Sonia Romo
2020-Sep-02
IT FDN 110 B - Foundations of Programming, Python
Assignment 08

Object Oriented Programming

Introduction

This week was the first real foray into object oriented programming. We looked at how to create classes to instantiate objects and all of the aspects of classes.

Labs

The labs this week (plus Dirk reviewing the labs during class) were extremely helpful in understanding the different features of classes.

LAB08-A

This code creates a class with three fields. It then creates the object and sets the values for each field. Lastly, it prints the class fields.

```python
#-----------------------------------------#
# Title: LAB080-A.py
# Desc: adding Class fields
# Change Log: (Who, When, What)
# SRomo, 2020-Aug-29, Created File, added fields
#-----------------------------------------#


class TrackInfo():

    # --- Fields --- #
    position = ''
    title = ''
    length = ''


objTrack1 = TrackInfo()
objTrack1.position = 2
objTrack1.title = 'cardigan'
objTrack1.length = '4:00'

print('Track position: {}'.format(objTrack1.position))
print('Track title: {}'.format(objTrack1.title))
print('Track length: {}'.format(objTrack1.length))
```

Figure 01 - LAB08-A code

```
In [97]: runfile('/Users/sonia/FDNPython/Mod_08/LAB08-A.py', wdir='/Users/sonia/
FDNPython/Mod_08')
Track position: 2
Track title: cardigan
Track length: 4:00
```

Figure 02 - LAB08-A results

LAB08-B

This code adds the Class constructor and adds attributes. It then creates two objects from the class with the three attributes and prints the objects.

/Users/sonia/FDNPython/Mod_08/LAB08-B.py

LAB08-B.py | LAB08-C.py | LAB08-D.py | LAB08-E.py | CD_Inventory.py

```python
#-----------------------------------------#
# Title: LAB080-B.py
# Desc: adding Class constructors
# Change Log: (Who, When, What)
# SRomo, 2020-Aug-30, Created File, added constructor
#-----------------------------------------#


class TrackInfo():

    # --- Fields --- #
    position = ''
    title = ''
    length = ''

    # --- Constructor --- #
    def __init__(self, pos, title, length):
        # --- Attribute --- #
        self.position = pos
        self.title = title
        self.length = length


objTrack1 = TrackInfo(1, 'cardigan', '4:00')
objTrack2 = TrackInfo(2, 'the 1', '3:59')

print('track 1: {}, {}, {}'.format(objTrack1.position, objTrack1.title, objTrack1.length))
print('track 2: {}, {}, {}'.format(objTrack2.position, objTrack2.title, objTrack2.length))
```

Figure 03 - LAB08-B code

```
In [98]: runfile('/Users/sonia/FDNPython/Mod_08/LAB08-B.py', wdir='/Users/sonia/
FDNPython/Mod_08')
track 1: 1, cardigan, 4:00
track 2: 2, the 1, 3:59
```

Figure 04 - LAB08-B results

LAB08-C

This code changes the Class fields to be different names than the attributes. It then creates two objects from the class with the three attributes and prints the objects.

/Users/sonia/FDNPython/Mod_08/LAB08-C.py

LAB08-C.py | LAB08-D.py | LAB08-E.py | CD_Inventory.py

```python
#------------------------------------------#
# Title: LAB080-C.py
# Desc: adding Class constructors
# Change Log: (Who, When, What)
# SRomo, 2020-Aug-30, Created File, added constructor
#------------------------------------------#


class TrackInfo():

    # --- Fields --- #
    fldPosition = ''
    fldTitle = ''
    fldLength = ''

    # --- Constructor --- #
    def __init__(self, pos, title, length):
        # --- Attribute --- #
        self.position = pos
        self.title = title
        self.length = length


objTrack1 = TrackInfo(1, 'cardigan', '4:00')
objTrack2 = TrackInfo(2, 'the 1', '3:59')

print('track 1: {}, {}, {}'.format(objTrack1.position, objTrack1.title, objTrack1.length))
print('track 2: {}, {}, {}'.format(objTrack2.position, objTrack2.title, objTrack2.length))
```

Figure 05 - LAB08-C code

```
In [99]: runfile('/Users/sonia/FDNPython/Mod_08/LAB08-C.py', wdir='/Users/sonia/
FDNPython/Mod_08')
track 1: 1, cardigan, 4:00
track 2: 2, the 1, 3:59
```

Figure 06 - LAB08-C results

LAB08-D

This code uses the setter and getter methods to establish and give access to private attributes.

