Python Variables –

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

Identifier 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 variable before using in the application. It allows us to create variable at 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.**

Multiple Assignments –

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.

- **Assigning single value to multiple variables –**

```
x=y=z=50
print (x)
print (y)
print (z)
```

- Assigning multiple values to multiple variables –

```
a,b,c=5,10,15
print (a)
print (b)
print (c)
```

Basic Fundamentals –

This section contains the basic fundamentals of Python like –

1. Tokens and their types
2. Comments

Tokens -

Tokens can be defined as a punctuator mark, reserved words and each individual word in a statement.Token is the smallest unit inside the given program.

There are following tokens in Python:

- ✓ Keywords
- ✓ Identifiers
- ✓ Literals
- ✓ Operators

Python Data Types –

Variables can hold values of different data types. **Python is a dynamically typed language** hence we need not define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.Python enables us to check the type of the variable used in the program. **Python provides us the *type()* function which returns the type of the variable passed.**

e.g,

```
A=10
b="Hi Python"
print(type(a));
print(type(b));
```

Standard data types -

A variable can hold different types of values. Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Numbers
2. String
3. List
4. Tuple
5. Dictionary

Python Keywords –

Python Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as variable.

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

Python Literals -

Literals can be defined as a data that is given in a variable or constant.Python support the following literals –

- **String literals:** String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes for a String.

  e.g. "Aman" , '12345'

  - **Types of Strings:**There are two types of Strings supported in Python -
    - **Single line String-** Strings that are terminated within a single line are known as Single line Strings.

      e.g. text1='hello'
    - **Multi line String -** A piece of text that is spread along multiple lines is known as multiple line String.There are two ways to create Multiline Strings –

      ✓ **Adding black slash at the end of each line –**

      text1='hello\

```
user'
```

- ✓ Using triple quotation marks -

```
c = '''Welcome
to GGI'''
```

- **Numeric Literals** – Numeric Literals are **immutable**. Numeric literals can belong to following four different numerical types.
  - **Int (Signed Integers)** – Numbers(can be both positive and negative) with no fractional part. e.g.: 100
  - **Long (Long Integers)** -Integers of unlimited size followed by lowercase or uppercase L e.g.: 87032845L
  - **Float (Floating Point)** -Real numbers with both integer and fractional part e.g.: 26.2
  - **Complex (complex)** -In the form of a+bj where a forms the real part and b forms the imaginary part of complex number. e.g.: 3.14j
- **Boolean Literals** -A Boolean literal can have any of the two values: True or False.
- **Special Literals** - Python contains one special literal i.e., **None**. None is used to specify to that field that is not created. It is also used for end of lists in Python.

```
g = 10
h = None
print (g)
print (h)
```

- **Literal Collections** - Collections such as tuples, lists and Dictionary are used in Python.

Python Operators –

The operator can be defined as a symbol which is responsible for a particular operation between two operands. Operators are the pillars of a program on which the logic is built in a particular programming language.Python provides a variety of operators described as follows.

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operators –

Operator	Description
+ (Addition)	It is used to add two operands. For example, if a = 20, b = 10 => a+b = 30
- (Subtraction)	It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value result negative. For example, if a = 20, b = 10 => a - b = 10
/ (divide)	It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a/b = 2
*** (Multiplication)**	It is used to multiply one operand with the other. For example, if a = 20, b = 10 => a * b = 200
% (reminder)	It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a%b = 0
**** (Exponent)**	It is an exponent operator represented as it calculates the first operand power to second operand.
// (Floor division)	It gives the floor value of the quotient produced by dividing the two operands.

Comparison Operators –

Operator	Description
==	If the value of two operands is equal, then the condition becomes true.
!=	If the value of two operands is not equal then the condition becomes true.
<=	If the first operand is less than or equal to the second operand, then the condition becomes true.
>=	If the first operand is greater than or equal to the second operand, then the condition becomes true.
>	If the first operand is greater than the second operand, then the condition becomes true.
<	If the first operand is less than the second operand, then the condition becomes true.

Assignment Operators -

Operator	Description
=	It assigns the the value of the right expression to the left operand.
+=	It increases the value of the left operand by the value of the right operand and assign the modified value back to left operand. For example, if a = 10, b = 20 => a+ = b will be equal to a = a+ b and therefore, a = 30.
-=	It decreases the value of the left operand by the value of the right operand and assign the modified value back to left operand. For example, if a = 20, b = 10 => a- = b will be equal to a = a- b and therefore, a = 10.
*=	It multiplies the value of the left operand by the value of the right operand and assign the modified value back to left operand. For example, if a = 10, b = 20 => a* = b will be equal to a = a* b and therefore, a = 200.
%=	It divides the value of the left operand by the value of the right operand and assign the reminder back to left operand. For example, if a = 20, b = 10 => a % = b will be equal to a = a % b and therefore, a = 0.

****=**	a**=b will be equal to a=a**b, for example, if a = 4, b =2, a**=b will assign 4**2 = 16 to a.
//=	A//=b will be equal to a = a// b, for example, if a = 4, b = 3, a//=b will assign 4//3 = 1 to a.

Bitwise Operator –

Operator	Description
& (binary and)	If both the bits at the same place in two operands are 1, then 1 is copied to the result. Otherwise, 0 is copied.
| (binary or)	The resulting bit will be 0 if both the bits are zero otherwise the resulting bit will be 1.
^ (binary xor)	The resulting bit will be 1 if both the bits are different otherwise the resulting bit will be 0.
~ (negation)	It calculates the negation of each bit of the operand, i.e., if the bit is 0, the resulting bit will be 1 and vice versa.
<< (left shift)	The left operand value is moved left by the number of bits present in the right operand.
>> (right shift)	The left operand is moved right by the number of bits present in the right operand.

Logical Operators –

Operator	Description
and	If both the expression are true, then the condition will be true. If a and b are the two expressions, a → true, b → true => a and b → true.
or	If one of the expressions is true, then the condition will be true. If a and b are the two expressions, a → true, b → false => a or b → true.
not	If an expression **a** is true then not (a) will be false and vice versa.

Membership Operators –

Operator	Description
In	It is evaluated to be true if the first operand is found in the second operand (list, tuple, or dictionary).
not in	It is evaluated to be true if the first operand is not found in the second operand (list, tuple, or dictionary).

Identity Operators –

Operator	Description
is	It is evaluated to be true if the reference present at both sides point to the same object.
is not	It is evaluated to be true if the reference present at both side do not point to the same object.

Operator Precedence –

Operator	Description
**	The exponent operator is given priority over all the others used in the expression.
~ + -	The negation, unary plus, and minus.
* / % //	The multiplication, divide, modules, reminder, and floor division.
+ -	Binary plus, and minus
>><<	Left shift. and right shift
&	Binary and.
^ |	Binary xor, and or
<= <>>=	Comparison operators (less than, less than equal to, greater than, greater then equal to).
<> == !=	Equality operators.
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Python Comments –

- Comments in Python can be used to explain any program code. It can also be used to hide the code as well.
- Comments are the most helpful stuff of any program. It enables us to understand the way, a program works. In python, any statement written along with # symbol is known as a comment. The interpreter does not interpret the comment.
- Comment is not a part of the program, but it enhances the interactivity of the program and makes the program readable.

Python supports two types of comments –

1. Single Line Comment –

```
# This is single line comment.
print "Hello Python"
```

2. Multi Line Comment –

```
''' This
    Is
    Multipline comment'''
```