Chapter 3: Creating Python Variables

Objectives

To discuss the concept of Python variables

- Explore the Python variables meaning along with discussing the means to create it, types, or the technique of assigning name
- Discuss the types of data present along with mentioning the means of deciding data for the variable

Introduction

One of the important aspects of working with any programming language is to understand how to store the data and manipulate it for completing the desired work. This chapter thus focuses on discussing this feature of Python by examining the Python variables, their types, and the procedure of naming variables along with assessing different built-in data types present in Python.

• Python Variables

Python variables are simply defined as the containers used for storing the values. In simple terms, it's the space that you keep in memory. There is no explicit command which needs to be entered for the creation of a variable. The sign of equal to (=) is used for the value assignment. For example – Two variables are created i.e. name and age

```
name = "Dinesh"
age = 20
```

Herein, the 'name' and 'age' are the variable names while 'Dinesh' and '20' are the variable values.

Python also allows the multiple assignments of a single value for different variables by using equal signs simultaneously. This could be written as:

```
x = y = z = 10
```

Python also provides the facility of deleting a created variable by using 'del'. For the above code, in case deletion of the variable message is done:

```
message = "I am learning Python"
print(message)
del message
print(message)
```

The output would be:

```
C:\Users\HP\Google Drive\Riya\2023\May\IAQS>sample.py
I am learning Python
Traceback (most recent call last):
  File "C:\Users\HP\Google Drive\Riya\2023\May\IAQS\sample.py", line 4, in <mod
le>
    print(message)
NameError: name 'message' is not defined
```

- **Types of variables**

As Python is completely an object-oriented language and not the typed one, there is no need to declare a variable or its type. Python is an inferred language. Therefore it can derive the type of variable by examining the stored data.

While working with the variables, different types of variables are discussed, like:

- Numbers – This variable type is used for storing numbers. There are two types of numbers i.e. floating point numbers which consist of decimals and integers which are whole numbers like height is a floating point while age is an integer.

```
height = 133.5
age = 20
```

- String – The variable used for storing text is called a string variable. The string variable could be defined by using a single or double quote. Herein, in the below example name and education are two string variables defined using double and single quotes respectively.

```
Name = "Ankit"
Education = 'Graduate'
```

The variable types also be classified based on the location for which the variable is created. The variable which is defined inside a function is called a local variable that cannot be accessed outside the function. For example – If a code is written as below wherein the product of two variables i.e. a and b need to be derived.

```
def product(a,b):
    product=a*b
    return product
print(product(3,4))
```

In the above example the value of a and b is not defined outside the function and the value is called using function only thus, the variables a and b are local variables that can only be used concerning function product.

However, in case when a variable can be accessed within any function without any restriction then the variable is said to be global. The case of a local variable can be rewritten in a global variable as:

```
a = 3
b = 4
def product():
    product=a*b
    return product
print(product())
```

Herein, as the variables a and b are defined outside the function product thus they are global variables and can be used anywhere in the code without having any boundary.

- **Variable name**

The variable name is simply defined as the identifier for the location wherein data is stored. In Python, every variable needs to be unique and thus, there is a set of rules which need to be followed for naming the variables i.e.

- The name of the variable should not be the same as the keywords of the language
- The variable name is case-sensitive thus, name and Name are two different variables
- The starting of a variable name should be from an underscore character or a letter
- The variable name could only consist of the underscores or alpha-numeric characters i.e. (0-9, A-Z, a-z, and _)
- No special character or number can be used as the variable name starting i.e. a variable name cannot be 1age or _email.

• Python Data types

The data types in Python are defined as the means of measuring the variable type. It generally explains the type of data stored within a variable. For example – the address stored in a variable would define alphanumeric characters while the experience of an employee is the numeric variable.

- **Built-in data types**

Python has the presence of many built-in data types. These built-in data types could be used to store different types of values, thus, the knowledge of each data type is relevant. The built-in data types are

- Numeric – The numeric data type is defined as the one wherein numeric values are stored. There are generally four types of numeric values i.e. integers, complex, float, and

long (the values exists in Python 2. x but is depreciated in Python 3. x). The sample for each of type is

```
a = 1
b = 2.0
c = 3L
d = 4j
print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

For the given values, the type could be derived which represents each numeric type. As with Python 3. x long doesn't work thus, its command is commented and the output is derived as

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

- String – The string data type consists of the character's contiguous set which is represented using quotes. Python allows to have the usage of double or single quotes. The sample code for each of the functions is presented below

```
a = "Learning"
b = "Python"
print(type(a))
print(type(b))
print(a+b)
print(a*2)
print(b[2])
print(b[2:4])
```

The output for the code is given below

```
<class 'str'>
<class 'str'>
LearningPython
LearningLearning
t
th
```

The code shows the type of a and b, concatenates both the variables, repeats the variable a, provides the value of b at index 2, and also slices the b variable from index value 2 to 4.

- Sequence - The sequence data types could be of three types i.e. list, tuple, and range. The list data type is the most versatile one wherein items are separated by a comma and written in square brackets [].Tuple data type is another sequence wherein enclosed with parenthesis (), the values are separated by a comma only. Lastly, the range is the in-built

function that returns the number sequence starting from 0 and increasing by 1 until the desired point is derived. The code for each step is given below

```
a = [1, 2]
b = [3]
c = (4, 5)
d = (6)
print(a)
print(a[1])
print(a+b)
print(b*2)
print(c)
print(c[1])
print(d*2)
for i in range(1,10,2):
    print(i)
```

In the above code, a and b are lists while c and d are tuples. As tuples can't be updated thus, concatenation for them can't be done. Herein i take the value from the range computed by having starting value of 1 while the ending point is 10 and an increment level of 2. The output for the functions is given below.

