Lists

Fall 2019

Sequences

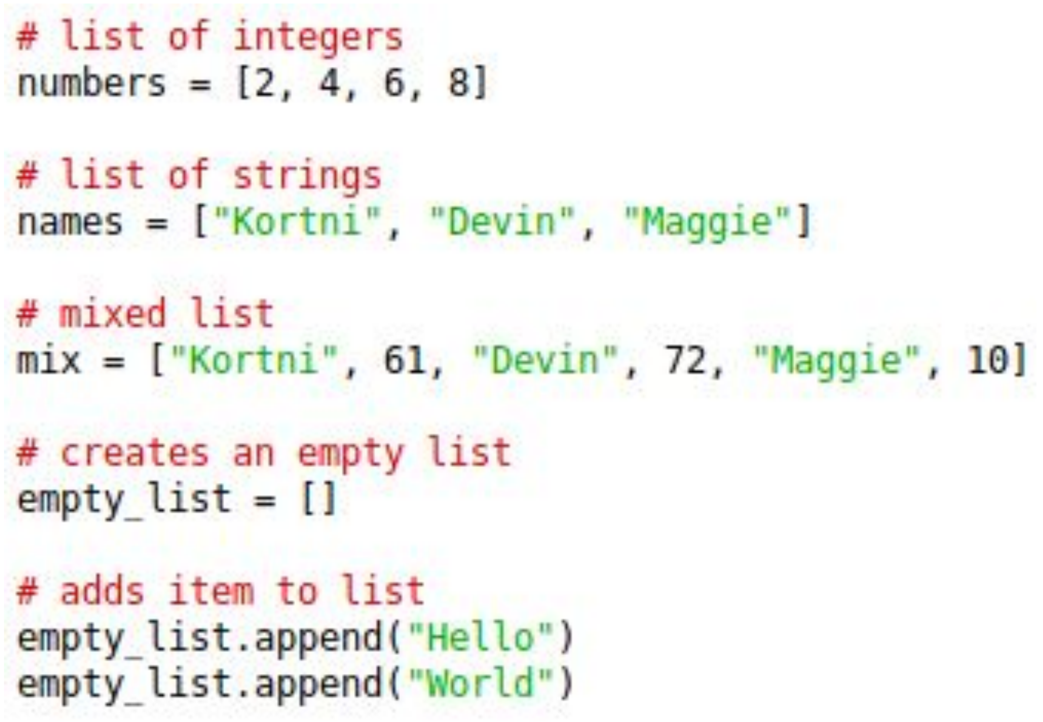
- An object that holds multiple items, stored one right after the other
    - Can use different operations to manipulate and view data
- Two main types of sequences
    - Lists
        - Items can be changed or added
    - Tuples
        - Items *cannot* be changed or added

Lists

- An object that contains multiple data items
- Lists are dynamic structures
    - Contents inside can be changed
    - New contents can be added
    - Items can also be removed from the list
- Python lists can contain multiple data types
    - Ex. can have a list that is ints and strings

Lists Anatomy

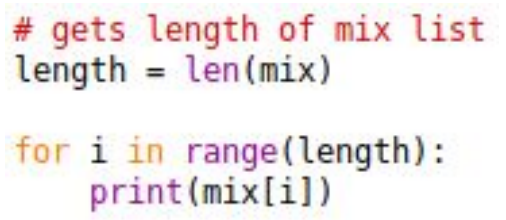
- Lists are denoted by square brackets
    - You put whatever items inside the brackets, separating each item with a comma
- You can create an empty list to add items to it later with just an empty set of brackets
    - You add items with the built-in Python append function



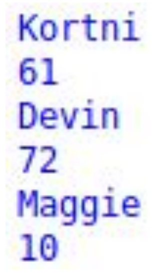
```
# list of integers
numbers = [2, 4, 6, 8]

# list of strings
names = ["Kortni", "Devin", "Maggie"]

# mixed list
mix = ["Kortni", 61, "Devin", 72, "Maggie", 10]

# creates an empty list
empty_list = []

# adds item to list
empty_list.append("Hello")
empty_list.append("World")
```

Displaying List Items

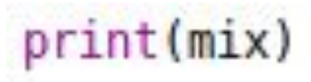
- If you print the list directly, it'll just print the entire contents in brackets
- To print the items one by one, you'll need to loop through the list by an "index" (in brackets)
  - Indices refers to what number item it corresponds to in the list
  - ***List indexing starts at 0, not 1***
    - Item 1: index 0, item 2: index 1



