**Python Data Types**

**Overview**

|  |
| --- |
| Data types are the classification or categorization of data items. Data types represent a kind of value which determines wh99at operations can be performed on that data. Numeric, non-numeric, sequence type and boolean(true or false) data are the most used data types. However, each programming language has its own classification largely reflecting its programming philosophy. |

Python has the following standard or built-in data types:

**Numeric**

A numeric value is any representation of data which has a numeric value. Python identifies three types of numbers:

* **Integer**: Positive or negative whole numbers (without a fractional part). Eg 15
* **Float**: Any real number with a fractional value. Eg 3.1416
* **Complex number**: A number with a real and imaginary component represented as *x+y*j. x and y are floats and j is -1(square root of -1 called an imaginary number)

**Boolean**

Data with one of two built-in values *True* or *False*. Notice that ‘T’ and ‘F’ are capital. *true* and *false* are also not valid booleans and Python will throw a *NameError* if you try to use them.

**Sequence Type**

A sequence is an ordered collection(every item has a position) of similar or different data types. Python has the following built-in sequence of data types:

* **String**: A string value is a collection of zero or more characters put in a single, double or triple quotes.
* **List**: A list is an ordered collection of one or more data items, not necessarily of the same type, put in *square brackets*.
* **Tuple**: A tuple in an ordered collection of one or more data items, not necessarily of the same type, put in *parentheses*.

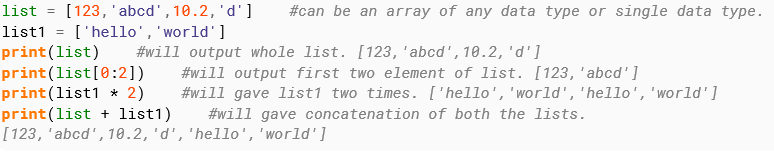
**Python Lists**

A list contains items separated by commas and enclosed within square brackets []. Creating a list is as simple as putting different comma-separated values between square brackets.

Elements in a list have the following characteristics:

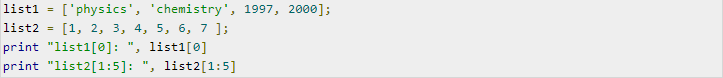
* They maintain their ordering unless explicitly re-ordered (eg, by sorting the list)
* They can be of any type, and types can be mixed.
* They are accessed via numeric (zero based) indices.

An example of a list is shown below:



**Accessing Values in Lists**

To access values in a list, use the square brackets for slicing along with the index or indices to obtain a value available at that index. For example:



When the above code is executed, it produces the following result:



**Updating Lists**

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

**Dictionary**

A dictionary is an unordered collection of data in a key:value pair form. A collection of such pairs is enclosed in *curly brackets.* For example:

|  |
| --- |
| {5: 'Claremont', 2: 'Newlands', 8: 'Rondebosch':, 1: 'Mowbray'} |

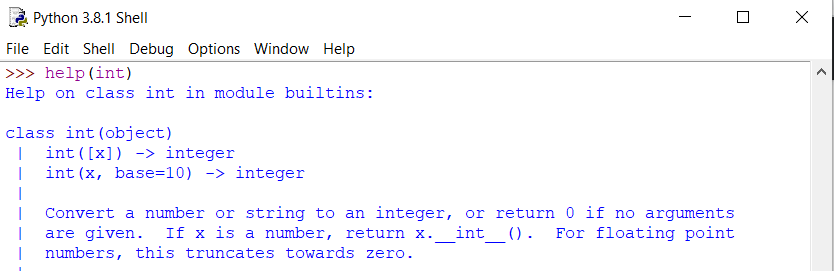
Elements in a dictionary have the following characteristics:

* Every entry has a key and a value.
* Ordering is not guaranteed
* Elements are accessed using key values.
* Values can be of any type (including other dict’s), and types can be mixed.

