Charles Hodges(hodges11@uw.edu)
August 15, 2021
IT FDN 110: Introduction to Programming (Python)
Assignment 06

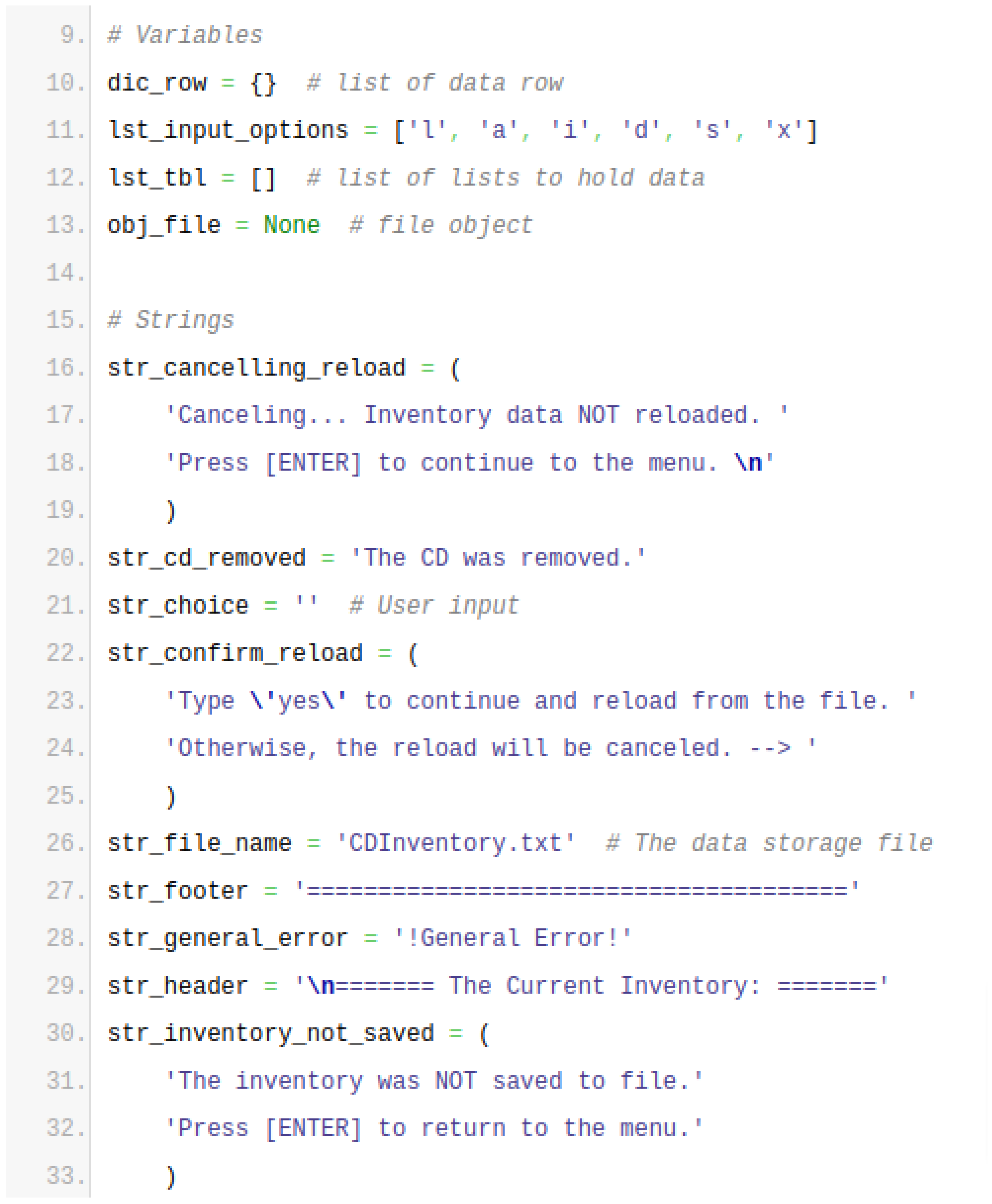
Modify an Example Solution of Assignment 05 by Adding Functions

Introduction

In this assignment, we were given a sample solution to assignment #05. We were asked to finish that assignment by adding functions into the already defined classes. Most of this assignment was simplifying the sample solution, adding a few functions, and a bit of formatting. The aspect that took the longest was modifying the variable names to match the introduced Google PyGuide's Naming Guidelines(external link)[1].

