* Introduction to python

Python 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 to emphasize code readability, and its syntax allows programmers to express concepts in fewer lines of code.

**History**

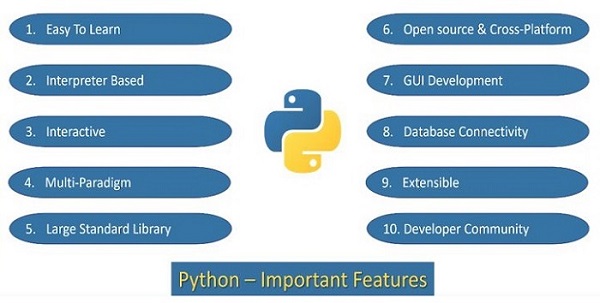
In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) which is situated in the Netherlands.

The inspiration for the name came from the BBC’s TV Show – ‘ **Monty Python’s Flying Circus’**, as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! He was the “Benevolent dictator for life” (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox.

The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with [Java](https://www.geeksforgeeks.org/java), [C++](https://www.geeksforgeeks.org/c-plus-plus)& [C](https://www.geeksforgeeks.org/c-programming-language). Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released, it had more than enough capability to provide classes with inheritance, several core data types of exception handling and functions.

* **Features of python**

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.



Python's most important features are as follows:

* [Easy to Learn](https://www.tutorialspoint.com/python/python_features.htm#easy-to-learn)
* [Dynamically Typed](https://www.tutorialspoint.com/python/python_features.htm#dynamically-typed)
* [Interpreter Based](https://www.tutorialspoint.com/python/python_features.htm#interpreter-based)
* [Interactive](https://www.tutorialspoint.com/python/python_features.htm#interactive)
* [Multi-paradigm](https://www.tutorialspoint.com/python/python_features.htm#multi-paradigm)
* [Standard Library](https://www.tutorialspoint.com/python/python_features.htm#standard-library)
* [Open Source and Cross Platform](https://www.tutorialspoint.com/python/python_features.htm#open-source-and-cross-platform)
* [GUI Applications](https://www.tutorialspoint.com/python/python_features.htm#gui-applications)
* [Database Connectivity](https://www.tutorialspoint.com/python/python_features.htm#database-connectivity)
* [Extensible](https://www.tutorialspoint.com/python/python_features.htm#extensible)

1. 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](https://www.tutorialspoint.com/python/python_basic_syntax.htm), 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.

1. Dynamically Typed

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

1. 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](https://www.tutorialspoint.com/machine_learning/index.htm) 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](https://www.tutorialspoint.com/cprogramming/index.htm), [C++](https://www.tutorialspoint.com/cplusplus/index.htm), [Java](https://www.tutorialspoint.com/java/index.htm), [C#](https://www.tutorialspoint.com/csharp/index.htm) 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.

1. 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.

1. Multi-paradigm

Python is a completely [object-oriented](https://www.tutorialspoint.com/python/python_oops_concepts.htm) language. Everything in a Python program is an [object](https://www.tutorialspoint.com/python/python_object_classes.htm). However, Python conveniently encapsulates its object orientation to be used as an imperative or procedural language such as C. Python also provides certain functionality that resembles functional programming. Moreover, certain third-party tools have been developed to support other programming paradigms such as aspect-oriented and logic programming.

1. 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](https://www.tutorialspoint.com/numpy/index.htm)
* [Pandas](https://www.tutorialspoint.com/python_pandas/index.htm)
* [Matplotlib](https://www.tutorialspoint.com/matplotlib/index.htm)
* Tkinter
* [Math](https://www.tutorialspoint.com/python/python_maths.htm)

1. Open Source and Cross Platform

Python's standard distribution can be downloaded from <https://www.python.org/downloads/> without any restrictions. You can download pre-compiled binaries for various operating system platforms. In addition, the source code is also freely available, which is why it comes under open source category.

Python is a cross-platform language. Pre-compiled binaries are available for use on various operating system platforms such as [Windows](https://www.tutorialspoint.com/windows10/index.htm), [Linux](https://www.tutorialspoint.com/unix/index.htm), Mac OS, [Android OS](https://www.tutorialspoint.com/android/index.htm). 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 behaviour makes Python a cross-platform language, and thus a Python program can be easily ported from one OS platform to other.

1. 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](https://www.tutorialspoint.com/pyqt/index.htm" \t "_blank), [WxWidgets](https://www.tutorialspoint.com/wxpython/index.htm" \t "_blank), [PySimpleGUI](https://www.tutorialspoint.com/pysimplegui/index.htm" \t "_blank) etc.

1. 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](https://www.tutorialspoint.com/mongodb/index.htm).

