Author: Patricia Foley

1.0 Objective

For Lab 2 the objective is to understand the following:

1.1 How to use dictionaries in python
1.2 How to set up classes including inheriting methods from a different class
1.3 Lastly to become familiar with the numpy library and the different functions offered in the library.

2.0 Features

The following are the features for each question:

2.1 Question 1: This program determine what books are available within a certain price range that the user enters.
2.2 Question 2: This program allows a user to display a contact name by either entering a name or number. In addition, the program allows a user to edit a contact by entering the name. Lastly, the user can also select to close out of the program.
2.3 Question 3: This program develops an airline booking reservation system and displays the relationships between the 5 different classes within the system.
2.4 Question 4: This program determines the number with the highest frequency in an array of random integers.

3.0 Configuration

Pycharm was used for the programing which was installed on Windows 10 64-bit operating system with Intel® Core™ i7-6600U CPU @ 2.60GHz processor. The installed RAM is 16.0 GB.

4.0 Input/Output

4.1 Question 1

Create a program using a dictionary to associate the cost with a book that is available for purchase. Prompt the user to enter a minimum and maximum value and then display the books within the selected price range.

4.1.1 Approach

To begin, a dictionary needs to be created that has the book titles as the keys and the costs as the value.

```
books = {'python' : 50, 'web': 30, 'c': 20, 'java': 40}
```

Second, the user needs to be prompted to enter a minimum and maximum value that they are willing to spend. To do this the function input() is used. Whenever the function input() is used, the input value is a string. However, in this case an integer is necessary. Therefore, int() should also be used.

```
minimum = int(input("Please enter a minimum value:"))
maximum = int(input("Please enter a maximum value:"))
```

Lastly, the program looks at each price and determines if the cost fits within the range chosen by the user. If the cost meets the criteria, then the book name is saved into a list. Once all prices are looked at the list is printed which represents all books that fall within the designated range.

4.1.2 Full Source Code

```
#This code displays the book names of any book that fits the range the customer
wants to spend
# Create a dictionary with the book names as the key and the cost as the value.
books = {'python' : 50, 'web': 30, 'c': 20, 'java': 40}
# Initialize a list
cost = []

# Ask the user to enter a minimum and maximum amount they would like to spend.
minimum = int(input("Please enter a minimum value:"))
maximum = int(input("Please enter a maximum value:"))

# Execute the loop for the number of items in the dictionary
for x in books:
    # Extract the values one at a time and place them in a list
    cost = books.get(x)
    # Determine if there is a value in the range the customer wants
    if minimum <= cost <= maximum:
        # Execute the loop for the values in the dictionary
        for bookname, value in books.items():
            # If the value is equal to the cost then print the book name
            if value == cost:
                print(bookname)
```

4.2 Question 2

Given a contact list which is saved as a dictionary, allow the user to do one of the following:

1. Display the contact list by entering a name
2. Display the contact list by entering a number
3. Edit a contact by entering a name
4. Exit the program

4.2.1 Approach

Since the user has multiple options and the program should continue to allow the user to select different options until they choose to exit the program, the code was setup with various functions. The functions that were developed were the following:

1. DisplayContactList () which shows all the contacts within the list.
2. UserSelection () which allows the user to choose which option they would like to select.

```
def UserSelection ():
    global UserInput
    #User can select either a, b, c, or d.
    UserInput = input("Select one of the following by entering a, b,
c, or d:"'\n'
                      "a) Display contact by name"'\n'
                      "b) Display contact by number"'\n'
                      "c) Edit contact by name"'\n'
```

```
                "d) Exit"'\n'
                "Your Selection:"'\n')
```

3. ValueInList (x) which determines if a name or number is in the contact list.

```
def ValueInList(x):
    NameToEdit = x
    InContacts = 'no'
    select = 0
    #Iterate for the number of dictionaries in the Contact List
    for w in ContactList:
        #Look at each dictionary individually
        single = ContactList[select]
        #If the user input can be located in the contact list than
initialize the while loops.
        if NameToEdit in single.values():
            InContacts = 'yes'
        select = select + 1
    #If user input cannot be found in the contact list. Then notify
the user the value cannot be found.
    if InContacts == 'no':
        print("Input is not in contact list")
    return InContacts
```

