# ASSINMENT-2

# 1.what are the data types in python? explain.

# Python Data Types

Data types are the classification or categorization of data items. Data types represent a kind of value which determines what operations can be performed on that data. Numeric, non-numeric and Boolean (true/false) data are the most used data types. However, each programming language has its own classification largely reflecting its programming philosophy.

Python has the following standard or built-in data types:

## Numeric

A numeric value is any representation of data which has a numeric value. Python identifies three types of numbers:

* Integer: Positive or negative whole numbers (without a fractional part)
* Float: Any real number with a floating point representation in which a fractional component is denoted by a decimal symbol or scientific notation
* Complex number: A number with a real and imaginary component represented as x+yj. x and y are floats and j is -1(square root of -1 called an imaginary number)

## Boolean

Data with one of two built-in values True or False. Notice that 'T' and 'F' are capital. true and false are not valid booleans and Python will throw an error for them.

## Sequence Type

A sequence is an ordered collection of similar or different data types. Python has the following built-in sequence data types:

* String: A string value is a collection of one or more characters put in single, double or triple quotes.
* List : A list object is an ordered collection of one or more data items, not necessarily of the same type, put in square brackets.
* Tuple: A Tuple object is an ordered collection of one or more data items, not necessarily of the same type, put in parentheses.

## Dictionary

A dictionary object is an unordered collection of data in a key:value pair form. A collection of such pairs is enclosed in curly brackets. For example: {1:"Steve", 2:"Bill", 3:"Ram", 4: "Farha"}

### type() function

Python has an in-built function type() to ascertain the data type of a certain value. For example, enter type(1234) in Python shell and it will return <class 'int'>, which means 1234 is an integer value. Try and verify the data type of different values in Python shell, as shown below.

# Briefly explain history of python?

[Python](https://www.geeksforgeeks.org/python-programming-language/) is a widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

# Explain operators in python?

Python language supports the following types of operators.

* Arithmetic Operators
* Comparison (Relational) Operators
* Assignment Operators
* Logical Operators
* Bitwise Operators
* Membership Operators
* Identity Operators

Let us have a look on all operators one by one.

## Python Arithmetic Operators

Assume variable a holds 10 and variable b holds 20, then −

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| + Addition | Adds values on either side of the operator. | a + b = 30 |
| - Subtraction | Subtracts right hand operand from left hand operand. | a – b = -10 |
| \* Multiplication | Multiplies values on either side of the operator | a \* b = 200 |
| / Division | Divides left hand operand by right hand operand | b / a = 2 |
| % Modulus | Divides left hand operand by right hand operand and returns remainder | b % a = 0 |
| \*\* Exponent | Performs exponential (power) calculation on operators | a\*\*b =10 to the power 20 |
| // | Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) − | 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0 |

## Python Comparison Operators

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.

Assume variable a holds 10 and variable b holds 20, then −

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| == | If the values of two operands are equal, then the condition becomes true. | (a == b) is not true. |
| != | If values of two operands are not equal, then condition becomes true. | (a != b) is true. |
| <> | If values of two operands are not equal, then condition becomes true. | (a <> b) is true. This is similar to != operator. |
| > | If the value of left operand is greater than the value of right operand, then condition becomes true. | (a > b) is not true. |
| < | If the value of left operand is less than the value of right operand, then condition becomes true. | (a < b) is true. |
| >= | If the value of left operand is greater than or equal to the value of right operand, then condition becomes true. | (a >= b) is not true. |
| <= | If the value of left operand is less than or equal to the value of right operand, then condition becomes true. | (a <= b) is true. |

## Python Assignment Operators

Assume variable a holds 10 and variable b holds 20, then −

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| = | Assigns values from right side operands to left side operand | c = a + b assigns value of a + b into c |
| += Add AND | It adds right operand to the left operand and assign the result to left operand | c += a is equivalent to c = c + a |
| -= Subtract AND | It subtracts right operand from the left operand and assign the result to left operand | c -= a is equivalent to c = c - a |
| \*= Multiply AND | It multiplies right operand with the left operand and assign the result to left operand | c \*= a is equivalent to c = c \* a |
| /= Divide AND | It divides left operand with the right operand and assign the result to left operand | c /= a is equivalent to c = c / a |
| %= Modulus AND | It takes modulus using two operands and assign the result to left operand | c %= a is equivalent to c = c % a |
| \*\*= Exponent AND | Performs exponential (power) calculation on operators and assign value to the left operand | c \*\*= a is equivalent to c = c \*\* a |
| //= Floor Division | It performs floor division on operators and assign value to the left operand | c //= a is equivalent to c = c // a |

## Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; Now in the binary format their values will be 0011 1100 and 0000 1101 respectively. Following table lists out the bitwise operators supported by Python language with an example each in those, we use the above two variables (a and b) as operands −

a = 0011 1100

b = 0000 1101

-----------------

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a  = 1100 0011

There are following Bitwise operators supported by Python language

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| & Binary AND | Operator copies a bit to the result if it exists in both operands | (a & b) (means 0000 1100) |
| | Binary OR | It copies a bit if it exists in either operand. | (a | b) = 61 (means 0011 1101) |
| ^ Binary XOR | It copies the bit if it is set in one operand but not both. | (a ^ b) = 49 (means 0011 0001) |
| ~ Binary Ones Complement | It is unary and has the effect of 'flipping' bits. | (~a ) = -61 (means 1100 0011 in 2's complement form due to a signed binary number. |
| << Binary Left Shift | The left operands value is moved left by the number of bits specified by the right operand. | a << 2 = 240 (means 1111 0000) |
| >> Binary Right Shift | The left operands value is moved right by the number of bits specified by the right operand. | a >> 2 = 15 (means 0000 1111) |

## Python Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| and Logical AND | If both the operands are true then condition becomes true. | (a and b) is true. |
| or Logical OR | If any of the two operands are non-zero then condition becomes true. | (a or b) is true. |
| not Logical NOT | Used to reverse the logical state of its operand. | Not(a and b) is false. |

## Python Membership Operators

Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below –

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| in | Evaluates to true if it finds a variable in the specified sequence and false otherwise. | x in y, here in results in a 1 if x is a member of sequence y. |
| not in | Evaluates to true if it does not finds a variable in the specified sequence and false otherwise. | x not in y, here not in results in a 1 if x is not a member of sequence y. |

## Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below −

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| is | Evaluates to true if the variables on either side of the operator point to the same object and false otherwise. | x is y, here is results in 1 if id(x) equals id(y). |
| is not | Evaluates to false if the variables on either side of the operator point to the same object and true otherwise. | x is not y, here is not results in 1 if id(x) is not equal to id(y). |

## Python Operators Precedence

The following table lists all operators from highest precedence to lowest.

|  |  |
| --- | --- |
|  |  |
| 1 | \*\*  Exponentiation (raise to the power) |
| 2 | ~ + -  Complement, unary plus and minus (method names for the last two are +@ and -@) |
| 3 | \* / % //  Multiply, divide, modulo and floor division |
| 4 | + -  Addition and subtraction |
| 5 | >> <<  Right and left bitwise shift |
| 6 | &  Bitwise 'AND' |
| 7 | ^ |  Bitwise exclusive `OR' and regular `OR' |
| 8 | <= < > >=  Comparison operators |
| 9 | <> == !=  Equality operators |
| 10 | = %= /= //= -= += \*= \*\*=  Assignment operators |
| 11 | is is not  Identity operators |
| 12 | in not in  Membership operators |
| 13 | not or and  Logical operators |

explain the features of python?

#### 1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

#### 2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

#### 3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

#### 4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

#### 5) Free and Open Source

Python language is freely available at [offical web address](https://www.python.org/).The source-code is also available. Therefore it is open source.

#### 6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

#### 7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

#### 8) Large Standard Library

Python has a large and broad library and prvides rich set of module and functions for rapid application development.

#### 9) GUI Programming Support

Graphical user interfaces can be developed using Python.

#### 10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

justify why python is interactive interpreted language?

interpreted

Unlike C/C++ etc, Python is an interpreted object-oriented programming language. By interpreted it is meant that each time a program is run the interpreter checks through the code for errors and then interprets the instructions into machine-readable bytecode.

An interpreter is a translator in computer's language which translates the given code line-by-line in machine readable bytecodes. And if any error is encounterd it stops the translation until the error is fixed. Unlike C language, which is a compiled programming language. The compiler translates the whole code in one-go rather than line-by-line. This is the reason why in C language, all the errors are listed during compilation only.

## Interactive Python

Python is interactive. When a Python statement is entered, and is followed by the Return key, if appropriate, the result will be printed on the screen, immediately, in the next line. This is particularly advantageous in the debugging process. In interactive mode of operation, Python is used in a similar way as the Unix command line or the terminal.

Interactive Python is very much helpful for the debugging purpose. It simply returns the >>> prompt or the corresponding output of the statement if appropriate and returns error for incorrect statements. In this way if you have any doubts like: whether a syntax is correct, whether the module you are importing exists or anything like that, you can be sure within seconds using Python interactive mode.