```
# -Exercises4.py *- coding: utf-8 -*-
"""
@author: Bill
"""
"""
LONG STRINGS
"""
#%%
lstr1 = "A very long string can be tied together using " + \
        "a backslash (also called the escape character)."
lstr2 = ("A very long string can also be tied together using "
         "parentheses to enclose the various parts.")
#%%
"""
BUILDING LISTS OF RANDOM NUMBERS; RETURNING FUNCTIONAL VALUES

Sometimes you need a list of numbers. Here is a way to build a list of pseudo-
random numbers. We again import the python library random. It furnishes a
random integer generator called randint().

Note the basic loop design pattern reappears.

The function make_random() below builds a list of random integers using the
randint() function from the random library.  Note that we do not print the list
out as we usually do.  Instead we use another function to call the make_random()
function and print it in the calling function.
"""
#%%

import random

def make_random():
    """ Make a list of 10 random integers between 1 and 100. """
    numlis = []
    for i in range(0,10):
        numlis.append(random.randint(1,100))
    return numlis

def call_make_random():
    """ Uses make_random to get a list of random numbers """
    random_integers = make_random()
    print("The list of random numbers is",random_integers)
#%%

"""
For testing your program, however, it can be deadly to have a different set of
random numbers each time it is run. That's a formula for madness: sometimes
your program may work and sometimes not, depending on the particular random
numbers that happen to be generated on that run. Consequently, it can be very
hard to track down the error.  For testing purposes, you can generate the same
random numbers over and over again by setting the random number generator's
seed to the same value each time.
"""
#%%

import random

def make_same_random():
    """ Make a list of 10 random integers that are the same each time """
    numlis = []
    random.seed(17)        # set the seed from which random numbers are made
    for i in range(0,10):
        numlis.append(random.randint(1,100))
    return numlis
```

```
def call_make_random():
    """ Uses make_same_random to get a list of random numbers """
    random_integers = make_same_random()
    print(random_integers)
    random_integers1 = make_same_random()
    print(random_integers1)

#%%
"""
Exercise:
Write a function make_random_real() that uses random.random() in place of
random.randint() to build a list of 10 random reals and returns that list.
random.random() generates a random number between 0 and 1.
Note: we want to return a list not print it.
Here is my run.  Your list of random reals will likely be different from mine:
In [2]: make_random_real()
Out[2]:
[0.6069930611672794,
 0.9812910762564292,
 0.4290994419220008,
 0.9957161016532591,
 0.005874475115656863,
 0.5329633233660277,
 0.7662656130982124,
 0.8460145442822357,
 0.05511562729749986,
 0.009731494763540849]

"""
"""
Solution:
"""
#%%









#%%
"""
End solution
"""

"""
Exercise:
Rewrite make_random_real() using random.seed() to get the same random reals
each time. Run the function twice to show that you get the same set of
"random" numbers. A correct solution will display the same list each time it
is run.  Return the list of random numbers rather than printing them.
Here are a couple of my runs using 17 as the seed:
make_same_random_real()
Out[45]:
[0.5219839097124932,
 0.8066907771186791,
 0.9604947743238768,
 0.2896253777644655,
 0.7661074377979527,
 0.7042198668434126,
 0.6613830572238304,
 0.11016204891721182,
 0.026936778790526805,
```

```
 0.3841711045442975]

make_same_random_real()
Out[46]:
[0.5219839097124932,
 0.8066907771186791,
 0.9604947743238768,
 0.2896253777644655,
 0.7661074377979527,
 0.7042198668434126,
 0.6613830572238304,
 0.11016204891721182,
 0.026936778790526805,
 0.3841711045442975]

"""
"""
Solution:
"""
#%%









#%%
"""
End solution
"""
"""
SORTING LISTS

Exercise: Sorting lists, including numbers as well as lower and upper case
letters and strings.

Do with me. Use the line of code below to create a list.
"""
#%%
numlist = [67, 54, 39, 47, 38, 23, 99, 91, 91, 70]
#%%
"""
We use a method of lists to sort numlist.  It permanently reorders the list.
"""
print(numlist)
numlist.sort()
print(numlist)
#%%
"""
Note that we already have a way of doing this sort. This doesn't permanently
change the list.
