New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Introduction to Python
Exercises, Session 4

Ex. 4.1 Open the file passwords.txt. Then, using the open filehandle, without looping, print out a list of all the lines in the file. (Hint: use the file readlines() method).

Expected Output:

```
['ap172:jk45$JK\n', 'jab44:lucky1\n', 'axe99:pepper11\n',
 'jk43:godaddy3\n', 'jb23:password\n']
```

Ex. 4.2 Extending the above program, again without looping, print out a list of just the 1st three lines of the file. (Hint: use a slice of the list returned from readlines()).

Expected Output:

```
['ap172:jk45$JK\n', 'jab44:lucky1\n', 'axe99:pepper11\n']
```

Ex. 4.3 Create an empty list. Opening and looping through the file student_db.txt, split each line into elements, and append just the state values (the 4th value in each row) to a list. Print the list object. (Hint: make sure the list is initialized before the loop begins and is printed after the loop ends. If either statement is found inside the loop, it will be executed multiple times.

Expected Output:

```
['state', 'NY', 'NY', 'NY', 'PA', 'NY', 'NJ', 'NJ']
```

Ex. 4.4 Extend the above program by omitting the first line from the file. The proper way to do this is to call readlines() on the file handle, assign this to a list variable, take a slice of the resulting list of lines omitting the first element (lines[1:]), and then loop through this list.

Expected Output:

```
['NY', 'NY', 'NY', 'PA', 'NY', 'NJ', 'NJ']
```

Ex. 4.5 Starting with an empty list, and then opening and looping through student_db.txt, append the 1st field (a student ID) to a list only if the state field is NY. Print the list object.

Expected Output:

```
['jk43', 'axe99', 'jab44', 'ap172']
```

Ex. 4.6 Opening and reading through revenue.txt, build a list from the 3rd field (the floating point value), then sum up the values using sum(). (Hint: sum() will not work unless you convert each element to a float value before you add it to the list, so do that with float().)

Expected Output:

```
662.01        # may also have a tiny remainder, i.e., 662.0100000000001
```

Ex. 4.7 Extending the above program, use len() to determine the length of the list, then use the formula sum(flist) / len(flist) to produce an average (where flist is the list of floats you compiled).

Expected Output:

```
94.5728571429    # your floating point remainder may be slightly different
```

Ex. 4.8 Given the following code:

```
values = [1, 2, 3, 4, 5]
```

Determine and print the median (middle) value without mentioning any values directly. (Hint: use len() to determine the length, then use that value to figure out the "halfway" index. Make sure your calculated index is an integer).

Expected Output:

```
3
```

Ex. 4.9 Given the following code:

```
values = [3, 1, 5, 2, 4]
```

Again, determine and print the median value. This time you'll need to use the sorted() function, which takes an unsorted list and returns a sorted list.

Expected Output:

```
3
```

Ex. 4.10 Given the following code:

```
values = [6, 1, 3, 2, 4, 5]
```

Determine the median value, which (in an even-number of elements) is the value halfway between the two "middle" values in a sorted list. (Hint: again use the len() of the list to calculate the indices of the middle two values.)

Expected Output:

```
3.5
```

Ex. 4.11 Given a list containing duplicate values, initialize an empty set() and add each element to it to produce a set of unique values. Print the whole set. (Note: just for practice, don't pass the list directly to the set() constructor -- loop through the list and add each element one at a time.)

Starter Code:

```
dupvals = [1, 3, 1, 1, 2, 3, 2, 1, 3]
```

Expected Output:

```
{1, 2, 3}
```

Ex. 4.12 Opening the revenue.txt text file, loop through and print each line of the file, but make sure there are no blank lines printed (hint: use rstrip() on each line to remove the newline character; print itself already adds a newline to the end of every line).

Expected Output:

```
Haddad's,PA,239.50
Westfield,NJ,53.90
The Store,NJ,211.50
Hipster's,NY,11.98
Dothraki Fashions,NY,5.98
Awful's,PA,23.95
The Clothiers,NY,115.20
```

Ex. 4.13 Given the following list:

```
mylist = [-5, -2, 1, -3, 1.5, 7, 9]
```

Loop through mylist and build a new list that is only positive numbers (i.e., greater than 0). Print the list.

Expected Output:

```
[1, 1.5, 7, 9]
```

Ex. 4.14 Given the following code:

```
sentence = "I could; I wouldn't.  I might?  Might I!"
```

Split this sentence into words (without altering or processing them). Print each word on a line.

Expected Output:

```
I
could;
I
wouldn't.
I
might?
Might
I!
```

Ex. 4.15 Extending the above code, use a single call to rstrip() to remove any punctuation. (Hint: you can place any punctuation characters to be removed together in a single string argument to rstrip(); any single character within the string will be removed.)

Expected Output:

```
I
could
I
wouldn't
I
might
Might
I
```

Ex. 4.16 Extending the above code, add an additional statement to lowercase each word.

Expected Output:

```
i
could
i
wouldn't
i
might
might
i
```

Ex. 4.17 Extending the above code, add each word to a set(), then print the entire set.

Expected Output:

```
{'i', 'could', 'might', "wouldn't"}
```

Ex. 4.18 Open the file pyku.txt. Print each word from the file on a separate line. (Hint: use read() on the filehandle to return a string, then use split() on the string to create a master list of words.)

Expected Output:

```
We're
out
of
gouda.
This
parrot
has
ceased
to
be.
Spam,
spam,
spam,
spam,
spam.
```

Ex. 4.19 Extending the previous solution, again strip each word of punctuation and lowercase each word and add each to a set. Print the set.

Expected Output:

```
{'be', 'has', 'parrot', 'to', 'this', 'of',
 'spam', 'ceased', "we're", 'gouda', 'out'}
```

Ex. 4.20 Extend the previous solution by adding the following code at the start:

```
text = "we're certainly out of gouda but Python is great."
```

Now split this text into words (no need to strip or lowercase). Removing the printing of the set and adding to the end of the code, loop through each word in text and check the word against the set created earlier (use the in operator to check for membership). If the word is not in the set, print it.

Expected Output:

```
certainly
but
Python
is
great.
```