3. WRITTEN RESPONSES

3 a.

3.a.i.

The purpose of the program is to use a bubble sort to order letters alphabetically, numbers based on value, and words based on the number of letters.

3.a.ii.

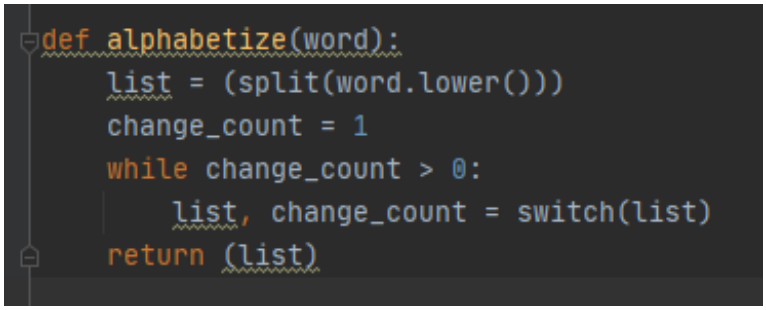
The video shows the program is to order the letters of a word alphabetically, order a group of numbers based on value, and order a group of words based on length.

3.a.iii.

The input of the program is a word, a list of numbers separated by commas, and a list of words separated by commas. The output is a list of the letters of the word in alphabetical order, a list of the numbers in increasing order, and a list of the words in the order of increasing size.

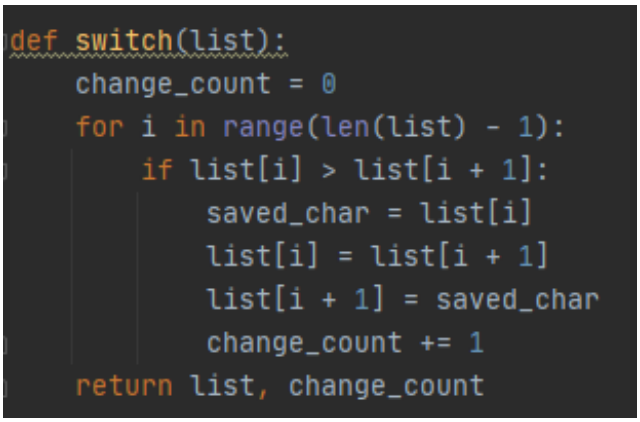
3 b.

3.b.i.



```
def alphabetize(word):
    list = (split(word.lower()))
    change_count = 1
    while change_count > 0:
        list, change_count = switch(list)
    return (list)
```

3.b.ii.



```
def switch(list):
    change_count = 0
    for i in range(len(list) - 1):
        if list[i] > list[i + 1]:
            saved_char = list[i]
            list[i] = list[i + 1]
            list[i + 1] = saved_char
            change_count += 1
    return list, change_count
```

3.b.iii.

The name of the list is list.

3.b.iv.

The data in this list represents lowercase letters of the inputted word.

3.b.v.

Decreases program complexity because without a list we would need a different variable for each letter and then have to compare all of those variables which is not feasible.

3 c.

3.c.i.

```
def switch(list):
    change_count = 0
    for i in range(len(list) - 1):
        if list[i] > list[i + 1]:
            saved_char = list[i]
            list[i] = list[i + 1]
            list[i + 1] = saved_char
            change_count += 1
    return list, change_count
```

3.c.ii.

```
list = (split(word.lower()))
```

3.c.iii.

The identified procedure stitches each letter in a list to get them in alphabetical order. The procedure may need to be called more than once to finish alphabetizing the list.

3.c.iv.

This procedure compares each letter of the list to the next letter in the list and switches them if the first letter is greater (ex: b>a) than the next letter. Whenever two letters are switched it adds one to the change count.

3 d.

3.d.i.

First call:

```
list = [b,a]
list, change_count = switch(list)
```

Second call:

```
list = [b,a,d,a]
list, change_count = switch(list)
```

3 d.ii.

Condition(s) tested by first call:

If the list [b,a] is entered the function will test whether b > a.

Condition(s) tested by second call:

If the list [b,a,d,a] is inputted the function will test wether b>a, d>b, and a>d.

3.d.iii.

Results of the first call:

If the list [b,a] is entered the function will return [a,b] change_count = 1. If the list [b,a,d,a] is inputted [a,b,a,d] and change_count = 2 will be returned

Results of the second call:

If the list [b,a,d,a] is inputted [a,b,a,d] and change_count = 2 will be returned.