"""
print(sorted(numlist))
#%%
"""
Below we make a random list of 10 letters of the alphabet. By setting the
random seed, we insure that it generates the same list every time it is run.
"""
#%%
def make_alpha_list():
```

```
    import random
    alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o',
                'p','q','r','s','t','u','v','w','x','y','z']
    random.seed(17)

    alpha_list = []
    for i in range(0,10):
        alpha_list.append(random.choice(alphabet))
    return alpha_list
#%%
"""
Here we give the variable alphlist the value returned by make_alpha_list()
"""
alphlist = make_alpha_list()
#%%
alphlist = ['q', 'n', 'z', 'j', 'l', 'j', 'f', 'y', 'w', 'w']
#%%
print(alphlist)
alphlist.sort()
print(alphlist)
#%%
Alphlist = ['e', 'F', 'h', 'A', 'D', 'F', 'b', 'D', 'b', 'J']
#%%
# notice following is unsatisfactory for sorting mixed upper/lower case
print(Alphlist)
Alphlist.sort()
print(Alphlist)
#%%
# specifying the proper key fixes the problem
print(Alphlist)
Alphlist.sort(key=str.lower)
print(Alphlist)
#%%
strlist = ['now', 'is', 'the', 'time', 'for', 'all',
    'good','men', 'to', 'come', 'to', 'the','aid', 'of', 'their', 'country']
#%%
print(strlist)
strlist.sort()
print(strlist)
#%%
Strlist = ['Now', 'is', 'The', 'time', 'For', 'All',
    'Good','men', 'To', 'come', 'to', 'the','aid', 'of', 'Their', 'country']
#%%
# Note that all capital letters sort before lower case
print(Strlist)
Strlist.sort()
print(Strlist)
#%%
# this treats all capital letters as if they were lower case
print(Strlist)
Strlist.sort(key=str.lower)
print(Strlist)
#%%

"""
DESCRIPTIVE STATISTICS

  Go to docs.python.org/3/ and search statistics. Go to that library.
  You can alternatively, go to Help>Python Documention in Spyder and search
  statistics in the index.  You see the same thing.
"""
#%%
nlist = [2, 4, 13, 3, 7, 8, 5]
nlisteven = nlist + [9]
rlist = [3.14, 2.71, -8.43, 5.25, 10.11, -23.78, 44.45]
```

```
rlisteven = rlist + [90.3]
#%%
import statistics
#%%
statistics.mean(nlist)
#%%
statistics.mean(rlist)
#%%
print(nlist)
print(sorted(nlist))    # print sorted version of nlisteven to see median
#%%
statistics.median(nlist)
#%%
print(nlisteven)
print(sorted(nlisteven))    # print sorted version of nlisteven to see median
#%%
statistics.median(nlisteven)  # note that it averages the two middle values
#%%
print(rlist)
print(sorted(rlist))       # print sorted version of rlist to see median
#%%
statistics.median(rlist)
#%%
print(rlisteven)
print(sorted(rlisteven))      # print sorted version of rlisteven to see median
#%%
statistics.median(rlisteven)  # note that it averages the two middle values
#%%
mlist = [2, 3, 4, 5, 4, 5, 3, 6, 1, 3, 4, 3]
#%%
print(mlist)
print(sorted(mlist))       # print sorted version of mlist to see mode
#%%
statistics.mode(mlist)
#%%
statistics.stdev(rlist)
#%%
statistics.variance(rlist)
#%%
"""
list functions  --- max, min, sum
"""
#%%
nlis = [3.14,-4,25,8,106,32]
print("the maximum of nlis is",max(nlis))
print("the minimum of nlis is",min(nlis))
print("the sum of nlis is",sum(nlis))
#%%
"""
USING DESCRIPTIVE STATISTICS; TRY/EXCEPT ERROR HANDLING
Example: Compute the statistics for the following list
"""
""" First let's generate a 100 random reals between 5000 and 6000 """
#%%
import random
stat_list = []
for i in range(0,100):
    stat_list.append(1000*random.random()+5000)

ilis = [3, 1, 5, 2, 1, 3, 7, 3]
#%%
def my_stats(slis):
    import statistics
    print("Mean: ", statistics.mean(slis))
    print("Median: ", statistics.median(slis))
```

```
    print("Mode: ", statistics.mode(slis))
#    try:
#        print("Mode: ", statistics.mode(slis))
#    except statistics.StatisticsError as e:
#        print("Mode error: ", e)
    print("Standard Deviation: ", statistics.stdev(slis))
    print("Variance: ", statistics.variance(slis))

#%%
"""
A simple example of try/except error catching. If the user inputs a number
then all is fine; if the user enters a non-number, then the exception is
caught and printed out.
