Unit:2 ; Class:5

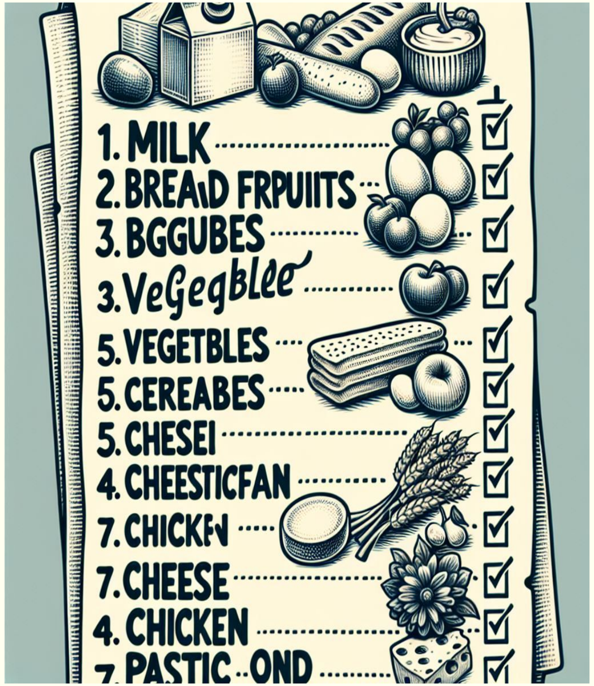
Lists & Tuples

Instructor: Musfique Ahmed

What are Lists?

- A list is a collection of items in a specific order.
- Lists can store items of different data types.
- Defined using square brackets [ ].

```
fruits = ["apple", "banana", "cherry"]
print(fruits)
```

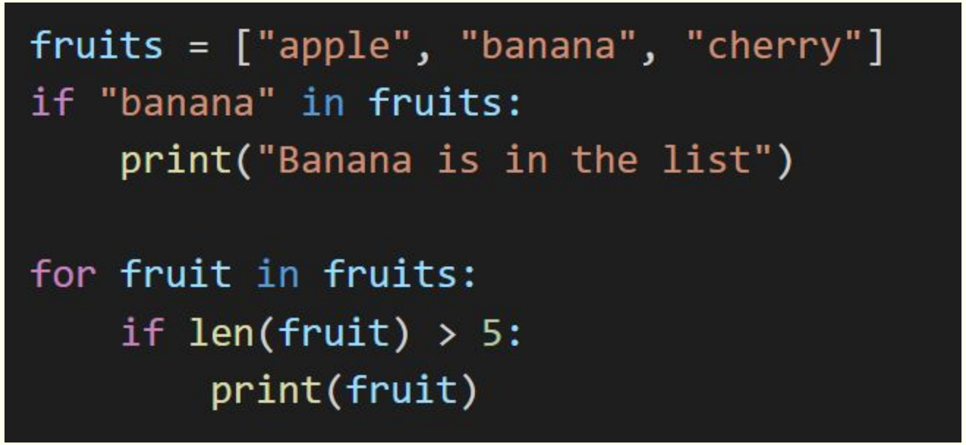


Practice Problems:

1. Create a list of your five favorite movies and print it.
2. Modify the second item in a list of cities.

Using Conditionals with Lists

- Check if an item exists in a list using `in` or `not in`.
- Use loops and conditionals to process lists.



