whole thing is enclosed in curly braces. Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

In the Dictionary functions like accessing values, updating the dictionary, adding new pairs and deleting the elements can be performed.

1. Create a file **dictionary.py** and type the below code. This program describes how to access, update, add and delete entries in dictionary.

A screen shot of a smart phone

Description automatically generated

1. In the terminal type the following to check the output for the above code

>>> **python3 dictionary.py**

A picture containing food, drawing

Description automatically generated

Note: If the key you ask for doesn't exist, you'll get an error. More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins.

Dictionary is unordered. Thus, if you compare two dictionaries with the same content but not the same order, it will return true.

1. Let’s find out what will happen when we compare two lists and two dictionaries with similar values but different order. Create a file **compare.py** and type the code below.

A screenshot of a cell phone

Description automatically generated

1. In the terminal type **python3 compare.py** to see the output for the above code.

A picture containing drawing

Description automatically generated

The output on the terminal should be **False** and **True** respectively, since order matters in List but not in Dictionary if it contains the same values.

**Looping in Dictionary**

A single Python dictionary can contain just a few key-value pairs or millions of pairs. Python lets you loop through a dictionary. You can loop through all a dictionary's key-value pairs, through its keys, or through its values.

1. Create a file **dict\_for.py** and type the code below. This program explains how multiple dictionaries are combined and used inside loop for accessing the key and value.

A screenshot of a cell phone

Description automatically generated

Type **python3 dict\_for.py** in the terminal for checking the output.

A close up of a sign

Description automatically generated

Note: For accessing only keys, use .keys() and for values use .values() in the loop.

1. Let’s create a program to find the richest man from the dictionary and save it as **richest.py**

A screenshot of a cell phone

Description automatically generated

Type “python3 dict\_for.py” in the terminal. The result would show Stark is the richest man on earth.



Note: The usage of income.get is similar to income.get(key). In the case above, income.get is just a reference to the income.get(key). The end=’ ‘ is just to say that you want a space after the end of the statement instead of a new line character.

1. One useful thing that you can get from Dictionary is counting how many times each character appears in the sentence. Make a program called **count\_character.py**and type the following.

A screenshot of text

Description automatically generated

Type **python3 count\_character.py** in the terminal.

The output would look like this with pprint(). While the print() will show result in horizontal format which is difficult to read. The count = {} is the empty dictionary.

A close up of a logo

Description automatically generated

**Set**

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

Example:

thisset = {"apple", "banana", "cherry"}

* **Accessing items:** You cannot access items in a set by referring to an index, since sets are unordered the items have no index. But you can loop through the set items using a for loop or ask if a specified value is present in a set, by using the in keyword.
* **Modify items:**Once a set is created, you cannot change its items, but you can add new items.
* **Add items:**To add one item to a set use the add() method. To add more than one item to a set use the update() method.
* **Remove items:**To remove an item in a set, use the remove(), or the discard() method.

Note: If the item to remove does not exist, remove() will raise an error and discard() will NOT raise an error. You can also use the pop(), method to remove an item, but this method will remove the last item. Remember that sets are unordered, so you will not know what item gets removed. The return value of the pop() method is the removed item.

**Joining two sets:**There are several ways to join two or more sets in Python. You can use the union() method that returns a new set containing all items from both sets, or the update() method that inserts all the items from one set into another.

**Constructor:**It is also possible to use the set() constructor to make a set.

1. Create a file **Set.py** and type the following program for understanding how joining of two sets work and constructor.

A screenshot of a cell phone

Description automatically generated

Type **python3 Set.py** in the terminal for the output.

A black sign with white text

Description automatically generated

Note: Both union() and update() will exclude any duplicate items.

**If you want to learn more about Python, please visit the following website and complete the tutorial:**

[**https://www.w3schools.com/python/**](https://www.w3schools.com/python/)