4. DisplaySelectedContact (x) which prints the contact that was selected by the user either entering the name or number of the contact.

```
def DisplaySelectedContact(x):
    #User enters a name or number
    if x == 'a':
        NameorNumber = input("Please Enter Contact Name:")
    elif x == 'b':
        NameorNumber = input("Please Enter Contact Number:")
    index = 0
    #Determine if input is in contact list.
    InContacts = ValueInList(NameorNumber)
    while InContacts == 'yes':
        #Iterate for the number of dictionaries in contact list
        for x in ContactList:
            #Save the single dictionary into a temporary list
            singledictionary = ContactList[index]
            #If the user input is found. Print the contact
            if NameorNumber in singledictionary.values():
                print(singledictionary)
            index = index + 1
            #Stop while loop after all dictionaries are looked at.
            if index == len(ContactList):
                InContacts = 'no'
```

5. PossibleEdits (w, d, i) which allows the user to edit the contact that was selected.

```
#Make changes to the value of the key that was selected.
def PossibleEdits(w,d,i):
    #If "a" was selected than an update to the name will occur.
    if w == 'a':
        update = input("What would you like to change the name to?:")
        key = 'name'
    # If "b" was selected than an update to the number will occur.
    elif w == 'b':
        update = input("What would you like to change the nnumber
to?:")
```

```
        key = 'number'
    # If "c" was selected than an update to the email will occur.
    elif w == 'c':
        update = input("What would you like to change the email
to?:")
        key = 'email'
    #Update the value in the dictionary
    d[key] = update
    #Update the contact list to have the correct value.
    ContactList[i] = d
    #Print the update
    print(ContactList[i])
    MakeSelection = 'no'
    return MakeSelection
```

6. EditContact () prompts the user to select what they would like to edit either the name, number, or email. If the user did not select a valid input, then notify the user to select again.

7.
```
def EditContact():
    #User inputs the name
    NameToEdit = input("Please Enter the Name you Would Like to
Edit:")
    index = 0
    CorrectSelection = ['a','b','c']
    MakeTheSelection = 'yes'
    #Check to see if the value is in the contact list.
    InContacts = ValueInList(NameToEdit)
    while InContacts == 'yes':
        #Interate for the number of dictionaries in the contact list.
        for x in ContactList:
            #Save the single dictionary into a temporary list
            singledictionary = ContactList[index]
            #See if the name is in the single dictionary
            if NameToEdit in singledictionary.values():
                while MakeTheSelection == 'yes':
                    #Ask the user what they would like to edit
                    WhatToEdit = input("Select one of the following
to edit by entering a, b, or c:"'\n'
                                       "a) Name"'\n'
                                       "b) Number"'\n'
                                       "c) Email"'\n'
                                       "Your Selection:")
                    #Determine if the input was valid
                    if WhatToEdit in CorrectSelection:
                        #Update the dictionary and contact list
accordingly
                        MakeTheSelection = PossibleEdits(WhatToEdit,
singledictionary, index)
                    #If input was not valid. Let the user know and
have them select again.
                    else:
                        print("Not the correct selection")
                        MakeTheSelection = 'yes'
            index = index + 1
            if index == len(ContactList):
                InContacts = 'no'
```

8. ExitContactList () which exits the program when selected.

```
        def ExitContactList():
            KeepSelecting = 'no'
            return KeepSelecting
```