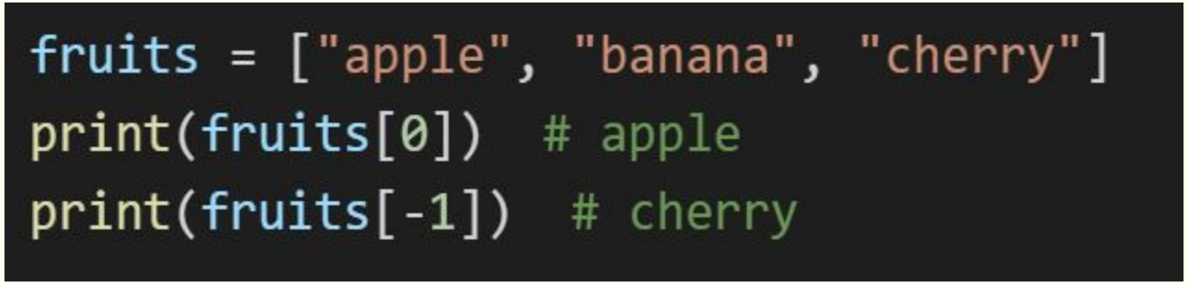
```python
fruits = ["apple", "banana", "cherry"]
if "banana" in fruits:
    print("Banana is in the list")

for fruit in fruits:
    if len(fruit) > 5:
        print(fruit)
```

Practice Problems:

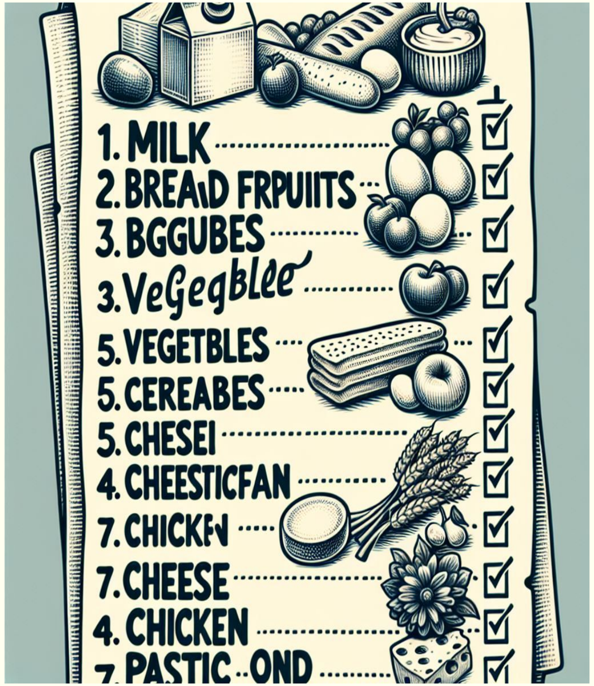
1. Write a program to check if "grape" exists in a list of fruits.
2. Iterate through a list of names and print the ones starting with "A".

Accessing List Elements

- Access elements using their index (starting at 0).
- Negative indices access elements from the end.



```python
fruits = ["apple", "banana", "cherry"]
print(fruits[0])  # apple
print(fruits[-1])  # cherry
```



Practice Problems:

1. Access the first and last elements of a list of colors.
2. Use a negative index to retrieve the second last item from a list of numbers.

try It yourself

Try these short programs to get some firsthand experience with Python's lists. You might want to create a new folder for each chapter's exercises to keep them organized .

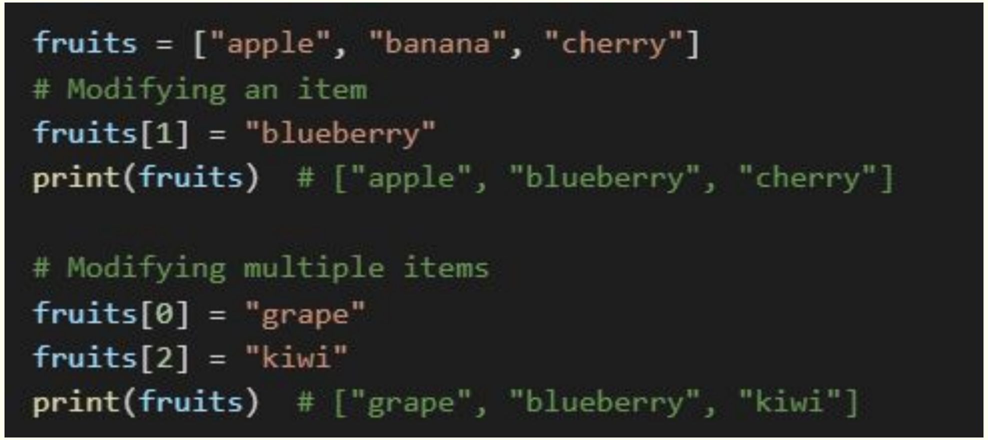
1. Names: Store the names of a few of your friends in a list called names . Print each person's name by accessing each element in the list, one at a time .

