# 1 Review of Python Basics

## 1.1 INTRODUCTION

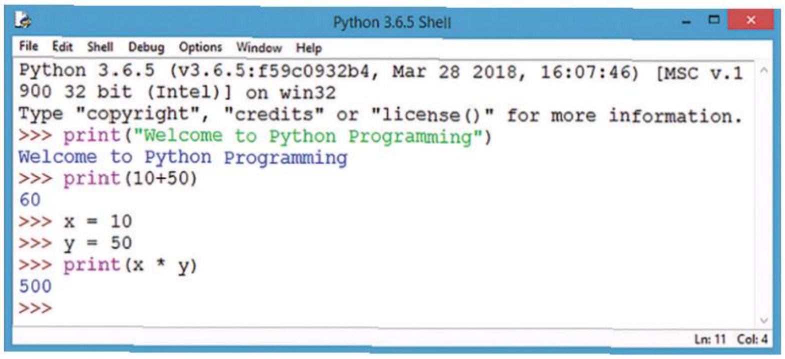
You are aware of the fact that Python is a powerful, modern, high-level programming language. You have learnt the basics of Python programming in Class XI. In this chapter, we will briefly recapitulate the Python concepts that you have already learnt in the previous class.

Python is an interpreted language as its programs are executed by an interpreter. Thus, Python interpreter should be installed on the computer system to write and run Python programs. We have also learnt that Python IDLE (Integrated Development and Learning Environment) provides two working modes—interactive mode (popularly known as Python shell) and script mode.

**CTM:** The Python IDLE tool offers an interactive and a more efficient platform to write your code in Python.

Using **Interactive (Python Shell window) mode**, the Python commands are directly typed at >>> (command prompt), and as soon as we press the Enter key, the interpreter displays the result(s) immediately, which is known as **Displaying**.

*For example,*



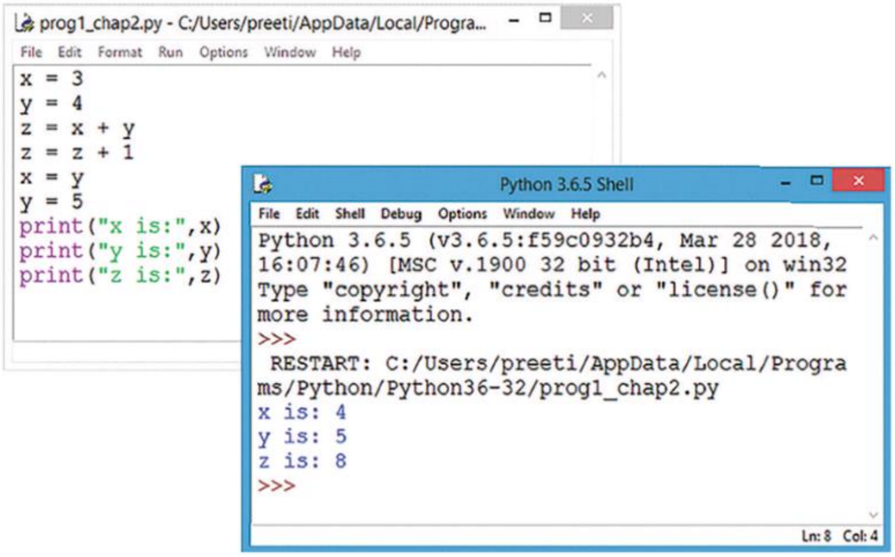
```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1
900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Welcome to Python Programming")
Welcome to Python Programming
>>> print(10+50)
60
>>> x = 10
>>> y = 50
>>> print(x * y)
500
>>>
```

**Fig. 1.1:** Working with Python Shell

## Script Mode

The drawback of interactive mode is that we cannot save the commands that we type. This can be overcome using the script mode. In order to switch over to script mode, click on the **File** menu option -> **New** option or press the shortcut key **Ctrl + N** from the shell window.

