north austin pythonistas

present

Python Builtin Containers: Lists

erik.oshaughnessy@gmail.com

Builtin Containers

- **Lists**
  - Ordered collection of items.
- Dictionary
  - Ordered collection of key/value pairs.
- Sets
  - Unordered collection of unique elements.

Lists: Ordered Collections


```
>> l = [1, ‘two’, 3.0]
>> l[0] == 1
True
>> l[1] == ‘two’
True
>> l[2] == 3.0
True
```

Constructing Lists

Using square brackets:

```
>> l = [] # empty list
>> l = ‘a’, ‘b’, ‘c’, ‘d’]
>> l = [expr for variable in sequence]
```

Using builtin **list** type:

```
>> l = list() # empty list
>> l = list(sequence)
>> l = list(expr for variable in sequence)
```

Adding Elements to Lists

Single items are added to the end of a list using the **append** method:

```
>> l = []
>> l.append(1)
>> l.append(2)
>> l
[1, 2]
```

Adding Elements to Lists (cont)

Use the **insert** method to add elements at specific indices:

```
>> l = [0, 1, 2, 3, 4]
>> l.insert(1, 'foo')
>> l
[0, 'foo', 1, 2, 3, 4]
```

Adding Elements to Lists (cont)

Use the **extend** method to add sequences of items to the end of a list:

```
>> l = []
>> numbers = [0, 1, 2, 3, 4]
>> l.extend(numbers)
[0, 1, 2, 3, 4]
```

Adding Elements to Lists (cont)

Lists can be added together to create new lists:

```
>> a = [0, 1, 2]
>> b = [3, 4, 5]
>> c = a + b
>> c
[0, 1, 2, 3, 4, 5]
```

Accessing Elements In Lists

Lists elements are accessed by their position (index number) in the list:

```
>> l = [1, 2, 3, 4, 5]
>> l[1] == 2
True
>> l[0] == 1
True
```

Accessing Elements In Lists (cont)

Lists indices can be positive or negative:

```
>> l = [1, 2, 3, 4, 5]
>> l[-1] == 5
True
>> l[-2] == 4
True
```

Changing Elements In Lists

```
>> l = [1, 2, 3, 4, 5]
>> l[0] = ‘foo’
>> l
[‘foo’, 2, 3, 4, 5]
>> l[5] = ‘bar’
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
```

Slices of Lists

A “slice” is a special kind of index that can be used to get sub-lists:

```
list_variable[start:stop:step]
```

```
>> l = [1, 2, 3, 4, 5]
>> l[1:3] # default step of 1
[2, 3]
```

Fun With Slices


```
>> l = [1, 2, 3, 4, 5]
>> m = l[:] # creates new copy of 'l'
>> l[1:]    # list trimming the first item
[2, 3, 4, 5]
>> l[:-1]   # list trimming the last item
[1, 2, 3, 4]
>> l[::-1]  # reversed copy of the list!
[5, 4, 3, 2, 1]
```

Removing Elements From Lists


```
>> l = [0, 1, 2, 3, 4]
>> l.pop(0)
0
>> l
[1, 2, 3, 4]
>> l.pop(-1)
4
>> l
[1, 2, 3]
```

Removing Elements From Lists (cont)

```
>> l = [‘a’, ‘b’, ‘c’, ‘d’, ‘c’]
>> l.remove(‘c’)
>> l
[‘a’, ‘b’, ‘d’, ‘c’]
>> l.remove(‘c’)
[‘a’, ‘b’, ‘d’]
```

Removing Elements From Lists (cont)

```
>> l = ['a', 'b', 'c', 'd',]
>> del(l[0])
>> l
['b', 'c', 'd']
>> del(l[-1])
>> l
['b', 'c']
```

Interrogating Lists

```
>> l = ‘a’, ‘b’, ‘c’, ‘d’]
>> len(l)          # number of items in l
4
>> ‘a’ in l        # test for membership
True
>> ‘z’ in l
False
```

Interrogating Lists (cont)

```
>> l = ['c', 'a', 'c', 'c', 'd']
>> l.count('c')
3
>> l.index('c')
0
>> l.pop('c')
>> l.index('c')
1
```

Re-arranging Lists

```
>> l = ‘q’, ‘w’, ‘e’, ‘r’, ‘t’, ‘y’]
>> l.sort()
>> l
[‘e’, ‘q’, ‘r’, ‘t’, ‘w’, ‘y’]
>> l.reverse()
>> l
[‘y’, ‘w’, ‘t’, ‘r’, ‘q’, ‘e’]
```

Lists in For Loops


```
>> l = ‘a’, ‘b’, ‘c’, ‘d’]
>> for value in l:
>>     print(value)

a
b
c
d
```

Lists in For Loops (cont)

```
>> l = ‘a’, ‘b’, ‘c’, ‘d’]
>> for index, value in enumerate(l):
>>     print(index, value)
0 a
1 b
2 c
3 d
```

Lists in For Loops (cont)


```
>> l = ‘a’, ‘b’, ‘c’, ‘d’]
>> for value in l[1:-1]:
>>     print(value)
b
c
>> for value in l[::2]:
>>     print(value)
a
c
```

Lists Comprehensions

List comprehensions are a very pythonic idiom for operating on a list and generating a new list:

```
new_list = [expr for value in old_list]
```

Lists Comprehensions (cont)

```
>> l = [0, 1, 2, 3, 4]
>> m = []
>> for value in l:
        m.append(value+2)
>> m
[2, 3, 4, 5, 6]
>> n = [value+2 for value in l]
>> m == n
True
```

Find More Help

Documentation:

https://docs.python.org/3/library/stdtypes.html#list

```
$ pydoc list            # shell command-line, python2
$ pydoc3 list           # shell command-line, python3
>> help(list)           # python[23] interpreter
# windows people help me out :)
```

Tutorials:

https://docs.python.org/3/tutorial/introduction.html#lists
https://docs.python.org/3/tutorial/datastructures.html#more-on-lists