LISTS AND TUPLES IN PYTHON

LISTS AND TUPLES IN PYTHON

What will you learn in this tutorial?

- The important characteristics of lists and tuples
- How to define each type
- Manipulation techniques
- Fundamentals of how and when to use each of them

LISTS AND TUPLES OVERVIEW

Python List Basics

- **A list is a collection of arbitrary objects, much like an array in other programming languages**
- **Lists are defined by enclosing a comma-separated sequence of objects in square brackets**

```
a = ['spam', 'egg', 'bacon', 'tomato']
```

LISTS AND TUPLES OVERVIEW

Important Characteristics of Lists

- **Lists are ordered**
- **Lists can contain any arbitrary objects**
- **List elements can be accessed by index**
- **Lists can be nested to arbitrary depth**
- **Lists are mutable**
- **Lists are dynamic**

LISTS AND TUPLES OVERVIEW

Python Tuple Basics

- **Tuples are identical to lists in all respects, except**
  - **Defined differently**
  - **Tuples are immutable**
  - **Unpacking and Packing**

LISTS AND TUPLES - TABLE OF CONTENTS

LISTS AND TUPLES - TABLE OF CONTENTS

LISTS - ORDERED AND ARBITRARY

A list is an ordered collection of objects

- **The order used when defining a list is maintained**
- **Lists with the same elements in a different order are not the same**

LISTS - ORDERED AND ARBITRARY

A list can contain arbitrary objects

- **The elements of a list can be the same type**
- **Or the elements can be of varying types**
- **Lists can contain complex objects:**
  - **Functions**
  - **Classes**
  - **Modules**

LISTS - ORDERED AND ARBITRARY

A list can contain any number of objects

- **From zero**
- **To as many as your computer's memory will allow**
- **A list with a single object is sometimes referred to as a singleton list**

LISTS - ORDERED AND ARBITRARY

The objects in a list don’t need to be unique

- **An object can appear multiple times within a list**

LISTS AND TUPLES - TABLE OF CONTENTS

LISTS - INDEXING AND SLICING

List elements can be accessed by index

- **Individual elements in a list can be accessed using an index in square brackets**

```
mylist[m]
```

- **List indexing is zero-based**

LISTS - INDEXING AND SLICING

List Indexing

```
a = ['spam', 'egg', 'bacon', 'tomato', 'ham', 'lobster']
```

'spam'	'egg'	'bacon'	'tomato'	'ham'	'lobster'
0	1	2	3	4	5

List Indices

LISTS - INDEXING AND SLICING

Negative Indexing

-6	-5	-4	-3	-2	-1
'spam'	'egg'	'bacon'	'tomato'	'ham'	'lobster'
0	1	2	3	4	5

LISTS - INDEXING AND SLICING

Slicing is indexing syntax that extracts a portion from a list

- **If** `a` **is a list** `a[m:n]` **returns the portion of** `a`
  - Starting with position `m`
  - And up to but not including position `n`

LISTS - INDEXING AND SLICING

List Slicing

```
a[2:5]
```

'spam'	'egg'	'bacon'	'tomato'	'ham'	'lobster'
0	1	2	3	4	5

List Indices

LISTS - INDEXING AND SLICING

Omitting the first and/or last index

- **Omitting the first index** `a[:n]` **starts the slice at the beginning of the list**
- **Omitting the last index** `a[m:]` **extends the slice from the first index** `m` **to the end of the list**
- **Omitting both indexes** `a[:]` **returns a copy of the entire list**
  - Unlike with a string, it's a copy, not a reference to the same object

LISTS - INDEXING AND SLICING

Specifying a Stride in a List Slice

- **Adding an additional `:` and a third index designates a stride (also called a step)**
- **For the slice** `[0:6:2]`

'spam'	'egg'	'bacon'	'tomato'	'ham'	'lobster'
0	1	2	3	4	5

LISTS AND TUPLES - TABLE OF CONTENTS

LISTS - OPERATORS AND BUILT-IN FUNCTIONS

The `in` Operator

- **A membership operator that can be used with lists**
  - Returns `True` if the first operand is contained within the second
  - Returns `False` otherwise
  - Also can be used as `not in`

