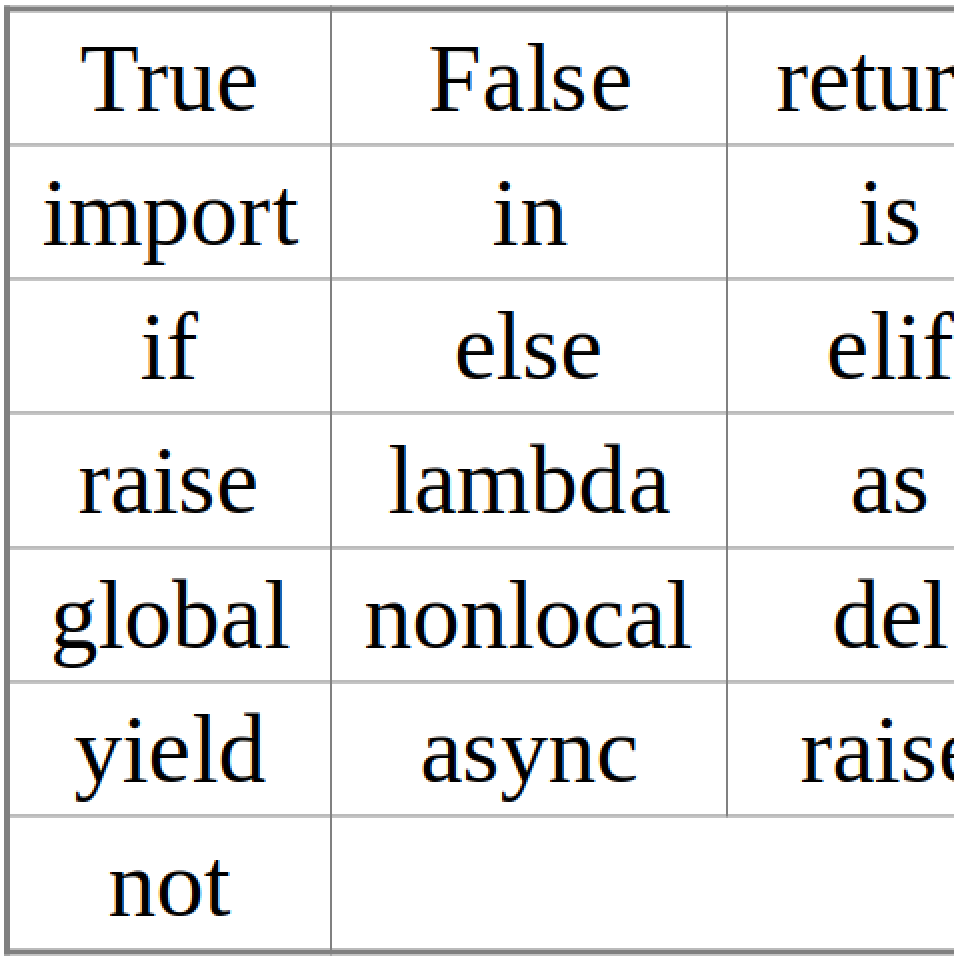
3.2 Variables and Data Types

In this lesson, we have looked at how Python lets us create variables, what are the naming conventions that we have to follow, and also the different data types in Python.

Python Variables   Naming Conventions and Styles

As we learned Python Variables should not start with digits, contain any special characters or be one of the keywords.

Here’s a list of reserved words, or what we call as keywords in Python 3.6. These have special meanings and purposes and we can’t use them for anything other than those purposes.



True	False	retur
import	in	is
if	else	elif
raise	lambda	as
global	nonlocal	del
yield	async	rais
not		

These keywords can change in different versions of Python. Some more may be added or some may be removed. You can always get the list of keywords in your current version by typing the following at the prompt.

```
>>> import keyword
>>> print(keyword.kwlist)
```

If you assign values to these keywords or any variable that doesn't follow the conventions that we mentioned, you will get syntax errors in your script.

To improve the readability of the names you can separate them with underscores as necessary, use Camel Case or Mixed case.

```
Eg:     first_name     FirstName    firstName
```

Data Types

Python variables hold values of different types. Following are a few of the built-in data types of Python.

- Numeric Types (int, float, complex)
- Text Type (str)
- Sequence Types (list, tuple, range)
- Mapping Type (dict)

The type of the variable is set based on the value that you assign to it. Remember that unlike in many other programming languages, Python changes the datatype of one variable to another if its value is set to another. The type of these variables can easily be obtained by using the type() function.

Eg:

var = 100 # This creates an integer variable with the assigned value

var = "Hello" #Type is changed to string in here

In case you need to specifically set a data type to a variable, you can use the constructor functions. Also, you can convert or cast the data type of one variable to another using them.

- int() - Constructs an integer from an integer literal, a float (by removing all decimals), or a string (if the string represents a whole number)
- float() - Constructs a float number from an integer, a float, or a string (if the string represents a float or an integer)

- str() - Constructs a string from a range of data types, including strings, integer, and float literals

Eg:

```
a.)
weight = 55.5
print(int(weight))
```

Here the weight which is assigned a float value will be turned to an integer. The output will be just 55.

b.)

```
weight = "55"
print(int (weight) +5)
```

In here the weight is converted to int and will display 60 as the output.

Mutable and Immutable Objects

You learnt that everything in Python is an Object, and each variable we use holds an object instance. A unique object id is assigned to an object when it is initiated. But this unique identity can change depending on the mutability of the objects. Mutable objects can be changed after being created, while Immutable ones cannot.

Objects of built-in types like list, set, dict are mutable.

Objects of built-in types like int, float, bool, str, tuple are immutable.

It is recommended to use mutable objects when there is a need to change the size or content of the object later in the program.

However, immutable objects are not always immutable. We learnt that a tuple in python is immutable. But the tuple consists of a sequence of names with unchangeable bindings to objects.

As an example consider a tuple consisting of a string and a list.

```
 t = ([10, 14, 15], 'John')
```

Strings are immutable, so we can’t change their value. But the contents of the list can be changed. Here the tuple itself isn’t mutable but contains items that are mutable.

Referece: Python Documentation