PYTHON

Practical Python Pro[illegible]
For Beginners and Experts

JONATHAN YATES

Chapter 3
Common Python Syntax

Python Identifiers

An identifier in any programming language is the name given to identify a variable, function, class, module or another object. In Python language, an identifier begins with an alphabetic letter A to Z or a to z or an underscore (_) followed by zero or more alphabetic letters, underscores and digits (0 to 9).

Python programming language does not allow special characters such as @, $, /, and % within identifiers. Python is a case sensitive programming language. Therefore, identifiers such as ‘Python’ and ‘python’ are two different identifiers in Python programming language.

Below are the naming conventions for identifiers in Python.

- Class name in Python always begins with an uppercase letter and all other Python identifiers starts with a lowercase letter.
- A Python identifier is private when such identifier begins with a single leading underscore.
- A Python identifier is strongly private when such identifier begins with two leading underscores.
- A Python identifier is a language-defined special name when such identifier ends with two trailing underscores.

Python Reserve Words

Reserve words in any programming language are special commands that compiler or interpreters understands, and these reserve words cannot be used as a constant or variable, or any other identifier names in that programming language.

Python has the following reserve words, and all such keywords contain lowercase letters

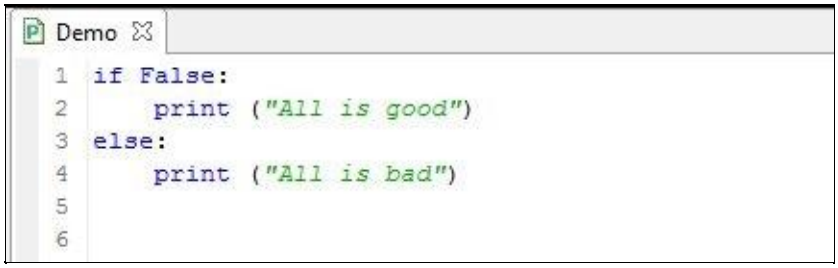
only.

and	def	exec	if	not	return
assert	del	finally	import	or	try
break	elif	for	in	pass	while
class	else	from	is	print	with
continue	except	global	lambda	raise	yield

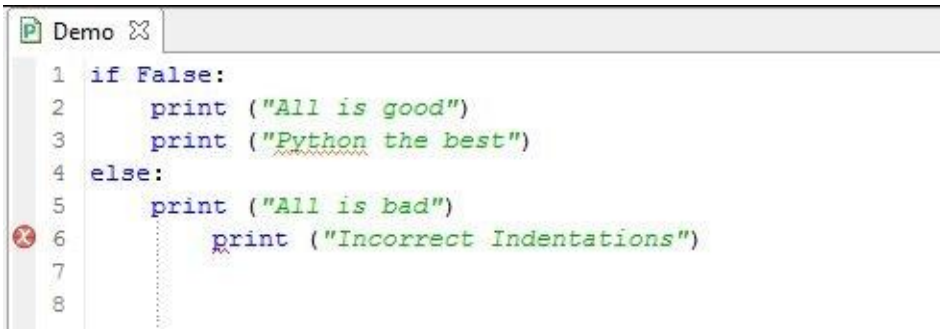
Python Keywords

Lines and Indentations

Any block of code in Python are denoted by line indentation, which is rigidly enforced. Python has no braces to denote blocks of code for class definitions and function definitions or flow control. The number of spaces used in an indentation can be variable but for all statements in a particular block, the number of spaces should always be same. For example, below, the block is correctly indented and therefore, there is no error.



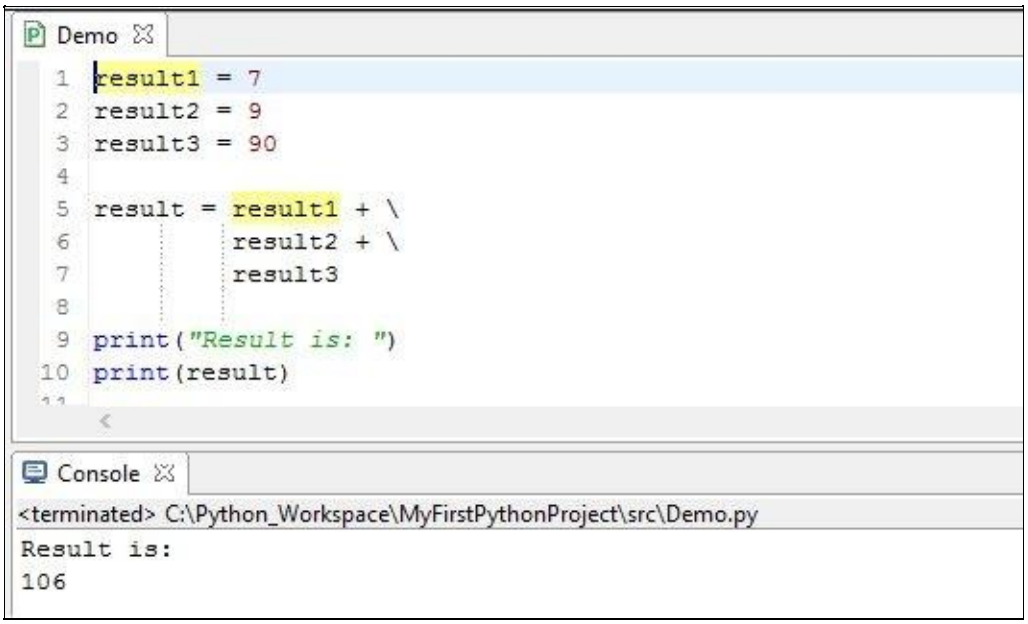
In the next example, since the last statement in the block is not properly indented, consequently, the block has an error.



