

Lists and Tuples

L555
Dept. of Linguistics, Indiana University
Fall 2010

Lists and Tuples

Sequence Basics

Indexing
Slicing
Caution!
Operations

List methods

Append, extend,
insert, remove, pop
Sort

Tuples

1 / 12

Sequences

- ▶ lists and tuples are containers for more than one element: sequences
 - ▶ example: `employee = ['Markus', 'Dickinson', 'assistant prof', 'MM317']`

Lists and Tuples

Sequence Basics

Indexing
Slicing
Caution!
Operations

List methods

Append, extend,
insert, remove, pop
Sort

Tuples

2 / 12

Sequences

- ▶ lists and tuples are containers for more than one element: sequences
 - ▶ example: `employee = ['Markus', 'Dickinson', 'assistant prof', 'MM317']`
 - ▶ another example for a sequence is a string

Lists and Tuples

Sequence Basics

Indexing
Slicing
Caution!
Operations

List methods

Append, extend,
insert, remove, pop
Sort

Tuples

2 / 12

Sequences

- ▶ lists and tuples are containers for more than one element: sequences
 - ▶ example: `employee = ['Markus', 'Dickinson', 'assistant prof', 'MM317']`
 - ▶ another example for a sequence is a string
 - ▶ each element in the sequence is assigned a position number, an **index** (starting from 0)
 - ▶ example: `employee[1]`

Lists and Tuples

Sequence Basics

Indexing
Slicing
Caution!
Operations

List methods

Append, extend,
insert, remove, pop
Sort

Tuples

2 / 12

Sequences

- ▶ lists and tuples are containers for more than one element: sequences
 - ▶ example: `employee = ['Markus', 'Dickinson', 'assistant prof', 'MM317']`
 - ▶ another example for a sequence is a string
- ▶ each element in the sequence is assigned a position number, an **index** (starting from 0)
 - ▶ example: `employee[1]`
- ▶ empty list: `[]`
- ▶ empty string: `''`

Lists and Tuples

Sequence Basics

Indexing
Slicing
Caution!
Operations

List methods

Append, extend,
insert, remove, pop
Sort

Tuples

2 / 12

Indexing

- ▶ accessing elements in a list is called **indexing**
 - ▶ example:
`greeting = 'hi there'`
`greeting[3]`

Lists and Tuples

Sequence Basics

Indexing
Slicing
Caution!
Operations

List methods

Append, extend,
insert, remove, pop
Sort

Tuples

3 / 12

Indexing

- ▶ accessing elements in a list is called **indexing**
 - ▶ example:


```
greeting = 'hi there'
greeting[3]
'hi there'[3]
```

Lists and Tuples

Sequence Basics

Indexing

Slicing

Caution!

Operations

List methods

Append, extend, insert, remove, pop

Sort

Tuples

Indexing

- ▶ accessing elements in a list is called **indexing**
 - ▶ example:


```
greeting = 'hi there'
greeting[3]
'hi there'[3]
```
 - ▶ indexing from the end: `greeting[-2]`

Lists and Tuples

Sequence Basics

Indexing

Slicing

Caution!

Operations

List methods

Append, extend, insert, remove, pop

Sort

Tuples

Indexing

- ▶ accessing elements in a list is called **indexing**
 - ▶ example:


```
greeting = 'hi there'
greeting[3]
'hi there'[3]
```
 - ▶ indexing from the end: `greeting[-2]`
 - ▶ adding sequences:


```
long_greeting = greeting + ' how are you'
```

Lists and Tuples

Sequence Basics

Indexing

Slicing

Caution!

Operations

List methods

Append, extend, insert, remove, pop

Sort

Tuples

Indexing

- ▶ accessing elements in a list is called **indexing**
 - ▶ example:


```
greeting = 'hi there'
greeting[3]
'hi there'[3]
```
 - ▶ indexing from the end: `greeting[-2]`
 - ▶ adding sequences:


```
long_greeting = greeting + ' how are you'
```
 - ▶ add multiple copies to a sequence: multiply
 - ▶ example:


```
comment = 'this is ' + 3 * 'very ' + 'good'
```

Lists and Tuples

Sequence Basics

Indexing

Slicing

Caution!