"""
#%%
def test_try():
    numb = input("Enter a number: ")
    print(type(numb))
    try:
        num = float(numb)
        print(num)
    except Exception as e:
        print("Exception was: ", e)
#%%
"""
Exercise:
Write a function temp_stat(temps) to compute the average, median, standard
deviation and variance of the temperatures in the table.  Print each out.
The following code generates the same temperatures each time because the seed
is set. Print the temperature list as the first line of the function.

Here is what my run on the table of temperatures built below looks like:

temp_stat(temperatures)
[52, 61, 45, 50, 44, 34, 57, 80, 91, 50, 38, 91, 84, 20, 55, 23, 83, 42, 44, 84]
Mean:  56.4
Median:  51.0
Standard deviation:  22.04397518836526
Variance:  485.9368421052631

"""
#%%
import random
random.seed(77)
temperatures = []
for i in range(0,20):
    temperatures.append(random.randint(20,95))
#%%
"""
Solution:
"""
#%%
def temp_stat(temps):
    """ prints the average, median, std dev, and variance of temps """
    pass # replace this pass (a do-nothing) statement with your code
```

```
#%%
"""
End solution
"""
"""
Exercise:
Add computing 'mode' to your solution to last exercise.  In the temperature
list that we constructed, there is no unique mode, so that the program will
crash unless you use try/except error handling.  See if you can add this
feature to your solution.

Note: if you change the seed to 277, then you will get a list that does have
a unique mode.  You might like to try that.

Here is a run of my solution:

temp_stat(temperatures)
[52, 61, 45, 50, 44, 34, 57, 80, 91, 50, 38, 91, 84, 20, 55, 23, 83, 42, 44, 84]
Mean:  56.4
Median:  51.0
Standard deviation:  22.04397518836526
Variance:  485.9368421052631
Mode error:  no unique mode; found 4 equally common values

"""
"""
Solution:
"""
#%%
def temp_stat(temps):
    """ computes the average, median, std dev, and variance of temps """
    pass # replace this pass (a do-nothing) statement with your code









#%%

"""
FORMATING -- A BRIEF INTRODUCTION

Let's define a string and use the format method of strings to make a spiffier
printout. Suppose we wanted to print a person's name, age, and weight. Here is
a very brief introduction using name, age, and weight to illustrate.
"""
#%%
nam1 = '"Teddy" Roosevelt'
nam2 = 'Theodore "Teddy" Roosevelt'
age = 60
wt = 199.1515115
#%%
# minimal formating -- {n} represents data item n --- notice misalignment
out1 = "name: {0}  age: {1}  weight: {2}"
#%%
print("name: {0}  age: {1}  weight: {2}".format(nam1,age,wt))
print("name: {0}  age: {1}  weight: {2}".format(nam2,age,wt))
#%%
```

```
# better formatting > means right align (< and ^ would be left and center)
# the numbers represent the minimum characters occupied by the datum.
out2 = "name: {0:>26}  age: {1:>4}   weight: {2:>10}"
#%%
print("name: {0:>26}  age: {1:>4}   weight: {2:>10}".format(nam1,age,wt))
print("name: {0:>26}  age: {1:>4}   weight: {2:>10}".format(nam2,age,wt))
#%%
# for the float (real number), allocate 5 spaces with 2 to the right of "."
out3 = "name: {0:>26}  age: {1:>4}   weight: {2:>5.2f}"
#%%
print("name: {0:>26}  age: {1:>4}   weight: {2:>5.2f}".format(nam1,age,wt))
print("name: {0:>26}  age: {1:>4}   weight: {2:>5.2f}".format(nam2,age,wt))
#%%
"""
Note: we can use the string variable out3 in the print statement:
"""
#%%
print(out3.format(nam1,age,wt))
print(out3.format(nam2,age,wt))
#%%
s = "my short string"
n = 5.1234
#%%
print("Start||{0:25}||End".format(s))
#%%
#%%
print("Start||{0:25}||End".format(n))
#%%
"""
Exercise:
Execute the following string and use it to try some of the following in the
print("Start||{???}||End".format(s)) statement that would enable you to see
what the format has done.  You can change what is in the {0:  } quickly,
Ctrl-Enter to execute and see the result.  Or you can up-arrow and modify
the previous version.