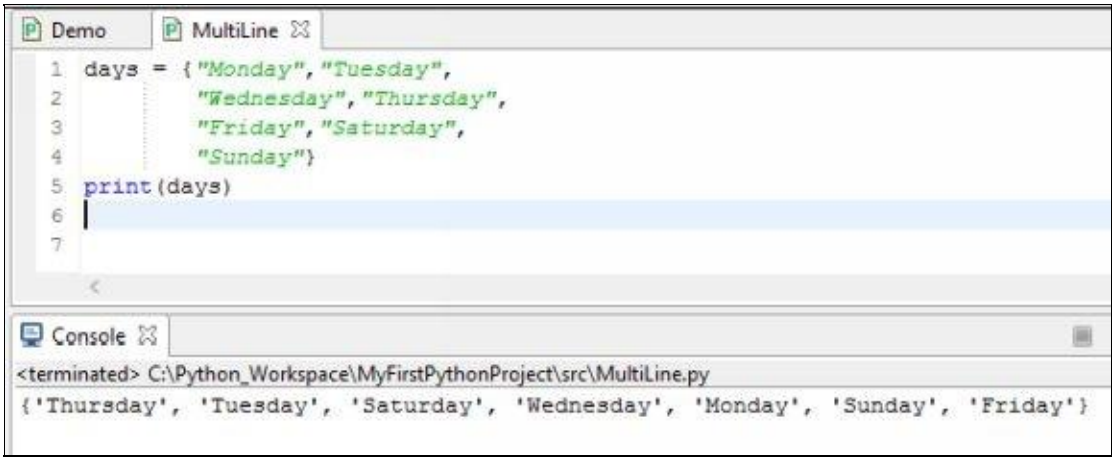
Therefore, the conclusion is that in Python programming language, all the continuous lines indented with the same number of spaces would form a block.

Representing a Statement as Multi-Line

Statements in the Python language ends with a new line. If the statement is required to be continued into the next line, then the line continuation character (\) is used in Python language. This line continuation character (\) denotes that the statement line should continue as shown in the below screenshot. In the below example, we have three variables result1, result2 and result3 and the final output is copied to the variable named result. Instead of writing the equation statement in a single line (result=result1+result2+result3), here, we have used line continuation character (\) so that, it could be written in three lines but represents a single statement in Python language.

