PYTHON
programming language

tutorialspoint

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

Today, Python is one of the most popular programming languages. Although it is a general-purpose language, it is used in various areas of applications such as Machine Learning, Artificial Intelligence, web development, IoT, and more.

This Python tutorial has been written for the beginners to help them understand the basic to advanced concepts of Python Programming Language. After completing this tutorial, you will find yourself at a great level of expertise in Python, from where you can take yourself to the next levels to become a world class Software Engineer.

This Python tutorial is based on the latest Python 3.11.2 version.

What is Python?

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-typed and garbage-collected 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).

Python supports multiple programming paradigms, including Procedural, Object Oriented and Functional programming language. Python design philosophy emphasizes code readability with the use of significant indentation.

This tutorial gives a complete understanding of Python programming language, starting from basic concepts to advanced concepts. This tutorial will take you through simple and practical approaches while learning Python Programming language.

Python Jobs

Today, Python is very high in demand, and all the major companies are looking for great Python programmers to develop websites, software components, and applications or to work with data science, AI, and ML technologies. When we were developing this tutorial in 2022, there was a high shortage of Python programmers, where the market demanded a greater number of Python programmers due to its applications in machine learning, artificial intelligence, etc.

Today, a Python programmer with 3-5 years of experience is asking for around $150,000 in an annual package, and this is the most demanding programming language in America. Though it can vary depending on the location of the job. It's impossible to list all of the companies using Python, to name a few big companies are:

- Google
- Intel
- NASA
- PayPal
- Facebook
- IBM
- Amazon
- Netflix
- Pinterest
- Uber
- Many more...

So, you could be the next potential employee for any of these major companies. We have developed great learning material for you to learn Python programming, which will help you prepare for the technical interviews and certification exams based on Python. So, start learning Python using this simple and effective tutorial from anywhere and anytime, absolutely at your pace.

Why to Learn Python?

Python is consistently rated as one of the world's most popular programming languages. Python is fairly easy to learn, so if you are starting to learn any programming language, then Python could be your great choice. Today, various schools, colleges, and universities are teaching Python as their primary programming language. There are many other good reasons that make Python the top choice of any programmer:

- Python is open source, which means it's available free of cost.
- Python is simple and easy to learn.
- Python is versatile and can be used to create different kinds of applications.
- Python has powerful development libraries, including AI, ML, etc.
- Python is much in demand and ensures a high salary.

Python is a MUST for students and working professionals to become great software engineers, especially when they are working in the web development domain. I will list down some of the key advantages of learning Python:

- **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.

Python "Hello, World!"

To start with Python programming, the very basic program is to print "Hello, World!" You can use the print() function. Below is an example of Python code to print "Hello, World!" –

```
# Python code to print "Hello, World!"

print ("Hello, World!")
```

Python Online Compiler

Our Python programming tutorial provides various examples to explain different concepts. We have provided Online Python Compiler/Interpreter. You can Edit and Execute almost all the examples directly from your browser without the need to set up your development environment.

Try to click the icon ⚙ to run the following Python code to print conventional "Hello, World!".

Below code box allows you to change the value of the code. Try to change the value inside print() and run it again to verify the result.

```
# This is my first Python program.

# This will print 'Hello, World!' as the output


print ("Hello, World!");
```

Careers with Python

If you know Python nicely, then you have a great career ahead. Here are just a few of the career options where Python is a key skill:

- Game developer
- Web designer
- Python developer
- Full-stack developer
- Machine learning engineer
- Data scientist
- Data analyst
- Data engineer
- DevOps engineer
- Software engineer
- Many more other roles

Characteristics of Python

Following are important characteristics of **Python Programming** –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Applications of Python

Python is a general purpose programming language known for its readability. It is widely applied in various fields.

- In Data Science, Python libraries like Numpy, Pandas, and Matplotlib are used for data analysis and visualization.
- Python frameworks like Django, and Pyramid, make the development and deployment of Web Applications easy.
- This programming language also extends its applications to **computer vision** and image processing.
- It is also favored in many tasks like Automation, Job Scheduling, GUI development, etc.

Features of Python

The latest release of Python is 3.x. As mentioned before, Python is one of the most widely used languages on the web. I'm going to list a few of them here:

- **Easy-to-learn –** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read –** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain –** Python's source code is fairly easy-to-maintain.
- **A broad standard library –** Python has a bulk of portable and cross-platform libraries and they are compatible with UNIX, Windows, and Macintosh.
- **Interactive Mode –** Python has support for an interactive mode that allows interactive testing and debugging of snippets of code.
- **Portable –** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable –** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases –** Python provides interfaces to all major commercial databases.
- **GUI Programming –** Python supports GUI applications that can be created and ported to many system calls, libraries, and operating systems, such as Windows, MFC, Macintosh, and the X Window system of Unix.
- **Scalable –** Python provides a better structure and support for large programs than shell scripting.