2. Greetings: Start with the list you used in Exercise 3-1, but instead of just printing each person's name, print a message to them . The text of each mes sage should be the same, but each message should be personalized with the person's name .

3. Your Own List: Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples . Use your list to print a series of statements about these items, such as "I would like to own a Honda motorcycle ."

Modifying Elements in a List

- Change the value of an item using its index.
- Use indexing to access the item and assign a new value.



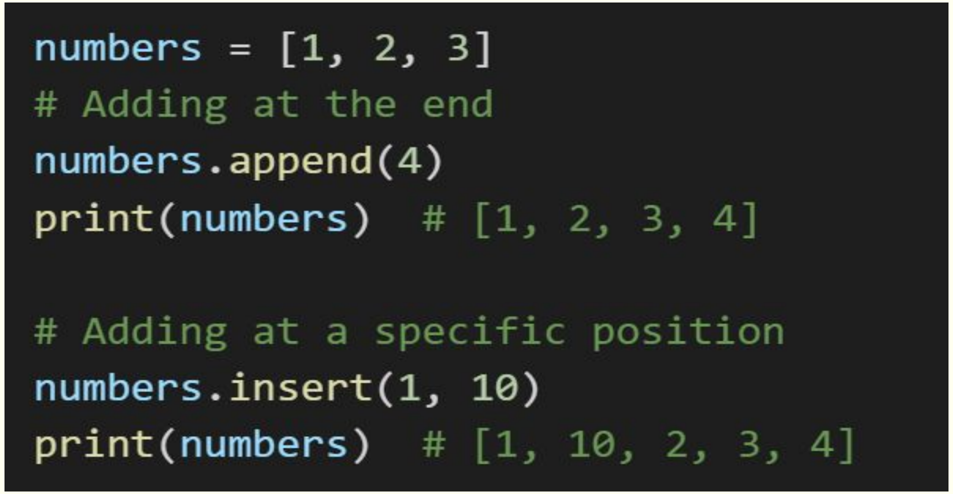
```python
fruits = ["apple", "banana", "cherry"]
# Modifying an item
fruits[1] = "blueberry"
print(fruits)  # ["apple", "blueberry", "cherry"]

# Modifying multiple items
fruits[0] = "grape"
fruits[2] = "kiwi"
print(fruits)  # ["grape", "blueberry", "kiwi"]
```

Practice Problems:

1. Change the third item in a list of cities to "Paris".
2. Modify the first and last items in a list of animals.

Adding Items to a List

- Use `append()` to add an item to the end of the list.
- Use `insert()` to add an item at a specific position.



