7 Python Fundamentals

In This Chapter

7.1 INTRODUCTION

You must have heard the term IPO – Input, Process, Output. Most (in fact, nearly all) daily life and computer actions are governed by IPO cycle. That is, there is certain Input, certain kind of Processing and an Output.

Do you know that *programs* make IPO cycle happen ?

Anywhere and everywhere, where you want to transform some kind of input to certain output, you have some kind of input to certain output, you have to have a *program*. A program is a set of instructions that govern the processing. In other words, a program forms the base for processing.

In this chapter, we shall be talking about all basic elements that a Python program can contain. You'll be learning about Python's basics like *character set, tokens, expressions, statements, simple input and output* etc. So, are we all ready to take our first sincere step towards Python programming ? And, here we go :-).

7.2 PYTHON CHARACTER SET

Character set is a set of valid characters that a language can recognize. A character represents any letter, digit or any other symbol. Python supports Unicode encoding standard. That means, Python has the following character set :

- Letters — A-Z, a-z
- Digits — 0-9
- Special symbols — space + - * / ** \ ( ) [ ] { } // = != == < , > . ' " "" , ; : % ! & # <= >= @ _ (underscore)
- Whitespaces — Blank space, tabs (→), carriage return (↵), newline, formfeed.
- Other characters — Python can process all ASCII and Unicode characters as part of data or literals.

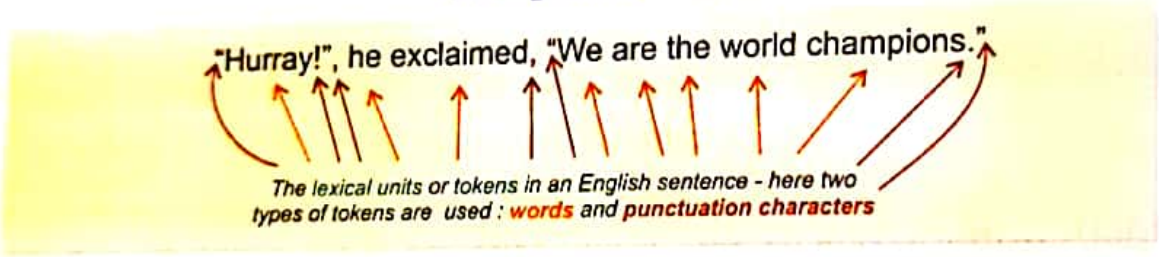
Every component of a Python program is created using the character set, be it statements or expressions or other components of a program.

7.3 TOKENS

In a passage of text, individual words and punctuation marks are called *tokens* or *lexical units* or lexical elements. The smallest individual unit in a program is known as a *Token* or a *lexical unit*.

Consider the following figure that tells what a *token* means.

An English Sentence



A Python Program

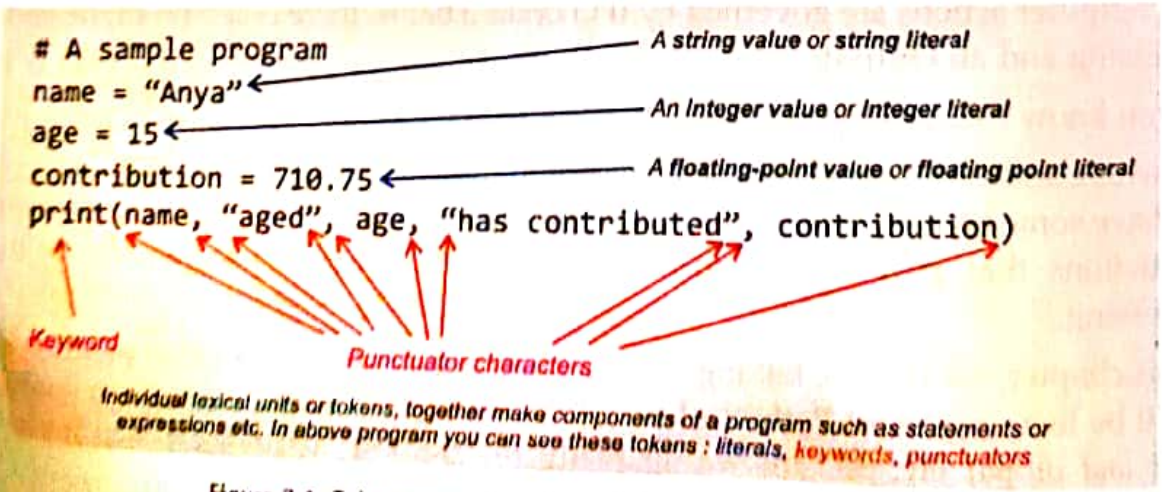


Figure 7.1 Tokens are smallest individual units in a program.

Python has following tokens :

(i) Keywords (ii) Identifiers (Names) (iii) Literals
(iv) Operators (v) Punctuators

Let us talk about these one by one.

