1.3 VARIABLES AND DATA TYPES

A variable is like a container that stores values you can access or change. The purpose of using variables is to allow the stored values to be used later on. We have learnt that any object or variable in Python is a name that refers to a value at a particular memory location and possesses three components:

- **A Value:** It represents any number or a letter or a string. To assign any value to a variable, we use assignment operator (=).
- **An Identity:** It refers to the address of the variable in memory which does not change once created. To retrieve the address (identity) of a variable, the command used is:

```
>>>id(variable_name)
```

- **A Type:** We are not required to explicitly declare a variable with its type. Whenever we declare a variable with some value, Python automatically allocates the relevant data type associated with it.

  Hence, the data type of a variable is according to the value it holds.

  For example, `>>> x = 20`

  The above statement signifies 'x' to be of integer type since it has been assigned an integer value 20.

  Data types are classified as follows (Fig.1.2):

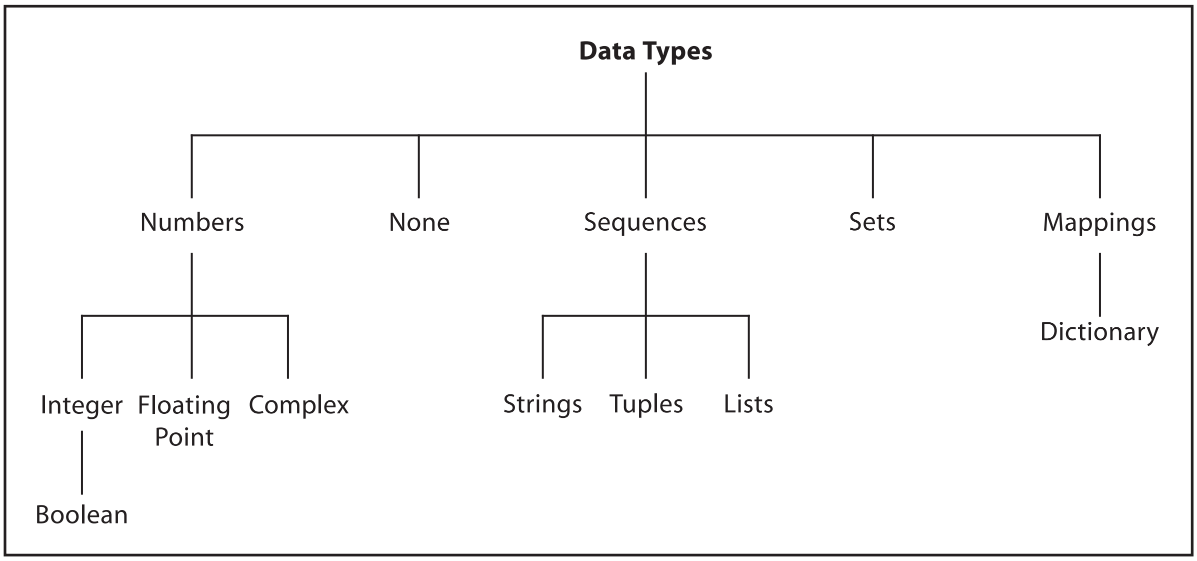
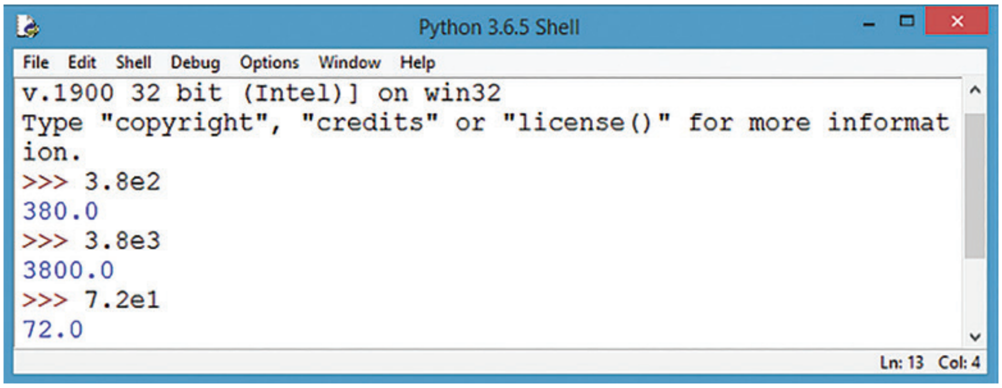


Fig. 1.2: Classification of Data Types in Python

1. **Number or Numeric Data Type:** Numeric data type is used to store numeric values. It is further classified into three subtypes:

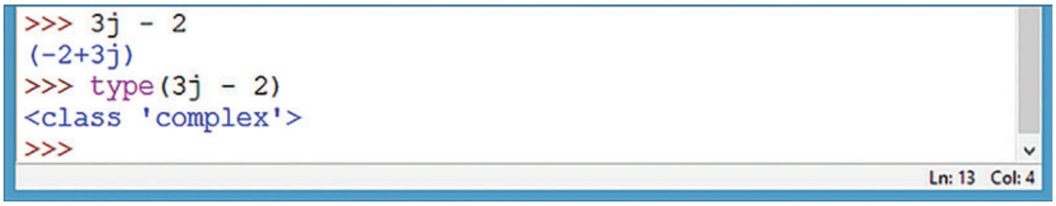
   (a) **Integer and Long:** To store whole numbers, *i.e.*, decimal digits without a fraction part. They can be positive or negative. **Examples:** 566, –783, –3, 44, etc.

(b) **Float/Floating Point:** Floating point numbers signify real numbers. This data type is used to store numbers with a fraction part. They can be represented in scientific notation where the uppercase or lowercase letter 'e' signifies the 10th power:



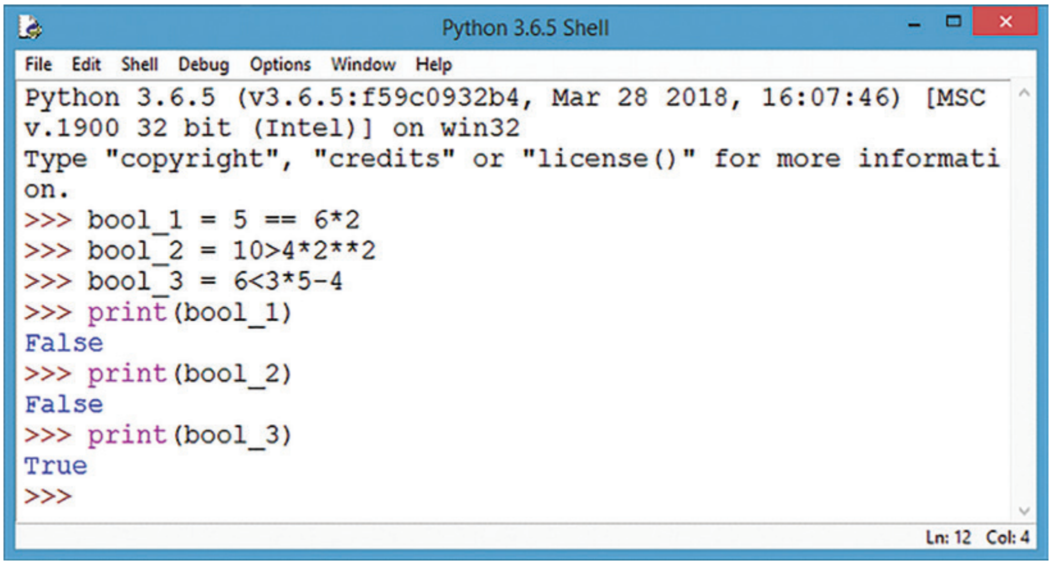
```
v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more informat
ion.
>>> 3.8e2
380.0
>>> 3.8e3
3800.0
>>> 7.2e1
72.0
```

(c) **Complex Numbers:** Complex numbers are pairs of real and imaginary numbers. They take the form 'a + bj', where 'a' is the float and 'b' is the real part of the complex number.



```
>>> 3j - 2
(-2+3j)
>>> type(3j - 2)
<class 'complex'>
>>>
```

(d) **Boolean:** Boolean data type is used in situations where comparisons to be made always result in either a true or a false value.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC
v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more informati
on.
>>> bool_1 = 5 == 6*2
>>> bool_2 = 10>4*2**2
>>> bool_3 = 6<3*5-4
>>> print(bool_1)
False
>>> print(bool_2)
False
>>> print(bool_3)
True
>>>
```

2. **None:** This is a special data type with an unidentified value. It signifies the absence of value in a situation, represented by None. Python doesn't display anything when we give a command to display the value of a variable containing value as None.

3. **Sequence:** A sequence is an ordered collection of items, indexed by integers (both positive as well as negative). The three types of sequence data types available in Python are Strings, Lists and Tuples, which we will discuss in successive topics.

4. **Sets:** Set is an unordered collection of values of any type with no duplicate entry. It is immutable.

5. **Mappings:** This data type is unordered and mutable. Dictionaries in Python fall under Mappings. A dictionary represents data in key-value pairs and accessed using keys, which are immutable. Dictionary is enclosed in curly brackets ({ }).