LECTURE 5

LISTS

MCS 260 Fall 2021

David Dumas

REMINDERS

- No class on Monday (Labor day)
- Homework 2 will be posted Thursday and due next **Wednesday at 10am** (24 hours later than usual due to holiday).

LISTS

A **list** is an object in Python that stores a sequence of values (of any type).

They are written in Python as comma-separated lists surrounded by square brackets.

```
>>> L = [4, "red", 2.2, [5,6]]  # Square bracket = list
>>> L
[4, 'red', 2.2, [5, 6]]
>>> type(L)
<class 'list'>
```

If `L` is a list, `len(L)` gives the number of elements.

```
>>> len(L)
4
```

The empty list exists and is written `[]`.

`list + list` will join lists.

```
>>> [1,2,3] + [4,5,6]
[1, 2, 3, 4, 5, 6]
```

`list * n` is equivalent to
`list + list + ... + list` (n times).

```
>>> [1,99]*3
[1, 99, 1, 99, 1, 99]
```

INDEXING

Items can be retrieved by 0-based index (position):

```
>>> L = [4,8,15,16,23,42]
>>> L[2]
15
>>> L[0]
4
>>> L[4]
23
```

MUTABILITY

Lists are **mutable**, meaning that the contents can be changed.

```
>>> L = [4,8,15,16,23,42]
>>> L[2] = 999
>>> L
[4, 8, 999, 16, 23, 42]
```

An element of a list can be deleted with the **del** keyword. Elements to the right move over.

```
>>> L = [4,8,15,16,23,42]
>>> del L[2]
>>> L
[4, 8, 16, 23, 42]
```

You can't access or assign list indices that don't exist:

```
>>> L = [4,8,15,16,23,42]
>>> L[6]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> L[6] = 121
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
```

You can add a new element to a list as follows:

```
>>> L.append(121)
>>> L
[4, 8, 15, 16, 23, 42, 121]
```

SEQUENCES

Lists are examples of Python **sequences**: ordered collections of elements that can be retrieved by index.

Strings are sequences, too. Indexing them retrieves a single character.

```
>>> "asdf"[2]      # string indexing
'd'
```

`len(string)` gives the number of characters.

NEGATIVE INDICES

Sequences allow negative indices, where `-1` refers to the last element, `-2` to the second to last, etc.

```
>>> [2,6,0,2,0,2,1][-1]
1
>>> "Oklahoma!"[-3]
'm'
```

In many cases, negative indices allow you to avoid using `len(...)`.

SLICES

Slices are a way to retrieve part of a sequence, e.g. a contiguous segment. The syntax

```
x[i:j]
```

retrieves elements of `x` with 0-based indices between `i` and `j`, *including* `i` *but not including* `j`.

Either `i` or `j` can be omitted; missing `i` is taken to be 0, missing `j` is taken to be just past the end.

```
>>> s = "learning Python"
>>> s[1:4]
'ear'
>>> s[:5]
'learn'
>>> s[5:]
'ing Python'
>>> s[:-1]
'learning Pytho'
>>> s[:]
'learning Python'
```

l	e	a	r	n	i	n	g		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

MEMBERSHIP TESTING

Python's `in` keyword is an operator that tests whether a value is one of the elements of a list.

```
>>> 5 in [2,3,0,2,2]
False
>>> 2 in [2,3,0,2,2]
True
>>> [1,2] in [[4,5,8],[1,2],[1,2,3]]
True
>>> 2 in [[4,5,8],[1,2],[1,2,3]]
False
```

For testing non-membership you can use

```
x not in L  # Equivalent to:  not (x in L)
```

IN OPERATOR FOR STRINGS

When used with strings, the `in` operator checks whether one string appears inside another:

```
>>> "e" in "team"
True
>>> "i" in "team"
False
>>> "car" in "vicarious"
True
>>> "rat" in "perambulate"
False
```

SEQUENCE CONVERSION

The function `list()` will convert another sequence to a list.

```
>>> list("abc")
['a', 'b', 'c']
```

Careful: `str()` exists but doesn't convert a sequence to a string in the corresponding way.

REFERENCES

- In *Downey*:
  - Lists are covered in Chapter 10 (includes more material than in today's lecture)

REVISION HISTORY

- 2021-08-31 Initial publication