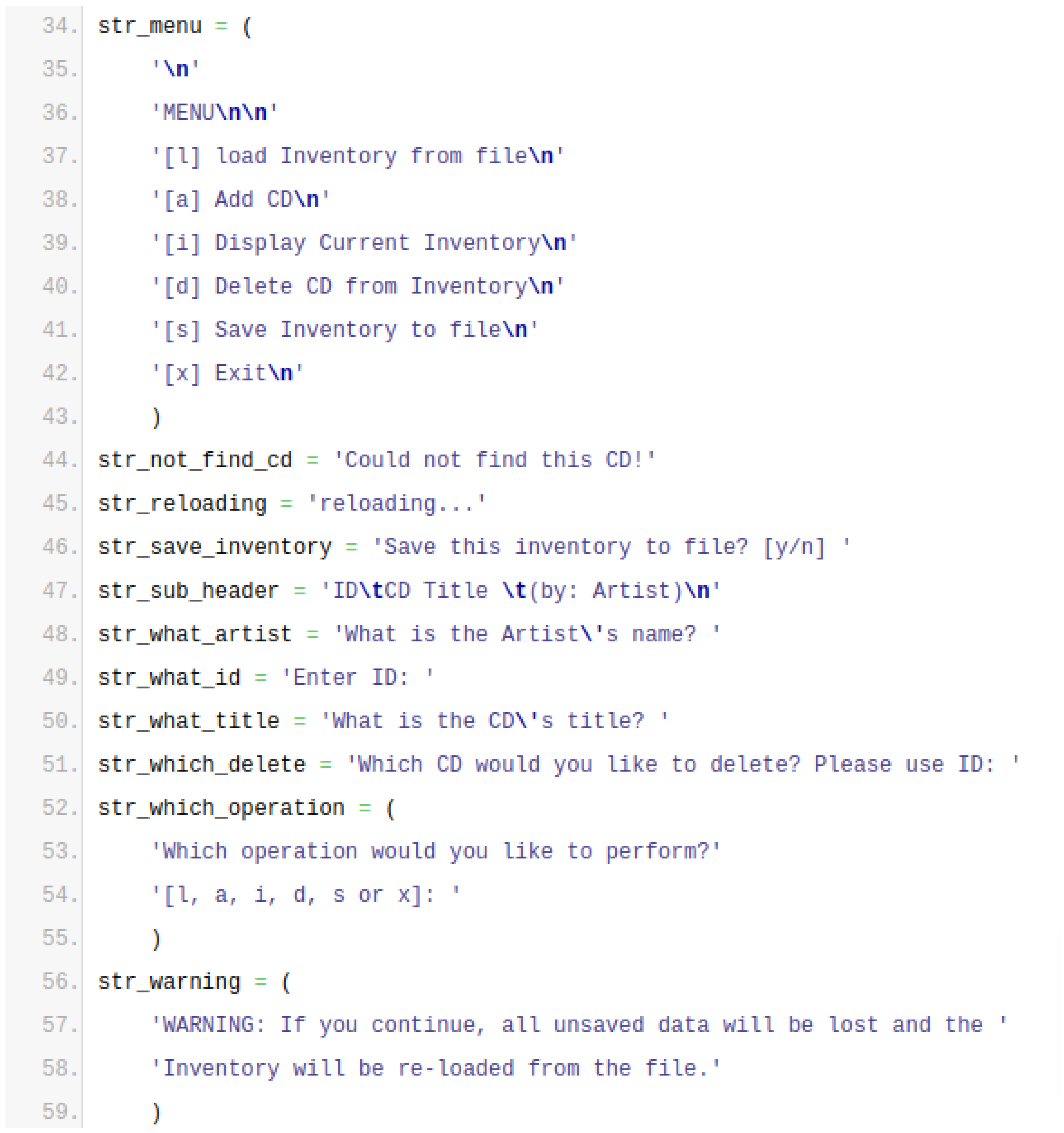
[1] Retrieved 2021-August-14

Non-String Variables & Strings



```python
# Variables
dic_row = {}  # list of data row
lst_input_options = ['l', 'a', 'i', 'd', 's', 'x']
lst_tbl = []  # list of lists to hold data
obj_file = None  # file object

# Strings
str_cancelling_reload = (
    'Canceling... Inventory data NOT reloaded. '
    'Press [ENTER] to continue to the menu. \n'
    )
str_cd_removed = 'The CD was removed.'
str_choice = ''  # User input
str_confirm_reload = (
    'Type \'yes\' to continue and reload from the file. '
    'Otherwise, the reload will be canceled. --> '
    )
str_file_name = 'CDInventory.txt'  # The data storage file
str_footer = '======================================'
str_general_error = '!General Error!'
str_header = '\n======= The Current Inventory: ======='
str_inventory_not_saved = (
    'The inventory was NOT saved to file.'
    'Press [ENTER] to return to the menu.'
    )
```



```python
str_menu = (
    '\n'
    'MENU\n\n'
    '[l] load Inventory from file\n'
    '[a] Add CD\n'
    '[i] Display Current Inventory\n'
    '[d] Delete CD from Inventory\n'
    '[s] Save Inventory to file\n'
