

A (debatably) Brief Introduction to Python



CHAPTER 3: LISTS AND DICTIONARIES

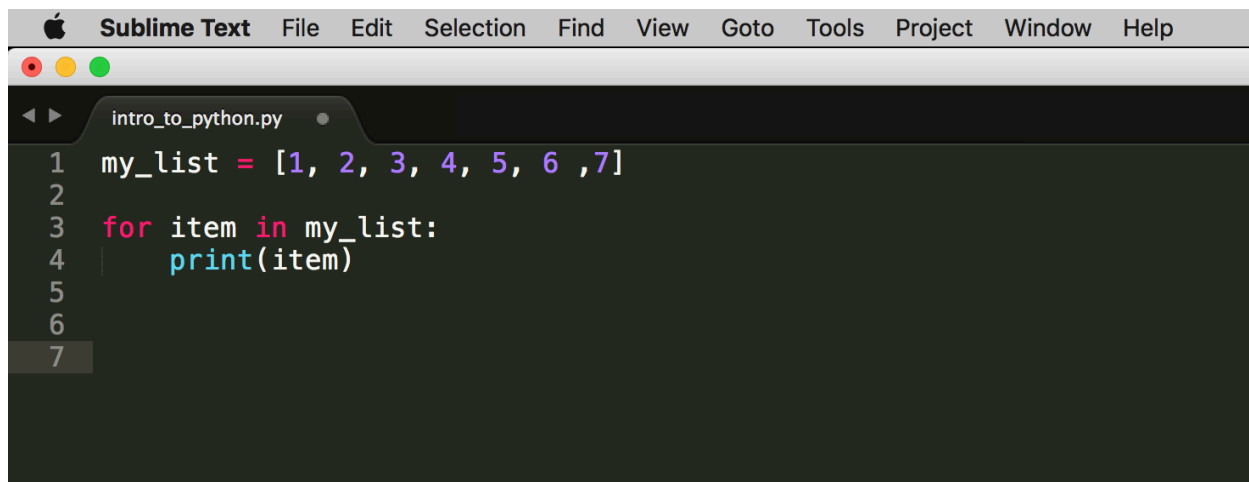
At long last, we're covering lists and dictionaries.

In effect, lists are basically collections of data. The lists part of this chapter should be really easy, because guess what? All of the indexing principles and methods that apply to strings **also** apply to lists? Why so? This is the main reason why strings aren't primitives in most other languages – strings are basically lists of characters! If you need a refresher, everything about indexing is located in chapter 2. Other than that, there are a few other, nice list functions that you should know:

- `len(<list>)` returns the length of a list.
- `<list>.append(value)` adds `value` to the end of a list.
- `<list>.index(value)` returns the index of the **first occurrence** of `value` in `<list>`.

Generally, we like to keep the same types of data within a list, however you can, for instance, store ints, strings, and booleans in the same list. Why don't we? It's not good programming style. Can we store multiple lists inside a list? Why, yes we can! In fact, we'll be dealing with 2-dimensional lists a lot in the near future.

Another main functionality of lists is the ability for programmers to **loop over them**. What does that mean? Let me show you an example so that you understand. Let's say I type in this code:

A screenshot of a Sublime Text editor window. The title bar shows the Apple logo, the text 'Sublime Text', and a menu bar with 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', 'Window', and 'Help'. Below the title bar is a tab labeled 'intro_to_python.py'. The editor area has a dark background with syntax-highlighted Python code:

```
1 my_list = [1, 2, 3, 4, 5, 6 ,7]
2
3 for item in my_list:
4     print(item)
5
6
7
```

Here's what the command line outputs:

```
Desktop — -bash — 80x24
Vedants-MacBook-Pro:Desktop Dant$ python intro_to_python.py
1
2
3
4
5
6
7
Vedants-MacBook-Pro:Desktop Dant$
```

Essentially, what we have is a modified **for loop**. We saw one of these in chapter 2, but we'll be going over these in greater detail. The loop above is **exactly** how you loop/iterate over a list. Here's a good question: how do you iterate over more than one list? Here:

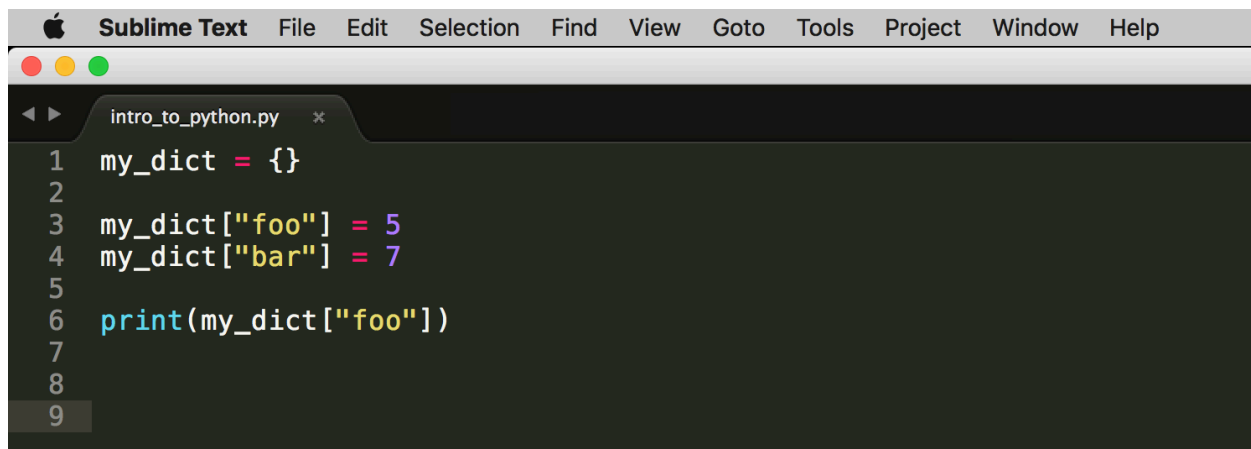
```
Sublime Text  File  Edit  Selection  Find  View  Goto  Tools  Project  Window  Help
intro_to_python.py x
1 my_list = [1, 2, 3, 4, 5, 6 ,7]
2 my_list_2 = [2, 4, 6, 8, 10, 12, 14]
3
4 for item, item2 in zip(my_list, my_list_2):
5     print(str(item) + " " + str(item2))
6
7
8 |
```

Also, there is an alternate way to loop over a list, and that is using a mixture of a for loop and indexing:

```
Sublime Text  File  Edit  Selection  Find  View  Goto  Tools  Project  Window  Help
intro_to_python.py x
1 my_list = [1, 2, 3, 4, 5, 6 ,7]
2
3 for i in range(len(my_list)):
4     print(my_list[i])
5
6 |
```

Look closely at these three loops, because you'll be using them often. **Very** often.

Now, we move on to dictionaries. These are very similar to lists, however they are indexed very differently. Instead of your standard index -> value system, you have a key -> value system. Here's how you would go about putting stuff in a dictionary:

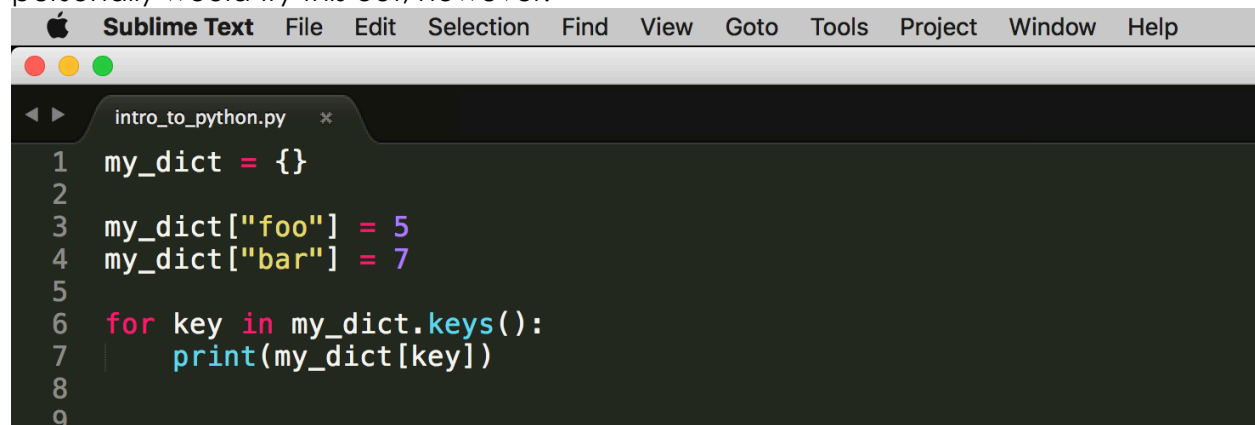


```
Sublime Text  File  Edit  Selection  Find  View  Goto  Tools  Project  Window  Help

intro_to_python.py x
1  my_dict = {}
2
3  my_dict["foo"] = 5
4  my_dict["bar"] = 7
5
6  print(my_dict["foo"])
7
8
9
```

Notice how you use curly brackets instead of square brackets. In order to add new elements to a dictionary, you don't have to use `append()`. Just declare them like in the above example.

Iterating over dictionaries is a little difficult. In fact, I don't expect any of you to do so. I personally would try this out, however:



```
Sublime Text  File  Edit  Selection  Find  View  Goto  Tools  Project  Window  Help

intro_to_python.py x
1  my_dict = {}
2
3  my_dict["foo"] = 5
4  my_dict["bar"] = 7
5
6  for key in my_dict.keys():
7      print(my_dict[key])
8
9
```

There are built-in functions for dictionaries, one of which is used in the example above. We're not going to cover them explicitly, however if you want to check some out, search the googles.

That just about wraps it up for lists and dictionaries, as well as for loops. We'll be covering conditionals in the next chapter, then finally programming style and program design in chapter 5.