MIT World Peace University

Python Programming
Second Year B. Tech, Semester 4

Different Operations on the List Data Structure

Assignment No. 4

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

February 28, 2023

Contents

1 Aim

Write a python program to create, append and remove etc. operation on list.

2 Objectives

1. To learn and implement List data structure.

3 Theory

3.1 Data structures in Python

The Different Data Structures in python are as follows:

1. *List*
   A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets [ ]
2. *Tuple*
   A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
3. *Dictionary*
   A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.
4. *Set*
   A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

3.2 Different Operations on Lists

The Different Operations that can be performed on a list are as follows:

1. *Creating a list*
   A list in Python is a collection of items in a particular order. You can make changes to elements in a list, add new elements to a list, and remove elements from a list.

```
>>> list1 = [1, 2, 3, 4]
>>> list2 = ['a', 'b', 'c', 'd']
>>> list3 = [1, 'a', 2, 'b']
```

2. *Accessing elements in a list*
   Lists are ordered collections, so you can access any element in a list by telling Python the position, or index, of the item desired. To access an element in a list, write the name of the list followed by the index of the item enclosed in square brackets.

```
>>> list1 = [1, 2, 3, 4]
>>> list1[0]
1
```

```
>>> list1[1]
2
>>> list1[2]
3
>>> list1[3]
4
```

3. *Adding elements to a list*
   Adding an element to the end of a list is as simple as appending the item to the list. You can use the append() method to add an item to a list.

```
>>> list1 = [1, 2, 3, 4]
>>> list1.append(5)
>>> list1
[1, 2, 3, 4, 5]
```

4. *Inserting elements into a list*
   You can insert a new element at any position in your list by using the insert() method. You do this by specifying the index of the new element and the value of the new item.

```
>>> list1 = [1, 2, 3, 4]
>>> list1.insert(2, 5)
>>> list1
[1, 2, 5, 3, 4]
```

5. *Removing elements from a list*
   There are several ways to remove items from a list, depending on the situation. If you know the position of the item you want to remove, you can use the del statement.

```
>>> list1 = [1, 2, 3, 4]
>>> del list1[2]
>>> list1
[1, 2, 4]
```

6. *Removing elements from a list*
   If you only know the value of the item you want to remove from the list, you can use the remove() method.

7. *Popping elements from a list*
   If you want to work with an item that's removed from a list, use the pop() method. This method removes the last item in a list, but it lets you work with that item after removing it.

```
>>> list1 = [1, 2, 3, 4]
>>> list1.pop(2)
3
>>> list1
[1, 2, 4]

```

8. *Sorting a list*
   Python's sort() method makes it relatively easy to sort a list. If you want to change the order of your list permanently, you can use the sort() method. The sort() method changes the order of a list permanently.

9. *Finding the length of a list*
   Finding the length of a list is as simple as using the len() function.

4 Code

[1]:
```python
# declaring an empty list
my_list = []
```

4.1 Trying different functions on the list

[2]:
```python
# append
print("How many values do you want to append to the list?")
values = int(input())
for i in range(values):
    print("Enter the value to append")
    value = int(input())
    my_list.append(value)
print("The list after appending is: ", my_list)
print("The length of the list is: ", len(my_list))
```

```
How many values do you want to append to the list?
Enter the value to append
Enter the value to append
Enter the value to append
Enter the value to append
The list after appending is:  [1, 2, 4, 3]
The length of the list is:  4
```

[3]:
```python
# insert
# Inserting a value at a specific index
my_list.insert(0, "Something to insert")
print("The list after inserting is: ", my_list)
print("The length of the list is: ", len(my_list))
```

```
The list after inserting is:  ['Something to insert', 1, 2, 4, 3]
The length of the list is:  5
```

[4]:
```python
# remove
my_list.remove("Something to insert")
print("The list after removing is: ", my_list)
print("The length of the list is: ", len(my_list))
```

```
The list after removing is:  [1, 2, 4, 3]
The length of the list is:  4
```

[5]:
```python
# pop

print("just removed", my_list.pop())
print("The list after popping is: ", my_list)
print("The length of the list is: ", len(my_list))
```

```
just removed 3
The list after popping is:  [1, 2, 4]
The length of the list is:  3
```

[6]:
```python
# sort

print("The list before sorting is: ", my_list)
print("The list after sorting is: ", my_list.sort())
```

```
The list before sorting is:  [1, 2, 4]
The list after sorting is:  None
```

[7]:
```python
# reverse

print("The list before reversing is: ", my_list)
print("The list after reversing is: ", my_list.reverse())
```

```
The list before reversing is:  [1, 2, 4]
The list after reversing is:  None
```

[9]:
```python
# index

print("The indices of the list are: ")
for _, i in enumerate(my_list):
    print(_, i)

print("The value at index 1 is: ", my_list.index(1))
```

```
The indices of the list are:
0 4
1 2
2 1
The value at index 1 is:  2
```

[10]:
```python
# count
print("The number of times 1 appears in the list is: ", my_list.count(1))
```

```
The number of times 1 appears in the list is:  1
```

[11]:
```python
# extend
print("The list before extending is: ", my_list)
print("The length of the list is: ", len(my_list))
my_list.extend([1, 2, 3])
print("The list after extending is: ", my_list)
```

```
The list before extending is:  [4, 2, 1]
The length of the list is:  3
The list after extending is:  [4, 2, 1, 1, 2, 3]
```

