Carrie West
CS161 – Jess
Project 5
June 8, 2024

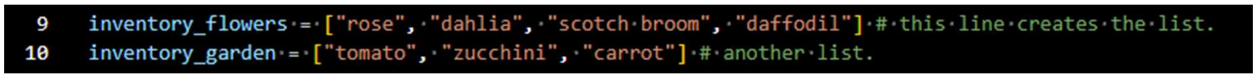
GitHub link:

https://github.com/CWestLBCC/CS161

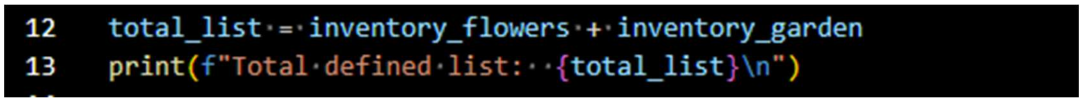
C_West_Project_5_Garden.py

For this project I created a program that manages a garden and the garden information. It will ask the user for an additional input for a flower. It will remove the noxious week Scotch Broom. It will also count how many different types of plants are in the garden. I will use this project and expand on it for the Final Project. I want to make sure every line item is covered for this Project #5 assignment that may not be in the final program.

1. Using lists.
   1) Create a list.
      In this section an inventory_flowers list and an inventory_garden list was created and then printed.
      Code:

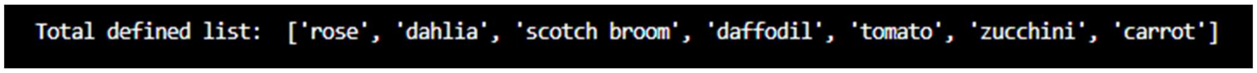


```python
inventory_flowers = ["rose", "dahlia", "scotch broom", "daffodil"] # this line creates the list.
inventory_garden = ["tomato", "zucchini", "carrot"] # another list.
```



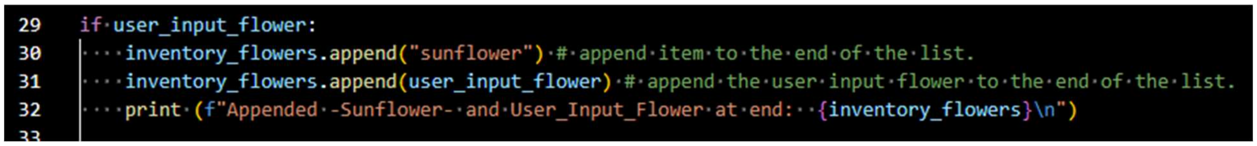
```python
total_list = inventory_flowers + inventory_garden
print(f"Total defined list:  {total_list}\n")
```

      Output:
      This output combines both the flowers and the garden lists together and prints them.



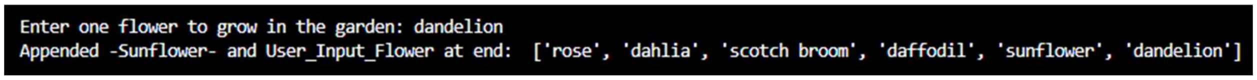
```
Total defined list:  ['rose', 'dahlia', 'scotch broom', 'daffodil', 'tomato', 'zucchini', 'carrot']
```

   2) Add more data to the list.
      In this section Sunflower was added to the end of the flowers list and also the User input of Dandelion.
      Code:



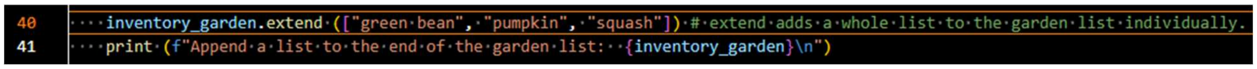
```python
if user_input_flower:
    inventory_flowers.append("sunflower") # append item to the end of the list.
    inventory_flowers.append(user_input_flower) # append the user input flower to the end of the list.
    print (f"Appended -Sunflower- and User_Input_Flower at end:  {inventory_flowers}\n")
```

