Ryan L. Blake

10 August 2021

IT FDN 100

Assignment 06

https://github.com/rblake50/IntroToProg-Python

**Functions and Code-Building**

# Introduction

Re-use of code is practical and helpful to reduce the amount of text in a script and allow readers to interpret code in a logical manner. Classes and functions offer a built-in way to re-use code across a script. In industry, it is rare that a code be written by a single person. This elevates the importance of both writing and reading intelligible code to foster collaboration. As such, building on someone else’s code represents an important topic. This document explores the topics of building on someone else’s code while taking advantage of classes and functions.

# Topics

* Classes and functions
* Building on someone else’s code

# Summary

Quite simply, functions take inputs (called parameters in the Python world) and then… do something. They “return” something at the end of the function, which can be a single variable or multiple. Functions are best used for operations that are repeated multiple times throughout a code. This allows the functions to be called and potentially conserve lines of code.

The basic function syntax is written below:

def some\_function(parameter1, parameter2):

some\_result = parameter1 + parameter2 # do something

return some\_result

Beyond this simple example, there are more possibilities. One important piece of functions involves what are called docstrings.

Text

Description automatically generated

Figure : docstring example (boxed in green) with function, parameter, and return descriptions.

Docstrings are strongly recommended, as they can help explain the code to another developer.

Classes can be thought of as parents to children like functions. Classes help segment code and keep things organized in a hierarchical manner.

Text

Description automatically generated

Figure 2: Class creation.

Intuitively, calling something like Processor.read\_data\_from\_file(strFileName, lstRows) would go within the class Processor and call the function read\_data\_from\_file. In the assignment, two classes are used: Processor and IO. Theses names are sensible, and the corresponding functions to processing and input/output fall within each class.

Most useful applications are complex; they would take years for a single person to write individually. Beyond production of a code, maintenance may be an even more important factor… What if the original developer takes on a new role? Thus, collaborating on code is a critical skill for developers.

This assignment involved completing seed code to accomplish the same task as Assignment 05 but in a more efficient manner. In completing the exercise, I observed a few findings. First, consistency with variable nomenclature is key. In following expected Python guidelines, “local” variables do not typically include the variable type shorthand (like str, int, or bool) but just the name instead. But when building on someone else’s code, you need to understand the conventions they used and how they appear in the code you are writing. Return variables in functions should be consistent with the variables within the function.

Second, comments are critical for assigning tasks. #TO DO appeared frequently in the seed code, which made it very easy to understand where development should occur. Also, pass was used in functions to allow the seed code to still run (albeit without logical success) for debugging.

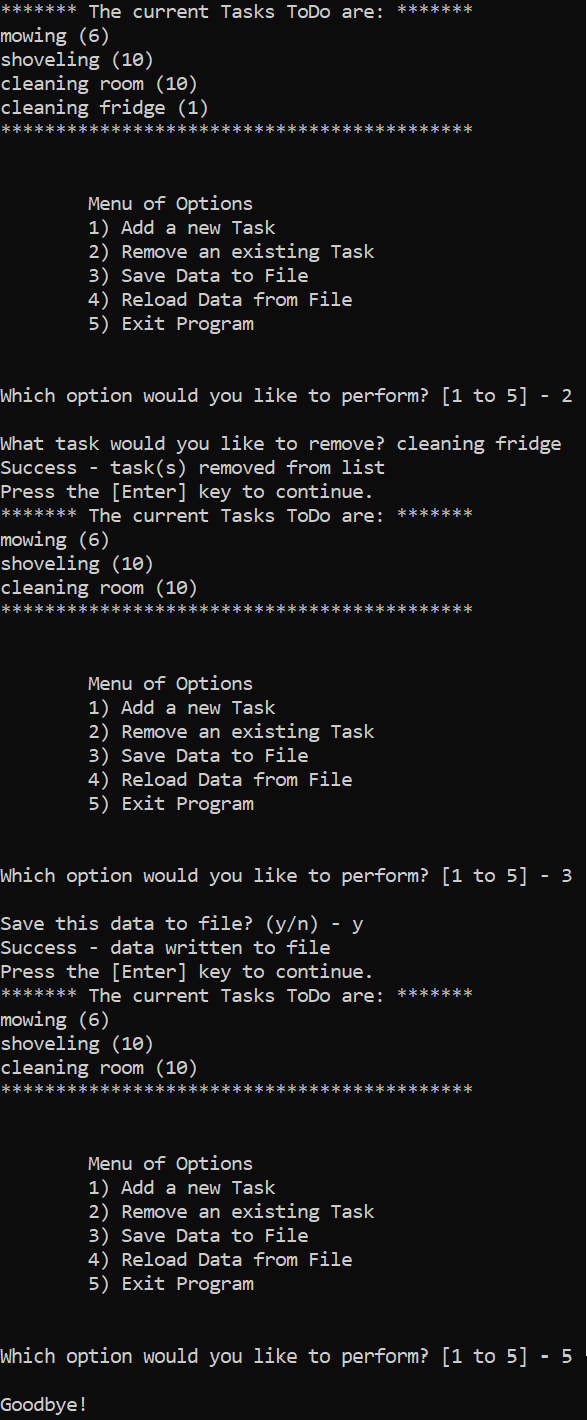
The code outputs are copied below:

