

Chapter 4 — Collections of Data

Introduction

As you saw in the last chapter, strings can be thought of as a collection of letters all connected to make something. This chapter we will introduce: lists, tuples, and dictionaries. All of these are ways we can create collections in Python.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Describe sequences of data.
- Use lists and tuples to store groups of values.
- Extract individual values and subsequences of values from a sequence.
- Manipulate lists to add new items, remove items, sort them, and reverse them.
- Apply the concept of key-value pairs to build a dictionary containing values about named entities.
- Understand that sequences may be values in a sequence, like any other value.

Prerequisites

This chapter will continue the use of literal values and variables from Chapter 1. List slice and extraction will re-emphasize the sub-string and character extraction from Chapter 3.

Sequences

Lists

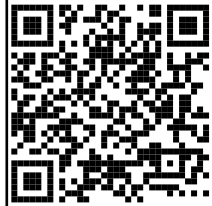
A list is a structure that holds zero or more values. The values may be any type, mixed, and unsorted. You may also assign a list to a variable, just like when you assign a number or a string. As you define a list the items will stay in the original order, until you change them.

```
[expression , expression... ]
```



`[]`
`list()`

A list is a dynamic sequence of values that may be added to, deleted from, sorted, and otherwise manipulated. You may specify zero or more values between the `[]` or create an empty list with the `list()` function.



<https://docs.python.org/3/library/stdtypes.html#list>
<http://bit.ly/2QPiEOq>

The following code creates two lists of strings and a list of numbers:

```
1| dwarfs = ["Doc", "Grumpy", "Happy", "Sleepy", "Bashful",  
2|         "Sneezy", "Dopey" ]  
3| suites = ["Spades", "Hearts", "Diamonds", "Clubs"]  
4| us_coins = [.01, .05, .10, .25, .50, 1.00]  
5| empty = []  
6| another_empty = list()
```

Tuple

A tuple is a static sequence of data, like a list but you can't change it. It is created with `()` instead of `[]`.

`(expression, expression...)`
`expression, expression...`
`expression,`
`(expression,)`
`()`

A tuple is a static sequence of values. Tuples are simply created by placing commas between two or more values. It is common and sometimes required to put parenthesis around tuples to avoid confusion.

To create a tuple with a single value, you must append a trailing comma. This lets the compiler know that the parenthesis are not for mathematical grouping.



<https://docs.python.org/3/library/stdtypes.html#tuples>
<http://bit.ly/2GHRWm3>



```
1| flavors = "vanilla", "chocolate", "strawberry", "pumpkin"  
2| print("We have", flavors, "of ice-cream.")  
3| print(flavors)
```

```
We have ('vanilla', 'chocolate', 'strawberry', 'pumpkin')  
flavors of ice-cream.
```

We can use tuples with the variable assignment operator to assign many values to many variables in a single statement.

```
a,b,c = 1,2,3  
print(a, b, c, a + b + c)
```

```
1 2 3 6
```

Individual Values and Slicing Lists and Tuples

Single Index

The square braces (indexing operator) are used to access (retrieve and set) values in a list. The first element is assigned the index of zero (0) and the last element is `len() - 1`. The last element may also be accessed from the end of the list by using the negative indexes.

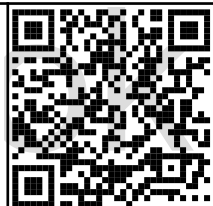
If you think about strings as simple collections of letters and symbols, the indexing operator works like it did with strings, it returns one value from the collection.

`sequence[pos_exp]`

Operator

Form of the subscript operator to extract the value at position `n`, where `n` is 0 to `len()-1`. If `n < 0` then extract an element from the right end of the sequence.

<https://docs.python.org/3/library/stdtypes.html#index-19>
<http://bit.ly/2CyCLaF>



```
1| dwarfs = ["Doc", "Grumpy", "Happy", "Sleepy", "Bashful",
```



```
"Sneezy", "Dopey" ]
2| print(dwarfs)
3| print(dwarfs[1])
4| dwarfs[6] = "Bubba"
5| print(dwarfs)

['Doc', 'Grumpy', 'Happy', 'Sleepy', 'Bashful', 'Sneezy',
 'Dopey']
Grumpy
['Doc', 'Grumpy', 'Happy', 'Sleepy', 'Bashful', 'Sneezy',
 'Bubba']
```

Slice — Two Indexes

A slice (range) may also be specified in the indexing operator. This works the same way that the indexing operator is used with strings to extract a sub string. You may also assign a list to a range from another list. The previous elements are removed and the new list is inserted at the starting location of the range.

