LT8

Lists

Data Types

1. Number
2. String
3. Tuple
4. List
5. Dictionary
6. Set

4 Collection Data Types in Python Programming Language

Collection Data Types

3. **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.

4. **List** is a collection which is ordered and changeable. Allows duplicate members.

5. **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

6. **Set** is a collection which is unordered and unindexed. No duplicate members.

Why Lists?

Tuples are immutable.

Lists are mutable.

Why Lists?

- Tuples are immutable.

```
tup = (1, 2, 3)
tup[0] = 5
TypeError: 'tuple' object does not
  support item assignment
```

- Lists are mutable.

```
lst = [1, 2, 3]
lst[0] = 5
>>> lst
>>> [5, 2, 3]
```

Creating a List

- Using square brackets:

```
empty_lst = []

lst = [1, 2, 3]

type(lst) → <class 'list'>
```

Creating a List

- Using a list() operator:

Syntax : list(seq)

`list()` converts any *sequence* into a list

```
empty_lst = list()

# converting a tuple to a list
lst1 = list((1, 2, 3))    → [1, 2, 3]

# converting a string to a list
lst2 = list('abc')    → ['a', 'b', 'c']
```

List Operations

- Indexing and Slicing:

```
lst = list(range(5)) → [0, 1, 2, 3, 4]

lst[4]   → 4
lst[2:] → [2, 3, 4]
lst[1:4:2] → [1, 3]
```

These operations are common to other sequences like tuple and string.

List Operations

- Change item value by assignment:

```
lst = list(range(5)) → [0, 1, 2, 3, 4]

lst[4] = 5

>>> lst  → [0, 1, 2, 3, 5]
```

Lists are mutable!

List Operations

- Membership – check if an item exists in a list:

```
lst = list(range(5)) → [0, 1, 2, 3, 4]

>>> 3 in lst        → True
>>> 5 not in lst    → True
```

List Operations

```
lst = [3, 1, 4, 7, 3]
```

- Length – no. of elements in the list:

```
>>> len(lst) → 5
```

- Sum :

```
>>> sum(lst) → 18
>>> sum('a','b') → Error
```

List Operations

- Maximum and Minimum :

```
lst1 = [3, 1, 4, 7, 3]
lst2 = ['e', 'M', 'E', '1']

>>> max(lst1)
>>> 7

>>> max(lst2)
>>> 'e'

>>> min(lst1)
>>> 1

>>> min(lst2)
>>> '1'
```

List Operations

```
lst = [3, 1, 4, 7, 3]
```

- index() : Locate the first index of the element

```
>>> lst.index(3) → 0
>>> lst.index(4) → 2
>>> lst.index(5) → Error!
```

- count() :

```
>>> lst.count(3) → 2
>>> lst.count(4) → 1
>>> lst.count(9) → 0
```

List Operations

```
lst = [0, 1, 2, 3, 4]
```

- reverse() :

```
>>> lst.reverse()
>>> lst  → [4, 3, 2, 1, 0]
```

- copy() :

```
>>> new_lst = lst.copy()
>>> new_lst   → [4, 3, 2, 1, 0]
```

`t = lst` **is just merely an assignment!**

List Operations

- Iteration – looping through a list: #Method 1

```
lst = [3,5,1,4,2]
for ele in lst:
    print(ele)
```

```
3
5
1
4
2
```

List Operations

- Iteration – looping through a list: #Method 2

```
lst = [3,5,1,4,2]
for i in range(len(lst)):
    print(lst[i])
```

```
3
5
1
4
2
```

List Operations

- Iteration – looping through a list: #Method 3

```
lst = [3,5,1,4,2]
while lst:              # while lst not empty
    print(lst[0])
    lst = lst[1:]
3
5
1
4
2
```

Note that lst is **mutated !!!**

List Operations

- Adding item(s) into a list – Concatenation:

```
x = [1,2,3]
y = ['a','b','c']
z = x + y

>>> z
>>> [1,2,3,'a','b','c']
```

List Operations

- Adding item(s) into a list – Repetition:

```
x = [1,2]
y = x * 3

>>> y
>>> [1,2,1,2,1,2]
```

List Operations

- Adding item(s) into a list – append() method:

```
lst = [1, 2, 3]
lst.append(4)
```

Can append only one item.

```
>>> lst → [1, 2, 3, 4]
```

lst is being **mutated !!!**

Cannot append into a tuple:

```
tup = (1, 2, 3)
tup = tup + (4,)
```

A new tuple is created!

```
>>> tup = (1,2,3)
>>> id(tup)
1474342251256
>>> tup = tup + (4,)
>>> id(tup)
1474342198872
```