4.2.2 Full Source Code

```
"""This code allows the user to select four different choices: display the contact
by entering the name, display the
contact by the number, edit the contact by entering the name and then what needs to
be updated, and exit the contact
list."""

#What values should the user initially use.
AcceptableEntries = ['a','b','c','d']
#Initiate the while loop.
KeepSelecting = 'yes'
global ContactList
#What is the contact list.
ContactList=[{'name': 'Rashmi', 'number': '8797989821', 'email': 'rr@gmail.com'},
             {'name': 'Saria', 'number': '9897989821', 'email': 'ss@gmail.com'}]

#Function to accept the user input.
def UserSelection ():
    global UserInput
    #User can select either a, b, c, or d.
    UserInput = input("Select one of the following by entering a, b, c, or d:"'\n'
                      "a) Display contact by name"'\n'
                      "b) Display contact by number"'\n'
                      "c) Edit contact by name"'\n'
                      "d) Exit"'\n'
                      "Your Selection:"'\n')

#Function to print the contact list.
def DisplayContactList():
    print(ContactList)

#Function to check if what was entered after the initial selections can be found in
the contact list.
def ValueInList(x):
    NameToEdit = x
    InContacts = 'no'
    select = 0
    #Iterate for the number of dictionaries in the Contact List
    for w in ContactList:
        #Look at each dictionary individually
        single = ContactList[select]
        #If the user input can be located in the contact list than initialize the
while loops.
        if NameToEdit in single.values():
            InContacts = 'yes'
        select = select + 1
    #If user input cannot be found in the contact list. Then notify the user the
value cannot be found.
    if InContacts == 'no':
        print("Input is not in contact list")
    return InContacts

#Display all contact information associated with a name or number
def DisplaySelectedContact(x):
    #User enters a name or number
    if x == 'a':
        NameorNumber = input("Please Enter Contact Name:")
```

```
    elif x == 'b':
        NameorNumber = input("Please Enter Contact Number:")
    index = 0
    #Determine if input is in contact list.
    InContacts = ValueInList(NameorNumber)
    while InContacts == 'yes':
        #Iterate for the number of dictionaries in contact list
        for x in ContactList:
            #Save the single dictionary into a temporary list
            singledictionary = ContactList[index]
            #If the user input is found. Print the contact
            if NameorNumber in singledictionary.values():
                print(singledictionary)
            index = index + 1
            #Stop while loop after all dictionaries are looked at.
            if index == len(ContactList):
                InContacts = 'no'

#Make changes to the value of the key that was selected.
def PossibleEdits(w,d,i):
    #If "a" was selected than an update to the name will occur.
    if w == 'a':
        update = input("What would you like to change the name to?:")
        key = 'name'
    # If "b" was selected than an update to the number will occur.
    elif w == 'b':
        update = input("What would you like to change the nnumber to?:")
        key = 'number'
    # If "c" was selected than an update to the email will occur.
    elif w == 'c':
        update = input("What would you like to change the email to?:")
        key = 'email'
    #Update the value in the dictionary
    d[key] = update
    #Update the contact list to have the correct value.
    ContactList[i] = d
    #Print the update
    print(ContactList[i])
    MakeSelection = 'no'
    return MakeSelection

#The user would like to edit a contact list. The user must enter the name of the
contact that they would like to update
def EditContact():
    #User inputs the name
    NameToEdit = input("Please Enter the Name you Would Like to Edit:")
    index = 0
    CorrectSelection = ['a','b','c']
    MakeTheSelection = 'yes'
    #Check to see if the value is in the contact list.
    InContacts = ValueInList(NameToEdit)
    while InContacts == 'yes':
        #Interate for the number of dictionaries in the contact list.
        for x in ContactList:
            #Save the single dictionary into a temporary list
            singledictionary = ContactList[index]
            #See if the name is in the single dictionary
            if NameToEdit in singledictionary.values():
                while MakeTheSelection == 'yes':
                    #Ask the user what they would like to edit
                    WhatToEdit = input("Select one of the following to edit by
entering a, b, or c:"'\n'
                                       "a) Name"'\n'
```

```
                                        "b) Number"'\n'
                                        "c) Email"'\n'
                                        "Your Selection:")
                        #Determine if the input was valid
                        if WhatToEdit in CorrectSelection:
                            #Update the dictionary and contact list accordingly
                            MakeTheSelection = PossibleEdits(WhatToEdit,
singledictionary, index)
                        #If input was not valid. Let the user know and have them select
again.
                        else:
                            print("Not the correct selection")
                            MakeTheSelection = 'yes'
                index = index + 1
                if index == len(ContactList):
                    InContacts = 'no'

#The user has no more information that they need or does not need to edit a contact
def ExitContactList():
    KeepSelecting = 'no'
    return KeepSelecting

#main program. keep looping till the user is done
while KeepSelecting == 'yes':
    #display the contact list
    DisplayContactList()
    # Function to accept the user input.
    UserSelection()
    #Check that the input was valid
    if UserInput in AcceptableEntries:
        #User would like to display contact by entering the name
        if UserInput == 'a':
            DisplaySelectedContact(UserInput)
        #User would like to display contact by entering the number
        elif UserInput == 'b':
            DisplaySelectedContact(UserInput)
        #User would like to edit a contact.
        elif UserInput == 'c':
            EditContact()
        #User would like to exit the program
        elif UserInput == 'd':
            KeepSelecting = ExitContactList()
    #If selection is not valid let the user know.
    else:
        print("User Entry is Not Valid. Please Select either a, b, c, or d")
```

