What is Python

**Python** is an object-oriented(OOPS), high level language, interpreted, dynamic and multipurpose programming language.

Python is *easy to learn* yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and *dynamic typing* with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas.

Python supports *multiple programming pattern*, including object oriented programming, imperative and functional programming or procedural styles.

Python is not intended to work on special area such as web programming. That is why it is known as *multipurpose* because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is *dynamically typed* so we can write a=10 to declare an integer value in a variable.

Python makes the development and debugging *fast* because there is no compilation step included in python development and edit-test-debug cycle is very fast.

Python Features

There are a lot of features provided by python programming language.

1) Easy to Use:

Python is easy to very easy to use and high level language. Thus it is programmer-friendly language.

2) Expressive Language:

Python language is more expressive. The sense of expressive is the code is easily understandable.

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 , Macintosh etc. Thus, Python is a portable language.

5) Free and Open Source:

Python language is freely available(www.python.org).The source-code is also available. Therefore it is open source.

6) Object-Oriented language:

Python supports object oriented language. Concept of classes and objects comes 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 your python code.

8) Large Standard Library:

Python has a large and broad library.

9) GUI Programming:

Graphical user interfaces can be developed using Python.

10) Integrated:

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

Python History

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
* *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by programming languages like:
  + ABC language.
  + Modula-3
* Python Version
* Python programming language is being updated regularly with new features and support. There are a lot of updation in python versions, started from 1994 to current date.

|  |  |
| --- | --- |
| **Python Version** | **Released Date** |
| Python 1.0 | January 1994 |
| Python 1.5 | December 31, 1997 |
| Python 1.6 | September 5, 2000 |
| Python 2.0 | October 16, 2000 |
| Python 2.1 | April 17, 2001 |
| Python 2.2 | December 21, 2001 |
| Python 2.3 | July 29, 2003 |
| Python 2.4 | November 30, 2004 |
| Python 2.5 | September 19, 2006 |
| Python 2.6 | October 1, 2008 |
| Python 2.7 | July 3, 2010 |
| Python 3.0 | December 3, 2008 |
| Python 3.1 | June 27, 2009 |
| Python 3.2 | February 20, 2011 |
| Python 3.3 | September 29, 2012 |

* A list of python versions with its released date is given below.

Python Applications

Python as a whole can be used in any sphere of development.

Let us see what are the major regions where Python proves to be handy.

1) Console Based Application

Python can be used to develop console based applications. For example: **IPython**.

2) Audio or Video based Applications

Python proves handy in multimedia section. Some of real applications are: TimPlayer, cplay etc.

3) 3D CAD Applications

Fandango is a real application which provides full features of CAD.

4) Web Applications

Python can also be used to develop web based application. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

5) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

6) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

Python Example

Python code is simple and easy to run. Here is a simple Python code that will print "Welcome to Python".

A simple python example is given below.

1. >>> a="Welcome To Python"
2. >>> **print** a
3. Welcome To Python
4. >>>

**Explanation:**

* Here we are using IDLE to write the Python code. Detail explanation to run code is given in Execute Python section.
* A variable is defined named "a" which holds "Welcome To Python".
* "print" statement is used to print the content. Therefore "print a" statement will print the content of the variable. Therefore, the output "Welcome To Python" is produced.

Python Variables

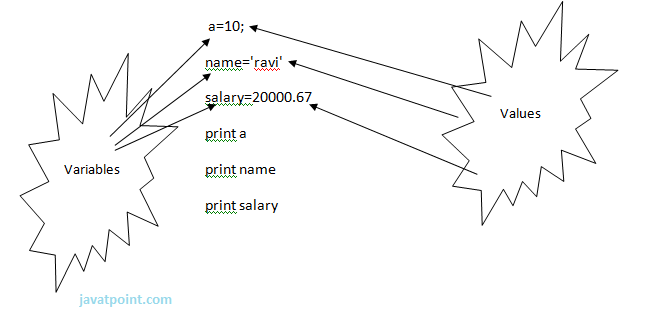
Variable is a name of the memory location where data is stored. Once a variable is stored that means a space is allocated in memory.

Assigning values to Variable:

We need not to declare explicitly variable in Python. When we assign any value to the variable that variable is declared automatically.

The assignment is done using the equal (=) operator.

**Eg:**



**Output:**

1. >>>
2. 10
3. ravi
4. 20000.67
5. >>>

Multiple Assignment:

Multiple assignment can be done in Python at a time.