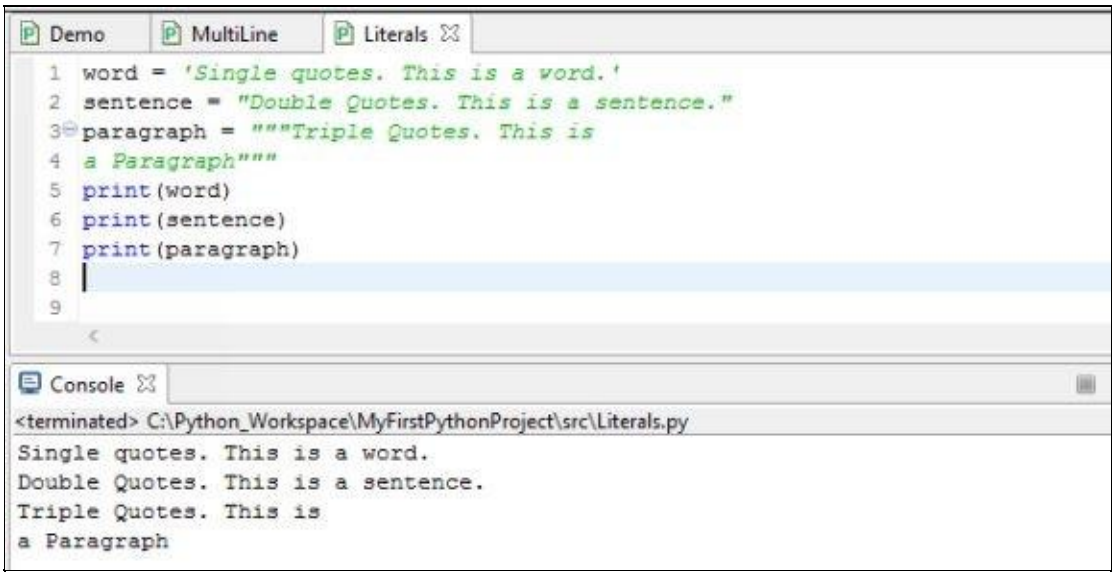


Also, a Python statement which is defined within braces (), {} and [] does not require the line continuation character (\) when written as a multi-line statement. This kind of Python statements are still interpreted as a single statement without the use of the line continuation character (\).



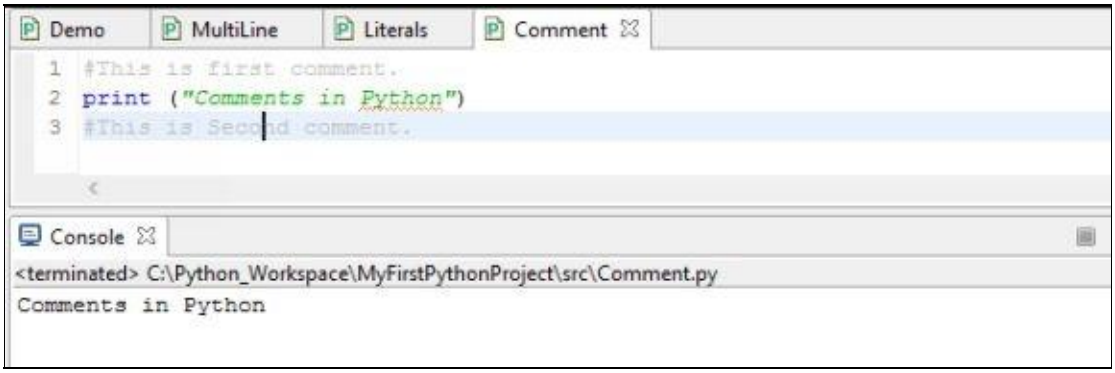
Quotation in Python

The Python language permits the use of single (‘), double (“) and triple (”’ or ”””) codes to represent a string literal, making sure that the same type of quote begins and ends that string. In the below example, single, double and triple codes are used to represent a string in a word, sentence or paragraph. When we print this variable, they print the string irrespective of single, double and triple codes used for representing string literal in Python language.



Comments in Python

Any comment in the Python language is represented by a hash sign (#) provided it is not used inside a string literal between codes (single, double or triple). All characters after the hash sign (#) and up to the end of the physical line are the part of comment and Python interpreter ignores this statement while interpreting the whole program. In the below example, the interpreter will just print the string present inside the print command and will ignore the parts mentioned after a sign before and after as comments.



Using Blank Lines

A blank line in the Python language is either a line with white spaces or a line with comments (i.e. statement starting with a hash sign (#)). The Python interpreter while interpreting a blank line, ignores it and no machine readable code will be generated. A multiline statement in Python is terminated after entering an empty physical line.

Waiting for the User

Using the Python programming language, we can set up the prompt which can accept a user's input. The following line of the program will display a prompt, which says "Press any key to exit", and waits for the user input or action.

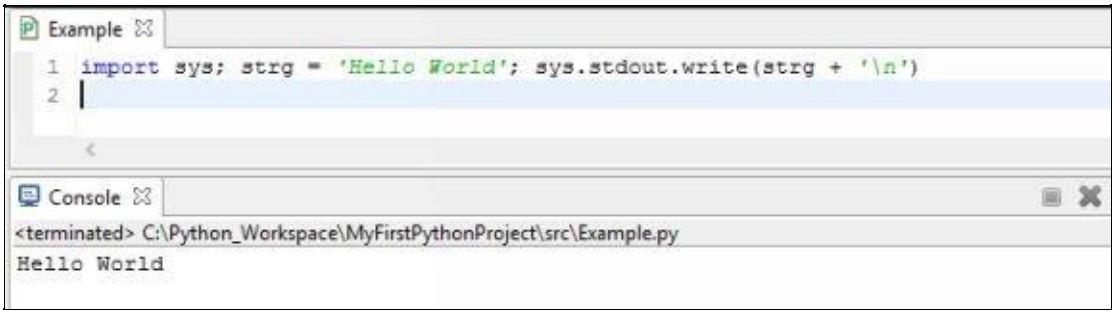
```
#! /usr/bin/python
raw_input ("\n\nPress any key to exit.")
```

Also, in the above statement, we have used "\n\n". This is used to create two new lines before displaying the actual line. Once the key is pressed by the user, the program will end. By doing this, we can keep a window console open until the user has finished his work with an application.

