Python Sequences

Vivek K. S., Deepak G.

Information Systems Decision Sciences (ISDS)
MUMA College of Business
University of South Florida
Tampa, Florida

2017

Objectives

- To understand built-in data sequence types such as Lists, Sets, Tuples and Dictionaries.
- To understand their applications and advantages.
- To understand iterators and constructs that support sequences.
- To learn the different operations that could be performed on these data types.

Lists

- A list is a collection of data elements arranged sequentially.
- They are mutable in nature, and data can inserted removed and appended.
- Lists are enclosed by a pair of squared brackets '[]'
- Lists in Python are **heterogeneous**. A list can contain any different type of data.
- Lists are especially good for keeping track of things that could change in the course of the program execution.
- A list could be created by a pair of squared-brackets [] or list().

Creating a List

A list could be created as follows:

```
cities = ['Chicago','San Francisco','New York']
# A list could also become an
# element of another list
metros = ['Dallas','Las Vegas', cities]
print(cities)
['Chicago', 'San Francisco', 'New York']
print(metros)
['Dallas', 'Las Vegas', ['Chicago',
'San Francisco', 'New York']]
```

- Changing the cities list here modifies the metros list as well.
- This is because, cities is just a reference to the block of memory containing "cities list" and changes to it is being reflected in the metros list.

Data Type Conversions in Lists

Converting a string to a list is easy as follows:

```
language = list('Python')
print(language)
['P', 'y', 't', 'h', 'o', 'n']

# Converting String to List using split()
birthday = '8/11/2017'
dob = birthday.split('/')
print(dob)
['8', '11', '2017']

# List can be converted to a String using join()
statement = ['I','love','Python']
sentence = ' '.join(statement)
print(sentence)
'I love Python'
```

Common List Operations

The following are the common operations in List.

- Adding elements to the list using add, append and insert.
- Indexing and Slicing for data retrieval.
- Modifying the list using indexing and Slicing techniques.
- Concatenating Lists.
- Removing items from the list using "del" remove and pop.
- Determining the membership of an item and counting the number of items.
- Searching and Sorting the list.

More Examples in Lists

More code examples are available at –
https://github.com/vivek14632/Python-Workshop/tree/master/Introducing%20Python/Chapter%203

Tuples

Similar to lists, tuples are sequences of arbitrary items.

- Tuples are the immutable collection sequence types in Python.
- Tuples can't be modified using add, delete, or changing of items once the tuple is created. A tuple is thus a constant list type.
- A tuple can be created using a pair of parenthesis ().
- Notice that it is different from how a list can be created by saying "list()".

Tuples

Often, there is a debate over how the word Tuple is to be pronounced. Guido put an end to this debate in one of his Tweets as follows:
"I pronounce tuple too-pull on Mon/Wed/Fri and tub-pull on Tue/Thu/Sat. On Sunday I don't talk about them. :) @avivby"

Tuple Unpacking

- Multiple variables can be assigned with values at one go using Tuples in Python.
- This underlying implementation has a bigger advantage in Python in a few operations such as swapping two/more variables.
- In other programming languages, particularly statically-typed language, it is more complicated than how it's done in Python.
- Without the use of a temporary variable, unlike in other languages, in Python, two variables can be swapped as follows:

```
a,b = b,a
```

More Examples in Tuples

More code examples are available at –
https://github.com/vivek14632/Python-Workshop/tree/master/Introducing%20Python/Chapter%203

Introducing Sets

- A set is an unordered collection of zero or more immutable Python data objects.
- Sets do not allow duplicates and are written as comma-delimited values enclosed in curly braces.
- The empty set is represented by set().
- Sets are *heterogeneous* similar to lists and tuples.
- Even though sets are not considered to be sequential, they do support a few of the familiar operations presented earlier in lists and tuples.

Applications and Operations Using Sets

- Common applications include membership testing, removing duplicates from a sequence, and computing mathematical operations such as intersection, union, difference, and symmetric difference.
- Like other collections, sets support x in set (membership), len(set), and "for x in set" (iteration).
- Being an unordered collection, sets do not record element position or order of insertion. Accordingly, sets do not support indexing, slicing, or other sequence-like behavior.

Frozen Sets

- There are currently two builtin set types in Python, set and frozenset.
- The set type is mutable – the contents can be changed using methods like add() and remove().
- Since it is mutable, it has no hash value and cannot be used as either a dictionary key or as an element of another set.
- The frozenset type is immutable and hashable – its contents cannot be altered after it is created; however, it can be used as a dictionary key or as an element of another set.

More Examples in Sets

More code examples are available at –
https://github.com/vivek14632/Python-Workshop/tree/master/Introducing%20Python/Chapter%203

Summary

- We understood Sequences in Python.
- We extensively experimented with Python's sequential data types such as Lists, Tuples and sets.
- We learned the difference between the different sequence types and how they help us in different situations.
- We understood and worked with the different methods that the sequence types provide.