7.4 BAREBONES OF A PYTHON PROGRAM

Let us take our discussion further. Now we are going to talk about the basic structure of a Python program – what all it can contain. Before we proceed, have a look at following sample code. Look at the code and then proceed to the discussion that follows. Don't worry if the things are not clear to you right now. They'll become clear when the discussion proceeds.

```
# This program shows a program's components
# Definition of function SeeYou() follows
def SeeYou() :
    print ("Time to say Good Bye !!")
# Main program-code follows now
a = 15
b = a - 10
print (a + 3)
if b > 5 :
    print ("Value of 'a' was more than 15 initially.")
else :
    print ("Value of 'a' was 15 or less initially.")
SeeYou()
```

Comments (begin with #) — Function — Statements — Expressions — Statements and expressions etc. are made of multiple types of individual units called **tokens** — Blocks — # colon means it's a block — Indentation (see indented lines) — Inline comments (comment beginning in the middle of a line) — # calling above defined function SeeYou()

As you can see that the above sample program contains various components like :

- expressions
- comments
- blocks and indentation
- statements
- function

Let us now discuss various components shown in above sample code.

(i) Expressions

An ***expression*** is any legal combination of symbols that ***represents a value.*** An expression represents something, which **Python evaluates** and which then produces a *value.*

Some examples of expressions are

```
15
2.9
```
} *expressions that are values only*

```
a + 5
(3 + 5) / 4
```
} *complex expressions that produce a value when evaluated*

EXPRESSIONS

An ***expression*** is any legal combination of symbols that ***represents a value.***

Now from the above sample code, can you pick out all expressions ?

These are : 15, $a - 10$, $a + 3$, $b > 5$

(ii) Statement

While an expression represents something, a statement is a programming instruction that does something *i.e.*, some action takes place.

Following are some examples of statements :

```
print ("Hello")    # this statement calls print function
if b > 5 :
    ⋮
```

STATEMENT

A statement is a programming instruction that does something *i.e.*, some action takes place.

While an expression is evaluated, a statement is executed *i.e.*, some action takes place. And it is **not necessary that a statement results in a value** ; it may or may not yield a value.

Some statements from the above sample code are :

```
a = 15
b = a - 10
print (a + 3)
if b < 5 :
    ⋮
```

NOTE

A statement executes and may or may not yield a value.

(iii) Comments

Comments are the additional readable information, which is read by the programmers but ignored by Python interpreter. In Python, comments begin with symbol # (Pound or hash character) and end with the end of physical line.

COMMENTS

Comments are the additional readable information to clarify the source code.

Comments in Python begin with symbol # and generally end with end of the physical line.

In the above code, you can see *four* comments :

(*i*) The physical lines beginning with # are the **full line comments**. There are *three* full line comments in the above program are :

```
# This program shows a program's components
# Definition of function SeeYou( ) follows
# Main program code follows now
```

(*ii*) The fourth comment is an ***inline comment*** as it starts in the middle of a physical line, after Python code (see below)

```
if b < 5 :# colon means it requires a block
```

NOTE

A *Physical line* is the one complete line that you see on a computer whereas a *logical line* is the one that Python sees as one full statement.

Multi-line Comments

What if you want to enter a ***multi-line comment*** or a ***block comment*** ?. You can enter a multi-line comment in Python code in *two* ways :

(*i*) Add a # symbol in the beginning of every physical line part of the multi-line comments, *e.g.*,

```
# Multi-line comments are useful for detailed additional information.
# Related to the program in question.
# It helps clarify certain important things.
```

(ii) Type comment as a triple-quoted multi-line string *e.g.,*

```
'''  Multi-line comments are useful for detailed additional
     information related to the program in question.
     It helps clarify certain important things
'''
```

This type of multi-line comment is also known as *docstring*. You can either use triple-apostrophe (''') or triple quotes (""") to write *docstrings*. The docstrings are very useful in documentation – and you'll learn about their usage later.

NOTE

Comments enclosed in triple quotes (" " ") or triple apostrophe (' ' ') are called docstrings.

(iv) Functions

A function is a code that has a name and it can be reused (executed again) by specifying its name in the program, where needed.

In the above sample program, there is one function namely **SeeYou( )**. The statements indented below its **def** statement are part of the function. [All statements indented at the same level below *def SeeYou( )* are part of *SeeYou( )*.] This function is executed in main code through following statement (Refer to sample program code given above)

```
SeeYou()      # function-call statement
```

FUNCTIONS

A **function** is a code that has a name and it can be reused (executed again) by specifying its name in the program, where needed.

Calling of a function becomes a statement *e.g.,* **print** is a function but when you call print( ) to print something, then that function call becomes a statement.

For now, only this much introduction of functions is sufficient. You will learn about functions in details in Class 12.

v) Blocks and Indentation

Sometimes a group of statements is part of another statement or function. Such a group of one or more statements is called **block** or **code-block** or **suite**. *For example,*

ur spaces together mark next indent-level

```
if b < 5 :
    print ("Value of 'b' is less than 5.")
    print ("Thank you.")
```

This is a block with all its statements at same indentation level.

Many languages such as C, C++, Java etc., use symbols like curly brackets to show blocks but Python does not use any symbol for it, rather it uses indentation.

Consider the following example :

```
if b < a :
    tmp = a
    a = b
    b = tmp
print ("Thank you")
```

This is a block, part of if statement. Notice, all statements in same block have same indentation level.

This statement is not part of if

BLOCK OR CODE-BLOCK OR SUITE

A group of statements which are part of another statement or a function are called *block* or *code-block* or *suite* in Python.