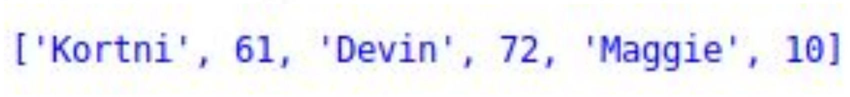
```
# gets length of mix list
length = len(mix)

for i in range(length):
    print(mix[i])
```



```
Kortni
61
Devin
72
Maggie
10
```



```
print(mix)
```



```
['Kortni', 61, 'Devin', 72, 'Maggie', 10]
```

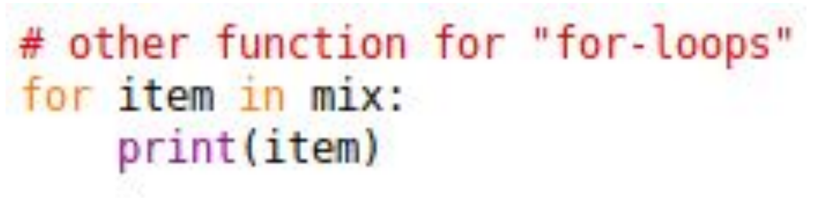
Displaying Subsets

- Can create a subset of a list using the colon in brackets
  - Can print or create a whole new list from the sublist
- Number before colon: starts at that index, includes the value at that index
- Number after colon: ends at that index, does not include value at that index

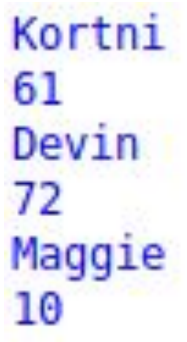
`numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`	
`print(numbers[:])`	`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
`print(numbers[:3])`	`[0, 1, 2]`
`print(numbers[1:])`	`[1, 2, 3, 4, 5, 6, 7, 8, 9]`
`print(numbers[1:5])`	`[1, 2, 3, 4]`
`print(numbers[:9])`	`[0, 1, 2, 3, 4, 5, 6, 7, 8]`

Displaying List Items

- To print items in a list, you can also use another form of the for-loop
  - This version of the for-loop only works with lists (and similar objects)
  - It was designed to make looping through lists easier
- This only works for *display* or using the list items; it will not work for altering the contents of the list



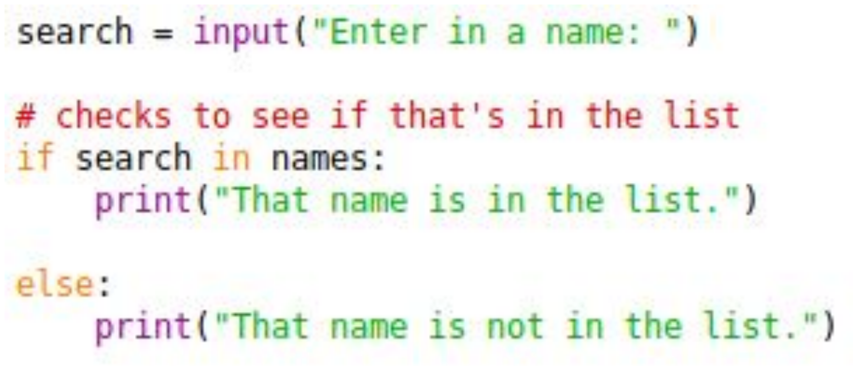
```
# other function for "for-loops"
for item in mix:
    print(item)
```



```
Kortni
61
Devin
72
Maggie
10
```

Searching Lists

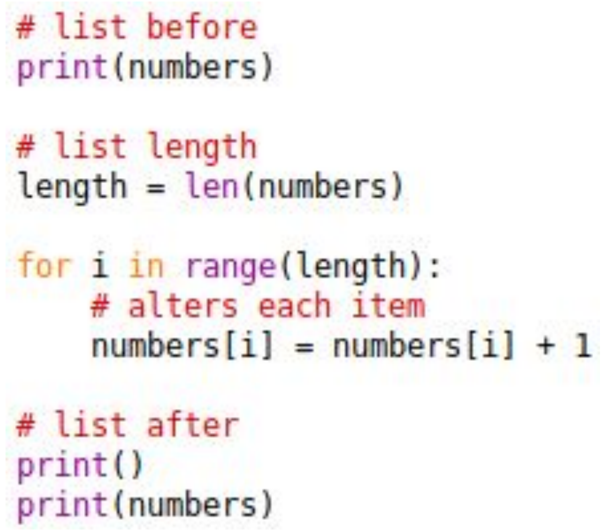
- You can check to see if an item is in the list via if-statements
- Check if a variable (either input or statically set) is in the list via “in” command, comparing it to the list you want to check
- You can check if something’s *not* in the list by adding “not” before the keyword “in”
    - if search not in names