    '[x] Exit\n'
    )
str_not_find_cd = 'Could not find this CD!'
str_reloading = 'reloading...'
str_save_inventory = 'Save this inventory to file? [y/n] '
str_sub_header = 'ID\tCD Title \t(by: Artist)\n'
str_what_artist = 'What is the Artist\'s name? '
str_what_id = 'Enter ID: '
str_what_title = 'What is the CD\'s title? '
str_which_delete = 'Which CD would you like to delete? Please use ID: '
str_which_operation = (
    'Which operation would you like to perform?'
    '[l, a, i, d, s or x]: '
    )
str_warning = (
    'WARNING: If you continue, all unsaved data will be lost and the '
    'Inventory will be re-loaded from the file.'
    )
```

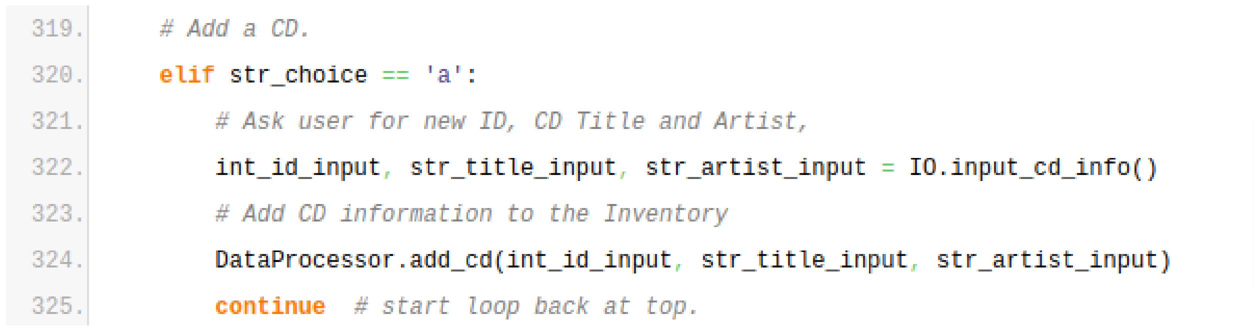
Figure 1 - Non-String Variables & Strings

Once again we have two alphabetized lists, non-string variables and strings. When I reviewed the sample solution which we were given, I noticed that the variable names did not match the Google PyGuide's naming convention, so I edited everything, as I abstracted the strings to a list at the top. I did take Laura's advice, and did not migrate all the tiny strings to that list, as readability was impacted by leaving some very tiny strings within the code.

