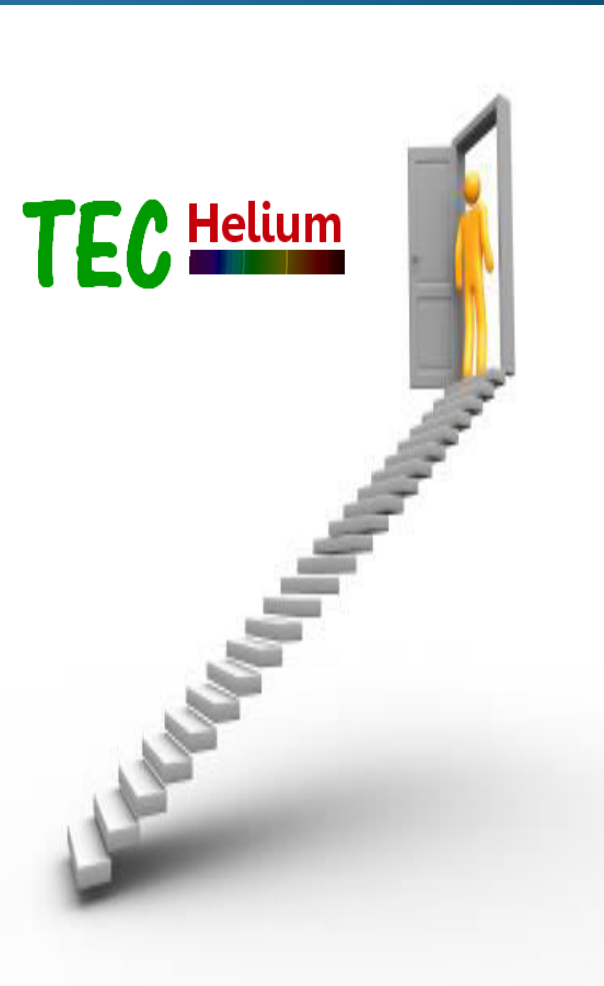


Lets Face it, now

Project Oriented Python

List Manipulation & Programs



List functions

- `list1 = ['apple', 'mango', 'banana']`
- `list1.append('papaya')` ## `['apple', 'mango', 'banana', 'papaya']`
- `list1.count(obj)` : Count how many time obj occurs
- `list1.extend(seq)` : add list1 and seq
- `list1.insert(index, obj)` Add obj at index
- `list1.pop()` : remove and return last element from list
- `list1.remove(obj)`
- `list1.reverse()`
- `list1.sort()`

The 'in' Operator

- Boolean test whether a value is inside a container:

```
>>> t = [1, 2, 4, 5]
>>> 3 in t
False
>>> 4 in t
True
>>> 4 not in t
False
```

- For strings, tests for substrings

```
>>> a = 'abcde'
>>> 'c' in a
True
>>> 'cd' in a
True
>>> 'ac' in a
False
```

Operations on Lists Only

```
>>> li = [1, 11, 3, 4, 5]
```

```
>>> li.append('a') # Note the method  
syntax
```

```
>>> li  
[1, 11, 3, 4, 5, 'a']
```

```
>>> li.insert(2, 'i')
```

```
>>> li  
[1, 11, 'i', 3, 4, 5, 'a']
```

The *extend* method vs +

- + creates a fresh list with a new memory ref
- *extend* operates on list `li` in place.

```
>>> li.extend([9, 8, 7])
>>> li
[1, 2, 'i', 3, 4, 5, 'a', 9, 8, 7]
```

- *Potentially confusing:*

- *extend* takes a list as an argument.
- *append* takes a singleton as an argument.

```
>>> li.append([10, 11, 12])
>>> li
[1, 2, 'i', 3, 4, 5, 'a', 9, 8, 7, [10, 11, 12]]
```

Operations on Lists Only

Lists have many methods, including index, count, remove, reverse, sort

```
>>> li = ['a', 'b', 'c', 'b']
```

```
>>> li.index('b')    # index of 1st occurrence  
1
```

```
>>> li.count('b')    # number of occurrences  
2
```

```
>>> li.remove('b')   # remove 1st occurrence
```

```
>>> li  
['a', 'c', 'b']
```

Thanks