Multiple Statements on a Single Line

The Python language allows to write multiple statements on a single line if they are

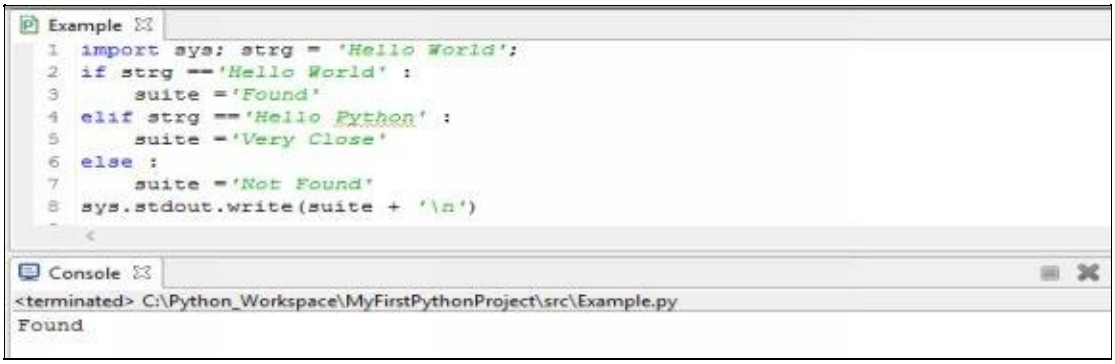
separated by a semicolon (;) as demonstrated in the example below.



Multiple Statement Groups as Suites and Header Line

In the Python language, a group of individual statements making a single code block are called suites. Whereas the compound or complex statements, such as if, def, while, and class require a suite and a header line.

Header line is the one that begins a statement (with the keyword like if, elif, else, etc.) and ends with a colon (:) and is followed by one or more lines which makes up the suite as demonstrated in the below example. Here, ***if strg==’Hello World’:*** is a header line which is followed by a suite (suite = ‘Found’).



Command Line Arguments

On UNIX OS, which has Python interpreter installed, we can take help and see all the lists of the functions. These are the basic ones. The below screenshot demonstrates the help command on the UNIX system and all the functions or short codes used.

```
$ python -h
```

usage: python [option] … [-c cmd | -m mod | file | -] [arg] …

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

[etc.]

Chapter 4

Types of Variables in Python

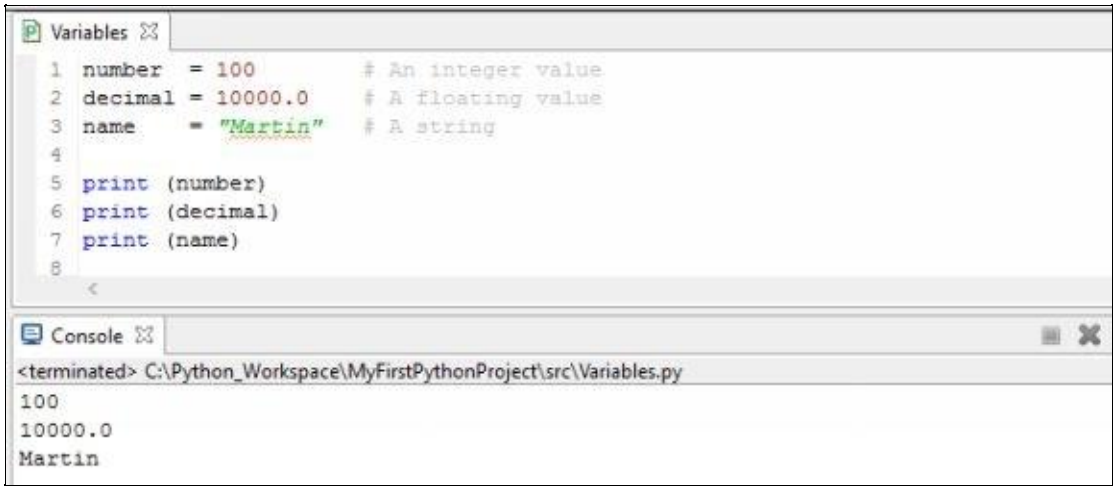
Variables in any programming language are the names of the reference to the reserved memory locations which are used to store values. Similarly, when we are creating a variable in Python then we are reserving some space in the memory.

These variables have their own data type. Based on the type, the interpreter allocates memory and decides what kind of data can be stored in these reserved memory locations. Characters, integers, decimals, etc. are the different data types which can be assigned to such variables.

Assigning Values to Variables

In the Python language, equal sign (=) is used to assign values to the variables. Such variables do not need explicit declaration. When we assign a value to a variable, the declaration or creation happens automatically.