Python Reference

The complete function and method references –

- Python Complete Reference
- Python Built-in Functions Reference
- Python Modules Reference
- Python Keywords Reference
- Python Cheatsheet

Python Practice

Practice Python from the below-given links:

- Python Quick Guide
- Python Online Quiz
- Python Interview Questions & Answers

Download Python

You can download Python from its official website: https://www.python.org/downloads/

Target Audience

This tutorial has been prepared for the beginners to help them understand the basics to advanced concepts of Python programming language. After completing this tutorial, you will find yourself at a great level of expertise in Python programming, from where you can take yourself to the next levels.

Prerequisites

Although it is a beginner's tutorial, we assume that the readers have a reasonable exposure to any programming environment and knowledge of basic concepts such as variables, commands, syntax, etc.

Python Questions & Answers

You can explore a set of Python Questions and Answers at **Python Questions & Answers**

Copyright & Disclaimer

© Copyright 2025 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial i

What is Python? i

Python Reference iv

Python Practice iv

Download Python iv

Target Audience iv

Prerequisites v

Python Questions & Answers v

Table of Contents vi

PYTHON BASICS 1

1. Python – Overview 2

2. Python - History and Versions 4

3. Python - Features 7

4. Python vs C++ 11

5. Python - Hello World Program 17

6. Python - Application Areas 20

7. Python Interpreter and Its Modes 23

8. Python - Environment Setup 28

9. Python - Virtual Environment 37

10. Python - Syntax 40

11. Python - Variables 47

12. Python - Data Types 57

13. Python - Type Casting 75

14. Python - Unicode System 85

15. Python - Literals 88

16. Python - Operators 94

17. **Python - Arithmetic Operators** 104

18. **Python - Comparison Operators** 115

19. **Python - Assignment Operators** 121

20. **Python - Logical Operators** 128

21. **Python - Bitwise Operators** 132

22. **Python - Membership Operators** 137

23. **Python - Identity Operators** 141

24. **Python Operator Precedence** 144

25. **Python - Comments** 147

26. **Python - User Input** 151

27. **Python - Numbers** 157

28. **Python - Booleans** 174

PYTHON CONTROL STATEMENTS 176

29. **Python - Control Flow** 177

30. **Python - Decision Making** 183

31. **Python - if Statement** 186

32. **Python if-else Statement** 189

33. **Python - Nested if Statement** 195

34. **Python - Match-Case Statement** 198

35. **Python - Loops** 201

36. **Python - For Loops** 203

37. **Python for-else Loops** 209

38. **Python - While Loops** 213

39. **Python - break Statement** 218

40. **Python - Continue Statement** 221

41. **Python - pass Statement** 224

42. **Python - Nested Loops** 226

PYTHON FUNCTIONS & MODULES 229

43. Python - Functions 230

44. Python - Default Arguments 243

45. Python - Keyword Arguments 246

46. Python - Keyword-Only Arguments 249

47. Python - Positional Arguments 251

48. Python - Positional-Only Arguments 254

49. Python - Arbitrary or, Variable-length Arguments 256

50. Python Variable Scope 259

51. Python - Function Annotations 266

52. Python - Modules 270

53. Python - Built-in Functions 281

PYTHON STRINGS 287

54. Python - Strings 288

55. Python Slicing Strings 298

56. Python - Modify Strings 304

57. Python - String Concatenation 307

58. Python - String Formatting 310

59. Python - Escape Characters 312

60. Python - String Methods 316

61. Python - String Exercises 321

PYTHON LISTS 323

62. Python - Lists 324

63. Python - Access List Items 328

64. Python - Change List Items 331

65. Python - Add List Items 333

66. Python - Remove List Items 335

67. **Python - Loop Lists** 338

68. **Python - List Comprehension** 342

69. **Python - Sort Lists** 346

70. **Python - Copy Lists** 349

71. **Python - Join Lists** 353

72. **Python - List Methods** 356

73. **Python - List Exercises** 359

PYTHON TUPLES 361

74. **Python - Tuples** 362

75. **Python - Access Tuple Items** 366

76. **Python - Update Tuples** 370

77. **Python - Unpack Tuple Items** 374

78. **Python - Loop Tuples** 376

79. **Python - Join Tuples** 379

80. **Python - Tuple Methods** 383

81. **Python Tuple Exercises** 386

PYTHON SETS 388

82. **Python - Sets** 389

83. **Python - Access Set Items** 394

84. **Python - Add Set Items** 397

85. **Python - Remove Set Items** 401

86. **Python - Loop Sets** 408

87. **Python - Join Sets** 412

