PYTHON VARIABLES

**Python Variables:-**

* Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value.
* In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type.
* Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore.
* It is recommended to use lowercase letters for the variable name. Rahul and rahul both are two different variables.

**Identidier naming:-** Variables are the example of identifiers. An Identifier is used to identify the literals used in the program.

The rules to name an identifier are given below:-

* The first character of the variable must be an alphabet or underscore ( \_ ).
* All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
* Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, \*).
* Identifier name must not be similar to any keyword defined in the language.
* Identifier names are case sensitive; for example, my name, and MyName is not the same.
* Examples of valid identifiers: a123, \_n, n\_9, etc.
* Examples of invalid identifiers: 1a, n%4, n 9, etc.

**Declaring Variable and Assigning Values:-**

* Python does not bind us to declare a variable before using it in the application. It allows us to create a variable at the required time.
* We don't need to declare explicitly variable in Python. When we assign any value to the variable, that variable is declared automatically.
* The equal (=) operator is used to assign value to a variable.

**Object References:-**

It is necessary to understand how the Python interpreter works when we declare a variable. The process of treating variables is somewhat different from many other programming languages.

Python is the highly object-oriented programming language; that's why every data item belongs to a specific type of class. Consider the following example.

print("John")

**Output:**

John

The Python object creates an integer object and displays it to the console. In the above print statement, we have created a string object. Let's check the type of it using the Python built-in **type()** function.

type("John")

**Output:**

<class 'str'>

In Python, variables are a symbolic name that is a reference or pointer to an object. The variables are used to denote objects by that name.

Let's understand the following example

a = 50

a

50

In the above image, the variable **a** refers to an integer object.

Suppose we assign the integer value 50 to a new variable b.

**a = 50**

**b = a**

a

50

b

100

Python manages memory efficiently if we assign the same variable to two different values.

**Object Identity:-**

In Python, every created object identifies uniquely in Python. Python provides the guaranteed that no two objects will have the same identifier. The built-in **id()** function, is used to identify the object identifier. Consider the following example.

a = 50

b = a

print(id(a))

print(id(b))

# Reassigned variable a

a = 500

print(id(a))

**Output:-**

140734982691168

140734982691168

2822056960944

We assigned the **b = a, a** and **b** both point to the same object. When we checked by the **id()** function it returned the same number. We reassign **a** to 500; then it referred to the new object identifier.

**Variable Names:-**

We have already discussed how to declare the valid variable. Variable names can be any length can have uppercase, lowercase (A to Z, a to z), the digit (0-9), and underscore character(\_). Consider the following example of valid variables names.

name = "Devansh"

age = 20

marks = 80.50

print(name)

print(age)

print(marks)

**Output:**

Devansh

20

80.5

Consider the following valid variables name.

name = "A"

Name = "B"

naMe = "C"

NAME = "D"

n\_a\_m\_e = "E"

\_name = "F"

name\_ = "G"

\_name\_ = "H"

na56me = "I"

print(name,Name,naMe,NAME,n\_a\_m\_e, NAME, n\_a\_m\_e, \_name, name\_,\_name, na56me)

**Output:**

A B C D E D E F G F I

In the above example, we have declared a few valid variable names such as name, \_name\_ , etc. But it is not recommended because when we try to read code, it may create confusion. The variable name should be descriptive to make code more readable.

The multi-word keywords can be created by the following method.

* **Camel Case -** In the camel case, each word or abbreviation in the middle of begins with a capital letter. There is no intervention of whitespace. For example - nameOfStudent, valueOfVaraible, etc.
* **Pascal Case -** It is the same as the Camel Case, but here the first word is also capital. For example - NameOfStudent, etc.
* **Snake Case -** In the snake case, Words are separated by the underscore. For example - name\_of\_student, etc.

**Multiple Assignment:-**

Python allows us to assign a value to multiple variables in a single statement, which is also known as multiple assignments.

We can apply multiple assignments in two ways, either by assigning a single value to multiple variables or assigning multiple values to multiple variables. Consider the following example.

**1. Assigning single value to multiple variables**

**Eg:**

x=y=z=50

print(x)

print(y)

print(z)

**Output:**

50

50

50

**2. Assigning multiple values to multiple variables:**

**Eg:**

a,b,c=5,10,15

print a

print b

print c

**Output:**

5

10

15

The values will be assigned in the order in which variables appear.

**Basic Fundamentals:-**

This section contains the fundamentals of Python, such as:

**i)Tokens and their types.**

**ii) Comments**

**a)Tokens:**

* The tokens can be defined as a punctuator mark, reserved words, and each word in a statement.
* The token is the smallest unit inside the given program.

There are following tokens in Python:

* Keywords.
* Identifiers.
* Literals.
* Operators.

We will discuss above the tokens in detail next tutorials.