1. 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](https://www.tutorialspoint.com/jupyter/ipython_introduction.htm" \t "_blank) (written in C#). Hence, it is possible to write and merge new functionality in these implementations with Java and C# respectively.

* Applications of Python

1. Operating Systems

The robust standard library of Python makes it perfect for building entire operating systems. The object-oriented design of the language ensures large projects are easily managed. Python is compatible with most operating systems and can be easily used to build native applications for Windows and Mac computers.

Example

While Python is not typically used to build entire operating systems, it is often used for scripting and automation tasks within operating systems. Python scripts can automate system administration tasks, manage files and directories, and interact with system APIs. Examples include writing scripts to automate backups, manage user accounts, or monitor system performance.

2. Web Development

Python offers several choices for complex web development projects. HTML and JavaScript are the main languages used to build the front end of an application. But Python-based web frameworks like Django, Pyramid, and Flask are used to handle backend or server-side functionality of sites and services like Spotify, Reddit, and Mozilla. Giant platforms like Google and YouTube depend largely on Python for critical infrastructure.

The standard library of Python also supports many Internet protocols like HTML and XML, JSON, Email processing, FTP, and IMAP.

Example

Python is widely used in web development for building dynamic websites, web applications, and APIs. Frameworks like Django and Flask provide powerful tools for building web applications, handling HTTP requests, and interacting with databases. Popular websites and web applications like Instagram, Pinterest, and Spotify are built using Python and its web frameworks.

3. Game Development

Just like for web development, Python offers an array of tools and libraries for game development. Would you believe, Battlefield 2 – one of the most popular shooting games of the early 2000s, was developed with the use of Python.

Python’s 2D and 3D game development libraries are PyGame, Pycap, Panda#D, Construct, PySoy, and PyOpenGL.

Python has been used to develop popular games, including Sims 4, World of Tanks, Eve Online, Mount & Blade, Doki Doki Literature Club, and Disney’s Toontown Online, to name a few.

Example

Python is increasingly used in the game development industry for building games, game engines, and game development tools. Libraries like Pygame provide a framework for building 2D games, while engines like Panda3D and Godot support the development of both 2D and 3D games. Python's simplicity and ease of use make it an attractive choice for prototyping and rapid game development.

4. Scientific and Numeric Computing

The Python ecosystem offers numerous tools and libraries that help scientists and researchers in scientific and numeric computing.

* SciPy is a set of packages for mathematics, science, and engineering
* Pandas is a library used for data analysis and modeling
* IPython is a strong interactive shell that provides hassle-free editing and recording of a work session and aid in visualizing and parallel computing.
* FreeCAD and Abaqus are real-life numerical and scientific applications built with Python

Example

Python is widely used in scientific computing and data analysis due to its rich ecosystem of libraries and tools. Libraries like NumPy, SciPy, and Pandas provide powerful tools for numerical computing, data manipulation, and statistical analysis. Python is also used in scientific research, engineering simulations, and data visualization tasks.

5. Artificial Intelligence and Machine Learning

The hottest buzzwords of the decade – Artificial Intelligence (AI) and Machine Learning are mostly about algorithms, code, and logic. Python, along with a few other programming languages, is increasingly being used for developing AI and ML-powered solutions. The scope and power of Python, along with its stability and security, make it ideal for running AI and ML systems.

Some important libraries for the job are:

* Scikit-Learn – for building various machine learning models
* SciPy – for scientific and technical computing
* TensorFlow – for state-of-the-art neural networks
* Keras – for artificial neural networks
* Pandas – for data analysis and manipulation

Example

Python is the preferred language for artificial intelligence (AI) and machine learning (ML) applications due to its simplicity, flexibility, and extensive libraries. Frameworks like TensorFlow, PyTorch, and scikit-learn provide tools for building and training machine learning models, neural networks, and deep learning algorithms. Python's popularity in AI and ML has led to its widespread adoption in areas like natural language processing, computer vision, and predictive analytics.

6. Desktop GUI

Python is an excellent choice for desktop GUI (Graphical User Interface) programming. The language offers numerous options for developers to build a fully functional GUI. The comprehensive syntax and modular programming approach of the Python framework help create a super-fast and responsive GUI.

Some prominent applications of Python tools for GUI development are PyQt, Tkinter, wxWidgets, Python GTK+, and Kivy. Standard applications like Dropbox and BitTorrent are primarily written in Python.

Example

Python can be used to develop desktop graphical user interface (GUI) applications using libraries like Tkinter, PyQt, and wxPython. These libraries provide tools for creating windows, dialogs, buttons, and other GUI components. Python's simplicity and cross-platform compatibility make it suitable for building desktop applications for various operating systems.

