**CS 11A: Technology & Computing Components I**

**H0S05 Dictionary, Set and Functions**

10/10/2020 Developed by Kim Nguyen

02/04/2022 Updated by Amanda Vaughan

07/20/2023 Reviewed by Asia Shvets

10/16/2023 Reviewed by Ajay Naik

01/30/2024 Reviewed by Chen Yuan

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.
* Version numbers may not match the most current version at the time of writing. If given the option to choose between the stable release (long-term support) or the most recent, please select the ****stable release**** rather than the beta-testing version.
* 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.
* If you are not familiar with a terminal, command line, and bash scripts, check out this video: <https://youtu.be/Dp7uw9c6QH8>
* All the steps and concepts in this tutorial are from references, so if you encounter problems, please ****try to read and compare the references to solve the problem****. If you still can't solve the problem, please contact your course TA.
* ****Avoid copy-pasting code from the book or the GitHub repository****. Instead, type out the code yourself. Resort to copy-pasting only when you are stuck and find things not working as expected.
* 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.

**Learning Outcomes**

Students will be able to:

* Understand the Dictionary data type and Set data type in Python
* Write a Python program that can manipulate data stored in Dictionary and Set.

**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)

**Preparation**

1. It is recommended that you create a folder named the current module number. You may store the folders wherever it is most convenient for you on your computer and can select a different naming convention if it makes more sense to you.
2. In Visual Studio Code, open the Module05 folder you just created.

A screenshot of a cell phone

Description automatically generated

**Dictionary Data Type**

A dictionary is a collection of many values. Unlike Lists, 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 pair and deleting the elements can be performed.

**Notes**

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

1. Under Module05 create a file **dictionary.py** and type the code below. This program describes how to access, update, add and delete 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 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 **python 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 as long as 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 **python dict\_for.py** in the terminal for checking the output.

A close up of a logo

Description automatically generated

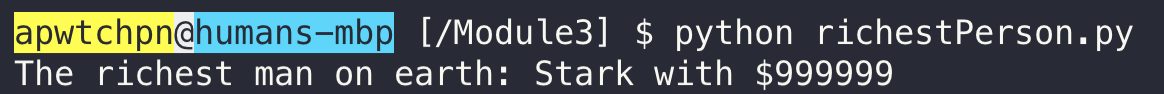
Note: For accessing only keys, use .keys() and for values use .values() in the loop.

1. 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 1: Add to the code above to show the person with the lowest income.**

**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.*

A screenshot of a cell phone

Description automatically generated

**Python Functions**

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

A function can be written to as a set of statements that take inputs and perform specific computation and produces output.

**Syntax:**

def functionname( parameters ):

code

return [expression]

**Note:** The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None. It is not mandatory to use a return statement.

A function can process some data and then return a value or set of values. The value the function returns is called a return value. The return statement takes a value from inside a function and sends it back to the line that called the function. Return values allow you to move much of your program's grunt work into functions, which can simplify the body of your program.

1. Create a file **print\_numbers.py** and type the following code.

A screenshot of a cell phone

Description automatically generated

In the terminal type **python print\_numbers.py** to check the output

A picture containing object, clock

Description automatically generated

A function doesn’t always have to return something, sometime, a function just prints something out, it is the same as return None. Example as below (You don’t have to write the below code):

A close up of a screen

Description automatically generated

Result:



**Passing Arguments**

Because a function definition can have multiple parameters, a function call may need multiple arguments. You can pass arguments to your functions in a number of ways.

Positional arguments: which need to be in the same order the parameters were written

Keyword arguments: where each argument consists of a variable name and a value

Arbitrary Number of Arguments: Sometimes you won't know ahead of time how many arguments a function needs to accept. In that case arbitrary can be used.

1. Create a file **positional.py** and type the following code.

A screenshot of a cell phone

Description automatically generated

In the terminal type **python positional.py** to check the output

A picture containing drawing

Description automatically generated

The definition shows that this function needs a type of animal and the animal's name. When we call describe\_pet(), we need to provide an animal type and a name, in that order. For example, in the function call, the argument ‘hamster’ is assigned to the parameter animal\_type and the argument ‘harry’ is assigned to the parameter pet\_name. In the function body, these two parameters are used to display information about the pet being described.

##### The order matters in positional arguments. The parameters specified in the function definition have to be in the same order as how the function is called. There won’t be any error showing when running the code, but the result might not be as you expected it to be.

**Note**: title() method is a built-in Python string method that returns a new string with the first character of each word capitalized, and all other characters in lowercase.

**Challenge:**

**Add to the code above to receive the following result:**

A picture containing drawing

Description automatically generated

**Arbitrary number of arguments**

Sometimes you'll want to accept an arbitrary number of arguments, but you won't know ahead of time what kind of information will be passed to the function. In this case, you can write functions that accept as many key-value pairs as the calling statement provides.

1. Create a file **arbitrary\_dict.py** and type the following code

A picture containing drawing

Description automatically generated

In the terminal type **python arbitrary\_dict.py** to check the output



The “menu” is the definition where user can pass the item and quantity, then it allows the user to pass in as many name-value pairs as they want. The double asterisks before the parameter \*\*restaurant cause Python to create an empty dictionary called “restaurant” and pack whatever name-value pairs it receives into this dictionary.

**Push your work to GitHub**

Access the Terminal at the location where you've generated the Python files.

Run the following commands to push your work to the GitHub repository:

git add --all

git commit -m "Submission for Module 5- FIRST and LAST Name"

git push origin master