Lists

Lists

A string is a sequence of characters. A *list* is a sequence of anything. As a string is an *object*, so is a list.

Lists

A string is a sequence of characters. A *list* is a sequence of anything. As a string is an *object*, so is a list.

A list value is specified by a pair of square brackets.

Lists

A string is a sequence of characters. A *list* is a sequence of anything. As a string is an *object*, so is a list.

A list value is specified by a pair of square brackets.

Example.

```
a = [1, 2, 3, 4]
```

Lists

Each element of a list is indexed by an integer, starting from 0.

To access a list `a`'s `i`th element, use `a[i]`.

Lists

Each element of a list is indexed by an integer, starting from 0.

To access a list `a`'s `i`th element, use `a[i]`.

Example.

```
>>> a = [1, 2, 3, 4]
>>> print a[2]
```


Lists

Each element of a list is indexed by an integer, starting from 0.

Lists

Each element of a list is indexed by an integer, starting from 0.

To access a list `a`'s `i`th element, use `a[i]`.

Example.

```
>>> a = [1, 2, 3, 4]
```

Lists

Each element of a list is indexed by an integer, starting from 0.

To access a list `a`'s `i`th element, use `a[i]`.

Example.

```
>>> a = [1, 2, 3, 4]
>>> print a[2]
3
```


Lists

Unlike strings, lists are *mutable*.

Example.

```
>>> b = ['trinity', 'College']
```


Lists

Unlike strings, lists are *mutable*.

Example.

```
>>> b = ['trinity', 'College']
>>> b[0] = 'Trinity'
```


Lists

Unlike strings, lists are *mutable*.

Lists

Unlike strings, lists are *mutable*.

Example.

```
>>> b = ['trinity', 'College']
>>> b[0] = 'Trinity'
>>> print b
['Trinity', 'College']
```


Lists

To traverse a list, use a *for* loop.

Example.

```
>>> list = ['Trinity', 'College']
```


Lists

Unlike strings, lists are *mutable*.

Example.

```
>>> b = ['trinity', 'College']
>>> b[0] = 'Trinity'
>>> print b
```


Lists

To traverse a list, use a *for* loop.

Lists

To traverse a list, use a *for* loop.

Example.

```
>>> list = ['Trinity', 'College']
>>> for element in list:
      print element,
Trinity College
```


Lists

To traverse a list, use a *for* loop.

Example.

```
>>> list = ['Trinity', 'College']
>>> for element in list:
      print element,
```


Lists

To *append* an element to a list,

```
>>> a = []
```


Lists

To *append* an element to a list,

Lists

To *append* an element to a list,

```
>>> a = []
>>> a.append(1)
>>> a.append(2)
```


Lists

To *append* an element to a list,

```
>>> a = []
>>> a.append(1)
```


Lists

To *append* an element to a list,

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


Lists

To *append* an element to a list,

```
>>> a = []
>>> a.append(1)
>>> a.append(2)
>>> print a
```

Lists

To *append* a list to another list,

```
>>> a = [1, 2, 3]
```


Lists

To *append* a list to another list,

```
>>> a = [1, 2, 3]
>>> b = [4, 5]
>>> a.append(b)
```


Lists

To *append* a list to another list,

Lists

To *append* a list to another list,

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

Lists

To *append* a list to another list,

```
>>> a = [1, 2, 3]
>>> b = [4, 5]
>>> a.append(b)
>>> print a
[1, 2, 3, 4, 5]
```


Lists

To *append* a list to another list,

```
>>> a = [1, 2, 3]
>>> b = [4, 5]
>>> a.append(b)
>>> print a
```


Lists

To *append* a list to another list,

```
>>> a = [1, 2, 3]
```


Lists

To *append* a list to another list,

Lists

To *append* a list to another list,

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


Lists

To *append* a list to another list,

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


Lists

To *append* a list to another list,

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


Lists

To *append* a list to another list,

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

Lists

To *delete* an element from a list,

```
>>> a = ['A', 'B', 'C', 'D, 'E']
```


Lists

To *delete* an element from a list,

```
>>> a = ['A', 'B', 'C', 'D, 'E']
>>> del a[3]
>>> print a
```


Lists

To *delete* an element from a list,

Lists

To *delete* an element from a list,

```
>>> a = ['A', 'B', 'C', 'D, 'E']
>>> del a[3]
```

Slicing strings

Strings can be *sliced* by specifying the first and last indices: `s[i:j]` is the substring of `s` starting from `i` through `j-1`.

Lists

To *delete* an element from a list,

```
>>> a = ['A', 'B', 'C', 'D, 'E']
>>> del a[3]
>>> print a
['A', 'B', 'C', 'E']
```


Slicing strings

Strings can be *sliced* by specifying the first and last indices: `s[i:j]` is the substring of `s` starting from `i` through `j-1`.

Example.

```
>>> t = 'trinity'
```


Slicing strings

Strings can be *sliced* by specifying the first and last indices: `s[i:j]` is the substring of `s` starting from `i` through `j-1`.