a) Use {0} to just print the string with no formating; {} does the same thing
b) Use {0:25} to print the string allowing a width of 25 characters for it
c) Use {0:>25} to print the string right-aligned in a space of 25 characters
d) Use {0:<25} to print the string left-aligned in a space of 25 characters
e) Use {0:^25} to print the string centered in a space of 25 characters

Here is what my output looked like:
Start||hello, there||End
Start||hello, there             ||End - minimum width 25
Start||             hello, there||End - width 25, right aligned
Start||hello, there             ||End - width 25, left aligned
Start||      hello, there       ||End - width 25, centered
"""
#%%
""" Execute this.  It is the string to work with: """
#%%
s = "hello, there"
#%%
""" The format statement to modify and work with: """
#%%
print("Start||{}||End".format(s))
#%%
"""
Solution:
"""
s = "hello, there"
```

```
"""
End Solution
"""
#%%

"""
A SMALL DATABASE APPLICATION USING WHAT WE'VE LEARNED.

The following uses a CSV file to store the data for a small telephone directory.
It is menu driven. That is, it presents a menu of choices. These choices are
to show the whole directory, create a new entry for the directory, delete an
entry from the directory, and edit an entry in the directory.
"""
#%%
# -phones.py *- coding: utf-8 -*-
"""
This was written as a basic program that did little at first.  Additional
features were added until it was finished.
Here is a first version.
phones.py
Version 1 -- builds the menu -- functions are empty
"""

def delete_phone():
    print("Deleting")

def edit_phone():
    print("Editing")

def save_phone_list():
    print("Saving")

def load_phone_list():
    print("Loading")

def show_phones():
    print("Showing phones")

def create_phone():
    print("Adding a phone")

def menu_choice():
    """ Find out what the user wants to do next. """
    print("Choose one of the following options?")
    print("   s) Show")
    print("   n) New")
    print("   d) Delete")
    print("   e) Edit")
    print("   q) Quit")
    choice = input("Choice: ")
    if choice.lower() in ['n','d', 's','e', 'q']:
        return choice.lower()
    else:
        print(choice +"?" + " That is an invalid option!!!")
        return None


def main_loop():

    load_phone_list()

    while True:
        choice = menu_choice()
        if choice == None:
```

```
            continue
        if choice == 'q':
            print( "Exiting...")
            break     # jump out of while loop
        elif choice == 'n':
            create_phone()
        elif choice == 'd':
            delete_phone()
        elif choice == 's':
            show_phones()
        elif choice == 'e':
             edit_phone()
        else:
            print("Invalid choice.")

    save_phone_list()


# The following makes this program start running at main_loop()
# when executed as a stand-alone program.
if __name__ == '__main__':
    main_loop()
#%%
# -phones.py *- coding: utf-8 -*-
"""
Version 2 -- show phones so we can see that the other functions work
"""

phones = [['Jerry Seinfeld', '(212) 344-3784'],
          ['Cosmo Kramer', '(212) 559-8185']]
name_pos = 0
phone_pos = 1
phone_header = [ 'Name', 'Phone Number']


def delete_phone():
    print("Deleting")

def edit_phone():
    print("Editing")


def save_phone_list():
    print("saving")

def load_phone_list():
    print("loading")

def show_phones():
    show_phone(phone_header, "")
    index = 1
    for phone in phones:
        show_phone(phone, index)
        index = index + 1
    print()

def show_phone(phone, index):
    outputstr = "{0:>3}  {1:<20}  {2:>16}"
    print(outputstr.format(index, phone[name_pos], phone[phone_pos]))

def create_phone():
    print("adding a phone")

def menu_choice():
    """ Find out what the user wants to do next. """
```

```
    print("Choose one of the following options?")
    print("   s) Show")
    print("   n) New")
    print("   d) Delete")
    print("   e) Edit")
    print("   q) Quit")
    choice = input("Choice: ")
    if choice.lower() in ['n','d', 's','e', 'q']:
        return choice.lower()
    else:
        print(choice +"?" + " That is an invalid option!!!")
        return None

def main_loop():

    load_phone_list()

    while True:
        choice = menu_choice()
        if choice == None:
            continue
        if choice == 'q':
            print( "Exiting...")
            break     # jump out of while loop
        elif choice == 'n':
            create_phone()
        elif choice == 'd':
            delete_phone()
        elif choice == 's':
            show_phones()
        elif choice == 'e':
             edit_phone()
        else:
            print("Invalid choice.")

