Lists basics

A *container* is a construct used to group related values together and contains references to other objects instead of data.

A `list` is a container created by surrounding a sequence of variables or literals with brackets [ ].

Example: `my_list = [10, 'abc',2.5]`

creates a new list variable `my_list` that contains the three items: `10`, `'abc'`, and `2.5`

A list item is called an *element* .

Lists basics

A `list` is also a sequence (just like `string`), meaning the contained elements are ordered by position in the list, known as the *element's index*, starting with 0.

```
>>> myList=[1,2,3,"a","hello",8]
>>> myList[2]
3
```

3 is at position 2

position 2

```
>>> my_list = [ ]
```

creates an empty list

Lists basics

A `list` is also a sequence (just like `string`), meaning the contained elements are ordered by position in the list, known as the *element's index*, starting with `0`.

```
>>> names = ["Jane","Margo","Sally"]
>>> names          print all elements of the list names
['Jane', 'Margo', 'Sally']

>>> names.append("Sam")  add another element to the list
>>> names                display the list
['Jane', 'Margo', 'Sally', 'Sam']
```

Lists basics

A `list` is also a sequence (just like `string`), meaning the contained elements are ordered by position in the list, known as the *element's index*, starting with 0.

```
>>> myList2=[1,2,3,5]
>>> myList2[2] = 7
>>> myList2
[1,2,7,5]

>>> len(myList2)          get the length of the list
4
```

Lists basics

A `list` is also a sequence (just like `string`), meaning the contained elements are ordered by position in the list, known as the *element's index*, starting with 0.

```
myList2:
[1,2,7,5]
```

```
>>> myList2.pop()          remove the last element
5

>>> myList2.pop(1)      remove the element with index 1
>>> myList2
[1,7]
```

Lists basics

A `list` is also a sequence (just like `string`), meaning the contained elements are ordered by position in the list, known as the *element's index*, starting with 0.

```
>>> myList3=[10,9,8,7,6,5,4,6,3,2,1]
>>> myList3.remove(6)  finds and removes first occurrence of value 6 in the list

>>> myList3
[10,9,8,7,5,4,6,3,2,1]

>>> myList3.index(8)    get the position/index of value 8
2
```

Lists basics

A `list` is also a sequence (just like `string`), meaning the contained elements are ordered by position in the list, known as the *element's index*, starting with `0`.

```
myList3:
[10,9,8,7,5,4,6,3,2,1]
```

```
>>> myList3.insert(3,12)
>>> myList3
[10,9,8,12,7,5,4,6,3,2,1]
```

Lists basics: in-class work

Assume we have two lists:

`a = [1,2,3,4]` and `b = ['a','b','c']`

Answer the following questions
(assume you are working with Python Shell):

1) How to display the last element of the list `a`?

2) How to remove the last element from the list `b`?

3) The function call `len(a)` returns the length of the list `a`. Write the command to display the sum of the lengths of lists `a` and `b` (using `len()`, `print()` and `+`).

Lists basics: in-class work

Still working with lists

`a = [1,2,3,4]` and `b = ['a','b','c']`

Answer the following questions
(assume you are working with Python Shell):

4) What will happen if you type the following statement:
`a+b` ?

5) How can we “save” the result of `(a+b)` so that we could use it sometime later in a program?

This OER material was produced as a result of the CS04ALL CUNY OER project.