/Users/sonia/FDNPython/Mod_08/LAB08-D.py

LAB08-D.py | LAB08-E.py | CD_Inventory.py

```python
#------------------------------------------#
# Title: LAB080-B.py
# Desc: adding Class constructors
# Change Log: (Who, When, What)
# SRomo, 2020-Aug-30, Created File, added constructor
#------------------------------------------#


class TrackInfo():

    # --- Fields --- #

    # --- Constructor --- #
    def __init__(self, pos, title, length):
        # --- Attribute --- #
        self.__position = pos
        self.__title = title
        self.__length = length

    # --- Properties --- #
    @property
    def position(self):
        return self.__position

    @position.setter
    def position(self, value):
        if type(value) == int:
            self.position = value
        else:
            raise Exception('Position must be an integer.')

    @property
    def title(self):
        return self.__title

    @title.setter
    def title(self, value):
        if type(value) == str:
            self.__title = value
        else:
            raise Exception('Title must be a string.')

    @property
    def length(self):
        return self.__length

    @length.setter
    def length(self, value):
        if type(value) == str:
            self.__length = value
        else:
            raise Exception('Length must be a string.')


objTrack1 = TrackInfo(1, 'cardigan', '4:00')
objTrack2 = TrackInfo(2, 'the 1', '3:59')

print('track 1: {}, {}, {}'.format(objTrack1.position, objTrack1.title, objTrack1.length))
print('track 2: {}, {}, {}\n'.format(objTrack2.position, objTrack2.title, objTrack2.length))


try:
    objTrack1.position = '1'
except Exception as e:
    print(e)
```

Figure 07 - LAB08-D code

```
In [100]: runfile('/Users/sonia/FDNPython/Mod_08/LAB08-D.py', wdir='/Users/sonia/FDNPython/Mod_08')
track 1: 1, cardigan, 4:00
track 2: 2, the 1, 3:59

Position must be an integer.
```

Figure 08 - LAB08-D results

LAB08-E

This code adds the `__str__` method to show the object's attributes in a nicely formatted way.

/Users/sonia/FDNPython/Mod_08/LAB08-E.py

LAB08-E.py | CD_Inventory.py

```python
#------------------------------------------#
# Title: LAB080-B.py
# Desc: adding Class constructors
# Change Log: (Who, When, What)
# SRomo, 2020-Aug-30, Created File, added constructor
#------------------------------------------#


class TrackInfo():

    # --- Fields --- #

    # --- Constructor --- #
    def __init__(self, pos, title, length):
        # --- Attribute --- #
        self.__position = pos
        self.__title = title
        self.__length = length

    # --- Properties --- #
    @property
    def position(self):
        return self.__position

    @position.setter
    def position(self, value):
        if type(value) == int:
            self.position = value
        else:
            raise Exception('Position must be an integer.')

    @property
    def title(self):
        return self.__title

    @title.setter
    def title(self, value):
        if type(value) == str:
            self.__title = value
        else:
            raise Exception('Title must be a string.')

    @property
    def length(self):
        return self.__length

    @length.setter
    def length(self, value):
        if type(value) == str:
            self.__length = value
        else:
            raise Exception('Length must be a string.')

    # --- Methods --- #

    def __str__(self):
        return ('{}, {}, {}'.format(self.position, self.title, self.length))



objTrack1 = TrackInfo(1, 'cardigan', '4:00')
objTrack2 = TrackInfo(2, 'the 1', '3:59')

print('1st: {}'.format(objTrack1.__str__()))
print('2st: {}\n'.format(objTrack2.__str__()))

print('1st: {}'.format(objTrack1))
```