4.3 Question 3

Develop an airline booking reservation system by implementing 5 different classes. In addition, the program should demonstrate inheritance both single inheritance and multiple inheritance. The program also should have a private variable, uses __init__(self,….) and the super() function associated with classes.

4.3.1 Approach

The five (5) different classes that were used were Flight, Person, Employee, Arrival Time, and Passenger. Each class has a constructor that initializes different attributes. The following is the Person class:

```
class Person:
    # Constructor to initialize the first and last name of the
person
    persontotal = 0
    def __init__(self, name, family):
        #Person's first name
        self.name = name
        #Person'ts last name
        self.lastname = family
```

The __init__ is the constructor used in python. The following is an example of how inheritance is used:

```
#Creating the Passenger Class and it inherits both the methods for
the Person class and ArrivalTime Class
class Passenger(Person,ArrivalTime):
    # Constructor to initialize the first and last name of the
passenger, the flight number, and the arrival time
    def __init__(self, name, family, number, time):
        #Constructor to set the same method's as Person Class.
        Person.__init__(self, name, family)
        #Constructor to set the same method's as ArrivalTime
Class.
        ArrivalTime.__init__(self, number, time)
```

To show whether or not the inheritance was success, hasattr() was used which returns “True” if the class does have the attribute.

4.3.2 Full Source Code

```
#Airline Booking Reservation System
#Creating the flight class
class Flight:
    #Constructor to initialize the flight number
    def __init__(self, number):
        #flight number
        self.number = number
#Creating the person class
class Person:
    # Constructor to initialize the first and last name of the person
    persontotal = 0
    def __init__(self, name, family):
        #Person's first name
        self.name = name
        #Person'ts last name
        self.lastname = family

        Person.persontotal = Person.persontotal + 1
    #Create a private variable that stores the number of people
    __privatetotal = persontotal
#Creating the employee class
class Employee:
    # Constructor to initialize the first and last name of the employee along with
the employee number
    def __init__(self, name, family, number):
        #Employee's first name
        self.name = name
        #Employee's last name
        self.family = family
        #Employee's number
        self.employeenumber = number
```

```
#Creating the Arrival Time Class and it inherits the Flight Class methods
class ArrivalTime(Flight):
    # Constructor to initialize the flight number and the arrival time of the
flight
    def __init__(self, number: object, time: object) -> object:
        #Using super function to set the method for the flight number
        super().__init__(number)
        #Arrival Time
        self.time = time

#Creating the Passenger Class and it inherits both the methods for the Person class
and ArrivalTime Class
class Passenger(Person,ArrivalTime):
    # Constructor to initialize the first and last name of the passenger, the
flight number, and the arrival time
    def __init__(self, name, family, number, time):
        #Constructor to set the same method's as Person Class.
        Person.__init__(self, name, family)
        #Constructor to set the same method's as ArrivalTime Class.
        ArrivalTime.__init__(self, number, time)

#Create an instance for the Flight Class
flight1 = Flight(123)
#Create an instance for the Person Class
person1 = Person('Jane', 'Doe')
#Create an instance for the Employee Class
employee1 = Employee('Bob', 'Smith', 567)
#Create an instance for the Arrival Time Class
arrival = ArrivalTime(456, '4:00 pm')
#Create an instance for the Passenger Class
passenger1 = Passenger('Jane', 'Doe', 123, '4:00 pm')

#Checking what attributes are in what class.
print('Does Flight, ArrivalTime, and Passenger classes have the attribute
"number":')
if (hasattr(flight1, 'number')) == (hasattr(arrival, 'number')):
    if (hasattr(passenger1, 'number')) == True:
        print('Yes they do.')
    else:
        print('No they do not.')

print('Does ArrivalTime and Passenger classes have the attribute "arrival time":')
if (hasattr(arrival, 'time')) == (hasattr(passenger1, 'time')):
    print('Yes they do.')
else:
    print('No they do not.')

print('Does Person and Passenger classes have attributes "name" and "family":')
if (hasattr(person1, 'name')) == (hasattr(passenger1, 'name')):
    if (hasattr(person1, 'family')) == (hasattr(passenger1, 'family')):
        print('Yes they do.')
    else:
        print('They only have the "name" attribute.')
else:
    print('No they do not.')
```