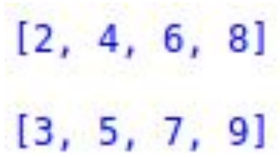
```
search = input("Enter in a name: ")

# checks to see if that's in the list
if search in names:
    print("That name is in the list.")

else:
    print("That name is not in the list.")
```

Manipulating List Items

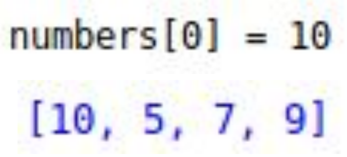
- You can change data in a list and place them back in their place
  - You can also just change the contents entirely
- You cannot use the “for … in list” loop to do this -- it only allows using the list items, not changing



```
# list before
print(numbers)

# list length
length = len(numbers)

for i in range(length):
    # alters each item
    numbers[i] = numbers[i] + 1

# list after
print()
print(numbers)
```



```
[2, 4, 6, 8]

[3, 5, 7, 9]
```

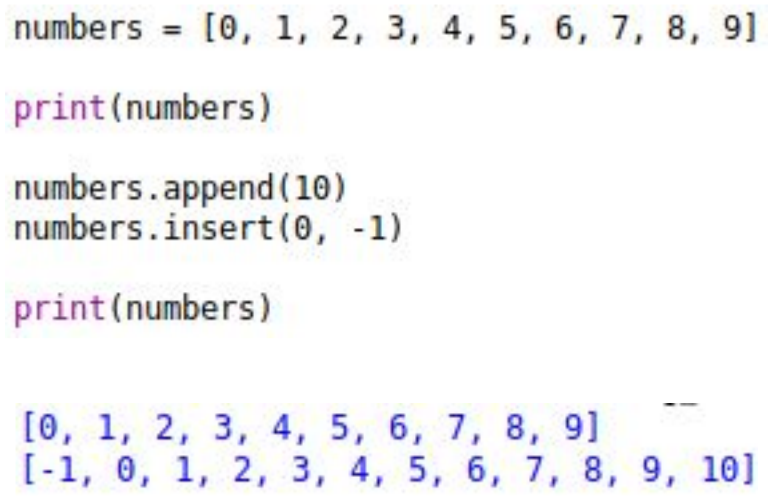


```
numbers[0] = 10
```

```
[10, 5, 7, 9]
```

Manipulating Lists

- Can add items to the end of a list using append
- Can also add items to specified indices using insert
    - First argument: index to be insert at
    - Second argument: item to be inserted

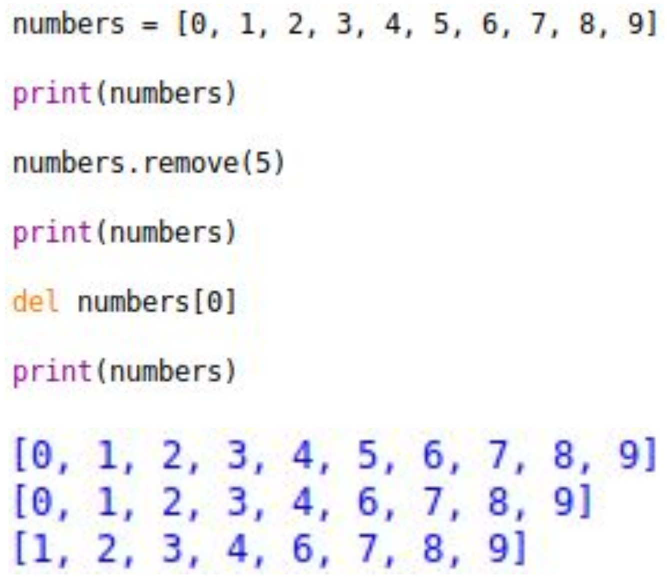


```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

print(numbers)

numbers.append(10)
numbers.insert(0, -1)

print(numbers)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Manipulating Lists

- Items can be removed from lists using two different methods
- remove
  - Removes a specific item from a list; if duplicates, only removes the first instance
- del
  - Removes an item from an index in the list