Figure 09 - LAB08-E code

```
In [101]: runfile('/Users/sonia/FDNPython/Mod_08/LAB08-E.py', wdir='/Users/sonia/FDNPython/Mod_08')
1st: 1, cardigan, 4:00
2st: 2, the 1, 3:59

1st: 1, cardigan, 4:00
```

Figure 10 - LAB08-E results

Homework

Homework this week was less stressful than last week. The live walkthrough of the labs in class helped solidify the concepts behind using Classes to create Objects.

After reviewing the pseudocode, I first started by creating the CD class code to be able to instantiate an object.

I then added code from the previous assignments, without yet modifying for the object, to build the frame for the code.

I then updated the `save_inventory`, `load_inventory`, and `show_inventory` functions to work with objects.

Lastly, I added a bit of error handling.

While the labs, class, and chapter 08 from [Python Programming for the Absolute Beginner]() (retrieved 2020-Sep-01) were helpful, [this article from Stack Overflow]() (retrieved 2020-Sep-01) also helped me in understanding the how to create objects from data in a `.txt` file.

```
In [105]: runfile('/Users/sonia/FDNPython/Assignment08/CD_Inventory.py', wdir='/Users/sonia/FDNPython/
Assignment08')
The file doesn't exist. No data could be loaded.

Menu

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


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


Enter ID: 1

What is the CD's title? folklore

What is the Artist's name? t swift
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       folklore (by:t swift)
======================================

Menu

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


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


Enter ID: 2

What is the CD's title? everywhere

What is the Artist's name? tim mcgraw
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================

Menu

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


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

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

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================


Save this inventory to file? [y/n] y
Your data was saved to the file!
Menu

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


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

Figure 11 - Spyder run #1

```
In [106]: runfile('/Users/sonia/FDNPython/Assignment08/CD_Inventory.py', wdir='/Users/sonia/FDNPython/
Assignment08')
Your data was loaded!

Menu

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


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

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

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================

Menu

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


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

Figure 12 - Spyder run #1

```
(base) MacBook-Pro:Assignment08 sonia$ python CD_Inventory.py
Your data was loaded!

Menu

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

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

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

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================

Menu

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

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

Enter ID: 3
What is the CD's title? margaritaville
What is the Artist's name? jimmy buffett
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
3       margaritaville (by:jimmy buffett)
======================================

Menu

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

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

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

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
3       margaritaville (by:jimmy buffett)
======================================

Save this inventory to file? [y/n] y
Your data was saved to the file!
Menu

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

Figure 13 - Terminal run #1

Summary

3-5 sentences on what you learned

Appendix

Complete homework source code:

```
#-----------------------------------------#
# Title: CD_Inventory.py
# Desc: Assignnment 08 - Working with classes
# Change Log: (Who, When, What)
# SRomo, 2020-Sep-01, created file, added CD class
# SRomo, 2020-Sep-02, adjusted save, load, and show functions, added error handling
#-----------------------------------------#

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

class CD():
    """Stores data about a CD:

    properties:
        intId: (int) with CD ID
        strTitle: (string) with the title of the CD
        strArtist: (string) with the artist of the CD
    methods:
        __str__: returns formatting of properties

    """
    # --- Constructor --- #
    def __init__(self, cdID, ttl, art):
        self.__intID = cdID
        self.__strTitle = ttl
        self.__strArtist = art


    # --- Properties --- #
    @property
    def intID(self):
        return self.__intID

    @intID.setter
    def intID(self, value):
        self.intID = value
```

```python

    @property
    def strTitle(self):
        return self.__strTitle

    @strTitle.setter
    def strTitle(self, value):
        self.strTitle = value

    @property
    def strArtist(self):
        return self.__strArtist

    @strArtist.setter
    def strArtist(self, value):
        self.strArtist = value

    # --- Method --- #
    def __str__(self):
        return ('{}, {}, {}'.format(self.intID, self.strTitle, self.strArtist))