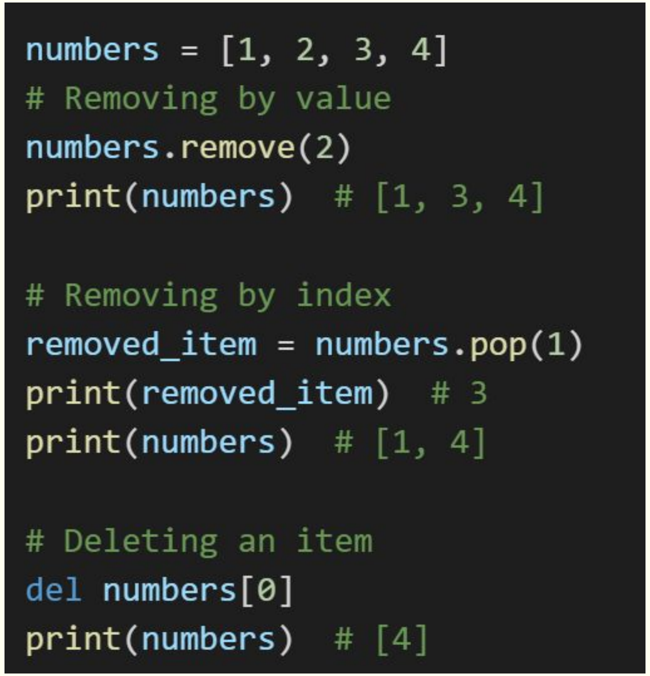
```python
numbers = [1, 2, 3]
# Adding at the end
numbers.append(4)
print(numbers)  # [1, 2, 3, 4]

# Adding at a specific position
numbers.insert(1, 10)
print(numbers)  # [1, 10, 2, 3, 4]
```

Practice Problems:

1. Start with an empty list and add three items using `append()`.
2. Add an item to the second position of a list using `insert()`.

Removing Items from a List

- Use `remove()` to remove an item by its value.
- Use `pop()` to remove an item by its index or the last item.
- Use `del` to delete an item or the entire list.



```python
numbers = [1, 2, 3, 4]
# Removing by value
numbers.remove(2)
print(numbers)  # [1, 3, 4]

# Removing by index
removed_item = numbers.pop(1)
print(removed_item)  # 3
print(numbers)  # [1, 4]

# Deleting an item
del numbers[0]
print(numbers)  # [4]
```

Practice Problems:

1. Remove an item by its value from a list of fruits.
2. Use `pop()` to remove and print the last item in a list.

try It yourself

The following exercises are a bit more complex than those in Chapter 2, but they give you an opportunity to use lists in all of the ways described .

3-4. Guest List: If you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you'd like to invite to dinner . Then use your list to print a message to each person, inviting them to dinner .

3-5. Changing Guest List: You just heard that one of your guests can't make the dinner, so you need to send out a new set of invitations . You'll have to think of someone else to invite .

- Start with your program from Exercise 3-4 . Add a print statement at the end of your program stating the name of the guest who can't make it .
- Modify your list, replacing the name of the guest who can't make it with the name of the new person you are inviting .
- Print a second set of invitation messages, one for each person who is still in your list .

3-6. More Guests: You just found a bigger dinner table, so now more space is available . Think of three more guests to invite to dinner .

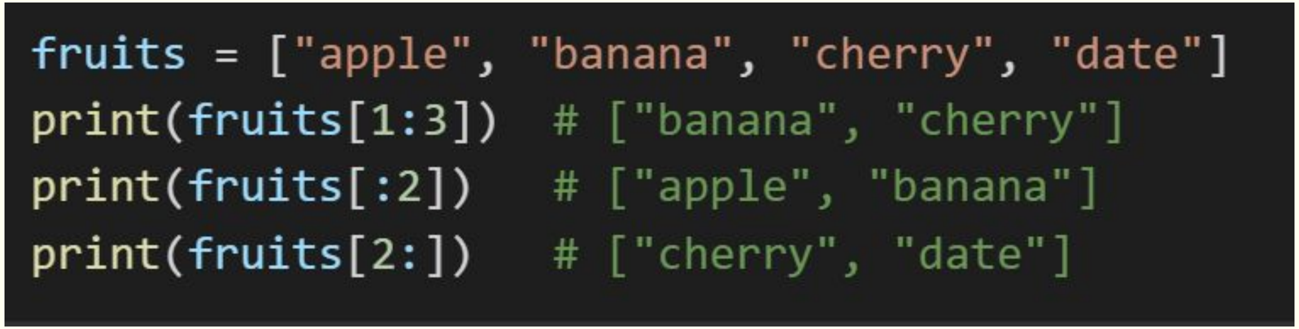
- Start with your program from Exercise 3-4 or Exercise 3-5 . Add a print statement to the end of your program informing people that you found a bigger dinner table .
- Use insert() to add one new guest to the beginning of your list .
- Use insert() to add one new guest to the middle of your list .
- Use append() to add one new guest to the end of your list .
- Print a new set of invitation messages, one for each person in your list .