5 Assignment Problems

5.1 1. Define a list called list_1 with four integer members, and find the output of the following

1) Access the first three elements from list_1 using forward indices: list_1[0:3]
2) Access the last element from list_1 using the len function: list_1[len(list_1) - 1]
3) Access the last two elements from list_1 by slicing: list_1[-2:]
4) Access the first two elements using backward indices list_1[:-2]
5) Reverse the elements in the string: list_1[::-1]

[1]:
```
list1 = [1, 2, 3, 4]
list1[0:3]
```

[1]:
```
[1, 2, 3]
```

[2]:
```
list1[len(list1) -1]
```

[2]:
```
4
```

[3]:
```
list1[-2:]
```

[3]:
```
[3, 4]
```

[4]:
```
list1[:-2]
```

[4]:
```
[1, 2]
```

[5]:
```
list1[::-1]
```

[5]:
```
[4, 3, 2, 1]
```

5.2 2. Accept 20 values from user and save it in list. Perform following operations on it

1) count similar elements of list.
2) count even and odd values of list .
3) count positive and negative values of list.

[27]:
```
import random
list = [random.randint(-5, 10) for i in range(20)]
list
```

[27]:
```
[7, -5, 3, 3, -2, 10, 3, 6, 7, 4, 5, -2, -4, 8, -4, 6, -2, -2, 7, -1]
```

[28]:
```
similar_values = {i: list.count(i) for i in list if list.count(i) > 1}
similar_values
```

```
[28]: {7: 3, 3: 3, -2: 4, 6: 2, -4: 2}
```

```
[29]: positive_values = {i:list.count(i) for i in list if i > 0}
      negative_values = {i:list.count(i) for i in list if i <= 0}
      print("Positive values: ", positive_values)
      print("Negative values: ", negative_values)
```

```
Positive values:  {7: 3, 3: 3, 10: 1, 6: 2, 4: 1, 5: 1, 8: 1}
Negative values:  {-5: 1, -2: 4, -4: 2, -1: 1}
```

5.3 3. Accept 10 values from user and save it in list. Perform following operations on it

1) Sort list in ascending order using sorted() function and display sorted list
2) sort list in descending order using sort() function
3) display length of list

```
[30]: list = [random.randint(-5, 10) for i in range(20)]
      list
```

```
[30]: [3, 7, -2, 8, 2, -2, 0, 0, -2, 4, 10, 1, 6, 9, 3, 9, 0, 8, 7, 10]
```

```
[31]: sorted(list)
```

```
[31]: [-2, -2, -2, 0, 0, 0, 1, 2, 3, 3, 4, 6, 7, 7, 8, 8, 9, 9, 10, 10]
```

```
[33]: list.sort(reverse=True)
      print(list)
```

```
[10, 10, 9, 9, 8, 8, 7, 7, 6, 4, 3, 3, 2, 1, 0, 0, 0, -2, -2, -2]
```

```
[34]: len(list)
```

```
[34]: 20
```

5.4 4. Accept two lists from user and merge them using + in a single lis

```
[38]: list_1 = [random.randint(-5, 10) for i in range(5)]
      list_2 = [random.randint(-5, 10) for i in range(5)]

      list_3 = list_1 + list_2

      print("List 1 is: ", list_1)
      print("List 2 is: ", list_2)
      print("The concatenated list is:", list_3)
```

```
List 1 is:  [-5, -2, 7, 7, -5]
List 2 is:  [-5, 4, 5, 0, -5]
The concatenated list is: [-5, -2, 7, 7, -5, -5, 4, 5, 0, -5]
```

6 Conclusion

The List data structure in python was studied in detail. The different operations that can be performed on a list were also studied. The code was written and tested to check the output. The differences between lists and tuples were also studied.

7 FAQ

1. ***Is a list mutable? Explain with Example.***
   Yes, a list is mutable. It is one of the most important data structures in Python. It is a collection of items in a particular order. You can make changes to elements in a list, add new elements to a list, and remove elements from a list. Tuples however, are immutable. They are like lists in that they are a collection of items in a particular order. However, you can't modify tuples.

   Example:

```
    >>> a = [1, 2, 3]
    >>> a[0] = 4
    >>> a
    [4, 2, 3]

```

2. ***What is the difference between append and extend?***
   The append() method adds its argument as a single element to the end of a list. The length of the list itself will increase by one. The extend() method iterates over its argument adding each element to the list and extending the list. The length of the list will increase by however many elements were in the iterable argument.

   Example:

```
    >>> a = [1, 2, 3]
    >>> a.append([4, 5])
    >>> a
    [1, 2, 3, [4, 5]]
    >>> a.extend([4, 5])
    >>> a
    [1, 2, 3, [4, 5], 4, 5]

```

3. ***What is the difference between remove and pop?***
   The remove() method removes the first matching value, not a specific index. The pop() method removes the item at the given index from the list and returns the removed item. If no index is specified, pop() removes and returns the last item in the list.

   Example:

```
    >>> a = [1, 2, 3, 4, 5]
    >>> a.remove(3)
    >>> a
    [1, 2, 4, 5]
    >>> a.pop(1)
    2
    >>> a
    [1, 4, 5]

```