88. **Python - Copy Sets** 416

89. **Python - Set Operators** 419

90. **Python - Set Methods** 424

91. **Python - Set Exercises** 426

PYTHON DICTIONARIES 427

92. Python - Dictionaries 428

93. Python - Access Dictionary Items 436

94. Python - Change Dictionary Items 442

95. Python - Add Dictionary Items 446

96. Python - Remove Dictionary Items 451

97. Python - Dictionary View Objects 455

98. Python - Loop Dictionaries 458

99. Python - Copy Dictionaries 461

100. Python - Nested Dictionaries 465

101. Python - Dictionary Methods 470

102. Python - Dictionary Exercises 471

PYTHON ARRAYS 473

103. Python - Arrays 474

104. Python - Access Array Items 479

105. Python - Add Array Items 482

106. Python - Remove Array Items 484

107. Python - Loop Arrays 486

108. Python - Copy Arrays 488

109. Python - Reverse Arrays 490

110. Python - Sort Arrays 493

111. Python - Join Arrays 496

112. Python - Array Methods 498

113. Python - Array Exercises 501

PYTHON FILE HANDLING 503

114. Python - File Handling 504

115. Python - Write to File 510

116. **Python - Read Files** 515

117. **Python - Renaming and Deleting Files** 523

118. **Python - Directories** 525

119. **Python - File Methods** 529

120. **Python OS File/Directory Methods** 531

121. **Python OS.Path Methods** 535

OBJECT ORIENTED PROGRAMMING 538

122. **Python - OOP Concepts** 539

123. **Python - Classes and Objects** 547

124. **Python - Class Attributes** 555

125. **Python - Class Methods** 559

126. **Python - Static Methods** 563

127. **Python - Constructors** 566

128. **Python - Access Modifiers** 571

129. **Python - Inheritance** 576

130. **Python - Polymorphism** 584

131. **Python - Method Overriding** 588

132. **Python - Method Overloading** 591

133. **Python - Dynamic Binding** 594

134. **Python - Dynamic Typing** 597

135. **Python - Abstraction** 599

136. **Python - Encapsulation** 601

137. **Python - Interfaces** 604

138. **Python - Packages** 607

139. **Python - Inner Classes** 611

140. **Python - Anonymous Class and Objects** 615

141. **Python - Singleton Class** 617

142. **Python - Wrapper Classes** 619

143. Python - Enums 620

144. Python - Reflection 624

PYTHON ERRORS & EXCEPTIONS 631

145. Python - Syntax Errors 632

146. Python - Exceptions Handling 635

147. Python - The try-except Block 644

148. Python - The try-finally Block 648

149. Python - Raising Exceptions 651

150. Python - Exception Chaining 655

151. Python - Nested try Block 658

152. Python - User-Defined Exceptions 661

153. Python - Logging 665

154. Python - Assertions 669

155. Python - Built-in Exceptions 671

PYTHON MULTITHREADING 679

156. Python - Multithreading 680

157. Python - Thread Lifecycle 688

158. Python - Creating a Thread 692

159. Python - Starting a Thread 697

160. Python - Joining the Threads 700

161. Python - Naming the Threads 703

162. Python - Thread Scheduling 707

163. Python - Thread Pools 711

164. Python - Main Thread 715

165. Python - Thread Priority 719

166. Python - Daemon Threads 725

167. Python - Synchronizing Threads 729

PYTHON SYNCHRONIZATION..734

168. Python - Inter-Thread Communication..735

169. Python - Thread Deadlock..741

170. Python - Interrupting a Thread..746

PYTHON NETWORKING..749

171. Python - Network Programming..750

172. Python - Socket Programming..751

173. Python - URL Processing..760

174. Python - Generics..766

PYTHON LIBRARIES..768

175. NumPy Tutorial..769

176. Python Pandas Tutorial..772

177. SciPy Tutorial..777

178. Matplotlib Tutorial..778

179. Django Tutorial..784

180. OpenCV Python Tutorial..787

PYTHON MISCELLANEOUS..788

181. Python - Date and Time..789

182. Python - math Module..807

183. Python - Iterators..813

184. Python - Generators..819

185. Python - Closures..825

186. Python - Decorators..829

187. Python - Recursion..835

188. Python - Regular Expressions..839

189. Python - PIP..854

190. Python - Database Access..859

191. Python - Weak References 869

192. Python - Serialization 875

193. Python - Templating 882

194. Python - Output Formatting 888

195. Python - Performance Measurement 897

196. Python - Data Compression 900

197. Python - CGI Programming 901

198. Python - XML Processing 916

199. Python - GUI Programming 927

200. Python - Command-Line Arguments 945

201. Python - Docstrings 952

202. Python - JSON 960

203. Python - Sending Email 965

204. Python - Further Extensions 974

205. Python - Tools/Utilities 983

206. Python - GUIs 987

PYTHON ADVANCED CONCEPTS 994

207. Python - Abstract Base Classes 995

