

Laura Qin  
August 28 2022  
IT FDN 110 A  
Assignment 08

# Object-Oriented Programming

## Introduction

For this assignment we are given a starter python file with some pseudocodes, and the goal is to complete the script using custom classes, including data classes and processing classes.

## Data Class

To tackle the challenge of writing an entire script from scratch and defining all the classes, properties and methods myself, I tried to work both forwards and backwards. First thing I did was completing the data class called CD. That way it's clearly defined how the data will be stored (as a list of CD objects with an integer as ID and strings as title and artist), so there's a good foundation to work with when I write out the rest of the script.

In `cd_id`'s setter, I implemented an if-else block to check whether the ID being passed into the setter is an integer or not. If it isn't, the setting won't assign the value to the `__cd_id` private attribute, and it will print out a message telling the user that the ID needs to be an integer. This way unexpected entry for ID can be handled properly.

```
class CD:
    '''Stores data about a CD:

    properties:
        cd_id: (int) with CD ID
        cd_title: (string) with the title of the CD
        cd_artist: (string) with the artist of the CD

    methods:
        None

    '''
    # -- Constructor -- #
    def __init__(self, id: int, title: str, artist: str):
        # -- Attributes -- #
```

```

        self.__cd_id = id
        self.__cd_title = title
        self.__cd_artist = artist

# -- Properties -- #
@property
def cd_id(self):
    '''Getter for cd_id.'''
    return self.__cd_id
@cd_id.setter
def cd_id(self, value):
    '''Setting for cd_id. Raises an exception if value passed in is not numeric.'''
    if type(value) != int:
        raise Exception('ID must be numeric.')
    else:
        self.__cd_id = value

@property
def cd_title(self):
    '''Getter for cd_title.'''
    return self.__cd_title.title()
@cd_title.setter
def cd_title(self, value):
    '''Setter for cd_title.'''
    self.__cd_title = value

@property
def cd_artist(self):
    '''Getter for cd_artist.'''
    return self.__cd_artist.title()
@cd_artist.setter
def cd_artist(self, value):
    '''Setter for cd_artist.'''
    self.__cd_artist = value

```

Script 1. The CD data class

## Code Structure

With the CD class defined, I started to work on the main part of the script. The purpose is to find out what functions are required, and what should be the inputs and outputs of those functions, as I write out the structure of the code.

Essentially, the main script first loads in a list of CD objects at the beginning. Then it prints out the menu, asks the user which action they would like to perform, completes the action, prints out the menu again, and repeats this over and over again until the user chooses to exit. This can be achieved through a while True loop, with an if-elif-else structure inside.

```
# Load data from file into a list of CD objects on script start
lstOfCDObjects = FileIO.load_inventory(strFileName)

while True:
    # Display menu to user
    IO.display_menu()

    # capture user's selection
    menu_choice = IO.get_menu_choice()

    # show user current inventory
    if menu_choice == 'i':
        IO.display_data(lstOfCDObjects)

    # let user add data to the inventory
    elif menu_choice == 'a':
        new_cd_obj = IO.get_cd()
        if new_cd_obj is not None:
            lstOfCDObjects.append(new_cd_obj)

    # let user save inventory to file
    elif menu_choice == 's':
        FileIO.save_inventory(strFileName, lstOfCDObjects)

    # let user load inventory from file
    elif menu_choice == 'l':
        load_yes_no = IO.get_load_yes_no()
        if load_yes_no == 'y':
            lstOfCDObjects = FileIO.load_inventory(strFileName)

    # let user exit program
```

```

elif menu_choice == 'x':
    break

# handle invalid menu choice
else:
    print('Invalid choice, please select again.')

