Deep Learning Lab 2

Spring 2018

Author:

Moe Almansoori

Objective:

Using some of the powerful Python language tools and features such as sets, lists, and dictionaries by developing python solutions for several popular and important situations such as searching within a dictionary, and nesting dictionaries, sets, and lists. Other important objective for this lab is to utilize the grate flexibility of the OOP programming style in python by using classes, inheritance, and different variables types. To do that, we went through four different tasks.

The objective for each task:

1- In this task we design a dictionary that contains books names and prices and the main task is being familiar with iterating, accessing, and searching the dictionary for a specific data item.
2- Here we use more advanced case of the dictionaries and lists in python as each value (or person name) has multiple corresponding values such as phone number, email address, and so on. So in this case we cant use the simple dictionary to represent this data structure but we need to use nested form of list/dictionary or dictionary/dictionary. Also here we edit the data inside the nested list/dictionary format.
3- Here we focus on using the power of the object oriented programming in python as it provide a great deal of time and effort saving and also efficiency. The task is focusing on using the classes, then inheritance and multiple inheritance. In addition to using the private and global variables.
4- Here we implemented a simple task to be familiar with numpy module that is mainly used in the machine learning in this class. So we generate random numbers and find the most frequent one among them.

Features:

The accomplished programs have been written in a simple programming structure and considering the logic sequence in results processing and proper transitioning between the steps. It is considered that when someone else reads the program to understand its commands line by line and or blocks functionality, by clearly separating the tasks and giving descriptive names for the variables and providing descriptive comments for the main parts of the code.

The fists task provides a simple and efficient solution for any library or online shopping search web application tool/website.

The second task provides the main part of the contacts management application that is used in any communication application, and definitely it can be improved to be more comprehensive.

The third task is focusing on utilizing the oop capabilities in python rather than the application itself which is a library system,

The fourth task is providing a very common application that can be used as a part of program in many cases and specially the statistical and machine learning applications which is our main topic for this class.

Configuration:

Similar working configuration had been used for all the Lab1 requirements, they are:

Python 3.6

OS: Widows 10

IDE: Eclipse Oxygen 2

Input/output (screenshots):

Requirement 1:

Lab2Req1

```python
# Lab #2 Requirement #1

# Initializing the minimum possible price as 0
UserRangeMin = 0
# Sking the user to enter their min and max price range
print('Please enter your price range. Minimum Range is zero default ..')
UserRangeMin=input('Min : ')
UserRangeMax=input('Max : ')
# Books names and prices in a dictionary format
Books = { "Wireless": 220, "Networking": 140, "Data Structure": 80, "Linux Server Admin": 260 \
          , "Client Server Programming with Linux": 285}
print('The book(s) that in the range of $' + str(UserRangeMin) + ' and $' + str(UserRangeMax) + ' is/are : ')
# Iterating through the dictionary and checking the value part (the book price) and print the matching book name and price
for key in Books:
    if Books[key] >= int(UserRangeMin) and Books[key] <= int(UserRangeMax):
        print('The book ('+ str(key) +') has a price of $' + str(Books[key]))
```

Console PyUnit

```
<terminated> Lab2Req1.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]
Please enter your price range. Minimum Range is zero default ..
Min : 80
Max : 280
The book(s) that in the range of $80 and $280 is/are :
The book (Wireless) has a price of $220
The book (Networking) has a price of $140
The book (Data Structure) has a price of $80
The book (Linux Server Admin) has a price of $260
```

Equirement 2:

Lab2Req2

```
Importing the used modules
mport gc
 The nested list/dictionary information of the contacts which contains name, phone number, and email address
ist1={"Moe":['Moe','8162352222','mkakh3@mail.umkc.edu'], "Mike":['Mike', '8168927366', 'sdlk@mail.umkc.edu']}
```

Console

```
<terminated> Lab2Req2.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]
To display contact by name press 1
To display contact by number press 2
To edit contact by name press 3
To quit the program press 4
your selection is : 1
Enter a name to display its contact : Mike
Name   :  Mike
Number :  8168927366
Email  :  sdlk@mail.umkc.edu
To display contact by name press 1
To display contact by number press 2
To edit contact by name press 3
To quit the program press 4
your selection is : 2
Enter a number to display its contact : 8168927366
Name   :  Mike
Number :  8168927366
Email  :  sdlk@mail.umkc.edu
To display contact by name press 1
To display contact by number press 2
To edit contact by name press 3
To quit the program press 4
your selection is : 4
Exiting the Contact List program ...

Exiting ...
End of the program, Thanks !
```