LISTS - OPERATORS AND BUILT-IN FUNCTIONS

The Concatenation (+) Operator

- **Concatenates the operands**

The Replication (*) Operator

- **Creates multiple concatenated copies**

LISTS - OPERATORS AND BUILT-IN FUNCTIONS

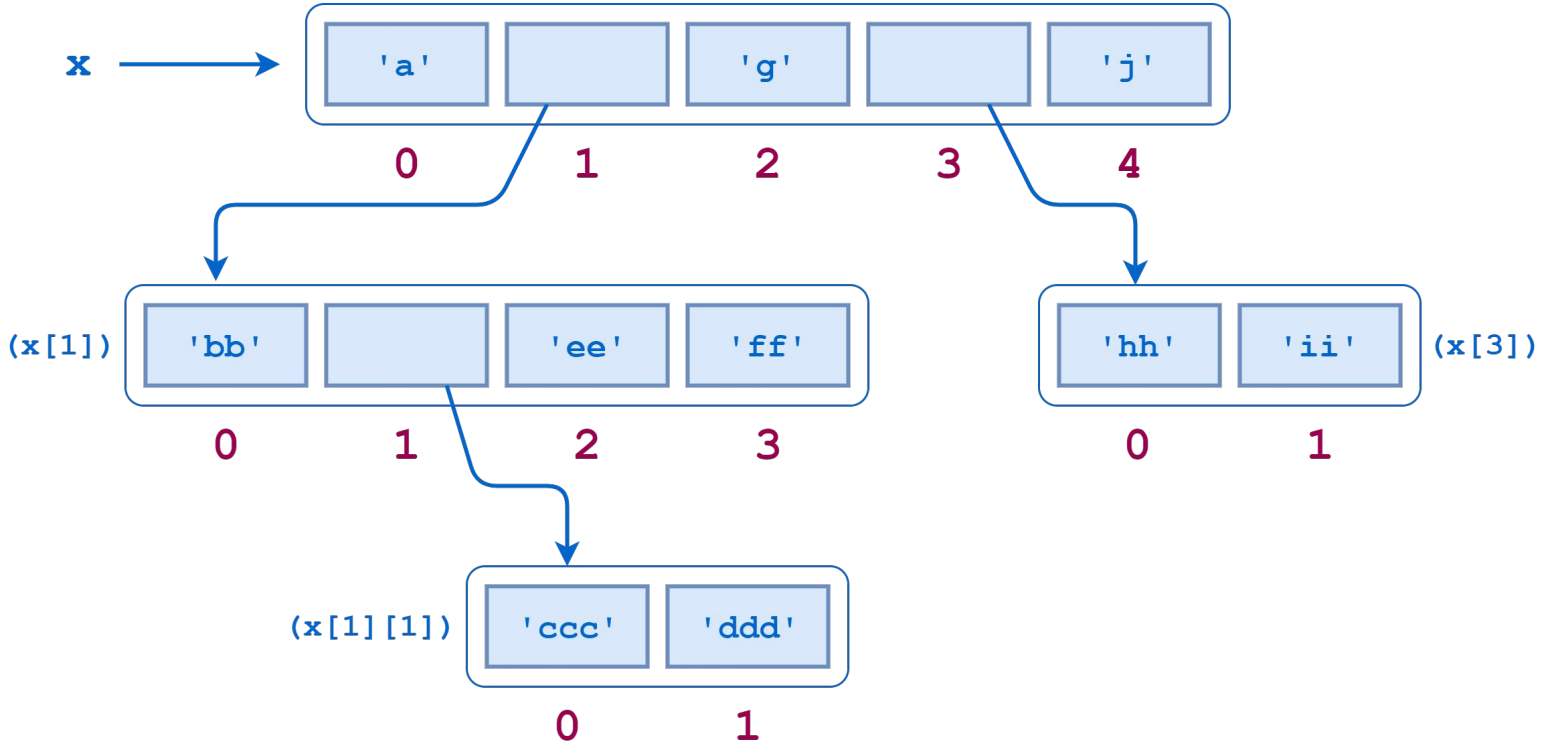
Python Built-in Functions that work with Lists

