**CS506 Programming for Computing**

**H0S02 Lists, Tuples, Dictionary, Set**

03/13/2025 Reviewed by Clark Ngo

04/10/2025 Reviewed by Naveena Moddu

04/10/2025 Reviewed by Ashwin Paturi

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

**Resources**

* Python crash course: a hands-on, project-based introduction to programming: Matthes, E. (2019): [Available online link](https://cityu.alma.exlibrisgroup.com/discovery/openurl?institution=01CITYUNIV_INST&rfr_id=info:sid%2Fsummon&rft_dat=ie%3D5152833400004251,language%3DEN&svc_dat=CTO&u.ignore_date_coverage=true&vid=01CITYUNIV_INST:Services)

**Learning Outcomes**

* **Section 1: Preparing your environment**
* **Section 2: Lists**
* **Section 3: Adding elements using ‘Append’**
* **Section 4: Inserting elements**
* **Section 5: Removing element**
* **Section 6: Using Loop in lists**
* **Section 7: Tuples**
* **Section 8: Dictionary Data Type**
* **Section 9: Looping in Dictionary**
* **Section 10: Set**

**Learning Outcomes**

**Notes**

If the **python** command is not functioning, try using **python3** instead.

**Section 1: Preparing your environment**

Get started with your virtual environment here: <https://cityuseattle.github.io/docs/git/github_codepsace/#codespaces>

**Section 2: Lists**

The list is the 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.

**Creating, accessing and updating values**

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

A black sign with white text

Description automatically generated

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. Python considers the first item in a list to be at position 0, not position 1

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

**>>> python List\_basic.py**

A close up of a black background

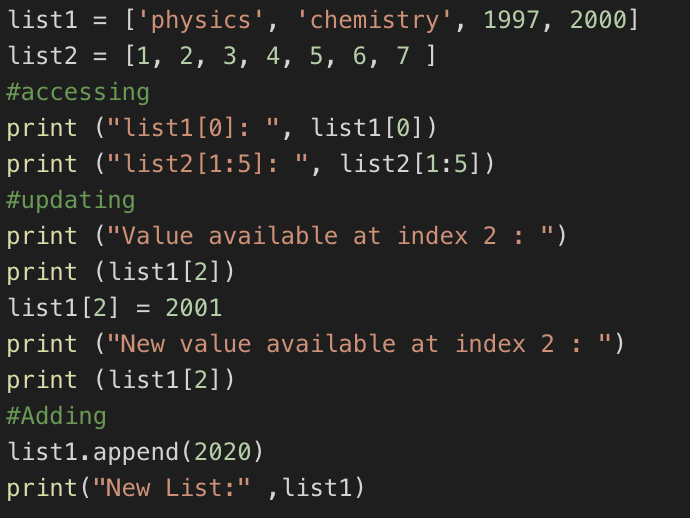
Description automatically generated

**Challenge 1: Explain the code above by commenting each line of code**

**Section 3: Adding elements using ‘Append’**

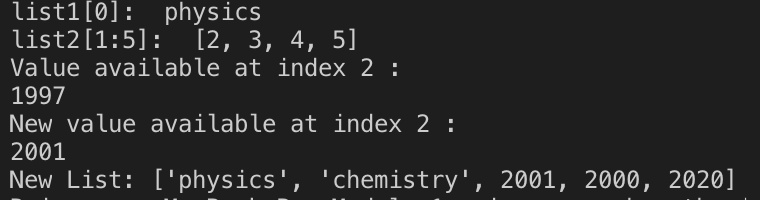
The simplest way to add a new element to a list is to *append* the item to the list. When you append an item to a list, the new element is added to the end of the list. The append() method makes it easy to build lists dynamically.

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



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

**>>> python List\_basic.py**



**Section 4: 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

**>>> python List\_basic.py**

A close up of a logo

Description automatically generated

**Section 5. Removing element**

There are 3 ways to remove an 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.
4. 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

>>> python 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 screenshot of a cell phone

Description automatically generated

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

>>>python 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

>>>python 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.

**Section 6: Using Loop in lists**

1. Let’s use a for loop to print out all the elements of a list. Create a file **replaceNegative.py** and enter the following code:

Text

Description automatically generated

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 consists 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 a negative value then that value is converted to absolute value and returned to list.