208. Python - Custom Exceptions 1000

209. Python - Higher Order Functions 1003

210. Python - Object Internals 1007

211. Python - Memory Management 1010

212. Python - Metaclasses 1014

213. Python - Metaprogramming with Metaclasses 1018

214. Python - Mocking and Stubbing 1021

215. Python - Monkey Patching 1024

216. Python - Signal Handling 1026

217. Python - Type Hints 1030

218. Python - Automation Tutorial 1040

219. Python - Humanize Package 1046

220. Python - Context Managers 1052

221. Python - Coroutines 1057

222. Python - Descriptors 1063

223. Python - Diagnosing and Fixing Memory Leaks 1071

224. Python - Immutable Data Structures 1076

Python Basics

1. Python – Overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation. 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.

Python is an open-source and cross-platform programming language. It is available for use under **Python Software Foundation License** (compatible to GNU General Public License) on all the major operating system platforms such as Linux, Windows and Mac OS.

To facilitate new features and to maintain that readability, the Python Enhancement Proposal (PEP) process was developed. This process allows anyone to submit a PEP for a new feature, library, or other addition.

The design philosophy of Python emphasizes on simplicity, readability and unambiguity. Python is known for its batteries included approach as Python software is distributed with a comprehensive standard library of functions and modules.

Python's design philosophy is documented in the Zen of Python. It consists of nineteen aphorisms such as –

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated

To obtain the complete Zen of Python document, type import this in the Python Shell –

```
>>>import this
```

This will produce following 19 aphorisms -

```
Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.
```

```
Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!
```

Python supports imperative, structured as well as object-oriented programming methodology. It provides features of functional programming as well.

Pythonic Code Style

Python leaves you free to choose to program in an object-oriented, procedural, functional, aspect-oriented, or even logic-oriented way. These freedoms make Python a great language to write clean and beautiful code.

Pythonic Code Style is actually more of a design philosophy and suggests to write a code which is:

- Clean
- Simple
- Beautiful
- Explicit
- Readable

The Zen of Python

The Zen of Python is about code that not only works, but is Pythonic. Pythonic code is readable, concise, and maintainable.

2. Python - History and Versions

History of Python

Python was developed by Guido van Rossum (a Dutch programmer) in the late 1980s and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages. Guido van Rossum wanted Python to be a high-level language that was powerful yet readable and easy to use.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Being the principal architect of Python, the developer community conferred upon him the title of **Benevolent Dictator for Life** (BDFL). However, in 2018, Rossum relinquished the title. Thereafter, the development and distribution of the reference implementation of Python is handled by a nonprofit organization **Python Software Foundation**.

Who Invented Python?

Python was invented by a Dutch Programmer Guido Van Rossum in the late 1980s. He began working on Python in December 1989 as a hobby project while working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands. Python's first version (0.9.0) was released in 1991.

Evolution of Python – The Major Python Versions

Following are the important stages in the history of Python –

Python 0.9.0

Python's first published version is 0.9. It was released in February 1991. It consisted of features such as classes with inheritance, exception handling, and core data types like lists and dictionaries.

Python 1.0

In January 1994, version 1.0 was released, armed with functional programming tools, features like support for complex numbers etc. and module system which allows a better code organization and reuse.

Python 2.0

Next major version – Python 2.0 was launched in October 2000. Many new features such as list comprehension, garbage collection and Unicode support were included with it. Throughout the 2000s, Python 2.x became the dominant version, gaining traction in industries ranging from web development to scientific research. Various useful libraries like NumPy, SciPy, and Django were also developed.