I am not entirely sure if we were expected to modify all of the non-string variables, and strings, to this convention, or not. But after we used the Google PyGuide for style examples in class, I assumed that we were going to use it further. Maybe we could get some guidance here. Finally, I was able to continue to practice line-wrapping successfully, so that was a bonus.

Args & Returns

There are 10 functions total, across the three classes. Of those 10 functions, only 2 return any values, and only 1 takes any arguments. Not sure if that was what was intended, but since the sample solution used global_var_names, and the current state of the Inventory whenever a menu selection was made, there were hardly any arguments or returns needed. I found that a bit strange, given the topic of the week. Maybe I am misunderstanding something.



```python
    # Add a CD.
    elif str_choice == 'a':
        # Ask user for new ID, CD Title and Artist,
        int_id_input, str_title_input, str_artist_input = IO.input_cd_info()
        # Add CD information to the Inventory
        DataProcessor.add_cd(int_id_input, str_title_input, str_artist_input)
        continue  # start loop back at top.
```

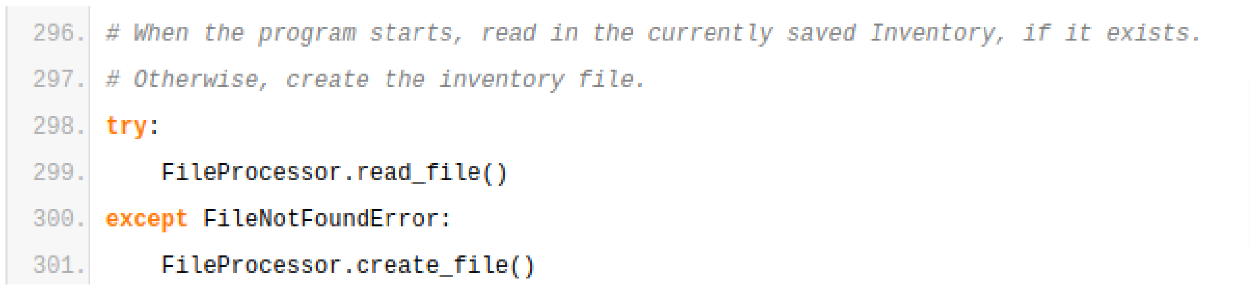
Figure 2 - Unpacking the tuple

The thing I did learn about returns, was that when returning multiple values, unpacking of the tuple is required. Initially, I had these two lines written like this:

`DataProcessor.add_cd(IO.input_cd_info())`

The issue I ran into was that the three returned values from the function "*IO.input_cd_info*" were arranged in a tuple, which was returned as the first value only. So I needed to unpack that tuple, immediately, into some local variables, and then pass them into the "*DataProcessor.add_cd()*" function. I was disappointed that I couldn't do that in one simple line. There were some other complex solutions that were described on StackOverflow, but, in the end, two simple lines were preferred to unnecessary complexity.