    save_phone_list()


# The following makes this program start running at main_loop()
# when executed as a stand-alone program.
if __name__ == '__main__':
    main_loop()
#%%
# -phones.py *- coding: utf-8 -*-
"""
Version 3 -- create phone; delete phone
"""

phones = [['Jerry Seinfeld', '(212) 344-3784'],
          ['Cosmo Kramer', '(212) 559-8185']]
name_pos = 0
phone_pos = 1
phone_header = [ 'Name', 'Phone Number']

def proper_menu_choice(which):
    if not which.isdigit():
        print ("'" + which + "' needs to be the number of a phone!")
        return False
    which = int(which)
    if which < 1 or which > len(phones):
        print ("'" + str(which) + "' needs to be the number of a phone!")
        return False
    return True

def delete_phone(which):
```

```
    if not proper_menu_choice(which):
        return
    which = int(which)

    del phones[which-1]
    print( "Deleted phone #", which)

def edit_phone():
    print("Editing")


def save_phone_list():
    print("saving")

def load_phone_list():
    print("loading")

def show_phones():
    show_phone(phone_header, "")
    index = 1
    for phone in phones:
        show_phone(phone, index)
        index = index + 1
    print()

def show_phone(phone, index):
    outputstr = "{0:>3}  {1:<20}  {2:>16}"
    print(outputstr.format(index, phone[name_pos], phone[phone_pos]))

def create_phone():
    print("Enter the data for a new phone:")
    newname = input("Enter name: ")
    newphone_num = input("Enter phone number: ")
    phone = [newname,newphone_num]
    phones.append(phone)
    print()

def menu_choice():
    """ Find out what the user wants to do next. """
    print("Choose one of the following options?")
    print("   s) Show")
    print("   n) New")
    print("   d) Delete")
    print("   e) Edit")
    print("   q) Quit")
    choice = input("Choice: ")
    if choice.lower() in ['n','d', 's','e', 'q']:
        return choice.lower()
    else:
        print(choice +"?" + " That is an invalid option!!!")
        return None


def main_loop():

    load_phone_list()

    while True:
        choice = menu_choice()
        if choice == None:
            continue
        if choice == 'q':
            print( "Exiting...")
            break     # jump out of while loop
        elif choice == 'n':
```

```
            create_phone()
        elif choice == 'd':
            which = input("Which item do you want to delete? ")
            print("which is ", which)
            delete_phone(which)
        elif choice == 's':
            show_phones()
        elif choice == 'e':
             edit_phone()
        else:
            print("Invalid choice.")

    save_phone_list()


# The following makes this program start running at main_loop()
# when executed as a stand-alone program.
if __name__ == '__main__':
    main_loop()
#%%
# -phones.py *- coding: utf-8 -*-
"""
Version 4 -- edits a phone
"""

phones = [['Jerry Seinfeld', '(212) 344-3784'],
          ['Cosmo Kramer', '(212) 559-8185']]
name_pos = 0
phone_pos = 1
phone_header = [ 'Name', 'Phone Number']

def proper_menu_choice(which):
    if not which.isdigit():
        print ("'" + which + "' needs to be the number of a phone!")
        return False
    which = int(which)
    if which < 1 or which > len(phones):
        print ("'" + str(which) + "' needs to be the number of a phone!")
        return False
    return True

def delete_phone(which):
    if not proper_menu_choice(which):
        return
    which = int(which)

    del phones[which-1]
    print( "Deleted phone #", which)

def edit_phone(which):
    if not proper_menu_choice(which):
        return
    which = int(which)

    phone = phones[which-1]
    print("Enter the data for a new phone. Press <enter> to leave unchanged.")