3-7. Shrinking Guest List: You just found out that your new dinner table won't arrive in time for the dinner, and you have space for only two guests .

- Start with your program from Exercise 3-6 . Add a new line that prints a message saying that you can invite only two people for dinner .
- Use pop() to remove guests from your list one at a time until only two names remain in your list . Each time you pop a name from your list, print a message to that person letting them know you're sorry you can't invite them to dinner .
- Print a message to each of the two people still on your list, letting them know they're still invited .
- Use del to remove the last two names from your list, so you have an empty list . Print your list to make sure you actually have an empty list at the end of your program

Slicing a List

- Access a range of elements using slicing.
- Syntax: `list[start:end]`
- Omitting `start` or `end` uses the start or end of the list.



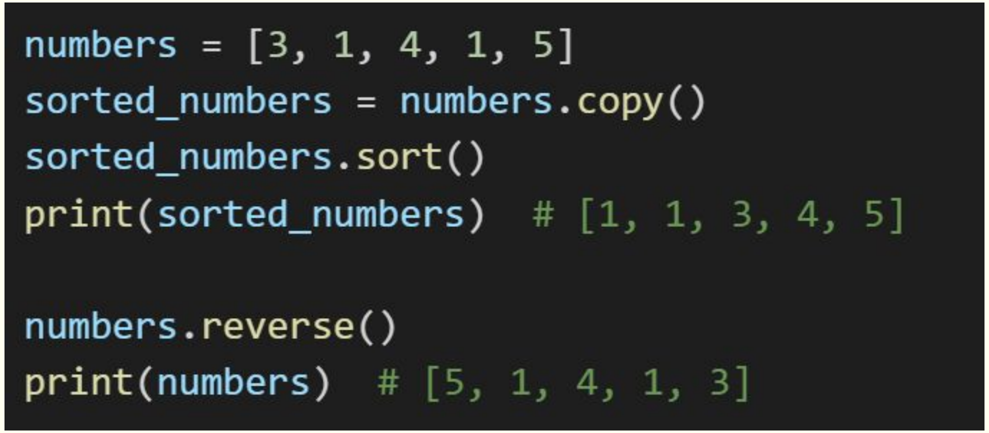
```python
fruits = ["apple", "banana", "cherry", "date"]
print(fruits[1:3])  # ["banana", "cherry"]
print(fruits[:2])   # ["apple", "banana"]
print(fruits[2:])   # ["cherry", "date"]
```

Practice Problems:

1. Slice the first three items from a list of numbers.
2. Create a list of five items and extract the middle three using slicing.

Additional List Operations

- **Copying Lists:**
  - Use `list.copy()` to create a duplicate.
- **Sorting:**
  - `sort()` to arrange items.
  - `reverse()` to reverse the order.



```
numbers = [3, 1, 4, 1, 5]
sorted_numbers = numbers.copy()
sorted_numbers.sort()
print(sorted_numbers)  # [1, 1, 3, 4, 5]

numbers.reverse()
print(numbers)  # [5, 1, 4, 1, 3]
```

Practice Problems:

1. Create a copy of a list of colors and sort it.
2. Reverse a list of numbers and print it.

try It yourself

3-8. Seeing the World: Think of at least five places in the world you'd like to visit .