The operand to the left of the equal sign (=) is the name of the variable and the operand to the right of the equal sign (=) is the value stored in the variable. This is demonstrated in the below example.



In the above example, the variable name ‘number’ has an integer value therefore, it behaves as an integer without any data type declaration. Similarly, the variable name ‘decimal’ has a floating value and variable name ‘name’ has a string value. Python is a THvery flexible language since it automatically determines the data type once the value is

assigned to the variable.

Multiple Assignment

The Python language allows the assignment of a single value to more than one variables and multiple values to multiple variables which are separated by commas in a single line as demonstrated in the below example.

In the first case (many-t0-one), the single value 1000 is assigned to many variables a, b, c and d.

In the second case (many-to-many), multiple values (“Jose”, “Patrick”, “Peter”) are assigned to multiple variables k, l and m. However, here is one to one mapping between a variable and a value, e.g. variable k will contain value as “Jose”, variable l will contain value as “Patrick” and variable m will contain value as “Peter”.

Standard Data Types in Python

In Python, the data is stored in memory which can be of many types. For example, a person’s birth year is stored as a numeric value and his or her qualifications are stored as alphanumeric characters. Depending on the type of value, the Python has different standard data types that are used to define the type of value a variable can contain.

Python language has five standard data types. We are going to discuss them in detail. These are:

Numbers

Strings

Lists

Tuples

Dictionary

Python Numbers

In Python language, the number data type are used to store numeric values. Numeric variable are created automatically in Python when we assign a numeric value to it as shown in the below example.

Python supports below four different numerical types.

int (signed integers)

long (long integers, they can also be represented in hexadecimal and octal)

float (floating point real values)

complex (complex numbers)

Below are the examples of number objects in Python language.

Int	Long	float	Complex
40	7965391L	0.0	8.14j
900	-0x29546L	17.90	675.j
-589	0455L	-31.9	23.8922e-36j

050	0xABDDAECCBEABCBFEACl	62.3+e68	.6776j
-0630	563213626792L	-560.	-.6844+0J
-0x1290	-032318432823L	-82.53e200	4e+86J
0x37	-5627995245529L	40.2-E52	7.59e-7j

Below are few things to note about Python number objects.

A complex number consists of an ordered pair of real floating-point numbers denoted by (real + img**j)**, where real and img are the real numbers and j is the imaginary number unit.

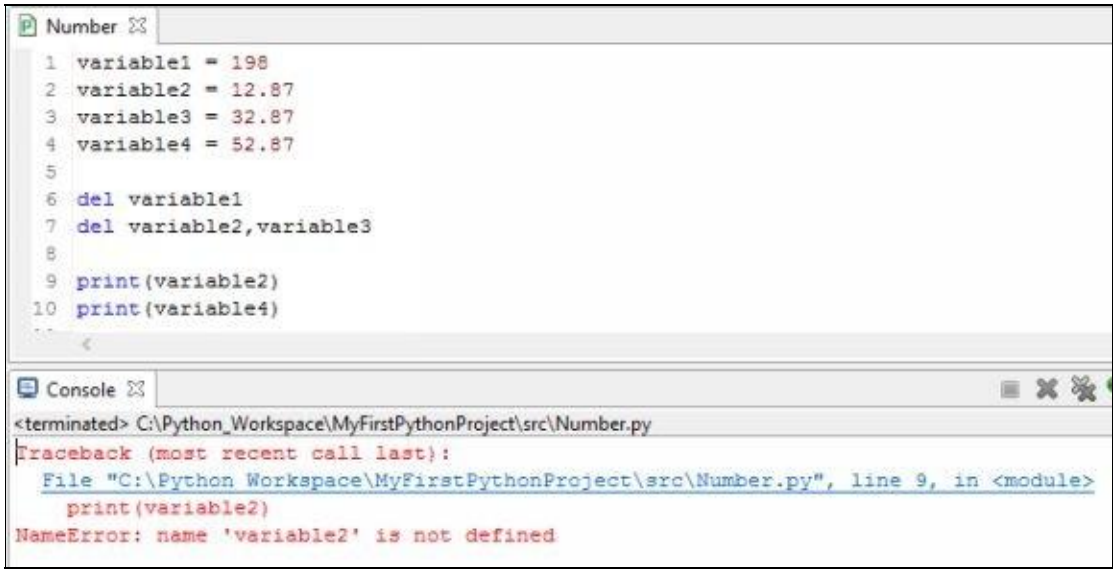
Python displays long integers (data type number) with an uppercase L.

Python language allows to use a lowercase L with long data type number, however it is recommended to use only an uppercase L in order to avoid confusion with the number 1.