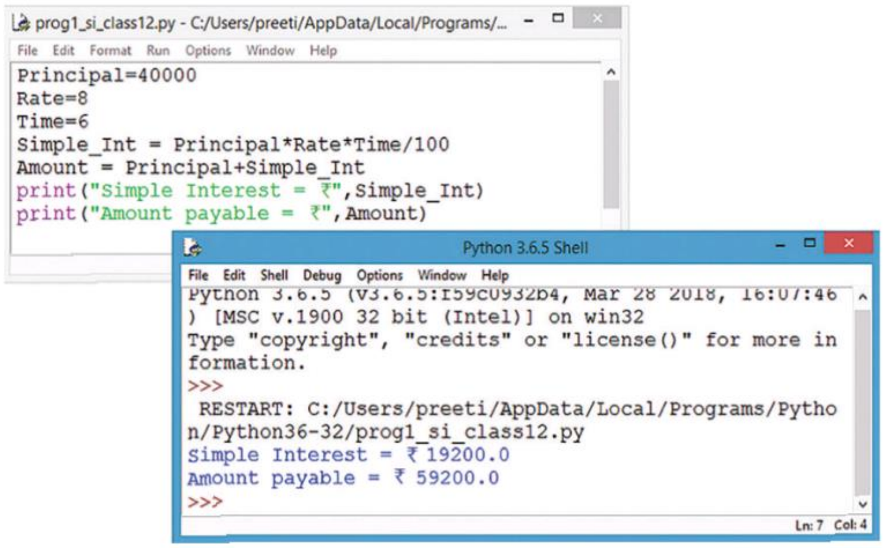
**Example 1:**


prog1_chap2.py - C:/Users/preeti/AppData/Local/Progra...
File Edit Format Run Options Window Help
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Ln: 8 Col: 4

```
x = 3
y = 4
z = x + y
z = z + 1
x = y
y = 5
print("x is:",x)
print("y is:",y)
print("z is:",z)
```

```
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 information.
>>>
 RESTART: C:/Users/preeti/AppData/Local/Progra
ms/Python/Python36-32/prog1_chap2.py
x is: 4
y is: 5
z is: 8
>>>
```

**Example 2:** Lokesh has taken a loan of ₹ 40000 from Vinod at the rate of 8% per annum. After 6 years, he wants to repay the loan in full including interest. Write the Python code (in script mode) to calculate and display the interest and total amount to be paid by Lokesh to settle his accounts.


prog1_si_class12.py - C:/Users/preeti/AppData/Local/Programs/...
File Edit Format Run Options Window Help
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Ln: 7 Col: 4

```
Principal=40000
Rate=8
Time=6
Simple_Int = Principal*Rate*Time/100
Amount = Principal+Simple_Int
print("Simple Interest = ₹",Simple_Int)
print("Amount payable = ₹",Amount)
```

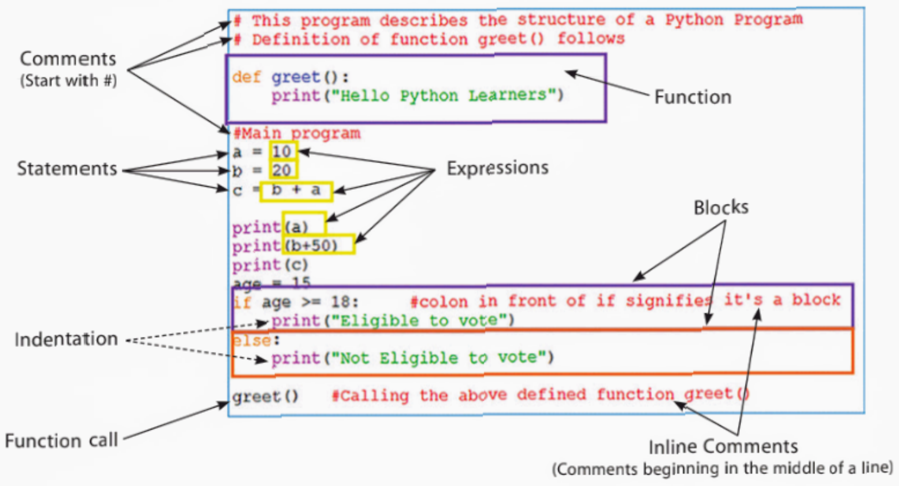
```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46
) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more in
formation.
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Pytho
n/Python36-32/prog1_si_class12.py
Simple Interest = ₹ 19200.0
Amount payable = ₹ 59200.0
>>>
```

## 1.2 STRUCTURE OF A PYTHON PROGRAM

A Python program constitutes several elements such as statements, expressions, functions, comments, etc., which are synchronized in the manner as shown below:

$$\begin{tikzcd}
\text{programs} \arrow[r, "\text{consist of}"] & \text{modules} \arrow[r, "\text{contain}"] & \text{statements} \arrow[d, "\text{contain}"] \\
& & \text{expressions} \arrow[d, "\text{create and process}"] \\
& & \text{objects}
\end{tikzcd}$$

Let us see the several components of a basic Python program.



```
# This program describes the structure of a Python Program
# Definition of function greet() follows

def greet():
    print("Hello Python Learners")

#Main program
a = 10
b = 20
c = b + a

print(a)
print(b+50)
print(c)
age = 15
if age >= 18:     #colon in front of if signifies it's a block
    print("Eligible to vote")
else:
    print("Not Eligible to vote")

greet()   #Calling the above defined function greet()
```

As shown in the snippet given above, the several components that a Python program holds are:

- **Expressions:** An expression **represents** something, like a number, a string, or an element. Any value is an expression.
- **Statements:** Anything that **does something** is a statement. Any assignment to a variable or function call is a statement. Any value contained in that statement is an expression.
- **Comments:** Comments are the additional information provided against a statement or a chunk of code for the better clarity of the code. Interpreter ignores the comments and does not count them in commands.