Python 3.0

Python 3.0, a completely revamped version of Python was released in December 2008. The primary objective of this revamp was to remove a lot of discrepancies that had crept in Python 2.x versions. Python 3 was backported to Python 2.6. It also included a utility named as python2to3 to facilitate automatic translation of Python 2 code to Python 3. Python 3 provided new syntax, Unicode support and improved integer division.

EOL for Python 2.x

Even after the release of Python 3, Python Software Foundation continued to support the Python 2 branch with incremental micro versions till 2019. However, it decided to discontinue the support by the end of year 2020, at which time Python 2.7.17 was the last version in the branch.

Current Version of Python

Meanwhile, more and more features have been incorporated into Python's 3.x branch. As of date, Python **3.11.2** is the current stable version, released in February 2023.

What's New in Python 3.11?

One of the most important features of Python's version 3.11 is the significant improvement in speed. According to Python's official documentation, this version is faster than the previous version (3.10) by up to 60%. It also states that the standard benchmark suite shows a 25% faster execution rate.

- Python 3.11 has a better exception messaging. Instead of generating a long traceback on the occurrence of an exception, we now get the exact expression causing the error.
- As per the recommendations of PEP 678, the add_note() method is added to the BaseException class. You can call this method inside the except clause and pass a custom error message.
- It also adds the **cbroot()** function in the maths module. It returns the cube root of a given number.
- A new module **tomllib** is added in the standard library. TOML (Tom's Obvious Minimal Language) can be parsed with tomllib module function.

Python in the Future

Python is evolving every day and Python 3.x is receiving regular updates. Python's developer community is focusing on performance improvements making it more efficient while retaining its ease of use.

Python is being heavily used for machine learning, AI, and data science, so for sure its future remains bright. It's role in these rapidly growing fields ensures that Python will stay relevant for years.

Python is also increasingly becoming the first programming language taught in schools and universities worldwide, solidifying its place in the tech landscape.

Frequently Asked Questions About Python History

1. Who created Python?

Python created by Guido Van Rossum, a Dutch Programmer.

2. Why Python is called Python?

Python does not have any relation to Snake. The name of the Python programming language was inspired by a British Comedy Group Monty Python.

3. When was Python's first version released?

Python's first version was released in February 1991.

4. What was the first version of Python?

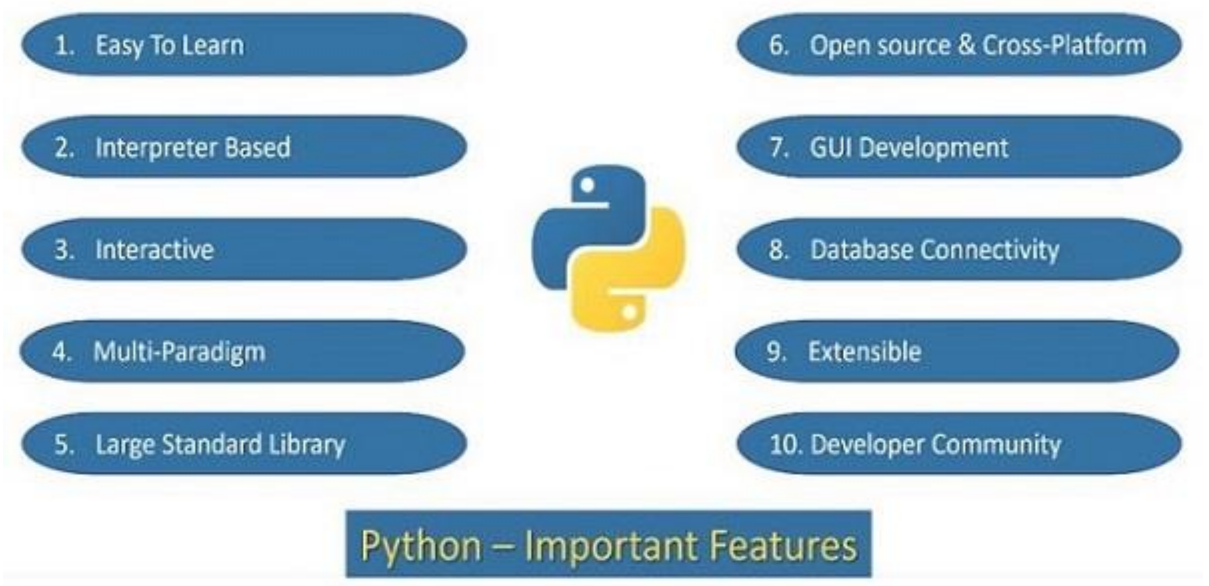
Python's first version was Python 0.9.0

5. When was Python 3.0 version released?

Python 3.0 version was released in December 2008.