There are two ways to assign values in Python:

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

**Eg:**

1. x=y=z=50
2. **print** x
3. **print** y
4. **print** z

**Output:**

1. >>>
2. 50
3. 50
4. 50
5. >>>

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

**Eg:**

1. a,b,c=5,10,15
2. **print** a
3. **print** b
4. **print** c

**Output:**

1. >>>
2. 5
3. 10
4. 15
5. >>>

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

Basic Fundamentals:

This section contains the basic fundamentals of Python like :

**i)Tokens and their types.**

**ii) Comments**

**a)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.

Tuples:

* Tuple is another form of collection where different type of data can be stored.
* It is similar to list where data is separated by commas. Only the difference is that list uses square bracket and tuple uses parenthesis.
* Tuples are enclosed in parenthesis and cannot be changed.

**Eg:**

>>> tuple=('rahul',100,60.4,'deepak')

>>> tuple1=('sanjay',10)

>>> tuple

('rahul', 100, 60.4, 'deepak')

>>> tuple[2:]

(60.4, 'deepak')

>>> tuple1[0]

'sanjay'

>>> tuple+tuple1

('rahul', 100, 60.4, 'deepak', 'sanjay', 10)

>>>

Dictionary:

* Dictionary is a collection which works on a key-value pair.
* It works like an associated array where no two keys can be same.
* Dictionaries are enclosed by curly braces ({}) and values can be retrieved by square bracket([]).

**Eg:**

>>> dictionary={'name':'charlie','id':100,'dept':'it'}

>>> dictionary

{'dept': 'it', 'name': 'charlie', 'id': 100}

>>> dictionary.keys()

['dept', 'name', 'id']

>>> dictionary.values()

['it', 'charlie', 100]

>>>

1. Python Keywords
2. Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword have a special meaning and a specific operation. List of Keywords used in Python are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

Identifiers

Identifiers are the names given to the fundamental building blocks in a program.

These can be variables ,class ,object ,functions , lists , dictionaries etc.

There are certain rules defined for naming i.e., Identifiers.

I. An identifier is a long sequence of characters and numbers.

II.No special character except underscore ( \_ ) can be used as an identifier.

III.Keyword should not be used as an identifier name.

IV.Python is case sensitive. So using case is significant.

V.First character of an identifier can be character, underscore ( \_ ) but not digit.

Python Literals

Literals can be defined as a data that is given in a variable or constant.

Python support the following literals:

**I. 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.

**Eg:**

"Aman" , '12345'

**Types of Strings:**

There are two types of Strings supported in Python:

a).Single line String- Strings that are terminated within a single line are known as Single line Strings.

**Eg:**

1. >>> text1='hello'

b).Multi line String- A piece of t ext that is spread along multiple lines is known as Multiple line String.

There are two ways to create Multiline Strings:

**1). Adding black slash at the end of each line.**

**Eg:**

1. >>> text1='hello\
2. user'
3. >>> text1
4. 'hellouser'
5. >

**2).Using triple quotation marks:-**

**Eg:**

1. >>> str2='''''welcome
2. to
3. SSSIT'''
4. >>> **print** str2
5. welcome
6. to
7. SSSIT
8. >>>

**II.Numeric literals:**

Numeric Literals are immutable. Numeric literals can belong to following four different numerical types.

|  |  |  |  |
| --- | --- | --- | --- |
| **Int(signed integers)** | **Long(long integers)** | **float(floating point)** | **Complex(complex)** |
| Numbers( can be both positive and negative) with no fractional part.eg: 100 | Integers of unlimited size followed by lowercase or uppercase L eg: 87032845L | Real numbers with both integer and fractional part eg: -26.2 | In the form of a+bj where a forms the real part and b forms the imaginary part of complex number. eg: 3.14j |

**III. Boolean literals:**

A Boolean literal can have any of the two values: True or False.

**IV. 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.

Eg:

1. >>> val1=10
2. >>> val2=None
3. >>> val1
4. 10
5. >>> val2
6. >>> **print** val2
7. None
8. >>>

**V.Literal Collections.**

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

**List:**

* List contain items of different data types. Lists are mutable i.e., modifiable.
* The values stored in List are separated by commas(,) and enclosed within a square brackets([]). We can store different type of data in a List.
* Value stored in a List can be retrieved using the slice operator([] and [:]).
* The plus sign (+) is the list concatenation and asterisk(\*) is the repetition operator.

**Eg:**

1. >>> list=['aman',678,20.4,'saurav']
2. >>> list1=[456,'rahul']
3. >>> list
4. ['aman', 678, 20.4, 'saurav']
5. >>> list[1:3]
6. [678, 20.4]
7. >>> list+list1
8. ['aman', 678, 20.4, 'saurav', 456, 'rahul']
9. >>> list1\*2
10. [456, 'rahul', 456, 'rahul']
11. >>>

Python Operators

Operators are particular symbols which operate on some values and produce an output.