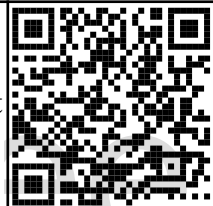
`sequence[start_exp: end_exp]`

Operator

The second form of the indexing operator (`[:]` square brackets), often called a slice, is used to extract groups of items and returns a sequence of the same type as the original sequence. The sub-sequence begins at position `start` and ends at `end-1`. If `start` is omitted begin at value 0, if `end` is omitted then include to end of the sequence.

<https://docs.python.org/3/library/stdtypes.html#index-19>

<http://bit.ly/2CyCLaF>




```
1| A = [1,2,3,4,5,6]
2| print(a)
3| print(a[2:5])
4| a[2:5] = [99]
5| print(a)
```

```
[1, 2, 3, 4, 5, 6]
[3, 4, 5]
[1, 2, 99, 6]
```



Delete Values from a List

To delete an item from a list, we use an unusually formatted statement named `del`. `del` is followed by a list item reference (or slice).

<pre><code>del sequence[pos_exp]</code></pre>	Statement
<pre><code>del sequence[start_exp: end_exp]</code></pre>	
<p>The <code>del</code> statement deletes the item or slice from a sequence. This can also be accomplished by setting the element/slice to an empty sequence, like <code>sequence[start_exp:end_exp] = []</code>.</p> <p>https://docs.python.org/3/library/stdtypes.html#index-22 http://bit.ly/2QTyveQ</p>	

```
1| x = ["a", "b", "c"]
2| print(x)
3| del x[1]
4| print(x)
```

```
['a', 'b', 'c']
['a', 'c']
```

Selected List Functions

List and Tuple Length

As we saw in the last chapter, with strings, the `len()` function will also tell us how many items are in a list.

<code>len(sequence)</code>	Function
----------------------------	----------



Returns the length in number of elements of a string, list, tuple, or dictionary. Same function that we use to return the length of a string.

<https://docs.python.org/3/library/functions.html#len>
<http://bit.ly/2ENddrZ>



```
1| suites = ["Spades", "Hearts", "Diamonds", "Clubs"]
2| nsuites = len(suites)
3| print("There are", nsuites, "suites in a standard deck of
   cards.")
```

There are 4 suites in a standard deck of cards.

Summation

Another handy function to use with lists (or tuples) is the `sum()` function. It simply adds all of the elements in a numeric list, and returns the total. In the example below, you can see where we combine the two functions to create the average value.

`sum(sequence)`

Function

Returns the total of the values on a sequence. Sequence must contain only numeric values (integer, float or mixed) or an error will be thrown.

<https://docs.python.org/3/library/functions.html#sum>
<http://bit.ly/2QPmZkB>



```
1| Grades = [.89, .96, 1.00, .76]
2| n = len(grades)
3| s = sum(grades)
4| print("With", n, "grades in the list you have an average
   of", s/n)
```

With 4 grades in the list you have an average of 0.9025



Selected List Methods

List methods work differently than string methods, they do not return new strings that you need to assign to a variable or use. These methods actually change the list in memory, directly. You will get unusual results if you try to assign these back to the original list.

Append

The first method you need to know is `.append()`. It does exactly what you would expect a method with name to do, it adds a value (or a list of values) to the end of the list. In the example below we can see that you have earned a 99% on a new assignment, and appended it to the list of grades.

<code>list.append(expression)</code>	Method of List
Add a new item or list to the end of an existing list.	
https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types http://bit.ly/2CBXhaz	

```
1| Grades = [.89, .96, 1.00, .76]
2| grades.append(.99)
3| n = len(grades)
4| s = sum(grades)
5| print("With", n, "grades in the list you have an average
   | of",s/n)
```

With 5 grades in the list you have an average of 0.92

Insert

You may want to do more than append an item, especially in a list where order is important. To do this we use the `.insert()` method. You will pass two values to insert: 1) the location to put the new value, and 2) the value to insert.



`list.insert(pos_expr, value_expr)`

Method of List

Insert a new item in the list at the specified index.

<https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>
<http://bit.ly/2CBXhaz>



```
1| people = ["Karen", "Jim", "Julie"]
2| print(people)
3| people.insert(0, "Amy")
4| people.insert(2, "Jane")
5| people.append("Will")
6| print(people)
```

```
['Karen', 'Jim', 'Julie']
['Amy', 'Karen', 'Jane', 'Jim', 'Julie', 'Will']
```

Pop

The pop method is a handy one to get a value from a list and to remove it at the same time. It is also used to easily implement a stack. A stack is a fundamental data structure used by programmers to create a last-in first-out way to access data. Think about a stack of playing cards, You draw off the top of the stack the last card that was placed on the stack.

