Data Structure List

June 8, 2024

[15]:
```
#List characteristics
"""Ordered: The items in a list have a defined order, and that order will not␣
 ↪change1.
Changeable (Mutable): You can change, add, and remove items in a list after it␣
 ↪has been created1.
Allow Duplicates: Since lists are indexed, lists can have items with the same␣
 ↪value1.
Indexing: List items are indexed, the first item has index [0], the second item␣
 ↪has index 1 etc1."""
```

[15]:
```
'Ordered: The items in a list have a defined order, and that order will not
change1.\nChangeable (Mutable): You can change, add, and remove items in a list
after it has been created1.\nAllow Duplicates: Since lists are indexed, lists
can have items with the same value1.\nIndexing: List items are indexed, the
first item has index [0], the second item has index 1 etc1.'
```

[ ]:
```
'''append(item): Adds an item to the end of the list1.
extend(iterable): Adds all the items from the iterable to the end of the list1.
insert(index, item): Inserts an item at a specific position in the list1.
remove(item): Removes the first occurrence of an item from the list1.
pop(index): Removes and returns the item at the given index1.
clear(): Removes all items from the list1.
index(item): Returns the index of the first occurrence of an item in the list1.
count(item): Returns the number of times an item appears in the list1.
sort(): Sorts the items of the list in place1.
reverse(): Reverses the order of items in the list1.
copy(): Returns a shallow copy of the list1.'''
```

[2]:
```
myList = ["apple","banana","Cherry"]
```

[3]:
```
#List Length
print(len(myList))
```

```
3
```

[4]:
```
print(type(myList))
```

```
<class 'list'>
```

[9]:
```
#positive Indexing
print(myList[0])
```

```
apple
```

[10]:
```
print(myList[1])
```

```
banana
```

[11]:
```
print(myList[2])
```

```
Cherry
```

[5]:
```
#Negative Indexing
print(myList[-1])
```

```
Cherry
```

[7]:
```
print(myList[-2])
```

```
banana
```

[8]:
```
print(myList[-3])
```

```
apple
```

[20]:
```
#use of in keyword
if "apple" in myList:
    print("Yes Apple is in fruits list")
```

```
Yes Apple is in fruits list
```

[14]:
```
#Change Item Value
myList[1]="blackcurrant"
myList
```

[14]: ['apple', 'blackcurrant', 'Cherry']

[17]:
```
#Change a Range of Item Values
myList[1:3]=["orange","kiwi", "mango"]
myList
```

[17]: ['apple', 'orange', 'kiwi', 'mango', 'mango']

[31]:
```
#Insert()
myList.insert(2,"watermelon")
myList
```

[31]: ['apple', 'orange', 'watermelon', 'kiwi', 'mango', 'mango', 'blackcurrent']

[21]:
```
#append()
myList.append("blackcurrent")
myList
```

[21]:
```
['apple',
 'orange',
 'watermelon',
 'watermelon',
 'kiwi',
 'mango',
 'mango',
 'blackcurrent']
```

[24]:
```
#extend()
FighterJets=["F-35 Lightning ll","F-22 Raptor","F-15EX","Block III F/A-18 Super␣
 ↪Hornet"]
FighterJetsRU=["su-35","su-57","MiG-29","Mig-35"]
FighterJets.extend(FighterJetsRU)
FighterJets
```

[24]:
```
['F-35 Lightning ll',
 'F-22 Raptor',
 'F-15EX',
 'Block III F/A-18 Super Hornet',
 'su-35',
 'su-57',
 'MiG-29',
 'Mig-35']
```

[32]:
```
#remove()
print("Before Remove ",myList)
myList.remove("watermelon")
print("After Remove ",myList)
myList
```

```
Before Remove  ['apple', 'orange', 'watermelon', 'kiwi', 'mango', 'mango',
'blackcurrent']
After Remove  ['apple', 'orange', 'kiwi', 'mango', 'mango', 'blackcurrent']
```