Starting the Program

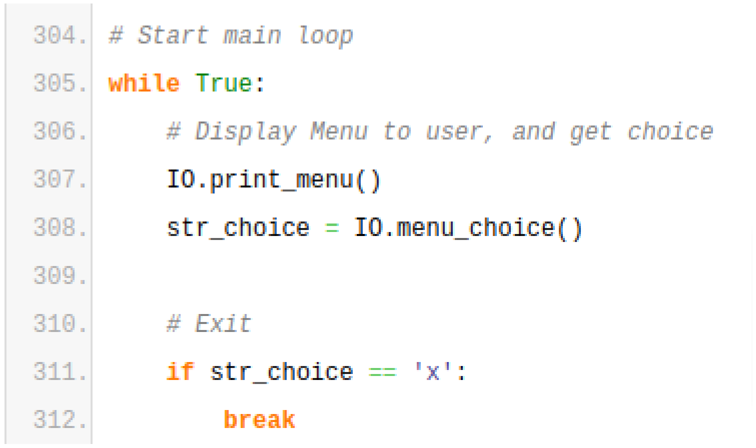


```python
# When the program starts, read in the currently saved Inventory, if it exists.
# Otherwise, create the inventory file.
try:
    FileProcessor.read_file()
except FileNotFoundError:
    FileProcessor.create_file()
```

Figure 3 - Starting the Program

To start the program, I used a Try:Except Block, which was, admittedly, a first for me. Since the ample solution had some text describing how they expected the program to start, we would need a couple of lines to accommodate that description. In this case, we try to read the current text file containing the Inventory. If the text file does not yet exist, we receive a “*FileNotFoundError*”. So we handle that error, keep the program running, and just create the file. While it does look a bit odd, it honors the sample solutions intended start.

The Main Loop

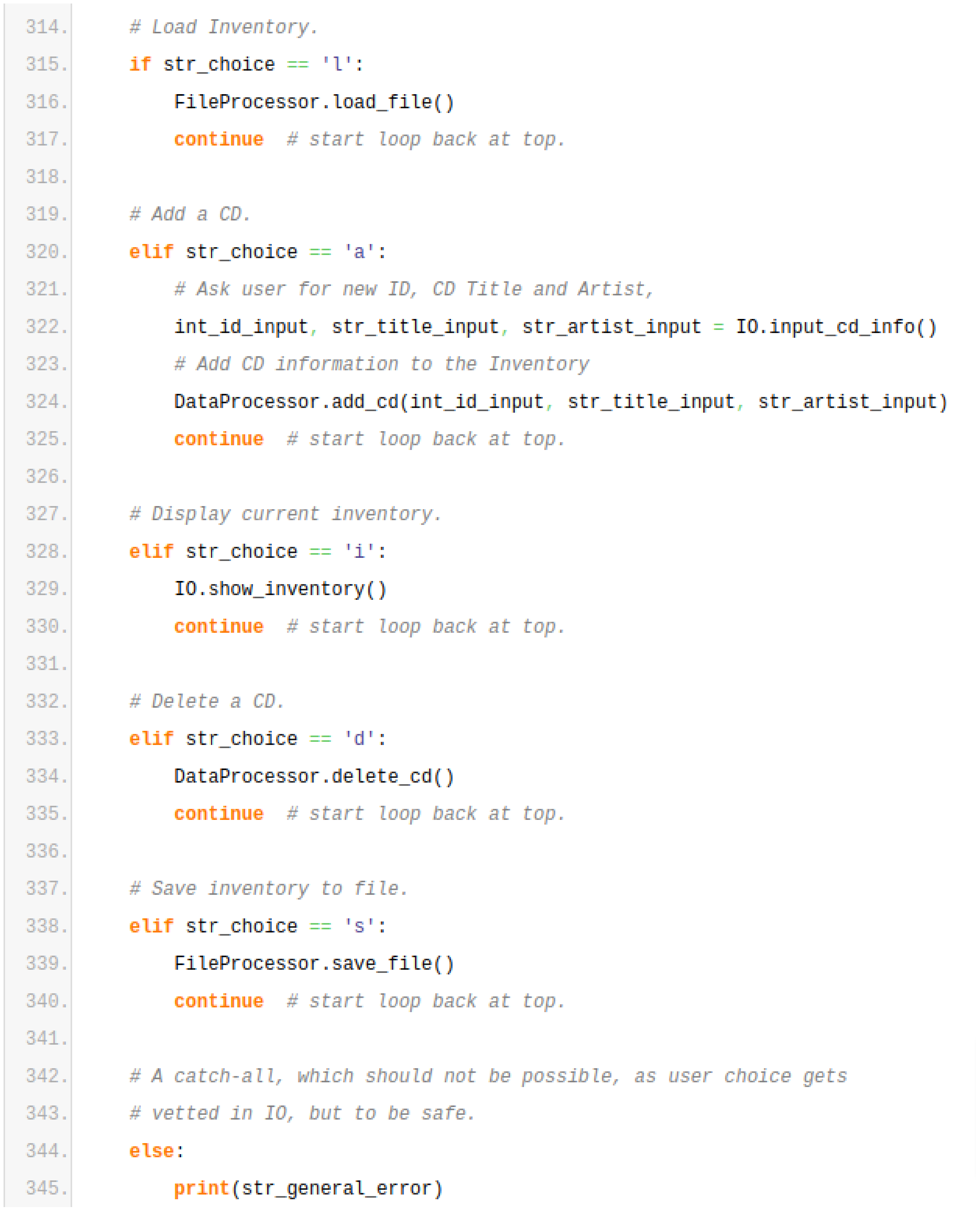


```
# Start main loop
while True:
    # Display Menu to user, and get choice
    IO.print_menu()
    str_choice = IO.menu_choice()

    # Exit
    if str_choice == 'x':
        break
```