```

Script 2. The main script

## Error Handling

Following Laura's suggestion, I moved the try-except blocks from the main script to within the functions, to keep them as close to the potential errors as possible. I also made the error easier to understand by printing out custom messages instead of using python's built-in error message. Here's a comparison of the error handling in my script from last week's assignment versus this week.

```

# 3.3 process add a CD
elif strChoice == 'a':
    try:
        # 3.3.1 Ask user for new ID, CD Title and Artist
        strID, strTitle, strArtist = IO.get_input_row()
        # 3.3.2 Add item to the table
        DataProcessor.add_row_to_data(strID, strTitle, strArtist, lstTbl)
        IO.show_inventory(lstTbl)
        continue # start loop back at top.
    except ValueError as e:
        print('\nInvalid ID, needs to be an integer')
        print('Detailed error message: ')
        print(type(e), e, e.__doc__, sep = '\n')
    except Exception as e:
        print('\nThere was a general error')
        print('Detailed error message: ')
        print(type(e), e, e.__doc__, sep = '\n')

```

Script 3. Error handling in my previous script

```

@staticmethod
def get_cd():
    '''Ask user to enter information of a new CD and return a new CD object

    Args:
        None
    '''

```

```

Returns:
    cd (a CD object): a new CD object populated with user input
'''

try:
    id = int(input('Enter ID: '))
    title = input('Enter title: ')
    artist = input('Enter artist: ')
    cd = CD(id, title, artist)
    return cd
except ValueError as e:
    print('\nID must be an integer.')
    return None
except Exception as e:
    print('\nThere was a general error. Here are the details: ')
    print(type(e), e, e.__doc__, sep = '\n')
    return None

```

Script 4. Error handling in the new script

Note that for the adding a CD part, in the main section of the new script I had to add a check to see if `new_cd_obj` is empty or not, before appending it to the data. That was to avoid having the display data function throw an error when an empty CD object was appended to the data.

## Processing Class

Lastly, I completed the two processing classes, `FileIO` and `IO`. For both classes, I borrowed code from the last assignment as much as possible, and did some modifications to make it work on the new data structure which is a list of CD objects. For example, here's a comparison between the data display method in the old script versus the new script. As seen here the new script utilizes the CD class's getter to grab the ID, title, and artist, before turning them into a string to display on the screen.

```

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

    Args:
        table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime.

    Returns:
        None.

    '''

```

```

print('\n===== The Current Inventory: =====')
print('ID\tCD Title (by: Artist)\n')
for row in table:
    print('{}\t{} (by: {})'.format(*row.values()))
print('=====')

```

Script 5. Data display method in the old script

```

@staticmethod
def display_data(lst_Inventory):
    '''Show what's currently in the data in memory.

    Args:
        lst_Inventory (list of CD objects): the CD inventory data being stored in
memory

    Returns:
        None
    '''
    print('\n===== The Current Inventory: =====')
    print('ID\tCD Title (by: Artist)\n')
    for cd in lst_Inventory:
        print('{}\t{} (by: {})'.format(str(cd.cd_id), cd.cd_title, cd.cd_artist))
    print('=====')

```

Script 6. Data display method in the new script

## Result

I ran the updated script in VS Code, it's working as expected. It prints out a menu, and asks the user to enter their selection. It was able to execute the user's request, and generate a text file that includes the data entered by the user. It also behaves as expected when run on the Terminal. When the file to read from doesn't exist, and when the user enters a non-numeric ID, instead of crashing it would print out some useful diagnostic information and skip to the beginning menu selection.

• (base) lauraqin@Lauras-MacBook-Pro Assignment\_08 % python CD\_Inventory.py

Menu

```
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
```

Which operation would you like to perform? [l, a, i, d, s or x]: i

```
===== The Current Inventory: =====
ID      CD Title (by: Artist)
```

```
1       The Dark Side Of The Moon (by:Pink Floyd)
2       Blood On The Tracks (by:Bob Dylan)
=====
```

Menu

```
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
```

Which operation would you like to perform? [l, a, i, d, s or x]: l

Loading from file will erase all unsaved data. Are you sure? [y/n]: y

File cannot be found.