|  |
| --- |
| C:\\_PythonClass\Assignment06\venv\Scripts\python.exe C:/\_PythonClass/Assignment06/Assigment06\_Starter.py  \*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*  mowing (6)  shoveling (10)  cleaning room (10)  cleaning fridge (1)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Menu of Options  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Exit Program    Which option would you like to perform? [1 to 5] - 1  What task would you like to add? doing homework  What priority does the task have? 4  Success - task and priority added to list  Press the [Enter] key to continue.  \*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*  mowing (6)  shoveling (10)  cleaning room (10)  cleaning fridge (1)  doing homework (4)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Menu of Options  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Exit Program    Which option would you like to perform? [1 to 5] - 2  What task would you like to remove? shoveling  Success - task(s) removed from list  Press the [Enter] key to continue.  \*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*  mowing (6)  cleaning room (10)  cleaning fridge (1)  doing homework (4)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Menu of Options  1) Add a new Task  2) Remove an existing Task  3) Save Data to File  4) Reload Data from File  5) Exit Program    Which option would you like to perform? [1 to 5] - 5  Goodbye! |

Also in the command shell:

A picture containing text, outdoor, screenshot

Description automatically generated



The code acts as intended.

# Appendix: Python Script

*# ---------------------------------------------------------------------------- #  
# Title: Assignment 06  
# Description: Working with functions in a class,  
# When the program starts, load each "row" of data  
# in "ToDoToDoList.txt" into a python Dictionary.  
# Add the each dictionary "row" to a python list "table"  
# ChangeLog (Who,When,What):  
# RRoot,1.1.2030,Created started script  
# RRoot,1.1.2030,Added code to complete assignment 5  
# rblake50,08.09.2021,Modified code to complete assignment 6  
# ---------------------------------------------------------------------------- #  
  
# Data ---------------------------------------------------------------------- #  
# Declare variables and constants*strFileName = **"ToDoFile.txt"** *# The name of the data file*objFile = **None** *# An object that represents a file*dicRow = {} *# A row of data separated into elements of a dictionary {Task,Priority}*lstTable = [] *# A list that acts as a 'table' of rows*strChoice = **""** *# Captures the user option selection*strTask = **""** *# Captures the user task data*strPriority = **""** *# Captures the user priority data*strStatus = **""** *# Captures the status of an processing functions  
  
# Processing --------------------------------------------------------------- #***class** Processor:  
 *""" Performs Processing tasks """* @staticmethod  
 **def** read\_data\_from\_file(file\_name, list\_of\_rows):  
 *""" Reads data from a file into a list of dictionary rows* **:param** *file\_name: (string) with name of file:* **:param** *list\_of\_rows: (list) you want filled with file data:* **:return***: (list) of dictionary rows  
 """* list\_of\_rows.clear() *# clear current data* file = open(file\_name, **"r"**)  
 **for** line **in** file:  
 task, priority = line.split(**","**)  
 row = {**"Task"**: task.strip(), **"Priority"**: priority.strip()}  
 list\_of\_rows.append(row)  
 file.close()  
 **return** list\_of\_rows, **'Success - data read from file'** @staticmethod  
 **def** add\_data\_to\_list(task, priority, list\_of\_rows):  
 *"""Adds a Task and Priority to the list of dictionary rows* **:param** *task: (string) of task to add* **:param** *priority: (string) of priority to add* **:param** *list\_of\_rows: (list) you want to add to* **:return***: (list) of dictionary rows  
 """* dicRow = {**"Task"**:task,**"Priority"**:priority}  
 list\_of\_rows.append(dicRow)  
 **return** list\_of\_rows, **'Success - task and priority added to list'** @staticmethod  
 **def** remove\_data\_from\_list(task, list\_of\_rows):  
 *"""Removes a desired task from the list* **:param** *task: (string) of task to remove* **:param** *list\_of\_rows: (list) of dictionary rows* **:return***: (list) of dictionary rows, (string) message  
 """* boolFound = **False  
 for** item **in** list\_of\_rows:  
 **if** task == item[**"Task"**]:  
 boolFound = **True** list\_of\_rows.remove(item)  
 strMsg = **'Success - task(s) removed from list'  
 if** boolFound == **False**:  
 strMsg = **'Failure - task not found'  
 return** list\_of\_rows, strMsg  
  