Figure 4 - Top of the Main Loop

Here is where we start the While loop, which allows the User to continue to interact with the script, until they are ready to exit. We print the menu of options, and call the function to request and accept the User’s selection. Also, if the User wants to exit, we put that option at the top.



```python
    # Load Inventory.
    if str_choice == 'l':
        FileProcessor.load_file()
        continue  # start loop back at top.

    # Add a CD.
    elif str_choice == 'a':
        # Ask user for new ID, CD Title and Artist,
        int_id_input, str_title_input, str_artist_input = IO.input_cd_info()
        # Add CD information to the Inventory
        DataProcessor.add_cd(int_id_input, str_title_input, str_artist_input)
        continue  # start loop back at top.

    # Display current inventory.
    elif str_choice == 'i':
        IO.show_inventory()
        continue  # start loop back at top.

    # Delete a CD.
    elif str_choice == 'd':
        DataProcessor.delete_cd()
        continue  # start loop back at top.

    # Save inventory to file.
    elif str_choice == 's':
        FileProcessor.save_file()
        continue  # start loop back at top.

    # A catch-all, which should not be possible, as user choice gets
    # vetted in IO, but to be safe.
    else:
        print(str_general_error)
```

Figure 5 - The rest of the Main Loop

The rest of the loop is made up of specific function calls, based on the User’s input. No matter what they select from the rest of the main loop’s options, they may continue to interact with the Inventory, even after they’ve completed a task.

Summary

In this program, we were given a sample solution to Assignment 05, and asked to amend it to include functions. We were asked to add some functions to the already defined classes, and format the script and output correctly. Most of the time that I spent on this assignment was on formatting variables to comply with the Google PyGuide. Hope that was correct.