3. Python - Features

Python is a feature-rich, high-level, interpreted, interactive, and object-oriented scripting language. Python is a versatile and very popular programming language due to its features such as readability, simplicity, extensive libraries, and many more. In this tutorial, we will learn about the various features of Python that make it a powerful and versatile programming language.



Features of Python

Python's most important features are as follows:

- Easy to Learn
- Dynamically Typed
- Interpreter Based
- Interactive
- Multi-paradigm
- Standard Library
- Open Source and Cross Platform
- GUI Applications
- Database Connectivity
- Extensible
- Active Developer Community

Easy to Learn

This is one of the most important reasons for the popularity of Python. Python has a limited set of keywords. Its features such as simple syntax, usage of indentation to avoid clutter of curly brackets, and dynamic typing that doesn't necessitate prior declaration of variable help a beginner to learn Python quickly and easily.

Dynamically Typed

Python is a dynamically typed programming language. In Python, you don't need to specify the variable at the time of the declaration. The types are specified at the runtime based on the assigned value due to its dynamically typed feature.

Interpreter Based

Instructions in any programming languages must be translated into machine code for the processor to execute them. Programming languages are either compiler based or interpreter based.

In case of a compiler, a machine language version of the entire source program is generated. The conversion fails even if there is a single erroneous statement. Hence, the development process is tedious for the beginners. The C family languages (including C, C++, Java, C# etc.) are compiler based.

Python is an interpreter based language. The interpreter takes one instruction from the source code at a time, translates it into machine code and executes it. Instructions before the first occurrence of error are executed. With this feature, it is easier to debug the program and thus proves useful for the beginner level programmer to gain confidence gradually. Python therefore is a beginner-friendly language.

Interactive

Standard Python distribution comes with an interactive shell that works on the principle of REPL (Read – Evaluate – Print – Loop). The shell presents a Python prompt >>>. You can type any valid Python expression and press Enter. Python interpreter immediately returns the response and the prompt comes back to read the next expression.

```
>>> 2*3+1

7

>>> print ("Hello World")

Hello World
```

The interactive mode is especially useful to get familiar with a library and test out its functionality. You can try out small code snippets in interactive mode before writing a program.

Multi-paradigm

Python is a completely object-oriented language. Everything in a Python program is an object. However, Python conveniently encapsulates its object orientation to be used as an imperative or procedural language – such as C. Python also provides certain functionalities that resemble functional programming. Moreover, certain third-party tools have been developed to support other programming paradigms such as aspect-oriented and logic programming.

Standard Library

Even though it has a very few keywords (only Thirty-Five), Python software is distributed with a standard library made of large number of modules and packages. Thus Python has out of box support for programming needs such as serialization, data compression, internet data handling, and many more. Python is known for its batteries included approach.

Some of the Python's popular modules are:

- NumPy
- Pandas
- Matplotlib
- Tkinter
- Math

Open Source and Cross Platform

Python's standard distribution can be downloaded from

https://www.python.org/downloads/https://www.python.org/downloads/ without any restrictions. You can download pre-compiled binaries for various operating systems. In addition, the source code is also freely available, which is why it comes under open source category.

Python software (along with the documentation) is distributed under Python Software Foundation License. It is a BSD style permissive software license and compatible to GNU GPL (General Public License).

Python is a cross-platform language. Pre-compiled binaries are available for use on various operating systems such as Windows, Linux, Mac OS, Android OS. The reference implementation of Python is called CPython and is written in C. You can download the source code and compile it for your OS platform.

A Python program is first compiled to an intermediate platform independent byte code. The virtual machine inside the interpreter then executes the byte code. This behavior makes Python a cross-platform language, and thus a Python program can be easily ported from one OS platform to other.

GUI Applications

Python's standard distribution has an excellent graphics library called TKinter. It is a Python port for the vastly popular GUI toolkit called TCL/Tk. You can build attractive user-friendly GUI applications in Python. GUI toolkits are generally written in C/C++. Many of them have been ported to Python. Examples are PyQt, WxWidgets, PySimpleGUI etc.

Database Connectivity

Almost any type of database can be used as a backend with the Python application. DB-API is a set of specifications for database driver software to let Python communicate with a relational database. With many third party libraries, Python can also work with NoSQL databases such as MongoDB.