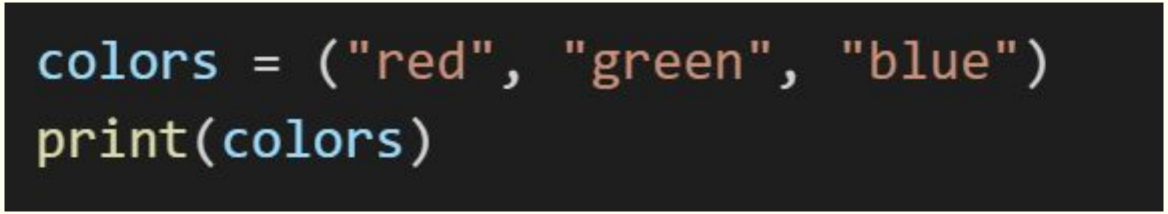
- Store the locations in a list . Make sure the list is not in alphabetical order .
- Print your list in its original order . Don't worry about printing the list neatly, just print it as a raw Python list .
- Use sorted() to print your list in alphabetical order without modifying the actual list .
- Show that your list is still in its original order by printing it .
- Use sorted() to print your list in reverse alphabetical order without changing the order of the original list .
- Show that your list is still in its original order by printing it again .
- Use reverse() to change the order of your list . Print the list to show that its order has changed .
- Use reverse() to change the order of your list again . Print the list to show it's back to its original order .
- Use sort() to change your list so it's stored in alphabetical order . Print the list to show that its order has been changed .
- Use sort() to change your list so it's stored in reverse alphabetical order . Print the list to show that its order has changed .

3-9. Dinner Guests: Working with one of the programs from Exercises 3-4 through 3-7 (page 46), use len() to print a message indicating the number of people you are inviting to dinner .

3-10. Every Function: Think of something you could store in a list . For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you'd like . Write a program that creates a list containing these items and then uses each function introduced in this chapter at least once

What are Tuples?

- A tuple is an immutable sequence of items.
- Defined using parentheses ( ).
- Once defined, the items in a tuple cannot be changed.



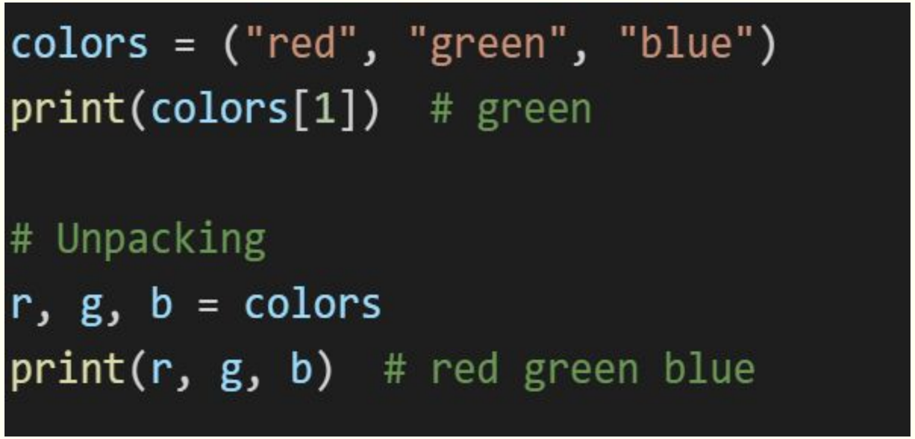
```python
colors = ("red", "green", "blue")
print(colors)
```

Practice Problems:

1. Create a tuple of three favorite foods and print it.
2. Try to modify an item in a tuple and observe the result.

Common Tuple Operations

- Access elements using indexing and slicing (same as lists).
- Use `len()` to find the length of a tuple.
- Tuples can be unpacked into variables.



```python
colors = ("red", "green", "blue")
print(colors[1])  # green

# Unpacking
r, g, b = colors
print(r, g, b)  # red green blue
```

Practice Problems:

1. Slice the first two items from a tuple of numbers.
2. Unpack a tuple of three items into separate variables.

Summary of Class 4

- Lists: Ordered, mutable collections.
- Tuples: Ordered, immutable collections.
- Common operations include indexing, slicing, and iteration.
- Conditionals can be used to process lists effectively.

