Sara O’Neal

May 11, 2021

Foundations of Python Programming (IT FDN 110 A)

Assignment 05

GitHub: <https://github.com/sarao3/IntroToProg-Python>

# Dictionaries in Python

## Introduction

The purpose of this paper is to provide an overview of how to complete the fifth assignment in the Foundations of Python Programming course, which covers how to work with dictionaries in Python.

## Difference between a List and Dictionary

Lists use an index (numeric) subscript and are denoted with square [] brackets. Dictionaries use key (character) subscripts and use curly {} brackets.

## Working with Dictionaries and Files

The following steps outline how to read two columns of data from a “ToDo list” text file, load the columns into a Python Dictionary object, add the dictionary rows to a Python List object to create a table, add and remove items and finally save the data back to the “ToDo List” text file.

1. Open a blank text file and add two different tasks and the priority of each, as shown in Figure 1. Save the text file in the same folder as the Python script you will be creating.

**Text

Description automatically generated with medium confidence**

**Figure 1:** Two different tasks and the priority of each added to a “ToDo List” text file

1. Open a new project in PyCharm to write the Python script. Set the relative path in PyCharm to the location where Assignment05.py script and “ToDo List” text file are saved.
2. In PyCharm, add a header that provides the title of the code, author of the code, a description of what the code does and a change log of who, what and when (Figure 2).

Text

Description automatically generated

**Figure 2:** Header for Assignment05 shown within the PyCharm IDE (Integrated Development Environment)

1. Declare the variables to be used throughout the script, as shown in Figure 3. This helps improve code readability.

Text

Description automatically generated

**Figure 3:** Bothstring and list variables are utilized in the Assignment05.py script

1. Load the data from the text file into memory using the “objFile” variable as a file handle and “strFile” as the variable to call up the “ToDo List” text file. Once the file is open, use the “row.split()” statement to return a list object. The list of data is then loaded into a dictionary object and retrieved from the list using an index of “0” for task and index of “1” for priority. “Strip()” is added at the end of the “dicRow” variable to remove a carriage return. The dictionary object is then appended to a data table (list of rows) called “lstTable.” The text file closes.

Graphical user interface, text, application, email

Description automatically generated

**Figure 4:** Load data from “ToDo List” text file into a Python list of dictionary rows

1. If the user selects option 1 from the menu, the data in the “lstTable” variable will show. The “for row in lstTable” will loop through all rows in the table (lstTable) and print out the value in the first and second column per the dictionary keys. Figure 5 shows the script to display the current items in the table.

Graphical user interface, text, application

Description automatically generated

**Figure 5:** Script to show the current data in the “lstTable” variable

1. If the user selects option 2 from the menu, the user can add another item to the “lstTable” variable. The user is prompted to input a task and priority. This dictionary object data is then appended to the “lstTable.” If the user enters “y” to continue adding items, the program will loop through this section of code until it breaks when the user enters “n”.

Graphical user interface, text, application

Description automatically generated

**Figure 6:** The script to add a new item to the “lstTable” variable

1. If the user selects option 3 to remove an item from the table, then the program will loop through each row looking for the task name that the user entered and would like to remove. Conditional “if” statements are used to determine if the task is in the table. If the task is not present, a message will be displayed to the user that the row or task was not found.

Graphical user interface, text, application

Description automatically generated

**Figure 7:** The script to remove an item from the “lstTable” variable

1. If the user selects option 4, the “ToDo List” text file is opened back up. For each row in the table (“lstTable”), the script writes the task and priority data into the text file. A new line is added after each task and priority, as shown by the “\n” in the script. Once all rows have been processed, the text file is closed.

Graphical user interface, text, application

Description automatically generated

**Figure 8:** The script to save data to the “ToDo List” text file

1. The data is written to the ToDoList.txt file using PyCharm.

Graphical user interface, text

Description automatically generated

**Figure 9:** Outcome of the Python script written to a text file for Assignment 05 using PyCharm

1. If the user selects option 5, the program will break and end if the user enters “y”. Otherwise, the user will be prompted to enter a number 1 through 5 from the menu.

Graphical user interface, text

Description automatically generated

**Figure 10:** The script to exit the program or return to the main menu

## Summary

This week, we learned how to work with dictionaries in Python. Programmers may find dictionaries easier to work with compared to other sequences because it replaces index subscripts with key (character) subscripts.