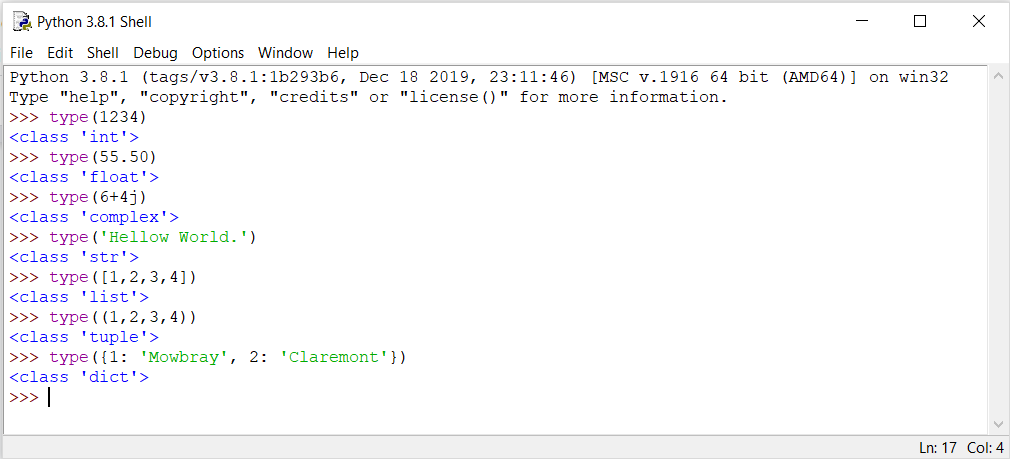
**type() function**

Python has an in-built function type() to check the data type of a certain value. For example enter type(1234) in Python shell and it will return <class 'int''>, which means 1234 is an integer value.

You can check the full details of a data type by calling the help function like so help(int)



Try and verify the data type of different values in Python shell, as shown below:



**Mutable and Immutable Objects**

Data objects of the above types are stored in a computer’s memory for processing. Some of these values can be modified during processing, but the contents of others can’t be altered once they are created in the memory.

Number values, strings, and tuple are immutable, which means their contents can’t be altered after creation.

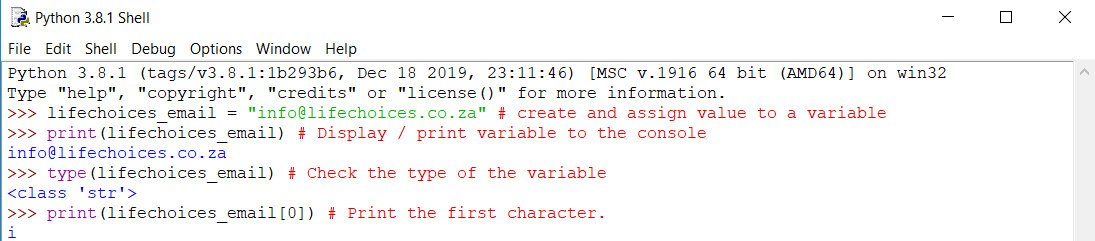
On the other hand, collection of items in a List or Dictionary can be modified. It is possible to add, delete, insert, and rearrange items in a list or dictionary. Hence they are mutable objects.

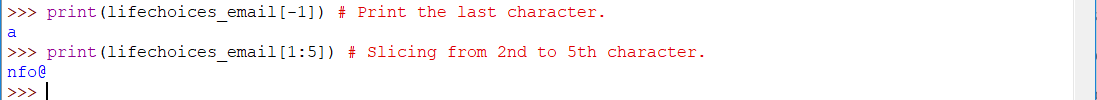
**How to access characters in a string**

We can access individual characters using it’s index and a range of characters using slicing. Index starts from 0. Trying to access a character out of index range will raise an *IndexError.* The index must be an integer. We can’t use float or other types, this will result in a *TypeError.*

Python allows negative indexing for its sequences.

The index of -1 refers to the last item, -2 to the second last item and so on. We can access a range of items in a string by using the slicing operator (:).

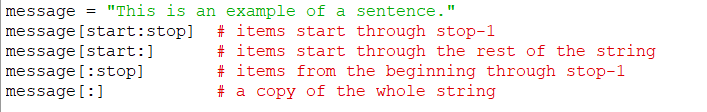




For more information regarding python strings especially combining strings, formatting strings and common python string methods, please have a look at the official python docs. <https://docs.python.org/3.8/library/string.html>

**Cutting and slicing strings.**

Slicing is used to retrieve the subset of values in an iterable. The basic slicing technique is to define a starting point, the stopping point, and the step size. The basic syntax used for slicing is shown below:



Below are some basic examples of python string slicing:



**Python Data Types Tasks**

**Task 1 (No coding) Basic concepts.**

1. Write a list of the differences between a list and a dictionary in Python.
2. How will you capitalize the first letter of a string?
3. What basic data types can you name in Python?
4. What is the output of print(str) if str = ‘“Hello World!”?
5. What is the output of print str[2:5] if str = “Hello World!”?
6. What is the output of print str[2:] if str = “Hello World!”?
7. How will you get all the keys from the dictionary?
8. How will you get the length of the string?

**Task 2 (Coding exercise)**

1. Write a program to get the largest number from a list.

Expected input: **[5,1,3,2,4]**

Expected output: **5**

1. Given a list of numbers with duplicates, how would you remove the duplicates so that the list become unique.

Expected input: **[10,2,45,3,5,7,2,10,45,8,10]**

Expected output**: [2,3,5,7,8,10,45]**