  Symbols used for writing comments include Hash (#) or Triple Double Quotation marks (""").
  **Hash (#)** is used in writing **single-line comments** that do not span multiple lines. **Triple Quotation Marks (''' or """)** are used to write **multiple-line comments**. Three triple quotation marks to start the comment and again three quotation marks to end the comment.
- **Functions:** Function is a set of instructions defined under a particular name, which once written can be called whenever and wherever required.
- **Block(s):** A block refers to a group of statements which are part of another statement or function. All statements inside a block are indented at the same level.

## 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):

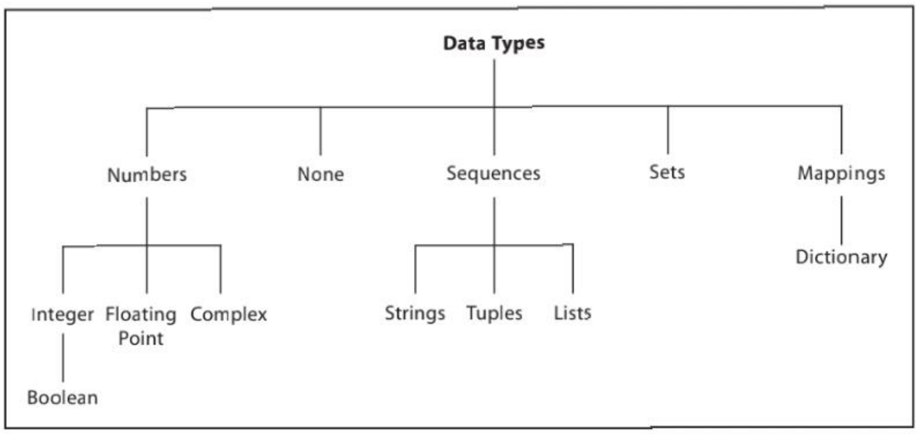
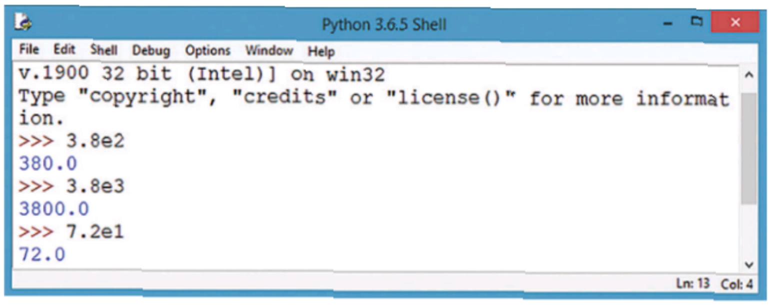


**Fig. 1.2:** Classification of Data Types in Python

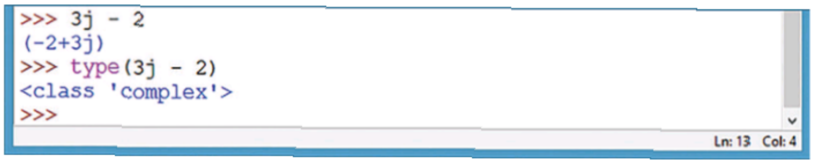
1. **Number or Numeric Data Type:** Numeric data type is used to store numeric values. It is further classified into three subtypes:
   (a) **Integer and Long:** To store whole numbers, *i.e.*, decimal digits without a fraction part. They can be positive or negative. **Examples:** 566, –783, –3, 44, 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:



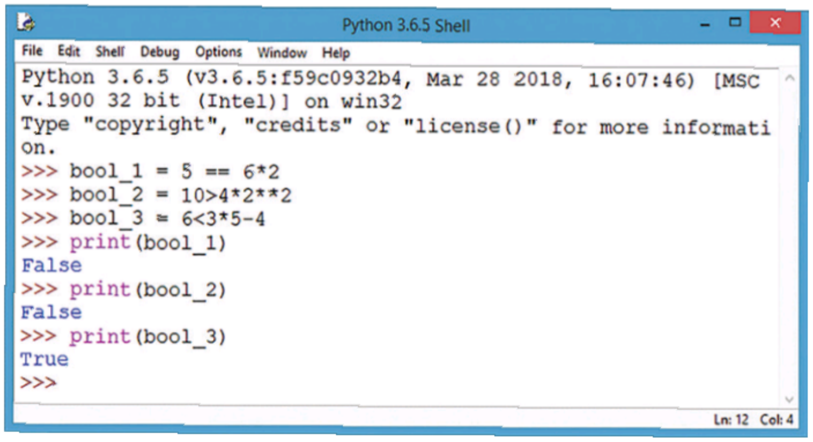
```
v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more informat
ion.
>>> 3.8e2
380.0
>>> 3.8e3
3800.0
>>> 7.2e1
72.0
```

(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 ({ }).

### 1.3.1 Dynamic Typing

One of the salient features of Python is dynamic typing. It refers to declaring a variable multiple times with values of different data types as and when required. It allows you to redefine a variable with different data types such as numeric, string, etc.

*For example,* consider the statement:

```
x = 20
```

The above statement signifies a variable 'x' of integer type as it holds an integer value. Now, suppose later in the program, you re-assign a value of different type to variable 'x' then, Python shall generate no error and allows the re-assignment with different set of values. *For example,*

```
x = 20
print(x)
x = "Computer Science"
print(x)
x = 3.276
print(x)
```

The above code on execution shall display the output as:

```
>>>20
>>>Computer Science
>>>3.276
```

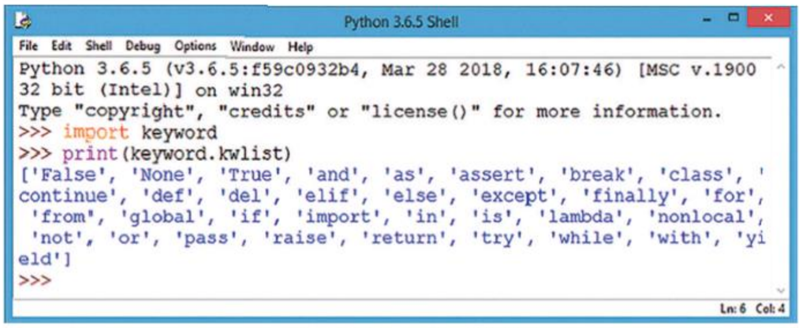
In the above example, we have assigned three different values to the variable 'x' with different types. This process is referred to as **Dynamic typing**.

**CTM:** Each and every element in Python is referred to as an object.

## 1.4 KEYWORDS

Keywords are the reserved words used by a Python interpreter to recognize the structure of a program. As these words have specific meanings for the interpreter, they cannot be used as variable names or for any other purpose. For checking/displaying the list of keywords available in Python, we have to write the following two statements:

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



```
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 information.
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', '
continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal',
 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yi
eld']
>>>
```

**Fig. 1.3:** Keywords in Python

**CTM:** All these keywords are in small letters, except for False, None, True, which start with capital letters.

## 1.5 MUTABLE AND IMMUTABLE TYPES

In certain situations, we may require changing or updating the values of certain variables used in a program. However, for certain data types, Python does not allow us to change the values once a variable of that type has been created and assigned values.

Variables whose values can be changed after they are created and assigned are called **mutable**.

Variables whose values cannot be changed after they are created and assigned are called **immutable**. When an attempt is made to update the value of an immutable variable, the old variable is destroyed and a new variable is created by the same name in memory.

Python data types can be classified into mutable and immutable as under:

- **Examples of mutable objects:** list, dictionary, set, etc.
- **Examples of immutable objects:** int, float, complex, bool, string, tuple, etc.

*For example,* int is an immutable type which, once created, cannot be modified.

Consider a variable 'x' of integer type:

```
>>>x = 5
```

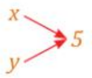
This statement will create a value 5 referenced by x.

$x \longrightarrow 5$

Now, we create another variable 'y' which is copy of the variable 'x'.

```
>>>y = x
```

The above statement will make y refer to value 5 of x. We are creating an object of type int. Identifiers x and y point to the same object.



Now, we give another statement as:

```
>>>x = x + y
```

The above statement shall result in adding up the value of x and y and assigning to x.

# Chapter 1: Review of Python basics (Computer Science)

After reading the documents write the answer of the following questions:

1. Write the use of the symbol # with an example.
2. What are the different types of variables and data types? Write with examples.
3. Give an example of none data type.
4. What is dynamic typing?
5. Differentiate between mutable and immutable types with an example.
6. Define: Operators, Keywords and Operands with example.