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

>>> python replaceNegative.py

A picture containing meter

Description automatically generated

**Challenge 2: Edit the program to replace any negative numbers in the list with positive ones. Expected result:**



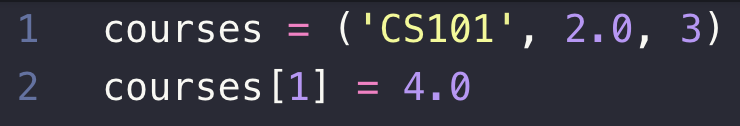
*Hint: you can use abs() function to turn negative numbers to positive*

**Section 7: Tuples**

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

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

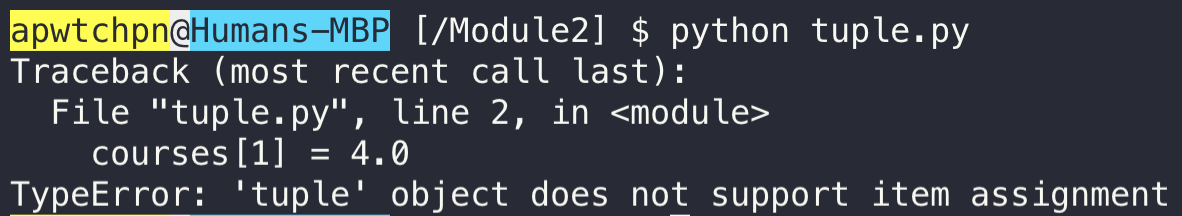
1. Create a**tuple.py** file and define a tuple with multiple data types and try to modify the grade by assigning 4.0 value to item at index 1 which is 2.0



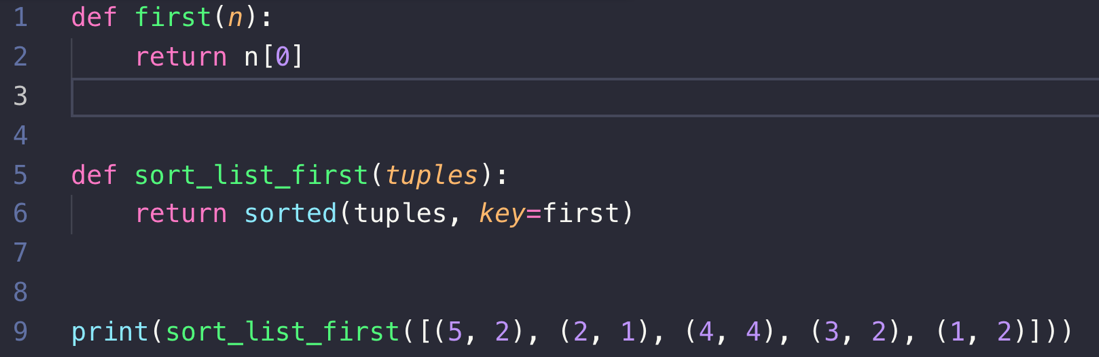
Note that you will get error message like because Tuples can’t be modified, **unlike** List

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

>>> python tuple.py



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



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

>>> python 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.

**Section 8: Dictionary Data Type**

A dictionary is a collection of many values. Unlike List, indexes 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 in a 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

>>> python dictionary.py

A picture containing 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 dictionary 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. Type the following in the terminal to check the output of the above code

>>> python compare.py

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 as long as it contains the same values.

**Section 9: 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 **python dict\_for.py** in the terminal for checking the output.

A close up of a logo

Description automatically generated

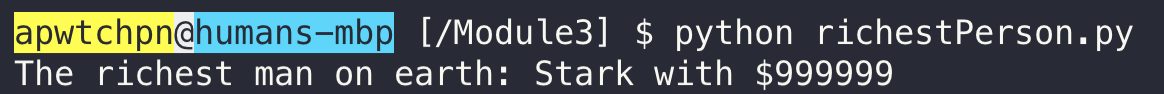
Note: For accessing only keys, use .keys() and for values use .values() in the loop.

2)Let’s Create a program to find the richest man from the dictionary and save it as **richest.py**

Text

Description automatically generated

Type “python richest.py” in the terminal. The result would show Stark is the richest man on earth.



**Challenge 3: Add to the code above to show the person with the lowest income.**

**Section 10: 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.

**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 **python 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.*

**Push your work to GitHub**

Follow instructions here: <https://cityuseattle.github.io/docs/git/codespaces_submission/>