[32]:
```
['apple', 'orange', 'kiwi', 'mango', 'mango', 'blackcurrent']
```

[33]:
```
#pop()
print("Before pop() ",myList)
myList.pop(1)
print("After pop() ",myList)
myList
```

```
Before pop()  ['apple', 'orange', 'kiwi', 'mango', 'mango', 'blackcurrent']
After pop()  ['apple', 'kiwi', 'mango', 'mango', 'blackcurrent']
```

[33]:
```
['apple', 'kiwi', 'mango', 'mango', 'blackcurrent']
```

[34]:
```
#del keyword emoves the specified index + can also delete the list completely
print("Before",FighterJets)
del FighterJets[3]
print("After",FighterJets)
```

```
Before ['F-35 Lightning ll', 'F-22 Raptor', 'F-15EX', 'Block III F/A-18 Super
Hornet', 'su-35', 'su-57', 'MiG-29', 'Mig-35']
After ['F-35 Lightning ll', 'F-22 Raptor', 'F-15EX', 'su-35', 'su-57', 'MiG-29',
'Mig-35']
```

[35]:
```
del FighterJetsRU #delete the list completely
```

[37]:
```
 '''Clear the List
The clear() method empties the list.

The list still remains, but it has no content'''

myFriend = ["Bulbul","Remon","Shovon","Mirza"]
myFriend.clear()
print("After Clear ",myFriend)
```

```
After Clear  []
```

[38]:
```
#Print all items in the list, one by one:
for items in FighterJets:
    print(items)
```

```
F-35 Lightning ll
F-22 Raptor
F-15EX
su-35
su-57
MiG-29
Mig-35
```

[39]:
```
#Print all items in the list by index:
for i in range(len(FighterJets)):
    print(FighterJets[i])
```

```
F-35 Lightning ll
F-22 Raptor
F-15EX
su-35
su-57
MiG-29
Mig-35
```

[41]:
```
'''The enumerate() function in Python is a built-in function that adds a␣
 ↪counter to an iterable and returns it'''
for index, items in enumerate(FighterJets):
    print(index,items)
```

```
0 F-35 Lightning ll
1 F-22 Raptor
2 F-15EX
3 su-35
4 su-57
5 MiG-29
6 Mig-35
```

[40]:
```
#help Keyword
help(myList)
```

```
Help on list object:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(…)
 |      x.__getitem__(y) <==> x[y]
```

```
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __iadd__(self, value, /)
|      Implement self+=value.
|
|  __imul__(self, value, /)
|      Implement self*=value.
|
|  __init__(self, /, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(self, /)
|      Return a reverse iterator over the list.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(self, /)
|      Return the size of the list in memory, in bytes.
|
|  append(self, object, /)
|      Append object to the end of the list.
```

```
 |
 |  clear(self, /)
 |      Remove all items from list.
 |
 |  copy(self, /)
 |      Return a shallow copy of the list.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  extend(self, iterable, /)
 |      Extend list by appending elements from the iterable.
 |
 |  index(self, value, start=0, stop=9223372036854775807, /)
 |      Return first index of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  insert(self, index, object, /)
 |      Insert object before index.
 |
 |  pop(self, index=-1, /)
 |      Remove and return item at index (default last).
 |
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
 |      order of two equal elements is maintained).
 |
 |      If a key function is given, apply it once to each list item and sort
them,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
```

```
|  Class methods defined here:
|
|  __class_getitem__(…) from builtins.type
|      See PEP 585
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __hash__ = None
```

[42]:
```
#sort
numbers=[45,3,67,4,65,32,90]
numbers.sort()
numbers
```

[42]:
```
[3, 4, 32, 45, 65, 67, 90]
```

[43]:
```
numbers=[45,3,67,4,65,32,90]
numbers.sort(reverse=True)
numbers
```

[43]:
```
[90, 67, 65, 45, 32, 4, 3]
```

[44]:
```
#copy
mainList=["123","123","123","123","123"]
temp=mainList.copy()
print("temp: ",temp)
```

```
temp:  ['123', '123', '123', '123', '123']
```

[ ]:
```

```