In Python, we can delete the reference to a number object (variable) by using the ‘del’ statement. Given below is the syntax of the ‘del’ statement.

```
del
variable1[,variable2[,variable3[….,variableN]]]]
```

Using the above statement, we can delete a single variable or multiple variables by using the ‘del’ statement as shown in the below example.



In the above example, since we have deleted variable2 using the ‘del’ command, this variable do not exist anymore when we tried to print it.

Strings in Python

In Python language, Strings are identified as a contiguous set of characters which are represented within the quotation marks. Python language permits the use of single (‘), double (“) and triple (”’ or ”””) codes to represent a string literal, making sure that the same type of quote begins and ends that string.

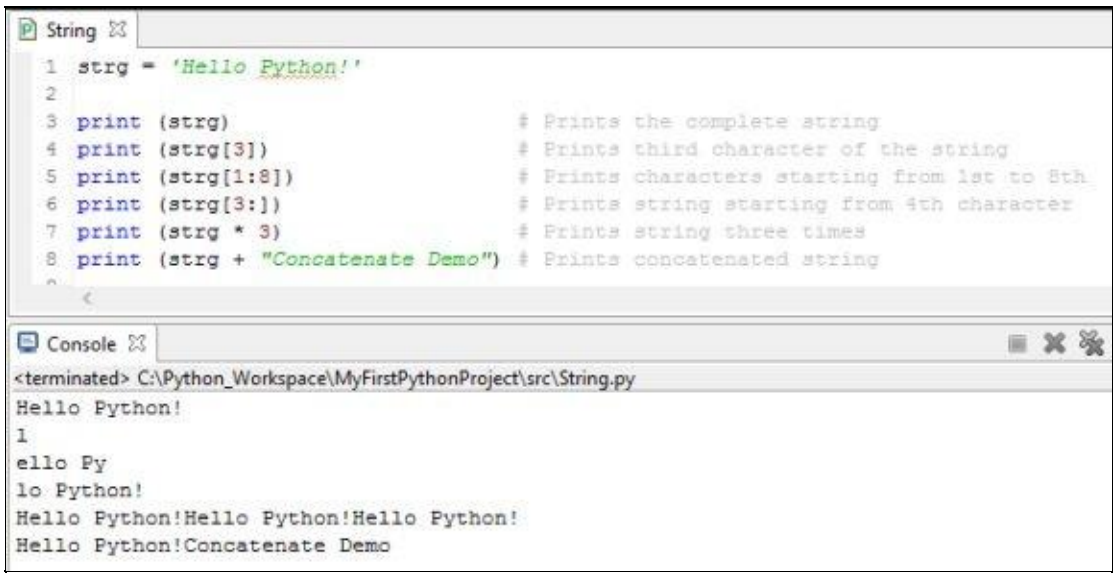
Strings in Python have below operators.

Slice operator ([ ] and [:]). By using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end, subsets of strings can be taken.

Plus (+) sign operator. By using the plus (+) sign operator, we can concatenate two or more strings.

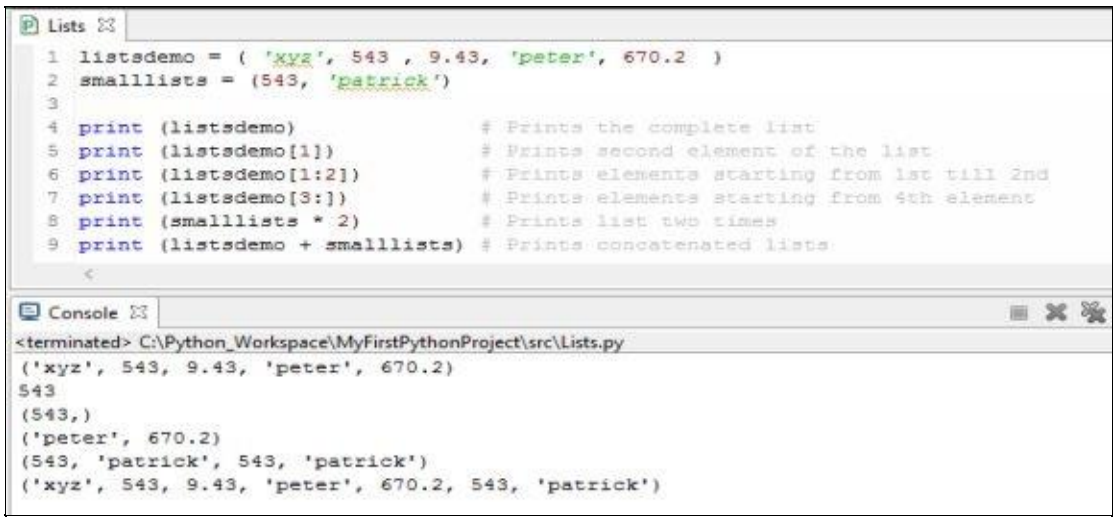
Asterisk (*) sign operator. Asterisk operator is the repetition operator. If we want to print string 3 times, then simply we can give command as print (string * 3).