```
[1, 2]
2
[1, 2, 3]
[3, 3]
(4, 5)
5
12
1
3
5
7
9
```

- Binary – The binary built-in data type consists of three things i.e. bytes, bytearray, and memoryview. The bytes and bytearray are the means of manipulating binary data while memoryview is used as the buffering protocol for accessing the other binary objects' memory without making any copy. The bytes are similar to strings just that with bytes prefix 'b' is added. Bytearray is created by bytearray() constructor and they are mutable objects.

- Mapping – The Python dictionary is the one used for representing mapping data type. It is a hash table type that works like associative arrays which consist of key and value pairs. Enclosed using the curly braces i.e. {}, with dictionary the values could be accessed and assigned using square brackets [].The sample for the mapping is

```
Map = {1:'a', 2:'b', 3:'c', 4:'d'}
print (Map)
print(Map[1])
print (Map.keys())
print (Map.values())
```

Herein, the output of the code prints the dictionary, the value of key 1, all keys, and all the values.

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}
a
dict_keys([1, 2, 3, 4])
dict_values(['a', 'b', 'c', 'd'])
```

- Boolean – The boolean data type is the built-in data type that can be used for the representation of the variable by two values i.e. True or False. The bool() function is the one that helps in the evaluation of the variable and has the results in the form of True or False.

- Set – The set in Python is defined as an unordered collection of the items which are enclosed using the curly braces {}. The Python set elements are mutable but the elements cannot be duplicated. Another form of set built-in data type is a frozenset which is defined as the immutable form of the set wherein no removal or addition could be done. The sample code for the types is

```
elements= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
print(elements)
print(type(elements))
for i in elements:
    print(i)
elements.remove(0)
print(elements)
frozen = frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
frozen.remove(0)
```

Herein, elements are the set which is initially printed along with its type mentioning, and looping through each element to print each value of the set. Further, as the set is mutable thus a value is removed, and then again set is printed. Now for another frozenset the feature of remove is tried but as it's not mutable thus the error is derived i.e.

```
    frozen.remove(0)
AttributeError: 'frozenset' object has no attribute 'remove'
```

Now, remove the frozen.remove(0) command from the code we can run the code

```
elements= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
print(elements)
print(type(elements))
for i in elements:
    print(i)
elements.remove(0)
print(elements)
frozen = frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
print(frozen)
```

and the results would be

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
<class 'set'>
0
1
2
3
4
5
6
7
8
9
{1, 2, 3, 4, 5, 6, 7, 8, 9}
frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
```

- None – The none is the data type that is used to show that object consists of no data. Though a variable None could be assigned but there is also the presence of methods that are used for calling the None type.

• Working with data types

The specification of the built-in data types mentions that there is the presence of data of various natures. Sometimes it's known to the coder while sometimes not known. Thus, there is the presence of a function to check the type of data. The type() function is the one which can be used for determining which type of data is stored in a variable. The example code for using the function is

```
data = "numeric"
number = 10
value = 15.0
print(type(data))
print(type(number))
print(type(value))
```

Herein, the type for each of the variables could be derived using type() function i.e.

```
<class 'str'>
<class 'int'>
<class 'float'>
```

Once the type is derived, often while working the requirement could be to convert the data type from one form to another. This process of conversion is known as data type conversion. These conversions could be of different types as shown in the below example

```
a = 10
b = 11.0
c = "4"
print(int(c))
print(int(b))
print(float(a))
print(float(c))
print(str(a))
print(str(b))
```

The output for the above code is

```
4
11
10.0
4.0
10
11.0
```

- Conversion to int – Herein, b is float and c is a string that is converted into an integer using int().
- Conversion to float –The example has a as an integer and c as a string which is converted into float using float().
- Conversion to string – The integer value of a and the floating value of b are converted into a string using str().

There is even the presence of many functions like int(), long(), float(), complex(), str(), tuple(), list(), set(), dict(), frozenset(), oct(), and hex(); which can be used for setting the data type. Thus, Python allows the creation of variables with the inclusion of the variable type needed by the user.

• Chapter Summary

- Python variables are simply defined as the containers used for storing the values.
- There are different types of variables like numeric and string which helps in storing various types of data
- The numeric variable is used for storing integer or floating values while the string is for textual data.
- Based on the location of storing variable, the Python variables are classified as local and global.
- Local variables can’t be accessed outside functions while global can be.
- The variable name is simply defined as the identifier for the location wherein data is stored, still, every variable needs to be unique in Python.
- A set of rules is specified for defining variable names.
- A mnemonic naming procedure is recommended while naming variables
- The data types in Python are defined as the means of measuring the variable type.
- The built-in data type is numeric, string, binary, sequence, mapping, none, set, and Boolean.
- The built-in data types provide the feature of converting the data type, setting the type of a value, and also determining the type of a variable.