Operations

List methods

Append, extend, insert, remove, pop

Sort

Tuples

Slicing

- ▶ accessing parts of segments is called **slicing**
 - ▶ example:


```
long_greeting[3:6]
```

 - ▶ the slice starts at the first index and goes up to the second (non-inclusive)!
 - ▶ count from the end:


```
long_greeting[-5:-1]
```
 - ▶ going all the way to the end:


```
long_greeting[4:]
```
 - ▶ starting at the beginning:


```
long_greeting[:6]
```
 - ▶ steps are given as optional third number:


```
long_greeting[1:6:2]
```

Lists and Tuples

Sequence Basics

Indexing

Slicing

Caution!

Operations

List methods

Append, extend, insert, remove, pop

Sort

Tuples

Caution

Initialization
Always initialize your variables! Otherwise you may end up with a random value.

Lists are Mutable
If you perform an operation on a list, it changes the list. In contrast, tuples and strings are immutable.

Lists and Tuples

Sequence Basics

Indexing

Slicing

Caution!

Operations

List methods

Append, extend, insert, remove, pop

Sort

Tuples

Operations on Sequences

- ▶ membership:
`employee = ['Markus, Dickinson', 'assistant prof', 'MM317']`
`'MM317' in employee`
- ▶ check length:
`len(employee)`
- ▶ minimum / 'smallest' element:
`nums = [5, 102, 13, 2, 99, 154, 7]`
`min(nums)`
`min(employee)`
- ▶ maximum:
`max(nums)`

Lists and Tuples

Sequence Basics
Indexing
Slicing
Caution!
Operations

List methods
Append, extend,
insert, remove, pop
Sort

Tuples

Lists

- ▶ change elements in list (allowed because it's mutable):
`employee[2] = 'full prof'`
- ▶ delete:
`del(employee[2])`

Lists and Tuples

Sequence Basics
Indexing
Slicing
Caution!
Operations

List methods
Append, extend,
insert, remove, pop
Sort

Tuples

Queues and Stacks

FIFO and LIFO

- LIFO Last in, first out (stack)
- FIFO First in, first out (queue)

Lists and Tuples

Sequence Basics
Indexing
Slicing
Caution!
Operations

List methods
Append, extend,
insert, remove, pop
Sort

Tuples

Queue and Stack Operations

- ▶ add at the end: `append`
`employee.append('Computational Linguistics')`
- ▶ retrieve from the end: `pop`
`employee.pop()`
 - ▶ This returns a value!

Lists and Tuples

Sequence Basics
Indexing
Slicing
Caution!
Operations

List methods
Append, extend,
insert, remove, pop
Sort

Tuples

Queue and Stack Operations

- ▶ add at the end: `append`
`employee.append('Computational Linguistics')`
- ▶ retrieve from the end: `pop`
`employee.pop()`
 - ▶ This returns a value!
- ▶ add at the beginning:
`employee.insert(0, 'Linguistics')`
- ▶ retrieve from the beginning:
`employee.pop(0)`

Lists and Tuples

Sequence Basics
Indexing
Slicing
Caution!
Operations

List methods
Append, extend,
insert, remove, pop
Sort

Tuples

Sorting

- ▶ sort destructively
`nums.sort()`
 - ▶ Caution: this does not return a value but modifies the list itself!
 - ▶ wrong: `nums.sorted = nums.sort()`
- ▶ non-destructive version:
`nums_sorted = sorted(nums)`

Lists and Tuples

Sequence Basics
Indexing
Slicing
Caution!
Operations

List methods
Append, extend,
insert, remove, pop
Sort

Tuples

More List Methods

- ▶ count how often the same element is in a list: `count`
`['a', 'rose', 'is', 'a', 'rose', 'is', 'a', 'rose'].count('rose')`
- ▶ add a list destructively: `extend`
`employee.extend(['a', 'rose', 'is', 'a', 'rose', 'is', 'a', 'rose'])`
- ▶ insert revisited, you can insert anywhere ...
`employee.insert(6, 'Linguistics')`
- ▶ find the first occurrence of an element in the list: `index`
`['a', 'rose', 'is', 'a', 'rose', 'is', 'a', 'rose'].index('rose')`

Tuples

Definition

Tuples are very similar to lists but are **immutable**. So once you create them, that's it!

- ▶ Indexing and slicing work with tuples just as with lists.
- ▶ Tuples do not support methods such as sorting.
- ▶ You can create them with parentheses:
`mytuple=(10,50, 'foo')`
- ▶ Why bother? Later more ;)