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

    properties:

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

    """
    @staticmethod
    def load_inventory(file_name, table):
        """Function to manage data ingestion from file to a list of objects

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

        Args:
            file_name (string): name of file used to read the data from
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        table.clear()  # this clears existing data and allows to load data from file

```

```
        try:
            with open(file_name, 'r') as objFile:
                for row in objFile:
                    data = row.strip().split(',')
                    objCD = CD(data[0],data[1],data[2])
                    table.append(objCD)
                print('Your data was loaded!\n')
        except FileNotFoundError:
            print('The file doesn\'t exist. No data could be loaded.\n')
        except:
            print('Something else went wrong.\n')


       @staticmethod
       def save_inventory(file_name, table):
           """Function to write data from the table to a file

           Args:
               file_name (string): name of file used to read the data from
               table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

           Returns:
               None
           """

           try:
               with open(file_name, 'w') as objFile:
                   for obj in table:
                       data = (str(obj.intID) + ',' + str(obj.strTitle) + ',' + str(obj.strArtist) + '\n')
                       objFile.write(data)
                   print('Your data was saved to the file!')
           except FileNotFoundError:
               print('The file doesn\'t exist. No data could be saved.\n')
           except:
               print('Something else went wrong.\n')


   # -- 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('Menu\n\n[l] Show inventory from file\n[a] Add CD to inventory')
        print('[i] Display Current Inventory\n[s] Save Inventory to file\n[x] exit\n')


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

        Args:
            None.

        Returns:
            choice (string): a lower case string 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, s or x]: ').lower().strip()
        print()  # Add extra space for layout
        return choice


    @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('======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
        for obj in table:
            print('{}\t{} (by:{})'.format(obj.intID,obj.strTitle,obj.strArtist))
        print('======================================\n')

    @staticmethod
    def add_cd():
        """Allows user to add a CD
```

```python

        Args:
            strID: user input for CD ID
            strTitle: CD title
            strArtist: artist name

        Returns:
            intID, strTitle, strArtist

        """
        strID = input('Enter ID: ').strip()
        strTitle = input('What is the CD\'s title? ').strip()
        strArtist = input('What is the Artist\'s name? ').strip()
        intID = int(strID)
        return intID, strTitle, strArtist


# -- Main Body of Script -- #
FileIO.load_inventory(strFileName, lstOfCDObjects) # Read in the currently saved inventory from txt file

while True:
    IO.print_menu() # Display Menu to user
    strChoice = IO.menu_choice() # Get user choice

    if strChoice == 'x': # process exit request
        break

    if strChoice == 'l': # process load inventory from file
        print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
        strYesNo = input('Do you want to continue? [y/n] ')
        if strYesNo.lower() == 'y':
            print('reloading...')
            FileIO.load_inventory(strFileName, lstOfCDObjects)
            IO.show_inventory(lstOfCDObjects)
        else:
            input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
            IO.show_inventory(lstOfCDObjects)
        continue  # start loop back at top.

    elif strChoice == 'a': # process add a CD
        intID, strTitle, strArtist = IO.add_cd() # User input for new ID, CD Title and Artist

        objCD = CD(intID, strTitle, strArtist) # Instantiate object

        lstOfCDObjects.append(objCD) # Append object to list

```

```
            IO.show_inventory(lstOfCDObjects) # Display inventory
            continue

        elif strChoice == 'i': # process display current inventory
            IO.show_inventory(lstOfCDObjects)
            continue

        elif strChoice == 's': # process save inventory to txt file
            IO.show_inventory(lstOfCDObjects) # Display current inventory and ask user for confirmation to save
            strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()

            if strYesNo == 'y':
                FileIO.save_inventory(strFileName, lstOfCDObjects)
            else:
                input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
            continue
        # catch-all should not be possible, as user choice gets vetted in IO, but to be save:
        else:
            print('General Error')
```