4.4 Question 4

Use the numpy library to generate a random array of 15 integers from 0 to 20. The program then should determine which integer has the highest frequency in the array and display the

number. If there are more than one number with the highest frequency both numbers will be displayed.

4.4.1 Approach

To generate a random array with 15 values of integers from 0 to 20 the following code was used:

```
a = np.random.randint(0, high = 20, size = (15))
```

Another array is create with the frequency of each number using bincount().

```
y = np.bincount(a)
```

In case there are multiple numbers with the same frequency each number in the frequency array is looked at and compared to the previous frequency. If the frequency number is larger than the previous frequency, the new frequency is saved into a list. If the list contains one number, then the following code is implemented:

```
z = np.argmax(y)
print('The most frequenct number is:')
print(z)
```

If the list contains more than one number then each number is displayed.

```
else:
    freqnumb = []
    yindex = 0
    # Iterate over each element in the array y
    for w in y:
        #Check to see if the value saved in temp is in y
        if temp[0] == w:
            #If so then place index of value into a list
            freqnumb.append(yindex)
        yindex = yindex + 1
    print('The most frequenct numbers are:')
    print(freqnumb)
```

4.4.2 Full Source Code

```
#Using numpy to create a random array
import numpy as np
#Create an array of 15 random interger numbers from 0 to 20.
a = np.random.randint(0, high = 20, size = (15))
print(a)
print('\n')
#Count the frequency of each number.
y = np.bincount(a)

index = 0
num = 0
temp = []

#Iterate over each element in the array y
for number in y:
    #If it is the first iteration than save the first element of y into a temporary
```

```
list
    if index == 0:
        temp.clear()
        temp.append(y[index])
    #Check to see if there is a different element that is larger than the previous
one.
    elif temp[num] < (y[index]):
        temp.clear()
        #If number in y is larger than number in temp. Clear temp and save number
in y into temp list.
        temp.append(y[index])
    #Check to see if there is an element that is equal to the number in the temp
list
    elif (index != 0) and (temp[num] == (y[index])):
        #Place the number into the temp list.
        temp.append(y[index])
    index = index + 1

#If temporary list only has one number than print argmax function will work.
if len(temp) == 1:
    #Find the largest number in y and print it
    z = np.argmax(y)
    print('The most frequenct number is:')
    print(z)
else:
    freqnumb = []
    yindex = 0
    # Iterate over each element in the array y
    for w in y:
        #Check to see if the value saved in temp is in y
        if temp[0] == w:
            #If so then place index of value into a list
            freqnumb.append(yindex)
        yindex = yindex + 1
    print('The most frequenct numbers are:')
    print(freqnumb)
```

5.0 Deployment

The programs are written in python 3 which can be installed from the following link: https://www.python.org/downloads/. Next, the source code can be downloaded from the following location: https://github.com/plfoley10/UMKC-CSEE-5590---Labs/tree/master/Lab2/Source. Lastly, run the program in python.

6.0 Limitation

Multiple programs use loops. By using loops this slows down the execution of the code.

6.1 For Question 1, the program ends without displaying “No books within the selected range.” In addition, if a user selected a range higher than the cost of books available a suggestion should be made to lower the range. Lastly, the user should be prompted again to see if the selection was what the user wanted to see.

6.2 For Question 2, the program does not check to see if the number of two individuals is the same. Since the number is for a phone two contacts would not have the same number.

6.3 For Question 3, the private variable is not able to display which is a characteristic of a private number. Which is beneficial when the public should not be able to access the

variable. However, for debugging purposes it would be easier if the private variable could be displayed.

6.4 For Question 4, a user cannot enter a random array. To fix this limitation a function could initially ask the user if the user would like to enter random numbers or have the array computer generated.

7.0 References

https://github.com