      Output:



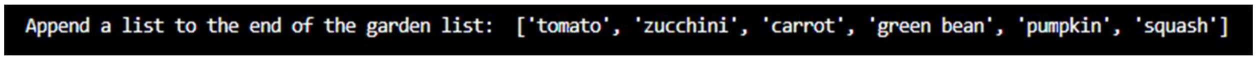
```
Enter one flower to grow in the garden: dandelion
Appended -Sunflower- and User_Input_Flower at end:  ['rose', 'dahlia', 'scotch broom', 'daffodil', 'sunflower', 'dandelion']
```

   3) Change data in an element of the list.
      This section added a whole list to the end of the existing garden list as individual items.
      Code:



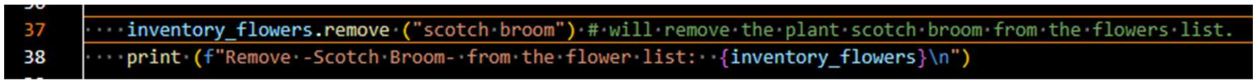
```python
    inventory_garden.extend (["green bean", "pumpkin", "squash"]) # extend adds a whole list to the garden list individually.
    print (f"Append a list to the end of the garden list:  {inventory_garden}\n")
```

      Output:



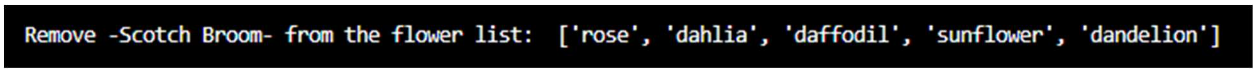
```
Append a list to the end of the garden list:  ['tomato', 'zucchini', 'carrot', 'green bean', 'pumpkin', 'squash']
```

4) Remove data from the list.
   This section removes the word Scotch Broom from the flowers list.
   Code:



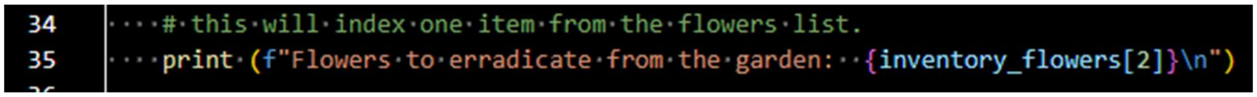
```
    inventory_flowers.remove ("scotch broom") # will remove the plant scotch broom from the flowers list.
    print (f"Remove -Scotch Broom- from the flower list:  {inventory_flowers}\n")
```

   Output:



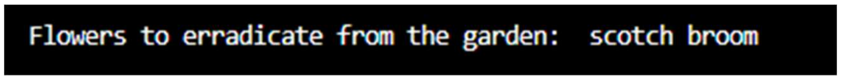
```
Remove -Scotch Broom- from the flower list:  ['rose', 'dahlia', 'daffodil', 'sunflower', 'dandelion']
```

5) Index the list to find some data stored within it.
   This code will find the item Scotch Broom within the flowers list.
   Code:



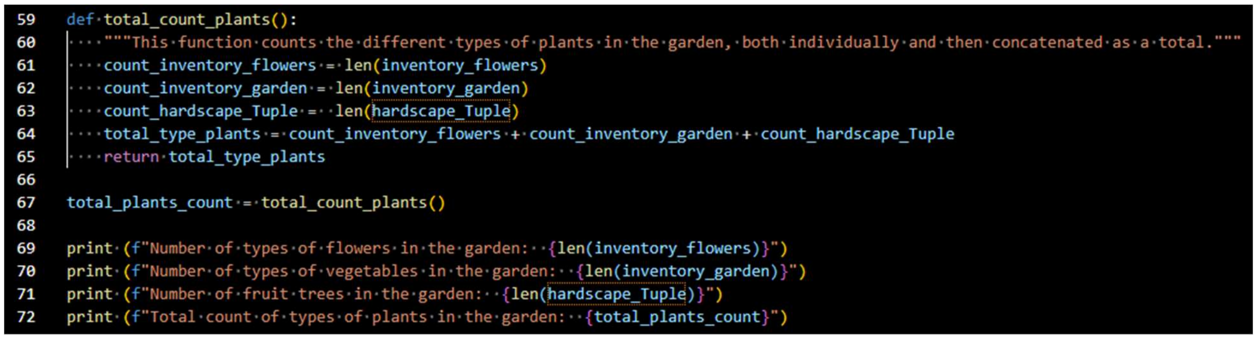
```
    # this will index one item from the flowers list.
    print (f"Flowers to erradicate from the garden:  {inventory_flowers[2]}\n")
```

   Output:



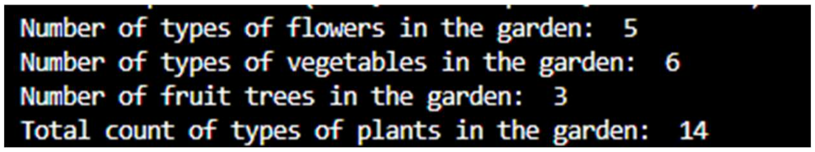
```
Flowers to erradicate from the garden:  scotch broom
```

6) Create a function that takes a list and accomplishes something similar to the built in functions (min, max, mean, sum, or comparison without using any built-in list methods).
   Count how many items are in each list and give a total for types of plants in the garden.
   Code:



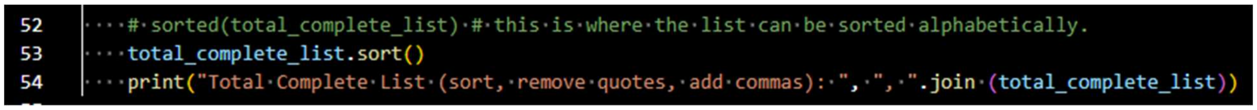
```
def total_count_plants():
    """This function counts the different types of plants in the garden, both individually and then concatenated as a total."""
    count_inventory_flowers = len(inventory_flowers)
    count_inventory_garden = len(inventory_garden)
    count_hardscape_Tuple =  len(hardscape_Tuple)
    total_type_plants = count_inventory_flowers + count_inventory_garden + count_hardscape_Tuple
    return total_type_plants

total_plants_count = total_count_plants()

print (f"Number of types of flowers in the garden:  {len(inventory_flowers)}")
print (f"Number of types of vegetables in the garden:  {len(inventory_garden)}")
print (f"Number of fruit trees in the garden:  {len(hardscape_Tuple)}")
print (f"Total count of types of plants in the garden:  {total_plants_count}")
```

   Output:

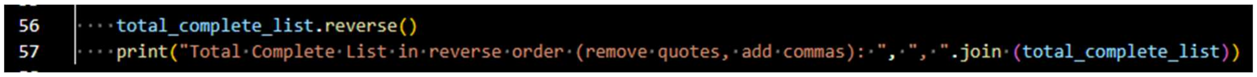


```
Number of types of flowers in the garden:  5
Number of types of vegetables in the garden:  6
Number of fruit trees in the garden:  3
Total count of types of plants in the garden:  14
```

7) Use a couple of methods on lists to accomplish some task.
   I used the .sort method to sort the list alphabetically. I then used the .reverse method to reverse the list.
   Code:

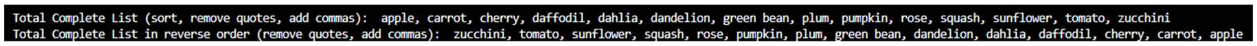


```
    # sorted(total_complete_list) # this is where the list can be sorted alphabetically.
    total_complete_list.sort()
    print("Total Complete List (sort, remove quotes, add commas): ", ", ".join (total_complete_list))
```