List Operations

- Adding item(s) into a list – extend() method:

Syntax : list.extend(seq)

```
lst = [1,2]
lst.extend([3,4]) → [1, 2, 3, 4]

lst.extend((5,6)) → [1, 2, 3, 4, 5, 6]

lst.extend('hi')
→ [1, 2, 3, 4, 5, 6, 'h', 'i']

lst.extend(6) → Error!
```

Concatenation vs Extend

```
lst1 = [1,2]
lst1.extend([3,4])
>>> lst1   → [1, 2, 3, 4]
```

Similar to ...

```
lst2 = [1, 2]
lst2 = lst2 + [3,4]
>>> lst2   → [1, 2, 3, 4]
```

Concatenation vs Extend

```
lst1 = [1,2]
lst1.extend('hi')
>>> lst1   → [1, 2, 'h', 'i']
```

But ...

```
lst2 = [1, 2]
lst2 = lst2 + 'hi' → Error!
```

Append vs Extend

```
lst1 = [1,2]
lst1.append('hi')
>>> lst1   → [1, 2, 'hi']


lst2 = [1,2]
lst2.extend('hi')
>>> lst2   → [1, 2, 'h', 'i']
```

Append vs Extend

```
lst1 = [1,2]
lst1.append([3,4])
>>> lst1    → [1, 2, [3,4]]


lst2 = [1,2]
lst2.extend([3,4])
>>> lst2    → [1, 2, 3, 4]
```

List Operations

- Adding item(s) into a list – insert() method:

Syntax : list.insert(index, new_item)

```
lst = [0,1]
lst.insert(0,'a') → ['a',0,1]

lst.insert(1,'b') → ['a','b',0,1]
lst.insert(4,'c') → ['a','b',0,1,'c']

lst.insert(9,'d')
→ ['a','b',0,1,'c','d']
```

List Operations

- Deleting item(s) from a list – remove() method:

Syntax : list.remove(item)

```
lst = [0,1,2,3,4,5,6,7,8,9]
lst.remove(4) → [0,1,2,3,5,6,7,8,9]

lst.remove(14) → Error! No such item.
```

It does not output the deleted item(s)!

List Operations

- Deleting item(s) from a list – pop() method:

Syntax : list.pop(index)

```
lst = [0,1,2,3,4,5,6,7,8,9]
>>> lst.pop()  → 0
>>> lst  → [1,2,3,5,6,7,8,9]

>>> lst.pop(4) → 6
>>> lst  → [1,2,3,5,7,8,9]
```

It will output the deleted item.

List Operations

- Deleting item(s) from a list – del function:

Syntax : del list[start:stop:step]

```
lst = [0,1,2,3,4,5,6,7,8,9]
del lst[1:9:2] → [0,2,4,6,8,9]

del lst[-1] → [0,2,4,6,8]
del lst[:] → []        same as del.clear()
del lst → lst no longer exist!
```

It does not output the deleted item(s)!

List Operations

```
lst = [3, 1, 4, 7, 2]
```

- Sorting a list:

```
>>> new_lst = sorted(lst)
>>> lst  → [3,1,4,7,2]
>>> new_lst  → [1,2,3,4,7]

>>> lst.sort()
>>> lst  → [1,2,3,4,7]
```

lst is being **mutated !!!**

List are Mutable

- Is this mutation?

```
lst = [1, 2, 3]
lst = lst + [4, 5, 6]
lst → [1, 2, 3, 4, 5, 6]
```

Answer: Yes!!

Mutable versus Immutable

```
lst = [1,2,3]
lst2 = lst
lst == lst2   → True
lst is lst2   → True
```

This is an assignment

Mutable

```
lst += [4,5,6]
lst  → [1,2,3,4,5,6]
lst2 → [1,2,3,4,5,6]
lst == lst2   → True
lst is lst2   → True
```

Mutable versus Immutable

```
tup = (1,2,3)
tup2 = tup
```

```
tup == tup2   → True
tup is tup2   → True
```

```
>>> tup = (1,2,3)
>>> tup2 = tup
>>> id(tup)
2797670539056
>>> id(tup2)
2797670539056
>>> tup += (4,5,6)
>>> tup
(1, 2, 3, 4, 5, 6)
>>> tup2
(1, 2, 3)
>>> id(tup)
2797670159944
>>> id(tup2)
2797670539056
```

A new tuple is created.

```
tup += (4,5,6)
tup      → (1,2,3,4,5,6)
tup2     → (1,2,3)
tup == tup2     → False
tup is tup2     → False
```

Python Lists: Summary

- Lists are *sequences* and can be used with all the *sequence* operations
- Lists are *mutable* and have *mutable* operations as well which are not common to tuples and strings