Extensible

The term extensibility implies the ability to add new features or modify existing features. As stated earlier, CPython (which is Python's reference implementation) is written in C. Hence one can easily write modules/libraries in C and incorporate them in the standard library. There are other implementations of Python such as Jython (written in Java) and IPython (written in C#). Hence, it is possible to write and merge new functionality in these implementations with Java and C# respectively.

Active Developer Community

As a result of Python's popularity and open-source nature, a large number of Python developers often interact with online forums and conferences. Python Software Foundation also has a significant member base, involved in the organization's mission to "**Promote, Protect, and Advance the Python Programming Language**"

Python also enjoys a significant institutional support. Major IT companies Google, Microsoft, and Meta contribute immensely by preparing documentation and other resources.

Apart from the above-mentioned features, Python has another big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

4. Python vs C++

Python is a general-purpose, high-level programming language. Python is used for web development, Machine Learning, and other cutting-edge software development technologies. Python is suitable for both new and seasoned C++ and Java programmers. **Guido Van Rossum** has created Python in 1989 at Netherlands' National Research Institute. Python was released in 1991.

C++ is a middle-level, case-sensitive, object-oriented programming language. Bjarne Stroustrup created C++ at Bell Labs. C++ is a platform-independent programming language that works on Windows, Mac OS, and Linux. C++ is near to hardware, allowing low-level programming. This provides a developer control over memory, improved performance, and dependable software.

Read through this article to get an overview of C++ and Python and how these two programming languages are different from each other.

What is Python?

Python is currently one of the most widely used programming languages. It is an interpreted programming language that operates at a high level. When compared to other languages, the learning curve for Python is much lower, and it is also quite straightforward to use.

Python is the programming language of choice for professionals working in fields such as Artificial Intelligence, Machine Learning (ML), Data Science, the Internet of Things (IoT), etc., because it excels at both scripting applications and as standalone programmers.

In addition to this, Python is the language of choice because it is easy to learn. Because of its excellent syntax and readability, the amount of money spent on maintenance is decreased. The modularity of the program and the reusability of the code both contribute to its support for a variety of packages and modules.

Using Python, we can perform –

- Web development
- Data analysis and machine learning
- Automation and scripting
- Software testing and many more

Features

Here is a list of some of the important features of Python –

- **Easy to learn –** Python has a simple structure, few keywords, and a clear syntax. This makes it easy for the student to learn quickly. Code written in Python is easier to read and understand.
- **Easy to maintain –** The source code for Python is pretty easy to keep up with.
- **A large standard library –** Most of Python's libraries are easy to move around and work on UNIX, Windows, Mac.
- **Portable –** Python can run on a wide range of hardware platforms, and all of them have the same interface.

Python Example

Take a look at the following simple Python program –

```
a = int(input("Enter value for a"))

b = int(input("Enter value for b"))


print("The number you have entered for a is ", a)

print("The number you have entered for b is ", b)
```

In our example, we have taken two variables "a" and "b" and assigning some value to those variables. Note that in Python, we don't need to declare datatype for variables explicitly, as the PVM will assign datatype as per the user's input.

- The **input()** function is used to take input from the user through keyboard.
- In Python, the return type of **input()** is string only, so we have to convert it explicitly to the type of data which we require. In our example, we have converted to int type explicitly through int( ) function.
- **print()** is used to display the output.

Output

On execution, this Python code will produce the following output –

```
Enter value for a 10

Enter value for b 20


The number you have entered for a is 10

The number you have entered for b is 20
```

What is C++?

C++ is a statically typed, compiled, multi-paradigm, general-purpose programming language with a steep learning curve. Video games, desktop apps, and embedded systems use it extensively. C++ is so compatible with C that it can build practically all C source code without any changes. Object-oriented programming makes C++ a better-structured and safer language than C.

Features

Let's see some features of C++ and the reason of its popularity.

- **Middle-level language –** It's a middle-level language since it can be used for both system development and large-scale consumer applications like Media Players, Photoshop, Game Engines, etc.

- **Execution Speed –** C++ code runs quickly because it's compiled and uses procedures extensively. Garbage collection, dynamic typing, and other modern features impede program execution.

- **Object-oriented language –** Object-oriented programming is flexible and manageable. Large apps can also be developed through this language.
- **Extensive Library Support –** C++ has a vast library. Third-party libraries are supported for fast development.