Appendix

```python
# ------------------------------------------#
# Title: CDInventory.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# Charles Hodges(hodges11@uw.edu), 2021-Aug-15, Created File
# ------------------------------------------#


# Variables
dic_row = {}  # list of data row
lst_input_options = ['l', 'a', 'i', 'd', 's', 'x']
lst_tbl = []  # list of lists to hold data
obj_file = None  # file object

# Strings
str_cancelling_reload = (
    'Canceling... Inventory data NOT reloaded. '
    'Press [ENTER] to continue to the menu. \n'
    )
str_cd_removed = 'The CD was removed.'
str_choice = ''  # User input
str_confirm_reload = (
    'Type \'yes\' to continue and reload from the file. '
    'Otherwise, the reload will be canceled. --> '
    )
str_file_name = 'CDInventory.txt'  # The data storage file
str_footer = '======================================'
str_general_error = '!General Error!'
str_header = '\n======= The Current Inventory: ======='
str_inventory_not_saved = (
    'The inventory was NOT saved to file.'
    'Press [ENTER] to return to the menu.'
    )
str_menu = (
    '\n'
    'MENU\n\n'
    '[l] load Inventory from file\n'
    '[a] Add CD\n'
    '[i] Display Current Inventory\n'
    '[d] Delete CD from Inventory\n'
    '[s] Save Inventory to file\n'
    '[x] Exit\n'
    )
str_not_find_cd = 'Could not find this CD!'
str_reloading = 'reloading...'
str_save_inventory = 'Save this inventory to file? [y/n] '
str_sub_header = 'ID\tCD Title \t(by: Artist)\n'
str_what_artist = 'What is the Artist\'s name? '
str_what_id = 'Enter ID: '
str_what_title = 'What is the CD\'s title? '
str_which_delete = 'Which CD would you like to delete? Please use ID: '
str_which_operation = (
    'Which operation would you like to perform?'
    '[l, a, i, d, s or x]: '
    )
str_warning = (
    'WARNING: If you continue, all unsaved data will be lost and the '
    'Inventory will be re-loaded from the file.'
    )


# -- PROCESSING -- #
class DataProcessor:
    """Processing the data in the table, before file interaction"""

    @staticmethod
    def add_cd(int_id_input, str_title_input, str_artist_input):
        """Function to manage data ingestion from User input of CD info.

       Accepts the User input of new CD information, and creates a dictionary
       object, which is appended to the list table which makes up the
       Inventory.

       Args:
           str_id_input (int):
           str_title_input (string):
           str_artist_input (string):

       Returns:
           None.
       """
        dic_row = {
            'ID': int_id_input,
            'Title': str_title_input.title(),
            'Artist': str_artist_input.title()
            }
        lst_tbl.append(dic_row)
        IO.show_inventory()

    @staticmethod
    def delete_cd():
```

```
  """Function to identify and then delete a CD from the Inventory.

  When the User selects a CD to delete, by ID, that CD is deleted from
  the Inventory.

  Args:
      None.

  Returns:
          None.
      """
    # Display Inventory to user
    IO.show_inventory()
    # Ask user which ID to remove
    int_id_del = int(input(str_which_delete).strip())
    # Search thru table and delete CD
    int_row_nr = -1
    bln_cd_removed = False
    for row in lst_tbl:
        int_row_nr += 1
        if row['ID'] == int_id_del:
            del lst_tbl[int_row_nr]
            bln_cd_removed = True
            break
    if bln_cd_removed:
        print(str_cd_removed)
    else:
        print(str_not_find_cd)
    # Display Inventory to user again
    IO.show_inventory()


class FileProcessor:
    """Processing the data to and from text file"""

    @staticmethod
    def read_file():
        """Function to manage data ingestion from file to a list of
         dictionaries.

        Reads the data from file identified by file_name into a 2D table
        (list of dicts) table one line in the file represents one dictionary
        row in table.

        Args:
            None.

        Returns:
            None.
        """
        # This code clears existing data, and loads data from file
        lst_tbl.clear()
        with open(str_file_name, 'r') as obj_file:
```

```python
            for line in obj_file:
                data = line.strip().split(',')
                dic_row = {
                          'ID': int(data[0]),
                          'Title': data[1],
                          'Artist': data[2]
                          }
                lst_tbl.append(dic_row)

    @staticmethod
    def load_file():
        """Function to manage data ingestion from file to a list of
          dictionaries, when initiated by the User, from the menu.

       Reads the data from file identified by file_name into a 2D table
       (list of dicts) table one line in the file represents one dictionary