Example.

```
>>> t = 'trinity'
>>> print t[0:4]
```

Slicing strings

Strings can be *sliced* by specifying the first and last indices: `s[i:j]` is the substring of `s` starting from `i` through `j-1`.

Example.

```
>>> t = 'trinity'
>>> print t[0:4]
trin
>>> print t[2:6]
```


Slicing strings

Strings can be *sliced* by specifying the first and last indices: `s[i:j]` is the substring of `s` starting from `i` through `j-1`.

Example.

```
>>> t = 'trinity'
>>> print t[0:4]
trin
```


Slicing lists

Lists can also be *sliced* by specifying the first and last indices: `a[i:j]` is the sublist of `a` starting from `i` through `j-1`.

Slicing strings

Strings can be *sliced* by specifying the first and last indices: `s[i:j]` is the substring of `s` starting from `i` through `j-1`.

Example.

```
>>> t = 'trinity'
>>> print t[0:4]
trin
>>> print t[2:6]
init
```

Slicing lists

Lists can also be *sliced* by specifying the first and last indices: `a[i:j]` is the sublist of `a` starting from `i` through `j-1`.

Example.

```
>>> a = ['A', 'B', 'C', 'D', 'E']
>>> print a[1:4]
```


Slicing lists

Lists can also be *sliced* by specifying the first and last indices: `a[i:j]` is the sublist of `a` starting from `i` through `j-1`.

Example.

```
>>> a = ['A', 'B', 'C', 'D', 'E']
```


Slicing lists

Lists can also be *sliced* by specifying the first and last indices: `a[i:j]` is the sublist of `a` starting from `i` through `j-1`.

Example.

```
>>> a = ['A', 'B', 'C', 'D', 'E']
>>> print a[1:4]
['B', 'C', 'D']
```


Slicing lists

Lists can also be *sliced* by specifying the first and last indices: `a[i:j]` is the sublist of `a` starting from `i` through `j-1`.

Example.

```
>>> a = ['A', 'B', 'C', 'D', 'E']
>>> print a[1:4]
['B', 'C', 'D']
>>> print a[2:3]
```

Slicing lists

Lists can also be *sliced* by specifying the first and last indices: `a[i:j]` is the sublist of `a` starting from `i` through `j-1`.

Example.

```
>>> a = ['A', 'B', 'C', 'D', 'E']
>>> print a[1:4]
['B', 'C', 'D']
>>> print a[2:3]
['C']
```


Lists

To *delete* a sublist from a list,

```
>>> a = ['A', 'B', 'C', 'D', 'E']
```


Lists

To *delete* a sublist from a list,

```
>>> a = ['A', 'B', 'C', 'D', 'E']
>>> del a[1:3]
```


Lists

To *delete* a sublist from a list,

```
>>> a = ['A', 'B', 'C', 'D, 'E']
>>> del a[1:3]
```

Lists

To *delete* a sublist from a list,

```
>>> a = ['A', 'B', 'C', 'D, 'E']
>>> del a[1:3]
>>> print a
['A', 'D', 'E']
```


Lists

To *delete* a sublist from a list,

```
>>> a = ['A', 'B', 'C', 'D, 'E']
>>> del a[1:3]
>>> print a
```


Strings and lists

Strings are immutable, but lists are *mutable*; so we often wish to covert strings to lists.

Example.

```
>>> t = 'trinity'
```


Strings and lists

Strings are immutable, but lists are *mutable*; so we often wish to covert strings to lists.

Strings and lists

Strings are immutable, but lists are *mutable*; so we often wish to covert strings to lists.

Example.

```
>>> t = 'trinity'
>>> l = list(t)
>>> print l
```


Strings and lists

Strings are immutable, but lists are *mutable*; so we often wish to covert strings to lists.

Example.

```
>>> t = 'trinity'
>>> l = list(t)
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Strings and lists

Strings are immutable, but lists are *mutable*; so we often wish to covert strings to lists.

Example.

```
>>> t = 'trinity'
>>> l = list(t)
>>> print l
['t', 'r', 'i', 'n', 'i', 't', 'y']
```

Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['Trinity', 'College']
>>> delimiter = ' '
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['Trinity', 'College']
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['Trinity', 'College']
>>> delimiter = ' '
>>> delimiter.join(t)
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['Trinity', 'College']
>>> delimiter = ' '
>>> delimiter.join(t)
'Trinity College'
```

Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['Trinity', 'College']
>>> delimiter = '*'
>>> delimiter.join(t)
'Trinity*College'
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['h', 'e', 'l', 'l', 'o']
>>> delimiter = ' '
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['h', 'e', 'l', 'l', 'o']
```

Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['h', 'e', 'l', 'l', 'o']
>>> delimiter = ' '
>>> delimiter.join(t)
'h e l l o'
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['h', 'e', 'l', 'l', 'o']
>>> delimiter = ' '
>>> delimiter.join(t)
```


Strings and lists

We can also covert lists of characters/words to strings, using a delimiter.

Example.

```
>>> t = ['h', 'e', 'l', 'l', 'o']
>>> delimiter = ''
>>> delimiter.join(t)
'hello'
```