New Datatypes:Tuples and Sets

Dan Fleck

Coming up: Passing a list as a parameter

Passing a list as a parameter

```
def class FamilyTree:
    # This class holds a list of family members in the variable tree
    … lots of code not shown …

    def getTree(self):
        return self.tree
```

Can a user of this class modify my family tree?

Yes… since lists are mutable:

```
theTree = family.getTree()
theTree.del('Roger')
```

this deletes Roger everywhere… remember we're passing a reference (an arrow) because tree is a mutable type so changing the data changes it everywhere!!!

Coming up: One way to fix it

One way to fix it

```
def class FamilyTree:
    # This class holds a list of family members
    … lots of code not shown …

    def getTree(self):
        return self.tree[:] # Return a COPY of the entire list
```

This works, but if someone gets this list they may want to modify it. The modification works fine, but then later on the tree “reverts” back to the original. What kind of error is this? (Syntax? Runtime? Logical?)

Coming up: The better way to fix it

The better way to fix it

Use the Python tuple data type:

```
def class FamilyTree:
    # This class holds a list of family members
    … lots of code not shown …

    def getTree(self):
        treeTuple = tuple(self.tree) # Copy the list into a tuple
        return treeTuple
```

A tuple is an immutable list. It cannot be changed, but all the other list operations apply (slicing and indexing).

Tuples

- Tuples are like **immutable lists**. You can slice and index them, but you cannot change them in any way:

  myTuple[1] = ‘Hello’

  TypeError: 'tuple' object does not support item assignment

- Doesn’t support appending or sorting either.

Tuples

```
myTuple = (2, 3, 4, 'A', 'B', 'C')
```

Create a tuple using parenthesis

```
>>> print myTuple[0]
>>> print myTuple[2:5]
```

Index/Slice with [ ] like all other sequences

Officially you can create a tuple by using commas without parenthesis:

```
myTuple = 3, 4, 5  # Don’t do this!
```

This works, but is not recommended. Use parenthesis to make it clear.

Small Tuples

- To create an empty tuple:
  myTuple = ()

- To create a one entry tuple
  myTuple = (1) # Error… that’s a math statement!

  Could we just create an empty tuple and append one thing to it?

No! Tuples are immutable!
Use: myTuple = (1,)

Sequence Types

- Tuples, lists, and strings are all sequence types in Python.
- A sequence is an ordered list of “things”.
- Python sequences all support slicing, indexing and some other things like:
  - x in seq : True if x is equal to any element in seq
  - x not in seq
  - len(seq)
  - A few others see: http://docs.python.org/lib/typesseq.html

Set Types

- Are mutable
- Cannot contain duplicates
- Are unordered – think of a set as a bag of unique things.
  - So can you slice and index sets?

No! They have no order. Asking for the “set element at position 23” doesn’t work because there are no positions!

- To the docs for the rest:
  - http://docs.python.org/lib/types-set.html

Set types

- A set is a different category than a sequence. A set is a group of *unique objects* (that does not allow duplicates). *New in Python version 2.4.*

- Just so you know, sets and sequences are very common in computer languages. The ideas here translate everywhere.

Coming up: Set Example

Set Example

- Sets cannot be created directly, but can be created from lists:

```
myList = [1,2, 3, 4, 5]
mySet = set(myList)   # To create a set from the list
print mySet
print type(mySet)
>>> set([1, 2, 3, 4, 5])
>>> <type 'set’>
```

Set Example

What if the list had duplicates?

```
myList = [1, 2, 3, 2, 4, 1, 5]
mySet = set(myList)
print mySet
print type(mySet)
>>> set([1, 2, 3, 4, 5])
>>> <type 'set’>
```

Same output, the duplicate values are REMOVED. This is because a set allows no duplicates!

Adding to sets

```
myList = [1, 2, 3, 2, 4, 1, 5]
mySet = set(myList)
mySet.add(1) # Try to add a duplicate
print mySet
>>> set([1, 2, 3, 4, 5])
mySet.add('AAA')
print mySet
>>> set([1, 2, 3, 4, 5, 'AAA'])
```

Nope… you cannot add a duplicate into a Set

You can add a new unique value though!

Set difference and union

- s.difference(t)
  - returns a new set with elements in s but not in t
- s.union(t)
  - Returns a new set elements from both s and t
- Lets try an example!

Using Sets

You need sets in situations where you don’t want duplicates.

One such place is the list of misses and used values in the Hangman game. Lets change Graphical Hangman to use sets instead of lists!

Coming up: Summary

Summary

- tuples are immutable lists
- Create them using ( ) instead of [ ], but slice and index using [ ].
- Strings, lists and tuples are all types of sequences
- A set is a data structure that does not allow duplicates.
- Sets are created from lists by using the built-in set( ) function

End of presentation