All of above operators are demonstrated in the below example.



Lists in Python

In Python language, a List is the most versatile compound data types. A list contains items which are separated by commas and enclosed within square brackets ([]). Lists are similar to arrays in C or C++ in some extents. The difference between arrays in C /C++ and lists in Python is that the former cannot have different datatype for elements while latter can have different datatype for elements.



Lists in Python have below operators.

Slice operator ([ ] and [:]). By using the slice operator ([ ] and [:]) with element position starting at 0 in the beginning of the list and working their way from -1 at the end, subsets of the list can be taken.

Plus (+) sign operator. By using the plus (+) sign operator, we can concatenate

two or more lists.

Asterisk (*) sign operator. Asterisk operator is the repetition operator. If we want to print a list 2 times, then simply we can give command as print (listsdemo * 2).

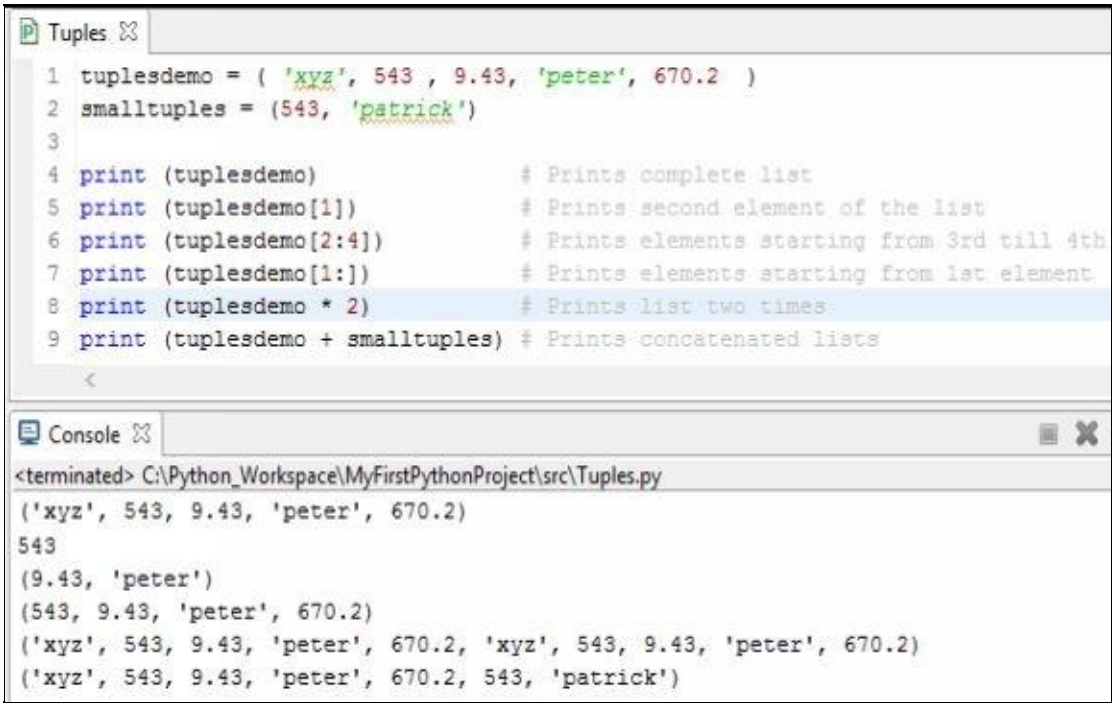
Tuples in Python

In Python language, a tuple is a sequence data type which is almost similar to the list. A tuple consists of a number of values which are comma separated. Unlike lists, tuples are enclosed within parentheses.

The main differences between tuples and lists are as follows.

Tuples are enclosed in parentheses (( )) whereas Lists are enclosed in brackets ([ ]).

Tuples are read-only lists as their elements and size cannot be changed, while Lists can be updated. We can change lists elements and size.



Tuples in Python have below operators.

Slice operator ([ ] and [:]). By using the slice operator ([ ] and [:]) with element position starting at 0 in the beginning of the tuple and working their way from -1 at

the end, subsets of the tuple can be taken.

Plus (+) sign operator. By using the plus (+) sign operator, we can concatenate two or more tuples.

Asterisk (*) sign operator. Asterisk operator is the repetition operator. If we want to print a tuple 2 times, then simply we can give command as print (tuplesdemo * 2).