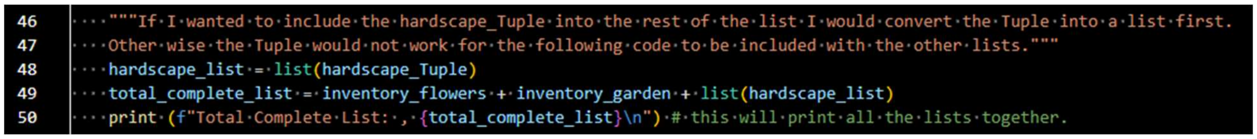
```
    total_complete_list.reverse()
    print("Total Complete List in reverse order (remove quotes, add commas): ", ", ".join (total_complete_list))
```

Output:



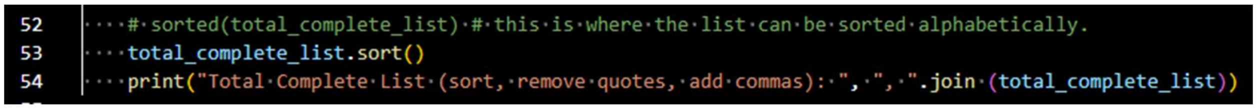
```
Total Complete List (sort, remove quotes, add commas):  apple, carrot, cherry, daffodil, dahlia, dandelion, green bean, plum, pumpkin, rose, squash, sunflower, tomato, zucchini
Total Complete List in reverse order (remove quotes, add commas):  zucchini, tomato, sunflower, squash, rose, pumpkin, plum, green bean, dandelion, dahlia, daffodil, cherry, carrot, apple
```

2. Use tuples in some of the tasks, make special note in code when a task cannot be performed exactly the same due to mutable versus immutable objects behavior.
   Code:
   A Tuple cannot be included with a list to be printed as sorted alphabetically. I converted the the hardscape_Tuple into the hardscape_list first and then concatenated the three lists together (inventory_flowers, inventory_garden and hardscape_list) so they could be printed as one line item that is then sorted alphabetically.



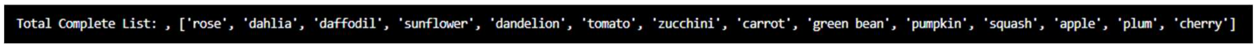
```
    """If I wanted to include the hardscape_Tuple into the rest of the list I would convert the Tuple into a list first.
    Other wise the Tuple would not work for the following code to be included with the other lists."""
    hardscape_list = list(hardscape_Tuple)
    total_complete_list = inventory_flowers + inventory_garden + list(hardscape_list)
    print (f"Total Complete List: , {total_complete_list}\n") # this will print all the lists together.
```

   This is the code to sort the list alphabetically.



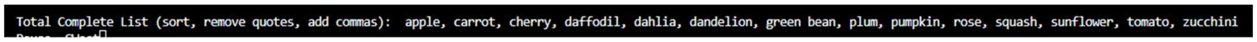
```
    # sorted(total_complete_list) # this is where the list can be sorted alphabetically.
    total_complete_list.sort()
    print("Total Complete List (sort, remove quotes, add commas): ", ", ".join (total_complete_list))
```

   Output:
   All three lists printed together.



```
Total Complete List: , ['rose', 'dahlia', 'daffodil', 'sunflower', 'dandelion', 'tomato', 'zucchini', 'carrot', 'green bean', 'pumpkin', 'squash', 'apple', 'plum', 'cherry']
```

   The final list sorted alphabetically.



```
Total Complete List (sort, remove quotes, add commas):  apple, carrot, cherry, daffodil, dahlia, dandelion, green bean, plum, pumpkin, rose, squash, sunflower, tomato, zucchini
```

3. Use both lists and tuples as arguments to functions, show how they behave differently and similarly.