       row in table.

       Args:
           None.

       Returns:
           None.
       """
        print(str_warning)
        str_yes_no = input(str_confirm_reload)
        if str_yes_no.lower() == 'yes':
            print(str_reloading)
            FileProcessor.read_file()
            IO.show_inventory()
        else:
            input(str_cancelling_reload)

    @staticmethod
    def save_file():
        """Function to save a file.

       When the User decides to write the current Inventory to a file, after
       any edits, this function is used.

       Args:
           None.

       Returns:
           None.
       """
        # Display current inventory and ask user for confirmation to save
        IO.show_inventory()
        str_yes_no = input(str_save_inventory).strip().lower()
        # Process choice
        if str_yes_no == 'y':
            # Save data
            obj_file = open(str_file_name, 'w')
            for row in lst_tbl:
                lst_values = list(row.values())
                lst_values[0] = str(lst_values[0])
                obj_file.write(','.join(lst_values) + '\n')
            obj_file.close()
        else:
            input(str_inventory_not_saved)

    @staticmethod
    def create_file():
        """Function to create a file if there is none, already.

       Since Write or Append are the only two ways to open/create a file
       if it has not yet been created, we use Append, as it will not
       overwrite any data, if it has already been created. This function
       creates and closes, or merely opens and closes the text file.

       Args:
           None.

       Returns:
           None.
       """
        obj_file = open(str_file_name, 'a')
        obj_file.close()


# -- PRESENTATION (Input/Output) -- #

class IO:
    """Handling Input / Output"""

    @staticmethod
    def print_menu():
        """Displays a menu of choices to the user

       Args:
           None.

       Returns:
           None.
       """
        print(str_menu)

    @staticmethod
    def menu_choice():
        """Gets user input for menu selection

       Args:
           None.

       Returns:
           None.
       """
        choice = ' '
        while choice not in lst_input_options:
            choice = input(str_which_operation).lower().strip()
        print()  # Add extra space for layout
        return choice

    @staticmethod
    def show_inventory():
        """Displays current inventory table

       Args:
           None.

       Returns:
           None.

       """
        print(str_header)
        print(str_sub_header)
        for row in lst_tbl:
            print('{}\t{} \t\t(by:{})'.format(*row.values()))
        print(str_footer)

    @staticmethod
    def input_cd_info():
        """Requests and receives CD information from the User.

       Args:
           None.

       Returns:
           int_id_input(int): ID Number
           str_title_input(string): CD Title
           str_artist_input(string): Artist Name
       """
        int_id_input = int(input(str_what_id).strip())
        str_title_input = input(str_what_title).strip()
        str_artist_input = input(str_what_artist).strip()
        return int_id_input, str_title_input, str_artist_input


# When the program starts, read in the currently saved Inventory, if it exists.
# Otherwise, create the inventory file.
try:
    FileProcessor.read_file()
except FileNotFoundError:
    FileProcessor.create_file()


# Start main loop
while True:
    # Display Menu to user, and get choice
```

```python
        IO.print_menu()
        str_choice = IO.menu_choice()

        # Exit
        if str_choice == 'x':
            break

        # Load Inventory.
        if str_choice == 'l':
            FileProcessor.load_file()
            continue  # start loop back at top.

        # Add a CD.
        elif str_choice == 'a':
            # Ask user for new ID, CD Title and Artist,
            int_id_input, str_title_input, str_artist_input = IO.input_cd_info()
            # Add CD information to the Inventory
            DataProcessor.add_cd(int_id_input, str_title_input, str_artist_input)
            continue  # start loop back at top.

        # Display current inventory.
        elif str_choice == 'i':
            IO.show_inventory()
            continue  # start loop back at top.

        # Delete a CD.
        elif str_choice == 'd':
            DataProcessor.delete_cd()
            continue  # start loop back at top.

        # Save inventory to file.
        elif str_choice == 's':
            FileProcessor.save_file()
            continue  # start loop back at top.

        # A catch-all, which should not be possible, as user choice gets
        # vetted in IO, but to be safe.
        else:
            print(str_general_error)
```