- `len()`
  - Returns the length of the list
- `min()`
  - Returns the object from the list with minimum value
- `max()`
  - Returns the object from the list with maximum value

LISTS AND TUPLES - TABLE OF CONTENTS

LISTS - NESTING

```
x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'],
'j']
```



LISTS AND TUPLES - TABLE OF CONTENTS

LISTS - MUTABLE AND DYNAMIC

Examples of Python object types that are immutable

- **Integer**
- **Float**
- **Strings**
- **Tuples**

LISTS - MUTABLE AND DYNAMIC

Lists are mutable

- **Once a list has been created, elements can be modified**
- **Individual values can be replaced**
- **The order of elements can be changed**

Lists are dynamic

- **Elements can be added and deleted from a list**

LISTS AND TUPLES - TABLE OF CONTENTS

LIST METHODS

Methods are similar to functions

- **A method is a specialized type of callable procedure that is tightly associated with an object.**
- **Like a function, a method is called to perform a distinct task**
- **But it is invoked on a specific object and has knowledge of its target object during execution**
- `obj.foo(<args>)`

LIST METHODS

String Methods vs List Methods

- **String Methods**
  - **Return a new string object that is modified**
  - **Leaving the original string object unchanged**
- **Most List Methods**
  - **Modify the target list in place**
  - **Do not return a new list**

LIST METHODS

- `mylist.append(<obj>)`
- `mylist.extend(<iterable>)`
- `mylist.insert(<index>, <obj>)`
- `mylist.remove(<obj>)`
- `mylist.clear()`
- `mylist.sort(<key=None>, <reverse=False> )`
- `mylist.reverse()`

LISTS AND TUPLES - TABLE OF CONTENTS

LIST METHODS

Continued - Methods with Return Values

- `mylist.pop(<index=-1>)` - returns the item removed
- `mylist.index(<obj>[, <start>[, <end>]])`
- `mylist.count(<obj>)`
- `mylist.copy()` - returns a shallow copy

LISTS AND TUPLES - TABLE OF CONTENTS

DEFINING AND USING TUPLES

Pronunciation

- **Pick your side**
  - **“Too-ple” like pupil or quadruple**
  - **“Tup-ple” like supple**
- **You will hear both**

DEFINING AND USING TUPLES

Python Tuple Basics

- **Tuples are identical to lists in all respects, except for the following**
  - **Tuples are defined by enclosing the elements in parentheses instead of square brackets**

```
a = ('spam', 'egg', 'bacon', 'tomato')
```

  - **Tuples are immutable**

DEFINING AND USING TUPLES

Why use a tuple instead of a list?

- **Program execution is faster when manipulating a tuple than it is for the equivalent list**
- **You don’t want data modified**
- **A Python dictionary requires keys that are of an immutable type**

