## What is Python Programming

**Python** is a widely used high-level programming language used for general-purpose programming. Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). An interpreted language, Python has a design philosophy which emphasizes code readability, and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

* Python is a **general-purpose interpreted, interactive, object-oriented, and high-level programming** language. It was created by **Guido van Rossum** during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).
* **First Release – 1991**
* Python is designed to be **highly readable**. It uses **English keywords frequently** where as other languages use punctuation, and it has fewer syntactical constructions than other languages.
* **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

### History of Python

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

#### Release Dates of Different Versions

**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

Python 3.4 - March 16, 2014

Python 3.5 - September 13, 2015

Python 3.6 - December 23, 2016

### Features of Python Programming

A simple language which is easier to learn  
Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

Free and open-source  
You can freely use and distribute Python, even for commercial use. Not only can you use and distribute softwares written in it, you can even make changes to the Python's source code.  
Python has a large community constantly improving it in each iteration.

Portability  
You can move Python programs from one platform to another, and run it without any changes.  
It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

Extensible and Embeddable  
Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.  
This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

A high-level, interpreted language  
Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.  
Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

Large standard libraries to solve common tasks  
Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .  
Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

Object-oriented  
Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.  
With OOP, you are able to divide these complex problems into smaller sets by creating objects

### Applications of Python

Web Applications

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS.

Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

Scientific and Numeric Computing

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on.

Also, the language is heavily used in machine learning, data mining and deep learning.

Creating software Prototypes

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must.

However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

Good Language to Teach Programming

Python is used by many companies to teach programming to kids and newbies.

It is a good language with a lot of features and capabilities. Yet, it's one of the easiest language to learn because of its simple easy-to-use syntax.

## Reasons to Choose Python as First Language

Simple Elegant Syntax  
  
Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

Even if you have never programmed before, you can easily guess that this program adds two numbers and prints it.

Not overly strict  
You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.  
  
Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

Expressiveness of the language  
Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of [Tic-tac-toe game](http://pastebin.com/7LTkj2V5) with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

Great Community and Support  
  
Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

## Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9). It helps differentiating one entity from another.

Naming conventions for Python identifiers −

* Class names start with an uppercase letter. All other identifiers start with a
* lowercase letter.
* Starting an identifier with a single leading underscore indicates that the identifier is private.
* Starting an identifier with two leading underscores indicates a strongly private identifier.
* If the identifier also ends with two trailing underscores, the identifier is a languagedefined special name.
* Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (\_). Names like myClass, var\_1 and print\_this\_to\_screen, all are valid example.
* An identifier cannot start with a digit. 1variable is invalid, but variable1 is perfectly fine.
* Keywords cannot be used as identifiers. The python keywords are

|  |  |  |
| --- | --- | --- |
| and | exec | raise |
| not | assert | def |
| or | finally | if |
| break | for | return |
| pass | class | del |
| from | print | import |
| continue | global | try |
| elif | else | with |
| in | is | except |
| while | lambda | yield |

Example

|  |
| --- |
| >>> global = 1  File "<interactive input>", line 1  global = 1  ^  SyntaxError: invalid syntax |

* We cannot use special symbols like !, @, #, $, % etc. in our identifier.

Example

|  |
| --- |
| >>> a@ = 0  File "<interactive input>", line 1  a@ = 0  ^  SyntaxError: invalid syntax |

* Identifier can be of any length.
* Python is a case-sensitive language. This means, Variable and variable are not the same. Always name identifiers that make sense.
* While, c = 10 is valid. Writing count = 10 would make more sense and it would be easier to figure out what it does even when you look at your code after a long gap.
* Multiple words can be separated using an underscore, this\_is\_a\_long\_variable.
* We can also use camel-case style of writing, i.e., capitalize every first letter of the word except the initial word without any spaces.