 @staticmethod  
 **def** write\_data\_to\_file(file\_name, list\_of\_rows):  
 *"""Writes data to text file* **:param***: file\_name: (object) of text file* **:param***: list\_of\_rows: (list) of dictionary rows  
 return: (list) of dictionary rows, (string) message  
 """* objFileText = open(file\_name,**"w"**)  
 **for** item **in** list\_of\_rows:  
 objFileText.write(item[**"Task"**]+**", "**+item[**"Priority"**]+**"\n"**)  
 objFileText.close()  
 **return** list\_of\_rows, **'Success - data written to file'***# Presentation (Input/Output) -------------------------------------------- #***class** IO:  
 *""" Performs Input and Output tasks """* @staticmethod  
 **def** print\_menu\_Tasks():  
 *""" Display a menu of choices to the user* **:return***: nothing  
 """* print(**'''  
 Menu of Options  
 1) Add a new Task  
 2) Remove an existing Task  
 3) Save Data to File   
 4) Reload Data from File  
 5) Exit Program  
 '''**)  
 print() *# Add an extra line for looks* @staticmethod  
 **def** input\_menu\_choice():  
 *""" Gets the menu choice from a user* **:return***: string  
 """* choice = str(input(**"Which option would you like to perform? [1 to 5] - "**)).strip()  
 print() *# Add an extra line for looks* **return** choice  
  
 @staticmethod  
 **def** print\_current\_Tasks\_in\_list(list\_of\_rows):  
 *""" Shows the current Tasks in the list of dictionaries rows* **:param** *list\_of\_rows: (list) of rows you want to display* **:return***: nothing  
 """* print(**"\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*"**)  
 **for** row **in** list\_of\_rows:  
 print(row[**"Task"**] + **" ("** + row[**"Priority"**] + **")"**)  
 print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
 print() *# Add an extra line for looks* @staticmethod  
 **def** input\_yes\_no\_choice(message):  
 *""" Gets a yes or no choice from the user* **:return***: string  
 """* **return** str(input(message)).strip().lower()  
  
 @staticmethod  
 **def** input\_press\_to\_continue(optional\_message=**''**):  
 *""" Pause program and show a message before continuing* **:param** *optional\_message: An optional message you want to display* **:return***: nothing  
 """* print(optional\_message)  
 input(**'Press the [Enter] key to continue.'**)  
  
 @staticmethod  
 **def** input\_new\_task\_and\_priority():  
 task = input(**"What task would you like to add? "**)  
 priority = input(**"What priority does the task have? "**)  
 **return** task, priority  
  
 @staticmethod  
 **def** input\_task\_to\_remove():  
 task = input(**"What task would you like to remove? "**)  
 **return** task  
  
*# Main Body of Script ------------------------------------------------------ #  
  
# Step 1 - When the program starts, Load data from ToDoFile.txt.*Processor.read\_data\_from\_file(strFileName, lstTable) *# read file data  
  
# Step 2 - Display a menu of choices to the user***while**(**True**):  
 *# Step 3 Show current data* IO.print\_current\_Tasks\_in\_list(lstTable) *# Show current data in the list/table* IO.print\_menu\_Tasks() *# Shows menu* strChoice = IO.input\_menu\_choice() *# Get menu option  
   
 # Step 4 - Process user's menu choice* **if** strChoice.strip() == **'1'**: *# Add a new Task* strTask, strPriority = IO.input\_new\_task\_and\_priority()  
 lstTable, strStatus = Processor.add\_data\_to\_list(strTask,strPriority,lstTable)  
 IO.input\_press\_to\_continue(strStatus)  
 **continue** *# to show the menu* **elif** strChoice == **'2'**: *# Remove an existing Task* strTask = IO.input\_task\_to\_remove()  
 lstTable, strStatus = Processor.remove\_data\_from\_list(strTask,lstTable)  
 IO.input\_press\_to\_continue(strStatus)  
 **continue** *# to show the menu* **elif** strChoice == **'3'**: *# Save Data to File* strChoice = IO.input\_yes\_no\_choice(**"Save this data to file? (y/n) - "**)  
 **if** strChoice.lower() == **"y"**:  
 lstTable, strStatus = Processor.write\_data\_to\_file(strFileName,lstTable)  
 IO.input\_press\_to\_continue(strStatus)  
 **else**:  
 IO.input\_press\_to\_continue(**"Save Cancelled!"**)  
 **continue** *# to show the menu* **elif** strChoice == **'4'**: *# Reload Data from File* print(**"Warning: Unsaved Data Will Be Lost!"**)  
 strChoice = IO.input\_yes\_no\_choice(**"Are you sure you want to reload data from file? (y/n) - "**)  
 **if** strChoice.lower() == **'y'**:  
 lstTable, strStatus = Processor.read\_data\_from\_file(strFileName,lstTable)  
 IO.input\_press\_to\_continue(strStatus)  
 **else**:  
 IO.input\_press\_to\_continue(**"File Reload Cancelled!"**)  
 **continue** *# to show the menu* **elif** strChoice == **'5'**: *# Exit Program* print(**"Goodbye!"**)  
 **break** *# and Exit*