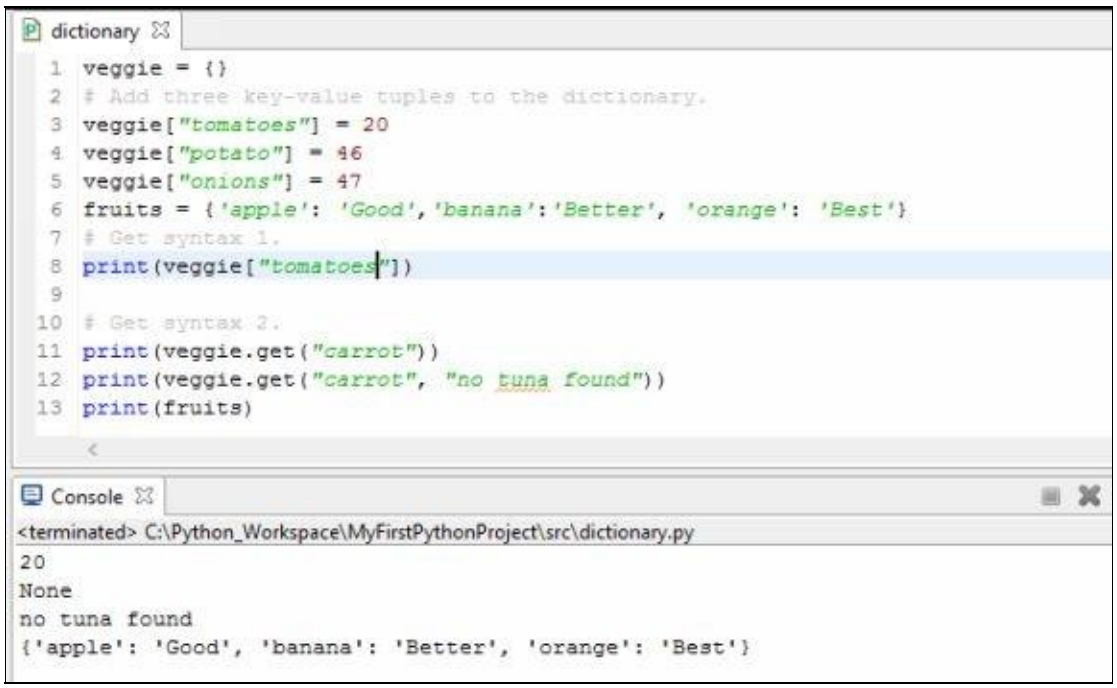
Dictionary in Python

A dictionary in Python represents hash table. A hash table (or hash map) is a data structure which is used to implement an associative array, a structure that can map keys to values. To compute an index of an array of buckets or slots, a hash table uses a hash function to procure the desired value. This concept in Python work like associative arrays or hashes found in Perl and consist of key-value pairs. Keys in Python dictionary can be of any data type, however mostly they are either numbers or strings. On the other hand, values in Python dictionary are Python objects.

In Python, syntax wise there are two ways dictionaries can be created which are mentioned below:

1. Dictionary name is given with curly braces ({ }) first (E.g. veggie = {}). Next we can define the key value pairs one by one as (E.g. veggie [“tomatoes”] = 20). Here, key is tomatoes and the value is 20.
2. Dictionary can also be defined with all key value pairs in one go within the curly braces ({}). (E.g. fruits = {‘apple’: ‘Good’,‘banana’:‘Better’, ‘orange’: ‘Best’}). Here, dictionary name is ‘fruits’, ‘apple’ is one of the key of such dictionary and ‘Good’ is the associated value with this key.

These syntaxes are demonstrated in the below example.



Data Type Conversion

While writing programming code, we may need to perform data type conversions. To support such operations, Python language has several built-in functions which are used to perform conversion from one data type to another. After conversion, these functions return a new object representing the converted value. Below is the list of Python built-in functions along with their operational description.

Function	Description
int(value [,Base])	This function converts value into an integer. “Base” specifies the base if value is a string.
long(value [,Base])	This function converts value into a long integer. “Base” specifies the base if value is a string.
chr(value)	This function converts an integer into a character.
	This function is used to create a

complex(real [,imag])	complex number.
dict(Value)	This function is used to create a dictionary. “Value” must be a sequence of (key, value) tuples.
eval(strg)	This function is used to evaluate a string which returns an object.
float(value)	This function converts value into a floating-point number.
frozenset(value)	This function converts value into a frozen set.
hex(value)	This function converts an integer value into a hexadecimal string.
list(value)	This function converts value to a list.
repr(value)	This function is used to convert an object value to an expression string.
oct(value)	This function is used to converts an integer value to an octal string.
ord(value)	This function is used to converts a single character to its integer value.
set(value)	This function is used to convert value into a set.
str(value)	This function is used to convert an object value into a string representation.
tuple(value)	This function is used to convert value into a tuple.

unichr(value)	This function is used to convert an integer value into a Unicode character.