Menu

```
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
```

Which operation would you like to perform? [l, a, i, d, s or x]: i

```
===== The Current Inventory: =====
ID      CD Title (by: Artist)
```

Menu

```
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
```

Which operation would you like to perform? [l, a, i, d, s or x]: a

```
Enter ID: 1
Enter title: the dark side of the moon
Enter artist: pink floyd
```

Menu

```
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
```

Which operation would you like to perform? [l, a, i, d, s or x]: a

```
Enter ID: 2
Enter title: blood on the tracks
Enter artist: bob dylan
```

Menu

```
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
```

Which operation would you like to perform? [l, a, i, d, s or x]: a

```

Which operation would you like to perform? [l, a, i, d, s or x]: a
Enter ID: id
ID must be an integer.
Menu
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
Which operation would you like to perform? [l, a, i, d, s or x]: i

===== The Current Inventory: =====
ID      CD Title (by: Artist)
1       The Dark Side Of The Moon (by:Pink Floyd)
2       Blood On The Tracks (by:Bob Dylan)
=====

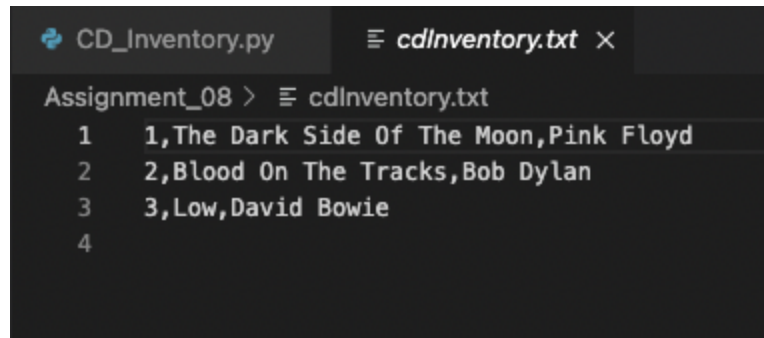
Menu
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
Which operation would you like to perform? [l, a, i, d, s or x]: a
Enter ID: 3
Enter title: low
Enter artist: david bowie
Menu
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
Which operation would you like to perform? [l, a, i, d, s or x]: s
Menu
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
Which operation would you like to perform? [l, a, i, d, s or x]: l
Loading from file will erase all unsaved data. Are you sure? [y/n]: y
Menu
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
Which operation would you like to perform? [l, a, i, d, s or x]: i

===== The Current Inventory: =====
ID      CD Title (by: Artist)
1       The Dark Side Of The Moon (by:Pink Floyd)
2       Blood On The Tracks (by:Bob Dylan)
3       Low (by:David Bowie)
=====

```

Figure 1. Running on VSCode





```
CD_Inventory.py  cdInventory.txt x
Assignment_08 > cdInventory.txt
1 1,The Dark Side Of The Moon,Pink Floyd
2 2,Blood On The Tracks,Bob Dylan
3 3,Low,David Bowie
4
```

Figure 2. cdInventory.txt after VSCode run

```
Assignment_08 — -zsh — 80x72
((base) lauraqin@Lauras-MacBook-Pro Assignment_08 % python CD_Inventory.py

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: i

===== The Current Inventory: =====
ID      CD Title (by: Artist)

1       The Dark Side Of The Moon (by:Pink Floyd)
2       Blood On The Tracks (by:Bob Dylan)
3       Low (by:David Bowie)
=====

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: a

Enter ID: asdf

ID must be an integer.

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: a

Enter ID: 4
Enter title: rumours
Enter artist: fleetwood mac

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: i

===== The Current Inventory: =====
ID      CD Title (by: Artist)

1       The Dark Side Of The Moon (by:Pink Floyd)
2       Blood On The Tracks (by:Bob Dylan)
3       Low (by:David Bowie)
4       Rumours (by:Fleetwood Mac)
=====
```

```

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: s

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: l

Loading from file will erase all unsaved data. Are you sure? [y/n]: y

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: i

===== The Current Inventory: =====
ID      CD Title (by: Artist)
1       The Dark Side Of The Moon (by:Pink Floyd)
2       Blood On The Tracks (by:Bob Dylan)
3       Low (by:David Bowie)
4       Rumours (by:Fleetwood Mac)
=====

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: x

(base) lauraqin@Lauras-MacBook-Pro Assignment_08 %