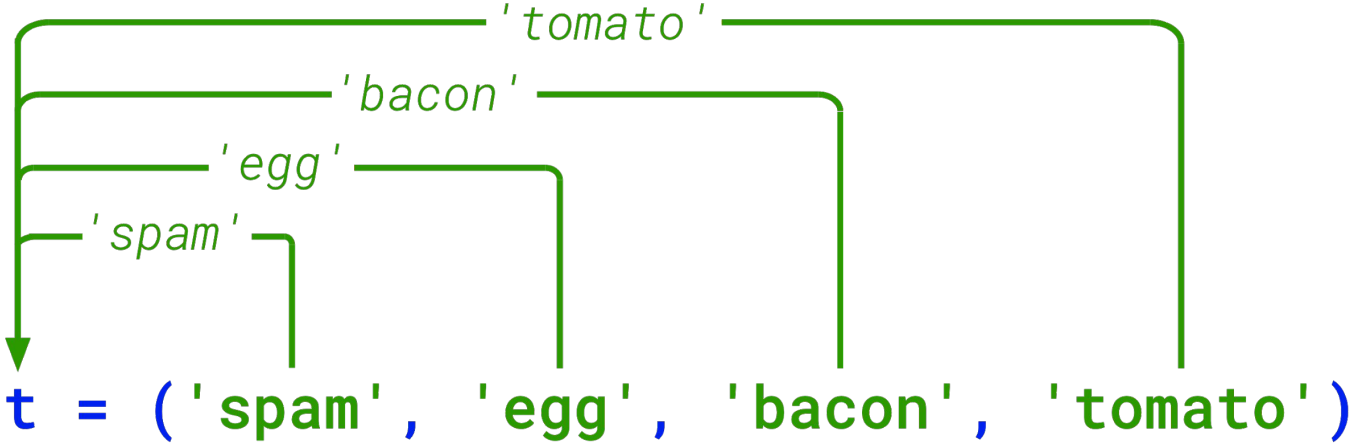
LISTS AND TUPLES - TABLE OF CONTENTS

TUPLE ASSIGNMENT, PACKING AND UNPACKING

Tuple Packing

- **A literal tuple containing several items can be assigned to a single object**

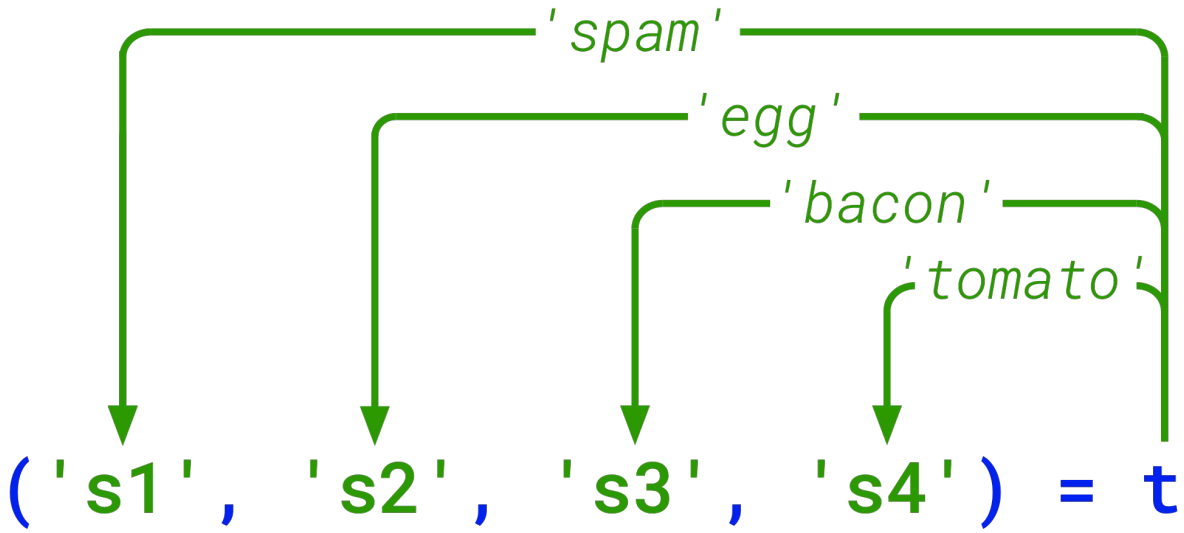
```
t = ('spam', 'egg', 'bacon', 'tomato')
```



TUPLE ASSIGNMENT, PACKING AND UNPACKING

Tuple Unpacking

- **Assigning the “packed” object to a new tuple, “unpacks” the individual items into the objects in the new tuple**



LISTS AND TUPLES - TABLE OF CONTENTS

CONGRATULATIONS
YOU’VE COMPLETED THE COURSE!

LISTS AND TUPLES
IN PYTHON

THANK YOU!

PRACTICE WITH WHAT YOU HAVE LEARNED