Project Idea: Grocery List Manager

Project Description:

Create a Python program that acts as a simple Grocery List Manager. The program should allow the user to:

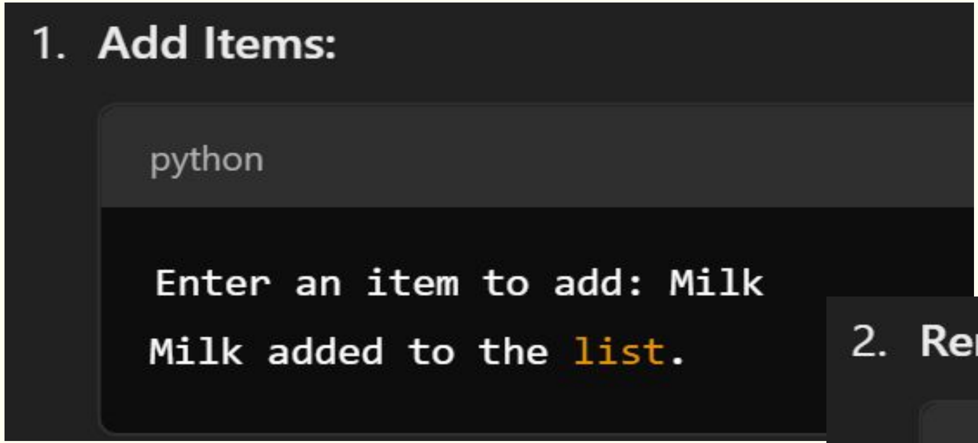
1. Add items to a grocery list.
2. Remove items from the list.
3. View the current list.
4. Check if a specific item is already in the list.
5. Exit the program.

Project Idea: Grocery List Manager

Requirements:

1. **Use Variables:** To store the list and other required values.
2. **Data Types:** Lists for storing items, and strings for user inputs.
3. **Input/Output:** Take user input to add, remove, or check items.
4. **Conditionals:** To handle choices and validate inputs.
5. **Lists:** For storing and manipulating the grocery items.
6. **Tuples:** For storing a sample tuple of predefined categories (e.g., fruits, vegetables, etc.).

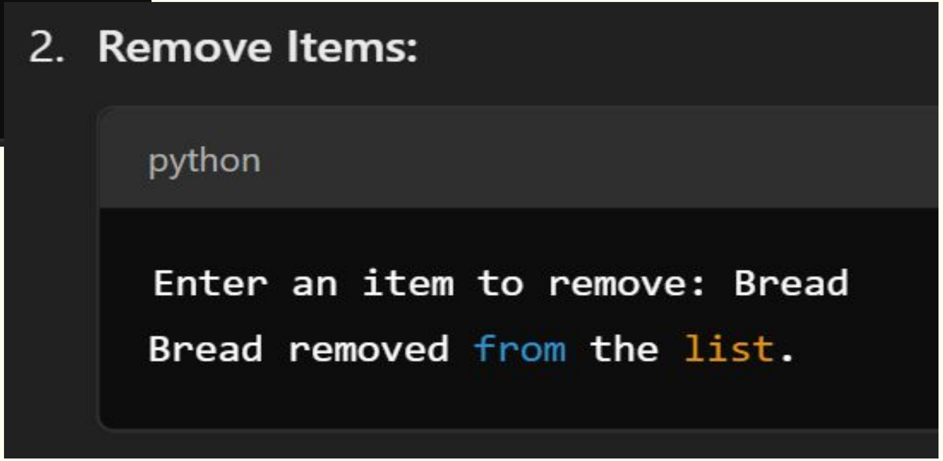
Sample Features:



1. **Add Items:**

python

```
Enter an item to add: Milk
Milk added to the list.
```

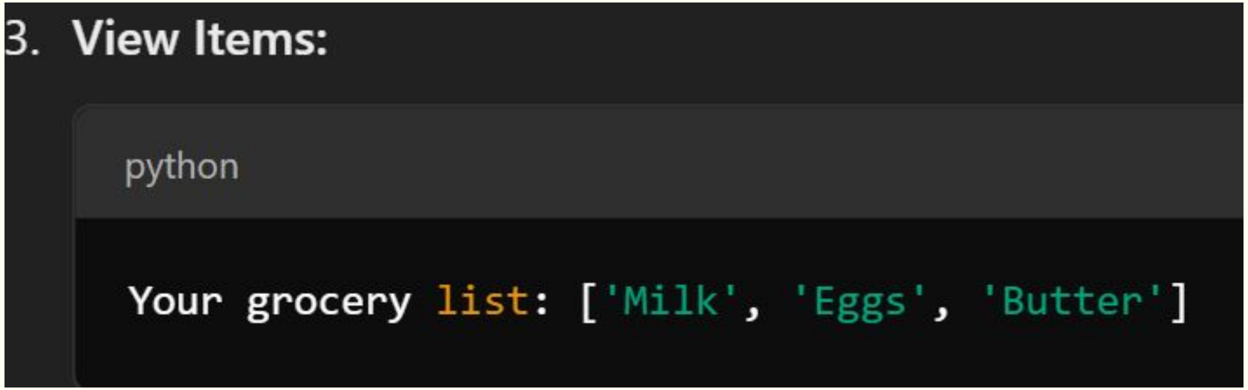


2. **Remove Items:**

python

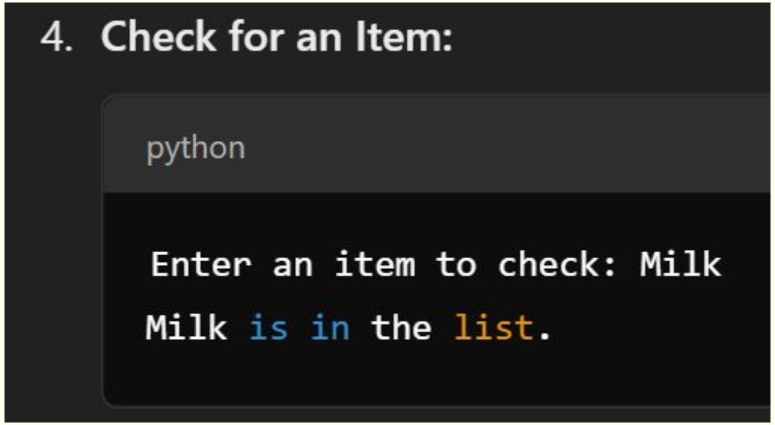
```
Enter an item to remove: Bread
Bread removed from the list.
```

Sample Features:


3. View Items:

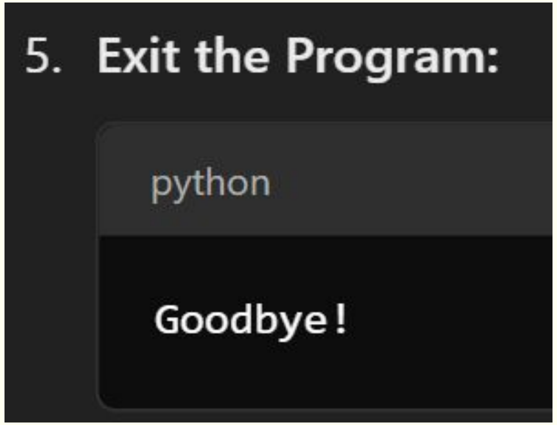
python

```
Your grocery list: ['Milk', 'Eggs', 'Butter']
```


4. Check for an Item:

python

```
Enter an item to check: Milk
Milk is in the list.
```


5. Exit the Program:

python

```
Goodbye!
```

Project Idea: Grocery List Manager

Additional Challenge Ideas:

- Allow users to categorize items (e.g., fruits, vegetables, dairy).
- Validate inputs to prevent duplicate entries.
- Display the total number of items in the list.

Thank You