`list.pop()`

`list.pop(pos_expr)`

Method of List

Returns the value at the end of the list and then removes it. You may also specify a location to extract/remove a value from. For example: `foo.pop(0)` would return and remove the first value in a list.

<https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>
<http://bit.ly/2CBXhaz>




```
1| Stack = []
2| # push three values onto the stack (using append)
3| stack.append(99)
4| stack.append(101)
5| stack.append(56)
6| #
7| v = stack.pop()
8| print("You popped",v,"and stack now contains", len(stack),
      "items.")
```

```
You popped 56 and stack now contains 2 items.
```

Remove

Sometimes you know the value you want to delete from a list, but you don't remember the index. The `remove` method finds the first occurrence of a value and deletes it for you.

`list.remove(expression)`

Method of List

Find the first occurrence of an element, by its value, and remove it from the list.

<https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>

<http://bit.ly/2CBXhaz>



```
1| parts = ["A56", "AX77", "BT91", "BWN18", "ZXT"]
2| parts.remove("BT91")
3| print(parts)
```

```
['A56', 'AX77', 'BWN18', 'ZXT']
```

Reverse

The `reverse` method does what you would expect it to do. It takes a list and changes it to an inverse of itself.



`list.reverse()`

Method of List

Reverses the items in a list.

<https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>
<http://bit.ly/2CBXhaz>



```
1| names = ['jo', 'bob', 'sue']
2| names.reverse()
3| print(names)
```

```
['sue', 'bob', 'jo']
```

Sort

Another commonly used list method is the sort. It will re-arrange a list into sorted order. If the list contains integers and floats the list will be sorted numerically. If the list contains strings then it will be sorted alphabetically.

`list.sort()`

Method of List

Sort the items in a list. A list of numeric items will be compared numerically, and a list of string values will be sorted alphabetically. Without overriding the default functionality of sort, you may not sort lists with a mix of strings and numbers.

<https://docs.python.org/3/library/stdtypes.html#list.sort>
<http://bit.ly/2T9TfM0>



```
1| A = [1, 6, 3, 4, 99]
2| a.sort()
3| print(a)
4| b = ['zip', 'zap', 'baz', 'buz']
5| b.sort()
6| print(b)
```



```
[1, 3, 4, 6, 99]
['baz', 'buz', 'zap', 'zip']
```

The Dictionary

A dictionary is like a list except that items are indexed by a name and not a number. In some languages it is called a hash, a map, or an associative array.

```
1| grades = {'jim': 78, 'sue':98, 'bob': 99, 'darla': 87}
2| print(grades["jim"])
```

```
78
```

Unlike lists and tuples, we call the map index a “key”. This is why dictionaries are also sometimes referred to as “key-value” collection. Keys can be any “immutable” type, including: numbers, strings, and tuples. In these examples we will use strings for the keys.

In addition to creating a map directly with values, You may create an empty map by using a pair of curly braces.

```
1| # make a list of what we need with quantity and measure
2| grocery_list = {}
3| grocery_list["milk"] = (1, "gal")
4| grocery_list["eggs"] = (1, "doz")
5| grocery_list["pop"] = (2, "12pk")
6| print (grocery_list)
7| what = "milk"
8| print ("we need", grocery_list[what], "of", what, ".")

{'milk': (1, 'gal'), 'eggs': (1, 'doz'), 'pop': (2, '12pk')}
we need (1, 'gal') of milk .
```

To set or extract a single value from a dictionary, use square brackets with the key value inside. If you try to get a value where the key does not exist, you will receive an error. If you assign a key that exists with a new value the old value will be overwritten. If you assign a new key a value, it will be created.

```
1| grades = {'jim': 78, 'sue':98, 'bob': 99, 'darla': 87}
2| grades["jim"] = 79
3| grades["sam"] = 68
```



```
4| print(grades)

{'jim': 79, 'sue': 98, 'bob': 99, 'darla': 87, 'sam': 68}
```

Selected Methods of Dictionaries

This introduction will discuss a few of the methods available for a dictionary.