```

Figure 3. Running on Terminal

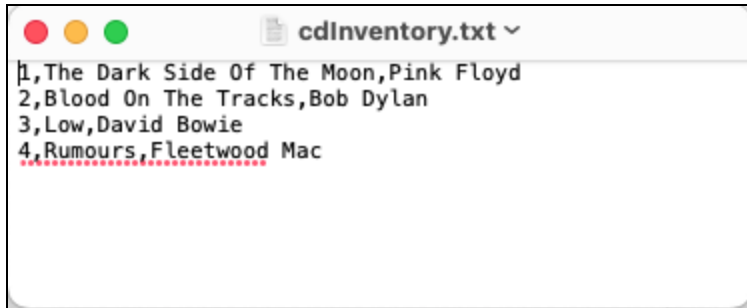


Figure 4. cdInventory.txt after Terminal run

## Summary

Through this assignment I was able to learn more about object-oriented programming, specifically how to use data classes to allow more control when dealing with data, and use process classes to make processing more organized and reusable. I was also able to improve error handling based on Laura's feedback from the previous assignment.

## Appendix

### 1. CD\_Inventory.py

```
#-----#
# Title: Assignmen08.py
# Desc: Assignment 08 - Working with classes
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, created file
# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08
# qinlaura, 2022-Aug-28, completed code
#-----#

# -- DATA -- #
strFileName = 'cdInventory.txt'
lstOfCDObjects = []

class CD:
    '''Stores data about a CD:

    properties:
        cd_id: (int) with CD ID
        cd_title: (string) with the title of the CD
        cd_artist: (string) with the artist of the CD'''
```

```

methods:
    None

'''
# -- Constructor -- #
def __init__(self, id: int, title: str, artist: str):
    # -- Attributes -- #
    self.__cd_id = id
    self.__cd_title = title
    self.__cd_artist = artist

# -- Properties -- #
@property
def cd_id(self):
    '''Getter for cd_id.'''
    return self.__cd_id
@cd_id.setter
def cd_id(self, value):
    '''Setting for cd_id. Raises an exception if value passed in is not numeric.'''
    if type(value) != int:
        raise Exception('ID must be numeric.')
    else:
        self.__cd_id = value

@property
def cd_title(self):
    '''Getter for cd_title.'''
    return self.__cd_title.title()
@cd_title.setter
def cd_title(self, value):
    '''Setter for cd_title.'''
    self.__cd_title = value

@property
def cd_artist(self):
    '''Getter for cd_artist.'''
    return self.__cd_artist.title()
@cd_artist.setter
def cd_artist(self, value):
    '''Setter for cd_artist.'''
    self.__cd_artist = value

```

```

# -- PROCESSING -- #
class FileIO:
    '''Processes data to and from file:

    properties:
        None

    methods:
        save_inventory(file_name, lst_Inventory): -> None
        load_inventory(file_name): -> (a list of CD objects)

    '''
    # -- Methods -- #
    @staticmethod
    def save_inventory(file_name, lst_Inventory):
        '''Save the inventory data into file.

        Args:
            file_name (string): name of file to save data to
            lst_Inventory (list of CD objects): the CD inventory data being stored in
memory

        Returns:
            None
        '''
        obj_file = open(file_name, 'w')
        for cd in lst_Inventory:
            line = ','.join([str(cd.cd_id), cd.cd_title, cd.cd_artist])
            obj_file.write(line + '\n')
        obj_file.close()

    @staticmethod
    def load_inventory(file_name):
        '''Load inventory data in file into memory.

        Args:
            file_name (string): name of file to load data from

        Returns:
            lst_Inventory (list of CD objects): the CD inventory data being stored in
memory
        '''