Lab2Req2

```
1  Importing the used modules
2 mport gc
3  The nested list/dictionary information of the contacts which contains name, phone number, and email address
4 ist1={"Moe":['Moe','8162352222','mkakh3@mail.umkc.edu'], "Mike":['Mike', '8168927366', 'sdlk@mail.umkc.edu']}
```

Console

Lab2Req2.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]

```
To display contact by name press 1
To display contact by number press 2
To edit contact by name press 3
To quit the program press 4
your selection is : 3
Enter a name to modify its contact info : Moe
Enter n to modify the number or e to modify the email ...
your selection is : n
The new number is : 8160000000

The updated list of contact is:
Name   :  Moe
Number :  8160000000
Email  :  mkakh3@mail.umkc.edu


Name   :  Mike
Number :  8168927366
Email  :  sdlk@mail.umkc.edu


To display contact by name press 1
To display contact by number press 2
To edit contact by name press 3
To quit the program press 4
your selection is :
```

Lab2Req2

```
1  Importing the used modules
2 mport gc
3  The nested list/dictionary information of the contacts which contains name, phone number, and email address
4 ist1={"Moe":['Moe','8162352222','mkakh3@mail.umkc.edu'], "Mike":['Mike', '8168927366', 'sdlk@mail.umkc.edu']}
```

Console

Lab2Req2.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]

```
To display contact by name press 1
To display contact by number press 2
To edit contact by name press 3
To quit the program press 4
your selection is : 3
Enter a name to modify its contact info : Moe
Enter n to modify the number or e to modify the email ...
your selection is : e
The new email is : mk@newwmail.com

The updated list of contact is:
Name   :  Moe
Number :  8162352222
Email  :  mk@newwmail.com


Name   :  Mike
Number :  8168927366
Email  :  sdlk@mail.umkc.edu


To display contact by name press 1
To display contact by number press 2
To edit contact by name press 3
To quit the program press 4
your selection is :
```

Requirement 3:

Lab2Req3

```
from builtins import str
```

Console

<terminated> Lab2Req3.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]

```
Adding 3 employees information into the Employees class :


 ===Employee #1===
Employee first name : Josh
Employee family name : Smith
Employee department : IT


 ===Employee #2===
Employee first name : Danny
Employee family name : Smith
Employee department : Library


 ===Employee #3===
Employee first name : Sara
Employee family name : Brown
Employee department : Registration


Adding 2 students information into the Students class :


 ===Student #1===
Student first name : Ahmed
Student family name : Albishri
Student department : Computer Science
```

Lab2Req3

```
from builtins import str
```

Console

<terminated> Lab2Req3.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]

```
 ===Student #1===
Student first name : Ahmed
Student family name : Albishri
Student department : Computer Science


 ===Student #2===
Student first name : Khalid
Student family name : Almalki
Student department : Telecommunication and Computer Networking


Adding 2 Faculty Members information into the FacultyMembers class,
which is inherited from the Employees class and add additional argument which is (the major).
So we customizing the inheritance here using super()
This additional attribute of the FacultyMembers class can be considered as a private data member


 ===Faculty Member #1===
Faculty Members first name : Alex
Faculty Members family name : Foard
Faculty Members department : Computer Science
Faculty Members field : Networking


 ===Faculty Member #2===
Faculty Members first name : Matt
Faculty Members family name : Douglas
Faculty Members department : Electrical Engineering
Faculty Members field : Power Electronics
```

Lab2Req3

```
from builtins import str
```

Console

<terminated> Lab2Req3.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]

```
 ===Faculty Member #2===
Faculty Members first name : Matt
Faculty Members family name : Douglas
Faculty Members department : Electrical Engineering
Faculty Members field : Power Electronics


Adding 2 Librarians information into the Librarians class :


 ===Librarian #1===
Librarian first name : Mike
Librarian family name : Allen
Librarian department : Computer Science


 ===Librarian #2===
Librarian first name : Michael
Librarian family name : Anderson
Librarian department : Telecommunication and Computer Networking


Adding 2 Books information into the Books class :


 ===Book title#1===
Book author : Corey Beard
Book Title : Wireless Communications
Book Edition : 2nd
Borrowed by : Ahmed Albishri
```