The values are known as Operands.

**Eg:**

1. 4 + 5 = 9

Here 4 and 5 are Operands and (+) , (=) signs are the operators. They produce the output 9.

**Python supports the following operators:**

1. Arithmetic Operators.
2. Relational Operators.
3. Assignment Operators.
4. Logical Operators.
5. Membership Operators.
6. Identity Operators.

**Arithmetic Operators:**

|  |  |
| --- | --- |
| **Operators** | **Description** |
| // | Perform Floor division(gives integer value after division) |
| + | To perform addition |
| - | To perform subtraction |
| \* | To perform multiplication |
| / | To perform division |
| % | To return remainder after division(Modulus) |
| \*\* | Perform exponent(raise to power) |

**eg:**

1. >>> 10+20
2. 30
3. >>> 20-10
4. 10
5. >>> 10\*2
6. 20
7. >>> 10/2
8. 5
9. >>> 10%3
10. 1
11. >>> 2\*\*3
12. 8
13. >>> 10//3
14. 3
15. >>>

**Relational Operators:**

|  |  |
| --- | --- |
| **Operators** | **Description** |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| == | Equal to |
| != | Not equal to |
| <> | Not equal to(similar to !=) |

**eg:**

1. >>> 10<20
2. True
3. >>> 10>20
4. False
5. >>> 10<=10
6. True
7. >>> 20>=15
8. True
9. >>> 5==6
10. False
11. >>> 5!=6
12. True
13. >>> 10<>2
14. True
15. >>>

**Assignment Operators:**

|  |  |
| --- | --- |
| **Operators** | **Description** |
| = | Assignment |
| /= | Divide and Assign |
| += | Add and assign |
| -= | Subtract and Assign |
| \*= | Multiply and assign |
| %= | Modulus and assign |
| \*\*= | Exponent and assign |
| //= | Floor division and assign |

**eg:**

1. >>> c=10
2. >>> c
3. 10
4. >>> c+=5
5. >>> c
6. 15
7. >>> c-=5
8. >>> c
9. 10
10. >>> c\*=2
11. >>> c
12. 20
13. >>> c/=2
14. >>> c
15. 10
16. >>> c%=3
17. >>> c
18. 1
19. >>> c=5
20. >>> c\*\*=2
21. >>> c
22. 25
23. >>> c//=2
24. >>> c
25. 12
26. >>>

**Logical Operators:**

|  |  |
| --- | --- |
| **Operators** | **Description** |
| And | Logical AND(When both conditions are true output will be true) |
| Or | Logical OR (If any one condition is true output will be true) |
| not | Logical NOT(Compliment the condition i.e., reverse) |

**eg:**

1. a=5>4 **and** 3>2
2. **print** a
3. b=5>4 **or** 3<2
4. **print** b
5. c=**not**(5>4)
6. **print** c

**Output:**

1. >>>
2. True
3. True
4. False
5. >>>

**Membership Operators:**

|  |  |
| --- | --- |
| **Operators** | **Description** |
| In | Returns true if a variable is in sequence of another variable, else false. |
| not in | Returns true if a variable is not in sequence of another variable, else false. |

**eg:**

a=10

b=20

list=[10,20,30,40,50];

**if** (a **in** list):

**print** "a is in given list"

**else**:

**print** "a is not in given list"

**if**(b **not** **in** list):

**print** "b is not given in list"

**else**:

**print** "b is given in list"

**Output:**

1. >>>
2. a **is** **in** given list
3. b **is** given **in** list
4. >>>

**Identity Operators:**

|  |  |
| --- | --- |
| **Operators** | **Description** |
| Is | Returns true if identity of two operands are same, else false |
| is not | Returns true if identity of two operands are not same, else false. |

**Example:**

a=20

b=20

**if**( a **is** b):

**print**  ?a,b have same identity?

**else**:

**print** ?a, b are different?

b=10

**if**( a **is** **not** b):

**print**  ?a,b have different identity?

**else**:

**print** ?a,b have same identity?

**Output:**

1. >>>
2. a,b have same identity
3. a,b have different identity
4. >>>

Python Comments

Python supports two types of comments:

**1) Single lined comment:**

In case user wants to specify a single line comment, then comment must start with ?#?

**Eg:**

1. # This is single line comment.

**2) Multi lined Comment:**

Multi lined comment can be given inside triple quotes.

**eg:**

1. ''' This
2. Is
3. Multipline comment'''

**eg:**

1. #single line comment
2. **print** "Hello Python"
3. '''''This is
4. multiline comment'''