TOKENS

The smallest individual unit in a program is known as a *Token* or a *lexical unit*

7.3.1 Keywords

Keywords are the words that convey a special meaning to the language compiler/interpreter. These are reserved for special purpose and must not be used as normal identifier names. Python programming language contains the following keywords :

KEYWORD

A ***keyword*** is a word having special meaning reserved by programming language

False	assert	del	for	in	or	while
None	break	elif	from	is	pass	with
True	class	else	global	lambda	raise	yield
and	continue	except	if	nonlocal	return	
as	def	finally	import	not	try	

7.3.2 Identifiers (Names)

Identifiers are fundamental building blocks of a program and are used as the general terminology for the names given to different parts of the program *viz.* variables, objects, classes, functions, lists, dictionaries etc. Identifier forming rules of Python are being specified below :

- An identifier is an arbitrarily long sequence of letters and digits.
- The first character must be a letter; the underscore (_) counts as a letter.
- Upper and lower-case letters are different. All characters are significant.
- The digits 0 through 9 can be part of the identifier except for the first character.
- Identifiers are unlimited in length. *Case* is significant *i.e.*, Python is case sensitive as it treats upper and lower-case characters differently.
- An identifier must not be a keyword of Python.
- An identifier cannot contain any special character except for underscore (_).

NOTE

Python is case sensitive as it treats upper and lower-case characters differently.

The following are some *valid* identifiers :

```
Myfile      DATE9_7_77
MYFILE      _DS
CHK         FILE13
Z2T0Z9 _    HJI3_JK
```

The following are some *invalid* identifiers :

`DATA-REC`	contains special character - (hyphen) (other than A - Z, a - z and _ (underscore))
`29CLCT`	Starting with a digit
`break`	reserved keyword
`My.file`	contains special character dot (.)

7.3.3 Literals / Values

Literals (often referred to as constant-Values) are data items that have a fixed value. Python allows several kinds of literals :

(*i*) String literals (*ii*) Numeric literals (*iii*) Boolean literals

(*iv*) Special Literal *None* (*v*) Literal Collections

STRING LITERALS

A *string literal* is a sequence of characters surrounded by quotes (single or double or triple quotes).

7.3.3A String Literals

The text enclosed in quotes forms a string literal in Python. *For example*, *'a'*, *'abc'*, *"abc"* are all string literals in Python. Unlike many other languages, both single character enclosed in quotes such as *"a"* or *'x'* or multiple characters enclosed in quotes such as *"abc"* or *'xyz'* are treated as String literals.

NOTE

In Python, one can form string literals by enclosing text in both forms of quotes – single quotes or double quotes.

As you can notice, one can form string literals by enclosing text in both forms of quotes – single quotes or double quotes. Following are some valid string literals in Python :

```
'Astha'   "Rizwan"   'Hello World'   "Amy's"   "129045"
'1-x-0-w-25'   "112FBD291"
```

Python allows you to have certain *nongraphic-characters* in String values. *Nongraphic characters* are those characters that cannot be typed directly from keyboard *e.g.*, backspace, tabs, carriage return etc. (No character is typed when these keys are pressed, only some action takes place.) These *nongraphic-characters* can be represented by using escape sequences. An escape sequence is represented by a backslash (\) followed by one or more characters.[1]

Following table (Table 7.1) gives a listing of escape sequences.

Table 7.1 *Escape Sequences in Python*

Escape sequence	What it does [Non-graphic character]	Escape sequence	What it does [Non-graphic character]
\\	Backslash (\)	\r	Carriage Return (CR)
\'	Single quote (')	\t	Horizontal Tab (TAB)
\"	Double quote (")	\uxxxx	Character with 16-bit hex value xxxx (Unicode only)
\a	ASCII Bell (BEL)	\Uxxxxxxxx	Character with 32-bit hex value xxxxxxxx (Unicode only)
\b	ASCII Backspace (BS)	\v	ASCII Vertical Tab (VT)
\f	ASCII Formfeed (FF)	\ooo	Character with octal value ooo
\n	New line character	\xhh	Character with hex value hh
\N{name}	Character named name in the Unicode[1] database (Unicode only)		

In the above table, you see sequences representing \, ', ". Though these characters can be typed from the keyboard but when used without escape sequence, these carry a special meaning and have a special purpose, however, if these are to be typed *as it is*, then escape sequences should be used. (In Python, you can also directly type a double-quote inside a single-quoted string and vice-versa. *e.g.* "anu's" is a valid string in Python.)

Check Point 7.1

1. What is meant by token ? Name the tokens available in Python.
2. What are keywords ? Can keywords be used as identifiers ?
3. What is an identifier ? What are the identifier forming rules of Python ?
4. Is Python case sensitive ? What is meant by the term 'case sensitive' ?
5. Which of the following are valid identifiers and why/why not :
   Data_rec, _data, 1 data, data1, my.file, elif, switch, lambda, break ?

String Types in Python

Python allows you to have *two* string types : (*i*) Single-line Strings (*ii*) Multiline Strings

(*i*) ***Single-line Strings (Basic strings).*** The strings that you create by enclosing text in single quotes (' ') or double quotes (" ") are normally single-line strings, *i.e.*, they must terminate in one line. To understand this, try typing the following in IDLE window and see yourselves :

```
Text1= 'hello↵
```