Lab2Req3

```python
from builtins import str

```

Console

<terminated> Lab2Req3.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]

```
 ===Librarian #2===
Librarian first name : Michael
Librarian family name : Anderson
Librarian department : Telecommunication and Computer Networking


Adding 2 Books information into the Books class :


 ===Book title#1===
Book author : Corey Beard
Book Title : Wireless Communications
Book Edition : 2nd
Borrowed by : Ahmed Albishri


 ===Book title #2===
Book author : Deep Medhi
Book Title : Network Routing
Book Edition : 3rd
Borrowed by : Khalid Almalki


Reteiving a private content form the Books class ...
Error ! retrieving __TempDepartment failed because it is a private variable of Books class


Can't retrieve this private content from Books class
```

Requirement 4:

Lab2Req4

```python
# Importing the required modules
import numpy as np
# Generating a vector of 15 random values between 0 and 20
RandArray = np.random.randint(20, size = 15)
print("The randomly generated array:", RandArray)
# Finding the most frequent element in this vector
MostFrequent = np.bincount(RandArray).argmax()
print("Most frequent element in this vector is : ", MostFrequent)
```

Console

```
<terminated> Lab2Req4.py [C:\Users\Surface User\AppData\Local\Programs\Python\Python36-32\python.exe]
The randomly generated array: [ 0  7  0 19  2  6  1  6  4  9  4 15  3 10 18]
Most frequent element in this vector is :  0
```

The implementation including code snippet

The code for all the four tasks is described in detail bellow

Requirement 1:

```
# Lab #2 Requirement #1

# Initializing the minimum possible price as 0
UserRangeMin = 0
# Sking the user to enter their min and max price
range
print('Please enter your price range. Minimum Range
is zero default ..')
UserRangeMin=input('Min : ')
UserRangeMax=input('Max : ')
# Books names and prices in a dictionary format
Books = { "Wireless": 220, "Networking": 140, "Data
Structure": 80, "Linux Server Admin": 260 \
         , "Client Server Programming with linux":
285}
print('The book(s) that in the range of $' +
str(UserRangeMin) + ' and $' + str(UserRangeMax) + '
is/are : ')
# Iterating through the dictionary and checking the
value part (the book price) and print the matching
book name and price
for key in Books:
    if Books[key] >= int(UserRangeMin) and Books[key]
<= int(UserRangeMax):
        print('The book ('+ str(key) +') has a price
of $' + str(Books[key]))
```

Requirement 2:

```
# Importing the used modules
import gc
# The nested list/dictionary information of the
contacts which contains name, phone number, and email
address
list1={"Moe":['Moe','8162352222','mkakh3@mail.umkc.ed
u'], "Mike":['Mike', '8168927366',
'sdlk@mail.umkc.edu']}
# An infinite loop to keep asking the user to enter
their choice
while True:
    # The available options that a user can use
    print('To display contact by name press 1\nTo
display contact by number press 2\nTo edit contact by
name press 3\nTo quit the program press 4')
    indexx = input('your selection is : ')
    # A glag to indicate if the user entered a name
that is not in the contacts list so it give an error
saying that
    flagg = 0
    # The case when the user enters 1 to search and
display contact info by name
    if indexx == str(1):
        name1 = input('Enter a name to display its
contact : ')
        for key in list1:
            if key == name1:
                flagg = 1
                print('Name   : ', list1[name1][0])
                print('Number : ', list1[name1][1])
                print('Email  : ', list1[name1][2])
        if flagg == 0:
```

```
            print('Ooops ! This name can not be
found, try again !')
    elif indexx == str(2):
        # The case when the user enters 2 to search
and display contact info by number
        num = input('Enter a number to display its
contact : ')
        for key in list1:
            if list1[key][1] == num:
                flagg = 1
                print('Name   : ', list1[key][0])
                print('Number : ', list1[key][1])
                print('Email  : ', list1[key][2])
        if flagg == 0:
            print('Ooops ! This name can not be
found, try again !')
    elif indexx == str(3):
        # The case when the user enters 3 to search
and edit contact info by number
        name1 = input('Enter a name to modify its
contact info : ')
        for key in list1:
            if key == name1:
                flagg = 1
                # User have to specify to edit the
number (select n) or the email (select e)
                print('Enter n to modify the number
or e to modify the email ...')
                indexxx = input('your selection is :
')
                if indexxx == 'n':
                    # Asking for the new number and
changing it
```

```
                    NewNumber = input('The new number
is : ')
                    list1[name1][1] = NewNumber
                elif indexxx == 'e':
                    # Asking for the new email and
changing it
                    NewEmail = input('The new email
is : ')
                    list1[name1][2] = NewEmail
                print('\nThe updated list of contact
is: ')
                for key in list1:
                    print('Name   : ', list1[key][0])
                    print('Number : ', list1[key][1])
                    print('Email  : ', list1[key][2])
                    print('\n')
        if flagg == 0:
            print('Ooops ! This name can not be
found, try again !')
    elif indexx == str(4):
        # when user enters 4 to exit
        print('Exiting the Contact List program ...')
        print("")
        print('Exiting ...')
        # Freeing up the used resources
        del list1
        gc.collect()
        print('End of the program, Thanks !')
        break
    else:
        # when user enters anything not 1,2,3,4
        print('Error ! you have to enter 1, 2, 3, or
4')
```