    print(phone[name_pos])
    newname = input("Enter phone name to change or press return: ")
    if newname == "":
        newname = phone[name_pos]

    print(phone[phone_pos])
    newphone_num = input("Enter new phone number to change or press return: ")
    if newphone_num == "":
```

```
        newphone_num = phone[phone_pos]

    phone = [newname, newphone_num]
    phones[which-1] = phone


def save_phone_list():
    print("saving")

def load_phone_list():
    print("loading")

def show_phones():
    show_phone(phone_header, "")
    index = 1
    for phone in phones:
        show_phone(phone, index)
        index = index + 1
    print()

def show_phone(phone, index):
    outputstr = "{0:>3}  {1:<20}  {2:>16}"
    print(outputstr.format(index, phone[name_pos], phone[phone_pos]))

def create_phone():
    print("Enter the data for a new phone:")
    newname = input("Enter name: ")
    newphone_num = input("Enter phone number: ")
    phone = [newname,newphone_num]
    phones.append(phone)

def menu_choice():
    """ Find out what the user wants to do next. """
    print("Choose one of the following options?")
    print("   s) Show")
    print("   n) New")
    print("   d) Delete")
    print("   e) Edit")
    print("   q) Quit")
    choice = input("Choice: ")
    if choice.lower() in ['n','d', 's','e', 'q']:
        return choice.lower()
    else:
        print(choice +"?" + " That is an invalid option!!!")
        return None


def main_loop():

    load_phone_list()

    while True:
        choice = menu_choice()
        if choice == None:
            continue
        if choice == 'q':
            print( "Exiting...")
            break     # jump out of while loop
        elif choice == 'n':
            create_phone()
        elif choice == 'd':
            which = input("Which item do you want to delete? ")
            print("which is ", which)
            delete_phone(which)
        elif choice == 's':
```

```
            show_phones()
        elif choice == 'e':
            which = input("Which item do you want to edit? ")
            print("which is ", which)
            edit_phone(which)
        else:
            print("Invalid choice.")

    save_phone_list()


# The following makes this program start running at main_loop()
# when executed as a stand-alone program.
if __name__ == '__main__':
    main_loop()
#%%
# -phones.py *- coding: utf-8 -*-
"""
Version 5 -- saves to a csv file called myphones.csv
"""
import csv


phones = [['Jerry Seinfeld', '(212) 344-3784'],
          ['Cosmo Kramer', '(212) 559-8185']]
name_pos = 0
phone_pos = 1
phone_header = [ 'Name', 'Phone Number']

def proper_menu_choice(which):
    if not which.isdigit():
        print ("'" + which + "' needs to be the number of a phone!")
        return False
    which = int(which)
    if which < 1 or which > len(phones):
        print ("'" + str(which) + "' needs to be the number of a phone!")
        return False
    return True

def delete_phone(which):
    if not proper_menu_choice(which):
        return
    which = int(which)

    del phones[which-1]
    print( "Deleted phone #", which)

def edit_phone(which):
    if not proper_menu_choice(which):
        return
    which = int(which)

    phone = phones[which-1]
    print("Enter the data for a new phone. Press <enter> to leave unchanged.")

    print(phone[name_pos])
    newname = input("Enter phone name to change or press return: ")
    if newname == "":
        newname = phone[name_pos]

    print(phone[phone_pos])
    newphone_num = input("Enter new phone number to change or press return: ")
    if newphone_num == "":
        newphone_num = phone[phone_pos]
```

```
    phone = [newname, newphone_num]
    phones[which-1] = phone


def save_phone_list():

    f = open("myphones.csv", 'w', newline='')
    for item in phones:
        csv.writer(f).writerow(item)
    f.close()

def load_phone_list():
    print("loading")

def show_phones():
    show_phone(phone_header, "")
    index = 1
    for phone in phones:
        show_phone(phone, index)
        index = index + 1
    print()

def show_phone(phone, index):
    outputstr = "{0:>3}  {1:<20}  {2:>16}"
    print(outputstr.format(index, phone[name_pos], phone[phone_pos]))

def create_phone():
    print("Enter the data for a new phone:")
    newname = input("Enter name: ")
    newphone_num = input("Enter phone number: ")
    phone = [newname,newphone_num]
    phones.append(phone)

def menu_choice():
    """ Find out what the user wants to do next. """
    print("Choose one of the following options?")
    print("   s) Show")
    print("   n) New")
    print("   d) Delete")
    print("   e) Edit")
    print("   q) Quit")
    choice = input("Choice: ")
    if choice.lower() in ['n','d', 's','e', 'q']:
        return choice.lower()
    else:
        print(choice +"?" + " That is an invalid option!!!")