Get

We saw above that, you can get a value from a dictionary by using the indexing operator (`[]`) and the key value. If we try to get a key that does not exist in a dictionary, we will get an error when we run our program.

```
1| grades = {'jim':63, 'darla': 78, 'bob': 93, 'sue':99}
2| print(grades['marc'])
```

```
Traceback (most recent call last):
  File "/dictionary_badkey.py", line 2, in <module>
    print(grades['marc'])
KeyError: 'marc'
```

This type of error may be easily avoided using the get method.

<code>dictionary.get(key)</code>	Method of Dictionary
<code>dictionary.get(key, default_value)</code>	
Get...	
https://docs.python.org/3.8/library/stdtypes.html#dict.get	

Keys

The `keys()` method returns an object called a view than can be converted to a list of only the keys from a map. You can then use these keys to access the individual values.



<code>dictionary.keys()</code>	Method of Dictionary
Returns a view containing the keys in a dictionary. A list can easily be created from this view. https://docs.python.org/3.8/library/stdtypes.html#dict.keys	

```
1| grades = {'jim':63, 'darla': 78, 'bob': 93, 'sue':99}
2| print(list(grades))
3| print(list(grades.keys()))
```

```
['darla', 'sue', 'jim', 'bob']
['darla', 'sue', 'jim', 'bob']
```

Values

The values method works like the keys method, except that it returns a view of the values in a dictionary.

<code>dictionary.values()</code>	Method of Dictionary
Returns a view containing the values in a dictionary. A list can easily be created from this view. https://docs.python.org/3.8/library/stdtypes.html#dict.values	

```
1| grades = {'jim':63, 'darla': 78, 'bob': 93, 'sue':99}
2| scores = list(grades.values())
3| print(scores)
4| print("the class average is", sum(scores)/len(scores))
```

```
[63, 78, 99, 93]
the class average is 83.25
```



Summary

Goes here

Important Terms

- append
- braces
- del
- dictionary
- index
- insert
- key value
- len
- list
- parenthesis
- pop
- remove
- reverse
- sequence
- slice
- sort
- sum
- tuple

Exercises

1. Create a program with a list of the following numbers: 1,6,8,4,3,9,4,2. Sort the list and print out the first element, the last element and the middle element. You should use the len() function for the last two values so that your program would work for a list of any length.
2. The concatenation (+) operator and the repeat operator (*) are defined over lists and tuples. These operators were not covered in the chapter, but they work like they did in the string chapter. Create two lists a containing the integers from 1 to 10 and the list b containing the integers 1,2, and 3. Add these two lists and store the result in c. Repeat the list c 4 times and store in variable d. Print c, d, and their lengths.
3. The Fibonacci Sequence is a progression of integers where the next value is the sum of the previous two values. Create a list with the two numbers 1 and 0. Write an expression that adds the first two numbers in a list and then inserts the result at the beginning of the list. Print the new list and you should have [1, 1, 0]. Copy and paste the expression and insert ten times and print the resulting list.
4. Let is create a simple game of "Battling Ships". Ask the human for a roe number from 0-7 and column number from 0-7 and print out the number in the "game board" list. A '1' represents a hit and a '0' represents a miss. Store the "game board" in a variable as a list that contains the following eight lists:

```
0, 0, 1, 1, 1, 0, 0, 0
1, 0, 0, 0, 0, 0, 1, 0
1, 0, 0, 0, 0, 0, 1, 0
1, 0, 0, 1, 0, 0, 0, 0
1, 0, 0, 1, 0, 0, 0, 0
```



```
1, 0, 0, 1, 0, 0, 0, 0  
0, 0, 1, 1, 1, 1, 0, 0  
0, 0, 0, 0, 0, 0, 0, 0
```

5. Create a dictionary with the item id codes as the key and the quantity on hand as the data. Ask the user for an item id and quantity to remove from inventory. Make the requested changes to the dictionary and print your results.

Item ID	Quantity on Hand
jhg786	8
tdr999 22	
abc123	44
br549	82

Word Search

Please support this work at <http://syw21.org>

```
b p s m l f s . j h e t t d  
l a o t n z j b r e g . m i  
. r v b r v a s n p a g m c  
v e i e k t u p l e p x s t  
s n n r e m o v e w p o p i  
e t l c y w f k i o e b q o  
q h e . d r . n i n r . n  
u e n . v . e w s j d a p a  
e s o l a y v b e m v c . r  
n i d e l x e n r n . e j y  
c s q q u y r k t n z s x u  
e t d a e s s l . l i s t i  
g s l i c e e c s u m k k x  
o t i n d e x h v s o r t t
```

append, braces, del, dictionary, index, insert, key value, len, list, parenthesis, pop, remove, reverse, sequence, slice, sort, sum, tuple

References