7. Business Applications

Applications of Python also include building ERP and ecommerce systems. Business applications are different from typical consumer software because they offer a set of specific features instead of a variety of features. Besides, they target a very tight-knit user group, usually an organization.

Python is perfect for delivering best-performance custom solutions for business applications as well as consumer applications.

Odoo is a well-rounded management software that offers numerous business applications that constitute a complete set of enterprise management applications.

Tryton is a three-tier high-level application platform designed for general purposes.

Example

Python is used in business applications for a wide range of purposes, including data analysis, automation, and web development. Python scripts can automate repetitive tasks, generate reports, and interact with business databases and APIs. Python's versatility and ease of integration with other technologies make it a valuable tool for building custom business applications.

8. Console-based Application

Console-based applications are text-based programs that interact with users through a command-line interface. These applications are typically used for tasks such as data processing, system administration, or automation scripts.

Example

A simple console-based calculator program written in Python. This program prompts the user to enter mathematical expressions and evaluates them to produce the result.

# Simple console-based calculator

while True:

 expression = input("Enter a mathematical expression (or 'quit' to exit): ")

 if expression.lower() == 'quit':

    break

try:

    result = eval(expression)

    print("Result:", result)

except:

    print("Invalid expression. Please try again.")

9. Audio or Video-based Applications

Python can be used to create applications that manipulate audio or video data, such as media players, editors, or streaming services. Libraries like PyDub and MoviePy provide tools for processing audio and video files in Python.

Example

A simple audio player application written in Python using the PyDub library. This application allows users to play, pause, and stop audio files.

from pydub import AudioSegment

from pydub.playback import play

# Load audio file

audio\_file = AudioSegment.from\_file("C:/mani/example.mp3")

# Play audio

play(audio\_file)

10. 3D CAD Applications

Python can be used in 3D computer-aided design (CAD) applications for tasks such as modeling, rendering, and simulation. Libraries like Blender provide a comprehensive set of tools for creating and manipulating 3D models programmatically.

Example

A script written in Python to create a simple 3D model using Blender's scripting interface. This script generates a cube and exports it to a file in the Wavefront OBJ format.

import bpy

# Create a cube

bpy.ops.mesh.primitive\_cube\_add(size=2)

# Export the cube to OBJ format

bpy.ops.export\_scene.obj(filepath="cube.obj")

11. Image Processing

Python is commonly used for image processing tasks such as image enhancement, segmentation, and object detection. Libraries like OpenCV and Pillow provide tools for manipulating and analyzing images in Python.

Example

A simple image processing application written in Python using the OpenCV library. This application loads an image, converts it to grayscale, and applies a Gaussian blur filter to smooth the image.

import cv2

# Load image

image = cv2.imread("example.jpg")

# Convert image to grayscale

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2YCrCb)

# Apply Gaussian blur

blurred\_image = cv2.GaussianBlur(gray\_image, (5, 5), 0)

# Display original and processed images

cv2.imshow("Original Image", image)

cv2.imshow("Blurred Image", blurred\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Python objects

In Python, **identity**, **type**, and **value** are three key characteristics that every object has. Understanding these properties is fundamental to how Python works under the hood.

Let's break them down:

**1. Identity**

* The **identity** of an object refers to its unique identity in memory. It is an identifier that distinguishes one object from another. Every object created in Python is stored at a specific location in memory, and its identity corresponds to that memory location.
* You can access an object's identity using the id() function. The value returned by id() is the memory address where the object is stored, but it is represented as an integer.
* The identity of an object is **constant** during the lifetime of the object.

**Example**:

a = "hello"

b = a

print(id(a))

print(id(b))

output:

1522483510448

1522483510448

* In this example, both a and b refer to the same string "hello", so their identities (memory addresses) are the same.

**2. Type**

* The **type** of an object refers to the class or data type the object belongs to. The type defines what operations can be performed on the object and how it behaves. Every object in Python has a type that can be accessed using the type() function.
* The type helps determine the structure and behavior of the object, such as whether it's an integer, string, list, or a custom class instance.

**Example**:

a = 10

b = "hello"

print(type(a))

print(type(b))

output:

<class 'int'>

<class 'str'>

* In this example, a is of type int, and b is of type str.

**3. Value**

* The **value** of an object refers to the actual data that the object holds. The value is the content of the object, such as a number, string, list, etc.
* The value of an object can change if the object is mutable (like a list), but for immutable objects (like strings and tuples), the value cannot be changed after creation.

**Example**:

a = 10

b = 20

print(a)

print(b)

a = 30

print(a)

output:

10

20

30