        return None


def main_loop():

    load_phone_list()

    while True:
        choice = menu_choice()
        if choice == None:
            continue
        if choice == 'q':
            print( "Exiting...")
            break     # jump out of while loop
        elif choice == 'n':
            create_phone()
        elif choice == 'd':
            which = input("Which item do you want to delete? ")
            print("which is ", which)
```

```
            delete_phone(which)
        elif choice == 's':
            show_phones()
        elif choice == 'e':
            which = input("Which item do you want to edit? ")
            print("which is ", which)
            edit_phone(which)
        else:
            print("Invalid choice.")

    save_phone_list()


# The following makes this program start running at main_loop()
# when executed as a stand-alone program.
if __name__ == '__main__':
    main_loop()

#%%
# -phones.py *- coding: utf-8 -*-
"""
This program maintains a database of names and phone numbers in a csv
file called myphones.csv.  It is run from the command line and is menu
driven.  To start it, save it in a directory and from the terminal run
>python phones.py
It can also be run from this editor in the usual way.
Version FINAL:
"""
import os
import csv


phones = []
name_pos = 0
phone_pos = 1
phone_header = [ 'Name', 'Phone Number']

def proper_menu_choice(which):
    if not which.isdigit():
        print ("'" + which + "' needs to be the number of a phone!")
        return False
    which = int(which)
    if which < 1 or which > len(phones):
        print ("'" + str(which) + "' needs to be the number of a phone!")
        return False
    return True

def delete_phone(which):
    if not proper_menu_choice(which):
        return
    which = int(which)

    del phones[which-1]
    print( "Deleted phone #", which)

def edit_phone(which):
    if not proper_menu_choice(which):
        return
    which = int(which)

    phone = phones[which-1]
    print("Enter the data for a new phone. Press <enter> to leave unchanged.")

    print(phone[name_pos])
    newname = input("Enter phone name to change or press return: ")
```

```
    if newname == "":
        newname = phone[name_pos]

    print(phone[phone_pos])
    newphone_num = input("Enter new phone number to change or press return: ")
    if newphone_num == "":
        newphone_num = phone[phone_pos]

    phone = [newname, newphone_num]
    phones[which-1] = phone


def save_phone_list():

    f = open("myphones.csv", 'w', newline='')
    for item in phones:
        csv.writer(f).writerow(item)
    f.close()

def load_phone_list():
    if os.access("myphones.csv",os.F_OK):
        f = open("myphones.csv")
        for row in csv.reader(f):
            phones.append(row)
        f.close()

def show_phones():
    show_phone(phone_header, "")
    index = 1
    for phone in phones:
        show_phone(phone, index)
        index = index + 1
    print()

def show_phone(phone, index):
    outputstr = "{0:>3}  {1:<20}  {2:>16}"
    print(outputstr.format(index, phone[name_pos], phone[phone_pos]))

def create_phone():
    print("Enter the data for a new phone:")
    newname = input("Enter name: ")
    newphone_num = input("Enter phone number: ")
    phone = [newname,newphone_num]
    phones.append(phone)

def menu_choice():
    """ Find out what the user wants to do next. """
    print("Choose one of the following options?")
    print("   s) Show")
    print("   n) New")
    print("   d) Delete")
    print("   e) Edit")
    print("   q) Quit")
    choice = input("Choice: ")
    if choice.lower() in ['n','d', 's','e', 'q']:
        return choice.lower()
    else:
        print(choice +"?")
        print("Invalid option")
        return None


def main_loop():

    load_phone_list()
```

```
    while True:
        choice = menu_choice()
        if choice == None:
            continue
        if choice == 'q':
            print( "Exiting...")
            break     # jump out of while loop
        elif choice == 'n':
            create_phone()
        elif choice == 'd':
            which = input("Which item do you want to delete? ")
            print("which is ", which)
            delete_phone(which)
        elif choice == 's':
            show_phones()
        elif choice == 'e':
            which = input("Which item do you want to edit? ")
            print("which is ", which)
            edit_phone(which)
        else:
            print("Invalid choice.")

    save_phone_list()


# The following makes this program start running at main_loop()
# when executed as a stand-alone program.
if __name__ == '__main__':
    main_loop()
#%%
```