Requirement 3:

```
from builtins import str

# creating a class named as Employees
class Employees:
    # Creating a constructor to initialize name,
family, department
    def __init__(self,nm,fam,dprt):
        self.name = nm
        self.family = fam
        self.department = dprt


class Students:
    # Creating a constructor to initialize name,
family, department
    def __init__(self,nm,fam,dprt):
        self.name = nm
        self.family = fam
        self.department = dprt


#Inheritance and using super to customize the
inheritance
class FacultyMembers(Employees):
    # Creating a constructor to initialize name,
family, department, and the field
    def __init__(self,nm,fam,dprt,field):
        # Using super for single inheritance to allow
adding new instances for the FacultyMembers class
(which is field)
```

```
        super().__init__(nm,fam,dprt)
        self.major = field


class Librarieans:
    # Creating a constructor to initialize name,
family, department
    def __init__(self,nm,fam,dprt):
        self.name = nm
        self.family = fam
        self.department = dprt

# Multi inheritance from Students,
Employees,Librarieans classes and using the super
constructor
class Books(Students, Employees,Librarieans):
    # Creating a constructor to initialize all the
data members from all the multi-inheritance here
    def __init__(self, auth, ttl, edi, field, nm,
fam, dprt):
        Students.__init__(self, nm, fam, dprt)
        Employees.__init__(self, nm, fam, dprt)
        Librarieans.__init__(self, nm, fam, dprt)
        self.author = auth
        self.title = ttl
        self.edition = edi
        self.major = field
        self.__TempDepartment = dprt
```

```
# Adding 3 employees information into the Employees
class
print("\n\nAdding 3 employees information into the
Employees class : ")
EmployeeDataHolder = Employees("Josh", "Smith", "IT")
print("\n\n ===Employee #1===" + "\nEmployee first
name : " + str(EmployeeDataHolder.name) + "\nEmployee
family name : " + str(EmployeeDataHolder.family) +
"\nEmployee department : " +
str(EmployeeDataHolder.department))
EmployeeDataHolder = Employees("Danny", "Smith",
"Library")
print("\n\n ===Employee #2===" + "\nEmployee first
name : " + str(EmployeeDataHolder.name) + "\nEmployee
family name : " + str(EmployeeDataHolder.family) +
"\nEmployee department : " +
str(EmployeeDataHolder.department))
EmployeeDataHolder = Employees("Sara", "Brown",
"Registration")
print("\n\n ===Employee #3===" + "\nEmployee first
name : " + str(EmployeeDataHolder.name) + "\nEmployee
family name : " + str(EmployeeDataHolder.family) +
"\nEmployee department : " +
str(EmployeeDataHolder.department))



# Adding 2 students information into the Students
class
print("\n\nAdding 2 students information into the
Students class : ")
StudentDataHolder = Students("Ahmed", "Albishri",
"Computer Science")
```

```
print("\n\n ===Student #1===" + "\nStudent first name
: " + str(StudentDataHolder.name) + "\nStudent family
name : " + str(StudentDataHolder.family) + "\nStudent
department : " + str(StudentDataHolder.department))
StudentDataHolder = Students("Khalid", "Almalki",
"Telecommunication and Computer Networking")
print("\n\n ===Student #2===" + "\nStudent first name
: " + str(StudentDataHolder.name) + "\nStudent family
name : " + str(StudentDataHolder.family) + "\nStudent
department : " + str(StudentDataHolder.department))
```

```
# Adding 2 Faculty Members information into the
FacultyMembers class that is inherited from the
Employees class and utilize the additional argument
which is (the major).\n so we inheriting all the
Employees class attributes using super() and adding
the additional one:
print("\n\nAdding 2 Faculty Members information into
the FacultyMembers class,\nwhich is inherited from
the Employees class and add additional argument which
is (the major).\nSo we customizing the inheritance
here using super()\nThis additional attribute of the
FacultyMembers class can be considered as a private
data member")
FacultyMemberDataHolder = FacultyMembers("Alex",
"Foard", "Computer Science", "Networking")
print("\n\n ===Faculty Member #1===" + "\nFaculty
Members first name : " +
str(FacultyMemberDataHolder.name) + "\nFaculty
Members family name : " +
str(FacultyMemberDataHolder.family) + "\nFaculty
```

```
Members department : " +
str(FacultyMemberDataHolder.department) + "\nFaculty
Members field : " +
str(FacultyMemberDataHolder.major))
FacultyMemberDataHolder = FacultyMembers("Matt",
"Douglas", "Electrical Engineering", "Power
Electronics")
print("\n\n ===Faculty Member #2===" + "\nFaculty
Members first name : " +
str(FacultyMemberDataHolder.name) + "\nFaculty
Members family name : " +
str(FacultyMemberDataHolder.family) + "\nFaculty
Members department : " +
str(FacultyMemberDataHolder.department) + "\nFaculty
Members field : " +
str(FacultyMemberDataHolder.major))


# Adding 2 Librarians information into the
Librarieans class
print("\n\nAdding 2 Librarians information into the
Librarians class : ")
LibDataHolder = Librarieans("Mike", "Allen",
"Computer Science")
print("\n\n ===Librarian #1===" + "\nLibrarian first
name : " + str(LibDataHolder.name) + "\nLibrarian
family name : " + str(LibDataHolder.family) +
"\nLibrarian department : " +
str(LibDataHolder.department))
LibDataHolder = Librarieans("Michael", "Anderson",
"Telecommunication and Computer Networking")
print("\n\n ===Librarian #2===" + "\nLibrarian first
name : " + str(LibDataHolder.name) + "\nLibrarian
```

```
family name : " + str(LibDataHolder.family) +
"\nLibrarian department : " +
str(LibDataHolder.department))



# Adding 2 books information into the Books class
print("\n\nAdding 2 Books information into the Books
class : ")
BookDataHolder = Books("Corey Beard", "Wireless
Communications", "2nd", "EECS", "Ahmed", "Albishri",
"Computer Science")
print("\n\n ===Book title#1===" + "\nBook author : "
+ str(BookDataHolder.author) + "\nBook Title : " +
str(BookDataHolder.title) + "\nBook Edition : " +
str(BookDataHolder.edition) + "\nBorrowed by : " +
str(BookDataHolder.name) + " " +
str(BookDataHolder.family))
BookDataHolder = Books("Deep Medhi", "Network
Routing", "3rd", "CS", "Khalid", "Almalki",
"Telecommunication and Computer Networking")
print("\n\n ===Book title #2===" + "\nBook author : "
+ str(BookDataHolder.author) + "\nBook Title : " +
str(BookDataHolder.title) + "\nBook Edition : " +
str(BookDataHolder.edition) + "\nBorrowed by : " +
str(BookDataHolder.name) + " " +
str(BookDataHolder.family))



print("\n\nReteiving a private content form the Books
class ...")
try:
```

```
    print(BookDataHolder.__TempDepartment)
except Exception:
    print("Error ! retrieving __TempDepartment failed because it is a private variable of Books class")
    print("\n\nCan't retrieve this private content from Books class")
```