```

```

try:
    lst_Inventory = []
    obj_file = open(file_name, 'r')
    for line in obj_file:
        l = line.strip().split(',')
        cd = CD(int(l[0]), l[1], l[2])
        lst_Inventory.append(cd)
    return lst_Inventory
except FileNotFoundError as e:
    print('\nFile cannot be found.')
    return []
except Exception as e:
    print('\nThere was a general error. Here are the details: ')
    print(type(e), e, e.__doc__, sep = '\n')
    return []

# -- PRESENTATION (Input/Output) -- #
class IO:
    '''Processes data to and from file:

    properties:
        None

    methods:
        print_menu(): -> None
        get_menu_choice(): -> str
        display_data(lst_Inventory): -> None
        get_cd(): -> a CD object
        get_load_yes_no(): -> str

    '''
    # -- Methods -- #
    @staticmethod
    def display_menu():
        '''Displays menu options.

        Args:
            None

        Returns:
            None

    '''

```

```

        print('\nMenu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current
Inventory')

        print('[s] Save Inventory to file\n[x] exit\n')

    @staticmethod
    def get_menu_choice():
        '''Get user's choice.

        Args:
            None

        Returns:
            choice (string): a lower case sting of the users input out of the choices
l, a, i, s or x
        '''
        choice = ' '
        while choice not in ['l', 'a', 'i', 's', 'x']:
            choice = input('Which operation would you like to perform? [l, a, i, d, s
or x]: ').lower().strip()
        print()
        return choice

    @staticmethod
    def display_data(lst_Inventory):
        '''Show what's currently in the data in memory.

        Args:
            lst_Inventory (list of CD objects): the CD inventory data being stored in
memory

        Returns:
            None
        '''
        print('\n===== The Current Inventory: =====')
        print('ID\tCD Title (by: Artist)\n')
        for cd in lst_Inventory:
            print('{}\t{} (by: {})'.format(str(cd.cd_id), cd.cd_title, cd.cd_artist))
        print('=====')

    @staticmethod
    def get_cd():
        '''Ask user to enter information of a new CD and return a new CD object
'''

```



```

    Args:
        None

    Returns:
        cd (a CD object): a new CD object populated with user input
    '''
    try:
        id = int(input('Enter ID: '))
        title = input('Enter title: ')
        artist = input('Enter artist: ')
        cd = CD(id, title, artist)
        return cd
    except ValueError as e:
        print('\nID must be an integer.')
        return None
    except Exception as e:
        print('\nThere was a general error. Here are the details: ')
        print(type(e), e, e.__doc__, sep = '\n')
        return None

    @staticmethod
    def get_load_yes_no():
        '''Print out a warning, and ask user if they want to proceed to load data from
file or not

    Args:
        None

    Returns:
        choice (str): whether the user wants to proceed with loading data, 'y' or
'n'
    '''
        choice = input('Loading from file will erase all unsaved data. Are you sure?
[y/n]: ')
        return choice

# -- Main Body of Script -- #

# Load data from file into a list of CD objects on script start
lstOfCDObjects = FileIO.load_inventory(strFileName)

```

```

while True:
    # Display menu to user
    IO.display_menu()

    # capture user's selection
    menu_choice = IO.get_menu_choice()

    # show user current inventory
    if menu_choice == 'i':
        IO.display_data(lstOfCDObjects)

    # let user add data to the inventory
    elif menu_choice == 'a':
        new_cd_obj = IO.get_cd()
        if new_cd_obj is not None:
            lstOfCDObjects.append(new_cd_obj)

    # let user save inventory to file
    elif menu_choice == 's':
        FileIO.save_inventory(strFileName, lstOfCDObjects)

    # let user load inventory from file
    elif menu_choice == 'l':
        load_yes_no = IO.get_load_yes_no()
        if load_yes_no == 'y':
            lstOfCDObjects = FileIO.load_inventory(strFileName)

    # let user exit program
    elif menu_choice == 'x':
        break

    # handle invalid menu choice
    else:
        print('Invalid choice, please select again.')

```

## 2. Github link

[https://github.com/NoraQin/Assignment\\_08](https://github.com/NoraQin/Assignment_08)