C++ Example

Let's understand the syntax of C++ through an example written below.

```
#include
using namespace std;

int main() {
   int a, b;
   cout << "Enter The value for variable a \n";
   cin >> a;
   cout << "Enter The value for variable b";
   cin >> b;
   cout << "The value of a is "<< a << "and" << b;
   return 0;
}
```

In our example, we are taking input for two variables "a" and "b" from the user through the keyboard and displaying the data on the console.

Output

On execution, it will produce the following output –

```
Enter The value for variable a
10
Enter The value for variable b
20
The value of a is 10 and 20
```

Comparison Between Python and C++ across Various Aspects

Both Python and C++ are among the most popular programming languages. Both of them have their advantages and disadvantages. In this tutorial, we shall take a look at their features which differentiate one from another.

Compiled vs Interpreted

Like C, C++ is also a compiler-based language. A compiler translates the entire code in a machine language code specific to the operating system in use and processor architecture.

Python is interpreter-based language. The interpreter executes the source code line by line.

Cross platform

When a C++ source code such as hello.cpp is compiled on Linux, it can be only run on any other computer with Linux operating system. If required to run on other OS, it needs to be recompiled.

Python interpreter doesn't produce compiled code. Source code is converted to byte code every time it is run on any operating system without any changes or additional steps.

Portability

Python code is easily portable from one OS to other. C++ code is not portable as it must be recompiled if the OS changes.

Speed of Development

C++ program is compiled to the machine code. Hence, its execution is faster than interpreter based language.

Python interpreter doesn't generate the machine code. Conversion of intermediate byte code to machine language is done on each execution of program.

If a program is to be used frequently, C++ is more efficient than Python.

Easy to Learn

Compared to C++, Python has a simpler syntax. Its code is more readable. Writing C++ code seems daunting in the beginning because of complicated syntax rules such as use of curly braces and semicolon for sentence termination.

Python doesn't use curly brackets for marking a block of statements. Instead, it uses indents. Statements of similar indent level mark a block. This makes a Python program more readable.

Static vs Dynamic Typing

C++ is a statically typed language. The type of variables for storing data need to be declared in the beginning. Undeclared variables can't be used. Once a variable is declared to be of a certain type, value of only that type can be stored in it.

Python is a dynamically typed language. It doesn't require a variable to be declared before assigning it a value. Since, a variable may store any type of data, it is called dynamically typed.

OOP Concepts

Both C++ and Python implement object oriented programming concepts. C++ is closer to the theory of OOP than Python. C++ supports the concept of data encapsulation as the visibility of the variables can be defined as public, private and protected.

Python doesn't have the provision of defining the visibility. Unlike C++, Python doesn't support method overloading. Because it is dynamically typed, all the methods are polymorphic in nature by default.

C++ is in fact an extension of C. One can say that additional keywords are added in C so that it supports OOP. Hence, we can write a C type procedure oriented program in C++.

Python is completely object oriented language. Python's data model is such that, even if you can adapt a procedure oriented approach, Python internally uses object-oriented methodology.

Garbage Collection

C++ uses the concept of pointers. Unused memory in a C++ program is not cleared automatically. In C++, the process of garbage collection is manual. Hence, a C++ program is likely to face memory related exceptional behavior.

Python has a mechanism of automatic garbage collection. Hence, Python program is more robust and less prone to memory related issues.

Application Areas

Because C++ program compiles directly to machine code, it is more suitable for system programming, writing device drivers, embedded systems and operating system utilities.

Python program is suitable for application programming. Its main area of application today is data science, machine learning, API development etc.

Difference Between Python and C++

The following table summarizes the differences between Python and C++ −

Criteria	Python	C++
Execution	Python is an interpreted-based programming language. Python programs are interpreted by an interpreter.	C++ is a compiler-based programming language. C++ programs are compiled by a compiler.
Typing	Python is a dynamic-typed language.	C++ is a static-typed language.
Portability	Python is a highly portable language, code written and executed on a system can be easily run on another system.	C++ is not a portable language, code written and executed on a system cannot be run on another system without making changes.
Garbage collection	Python provides a garbage collection feature. You do not need to worry about the memory management. It is automatic in Python.	C++ does not provide garbage collection. You have to take care of freeing memories. It is manual in C++.
Syntax	Python's syntaxes are very easy to read, write, and understand.	C++'s syntaxes are tedious.
Performance	Python's execution performance is slower than C++'s.	The speed of the execution of C++ codes is faster than Python codes.

Application areas	Python's application areas are machine learning, web applications, and more.	C++'s application areas are embedded systems, device drivers, and more.

==========

End of ebook preview

If you liked what you saw…

Buy it from our store @ https://store.tutorialspoint.com