Carrie West
CS161 – Jess
Final Project
June 11, 2024

GitHub link will have the most current version:

https://github.com/CWestLBCC/CS161

Filename: C_West_Final_Project.py

Readme Document: C_West_Final_Project_README.pdf

Any updated version would be filename: C_West_Final_Project_UPDATE.py

For this project I created a program that manages a community garden and the garden information.

Create a document that shows both an image of your program demonstrating each of the topics and an image of source code that makes it work.

1. Understanding of design
   The program is well formatted and easy to read and understand.
2. Code Organization
   This code is organized at the end of the program. It has a good, easy to read and understand docstring describing the function. The following print statements are formatted together for ease of use and understanding.
   Code:

```
74  def total_count_plants():
75      """This function counts the different types of plants in the garden, both individually and then concatenated as a total."""
76      count_inventory_flowers = len(inventory_flowers)
77      count_inventory_garden = len(inventory_garden)
78      count_tree_Tuple = len(tree_Tuple)
79      total_type_plants = count_inventory_flowers + count_inventory_garden + count_tree_Tuple
80      return total_type_plants
81
82  total_plants_count = total_count_plants()
83
84  print(f"Number of types of flowers in the garden:  {len(inventory_flowers)}")
85  print(f"Number of types of vegetables in the garden:  {len(inventory_garden)}")
86  print(f"Number of fruit trees in the garden:  {len(tree_Tuple)}")
87  print(f"Total count of types of plants in the garden:  {total_plants_count}")
88  print(f"Enjoy gardening this season!")
```

   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
Enjoy gardening this season!
```

3. Comments and docstrings

This is an example of a good docstring at the beginning of the program followed by an example of a comment.

```
"""C_West_Final_Project.py
West C
This program involves garden information for a community garden.
"""
import string # imports the Python string module.
```

4. Variables, if statements, loops (for and while loops)

This code is a while and if else loop to ensure valid input from the user. It also gives the user 3 tries to enter valid input before the program ends.

Code:

```
21  def get_value_flower_input():
22      '''This function asks for user input for a flower and verifies that it is an alphabetic entry, given three tries.'''
23      tries = 3
24      while tries > 0:
25          user_input_flower = input ("Enter one flower to grow in the garden: ").lower() #ensures user input is lower case
26          if user_input_flower.isalpha(): # ensures the user input is alphabetical
27              return user_input_flower
28          else:
29              print("This is an incorrect entry, please try again.")
30              tries -= 1
31      print("Incorrect entry, this program will end.")
32      return None
33  user_input_flower = get_value_flower_input()
```

Output:

```
Enter one flower to grow in the garden: D@ND3LION
This is an incorrect entry, please try again.
Enter one flower to grow in the garden: dAN$#
This is an incorrect entry, please try again.
Enter one flower to grow in the garden: dA6
This is an incorrect entry, please try again.
Incorrect entry, this program will end.
```

5. Collections (lists, tuples, sets, dictionaries)

Code:

This is the code for a basic lists.

```
10  #Creating lists.
11  inventory_flowers = ["rose", "dahlia", "scotch broom", "daffodil"]
12  inventory_garden = ["tomato", "zucchini", "carrot"]
```

```
18  total_list = inventory_flowers + inventory_garden
19  print(f"Current plants grown in the garden:  {total_list}\n")
```

This code will add a whole list to the garden list individually.

```
50      inventory_garden.extend (["green bean", "pumpkin", "squash"]) # extend adds a whole list to the garden list individually.
51      print (f"This gardening season will also add to the garden:  {inventory_garden}\n")
```

This is the code for a basic Tuple.

```
53      tree_Tuple = ("apple", "plum", "cherry") # Tuple list that will remain unchanged.
54      print (f"The trees currently grown in this garden are: {tree_Tuple}\n")
```

Output:

Output for the basic lists.

```
Current plants grown in the garden:  ['rose', 'dahlia', 'scotch broom', 'daffodil', 'tomato', 'zucchini', 'carrot']
```

Output for extending a whole list individually.