Requirement 4:

```
# Importing the required modules
import numpy as np
# Generating a vector of 15 random values between 0 and 20
RandArray = np.random.randint(20, size = 15)
print("The randomly generated array:", RandArray)
# Finding the most frequent element in this vector
MostFrequent = np.bincount(RandArray).argmax()
print("Most frequent element in this vector is : ", MostFrequent)
```

Explain about the deployment

It started with understanding the problem and figuring out what a good way to provide the required results. Then writing down a pseudo code and what is the input type and format. I was trying to avoid using any unnecessary programming structures and seek the simplicity. The startup program can be containing unnecessary loops or variables, but with several testing and polishing the code, the programs because more effective and easy to follow and understand.

Limitation

The programs that been developed for this lab purpose still need more improvements to serve in real life environment. For example, the contacts program can be improved more by including changing the name and adding new contacts. Other thing is these programs does not save the results permanently such as in a file.

References

- https://stackoverflow.com/questions/3294889/iterating-over-dictionaries-using-for-loops
- https://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3/Dictionaries
- https://www.youtube.com/watch?v=jzotq6GNia0
- https://stackoverflow.com/questions/5904969/how-to-print-a-dictionarys-key
- https://hackernoon.com/understanding-the-underscore-of-python-309d1a029edc
- http://www.pp.rhul.ac.uk/~george/PH2150/html/node47.html
-