```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

print(numbers)

numbers.remove(5)

print(numbers)

del numbers[0]

print(numbers)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 6, 7, 8, 9]
[1, 2, 3, 4, 6, 7, 8, 9]
```

Lists: Notes

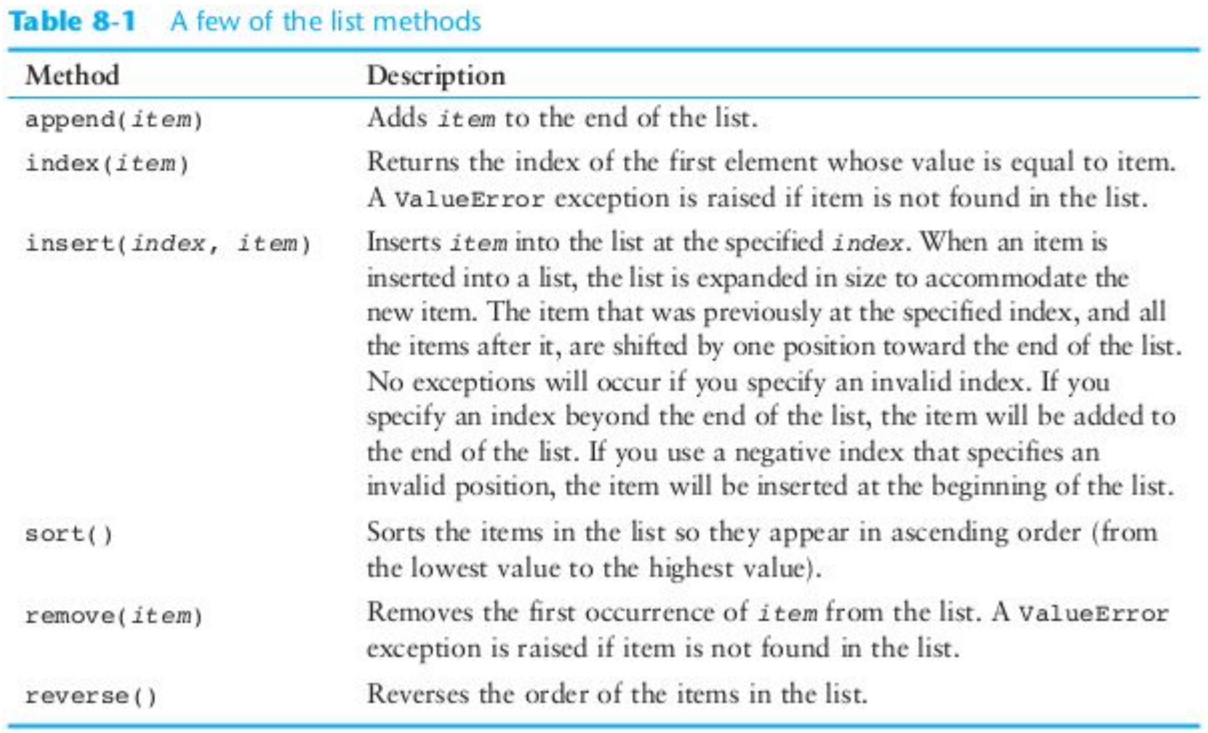


Table 8-1 A few of the list methods

Method	Description
`append(item)`	Adds *item* to the end of the list.
`index(item)`	Returns the index of the first element whose value is equal to item. A `ValueError` exception is raised if item is not found in the list.
`insert(index, item)`	Inserts *item* into the list at the specified *index*. When an item is inserted into a list, the list is expanded in size to accommodate the new item. The item that was previously at the specified index, and all the items after it, are shifted by one position toward the end of the list. No exceptions will occur if you specify an invalid index. If you specify an index beyond the end of the list, the item will be added to the end of the list. If you use a negative index that specifies an invalid position, the item will be inserted at the beginning of the list.
`sort()`	Sorts the items in the list so they appear in ascending order (from the lowest value to the highest value).
`remove(item)`	Removes the first occurrence of *item* from the list. A `ValueError` exception is raised if item is not found in the list.
`reverse()`	Reverses the order of the items in the list.