```
This gardening season will also add to the garden:  ['tomato', 'zucchini', 'carrot', 'green bean', 'pumpkin', 'squash']
```

Output for the basic Tuple.

```
The trees currently grown in this garden are: ('apple', 'plum', 'cherry')
```

6. Functions, parameters, arguments, default parameters, named arguments.
   Code:
   This is an example of a function with arguments.

```
def get_value_flower_input():
    '''This function asks for user input for a flower and verifies that it is an alphabetic entry, given three tries.'''
    tries = 3
    while tries > 0:
        user_input_flower = input("Enter one flower to grow in the garden: ").lower() #ensures user input is lower case
        if user_input_flower.isalpha(): # ensures the user input is alphabetical
            return user_input_flower
        else:
            print("This is an incorrect entry, please try again.")
            tries -= 1
    print("Incorrect entry, this program will end.")
    return None
user_input_flower = get_value_flower_input()
```

Output:

```
Enter one flower to grow in the garden: D@ND3LION
This is an incorrect entry, please try again.
Enter one flower to grow in the garden: dAN$#
This is an incorrect entry, please try again.
Enter one flower to grow in the garden: dA6
This is an incorrect entry, please try again.
Incorrect entry, this program will end.
```

7. Simple classes, attributes, and methods.
   The most current copy of work will be on GitHub. I will continue to work on this as I want to learn the topic and right now the code is not working.

8. User IO, simple file IO, input validation.
   This code is a “while” and “if else” loop to ensure valid input from the user.
   Code:

```
21  def get_value_flower_input():
22      '''This function asks for user input for a flower and verifies that it is an alphabetic entry, given three tries.'''
23      tries = 3
24      while tries > 0:
25          user_input_flower = input ("Enter one flower to grow in the garden: ").lower() #ensures user input is lower case
26          if user_input_flower.isalpha(): #ensures the user input is alphabetical
27              return user_input_flower
28          else:
29              print("This is an incorrect entry, please try again.")
30              tries -= 1
31      print("Incorrect entry, this program will end.")
32      return None
33  user_input_flower = get_value_flower_input()
```

Output:

When user input is an invalid input this error will show and give the user 3 tries before the program ends.

```
Enter one flower to grow in the garden: D@ND3LION
This is an incorrect entry, please try again.
Enter one flower to grow in the garden: dAN$#
This is an incorrect entry, please try again.
Enter one flower to grow in the garden: dA6
This is an incorrect entry, please try again.
Incorrect entry, this program will end.
```

```
Enter one flower to grow in the garden: d@nd3lion
This is an incorrect entry, please try again.
Enter one flower to grow in the garden: DANDELION
```

Even though DANDELION was in uppercase letters the output is in lowercase and alphabetical.

```
Enter one flower to grow in the garden: DANDELION
You chose a new plant to add to this garden.  The new list of flowers is:  ['rose', 'dahlia', 'scotch broom', 'daffodil', 'dandelion']
```

9. Recursive function calls.

   A recursive function call will call itself. At line 21 the function get_value_flower_input () is empty and will call itself recursively at line 33.

   Code:

```
21  def get_value_flower_input():
22      '''This function asks for user input for a flower and verifies that it is an alphabetic entry, given three tries.'''
23      tries = 3
24      while tries > 0:
25          user_input_flower = input ("Enter one flower to grow in the garden: ").lower() #ensures user input is lower case
26          if user_input_flower.isalpha(): #ensures the user input is alphabetical
27              return user_input_flower
28          else:
29              print("This is an incorrect entry, please try again.")
30              tries -= 1
31      print("Incorrect entry, this program will end.")
32      return None
33  user_input_flower = get_value_flower_input()
```

10. Understanding of basic testing.

    Code should be tested through out the development phase. Below is one example of testing the code while developing the program.

    Code:

```
14  #·This·is·an·example·of·basic·testing·of·the·code·as·it·is·developing.
15  #total_list·=·inventory_flowers·+·inventory_garden·
16  #print(f"Total·defined·list:··{total_list}\n")
```