# Lecture-01: Introduction to Python

## History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3

## Introduction:

Python is currently one of the most popular dynamic programming languages, along with Perl, Tcl, PHP, and newcomer Ruby. Although it is often viewed as a "scripting" language, it is really a general purpose programming language along the lines of Lisp or Smalltalk (as are the others, by the way). Today, Python is used for everything from throw-away scripts to large scalable web servers that provide uninterrupted service 24x7. It is used for GUI and database programming, client- and server-side web programming, and application testing. It is used by scientists writing applications for the world's fastest supercomputers and by children first learning to program.

## A Bird's Eye View of Python

When one is first exposed to Python, they are often struck by way that Python code looks, at least on the surface, similar to code written in other conventional programming languages such as C or Pascal. This is no accident---the syntax of Python borrows heavily from C. For instance, many of Python's keywords (if, else, while, for, etc.) are the same as in C, Python identifiers have the same naming rules as C, and most of the standard operators have the same meaning as C. Of course, Python is obviously not C and one major area where it differs is that instead of using braces for statement grouping, it uses indentation. For example, instead of writing statements in C like this

```
if (a < b) {
      max = b;
  } else {
      max = a;
  }
```

Python just dispenses with the braces altogether (along with the trailing semicolons for good measure) and uses the following structure

```
if a < b:
      max = b
  else:
      max = a
```

The other major area where Python differs from C-like languages is in its use of dynamic typing. In C, variables must always be explicitly declared and given a specific type such as int or double. This information is then used to perform static compile-time checks of the program as well as for allocating memory locations used for storing the variable's value. In Python, variables are simply names that refer to objects. Variables do not need to be declared before they are assigned and they can even change type in the middle of a program. Like other dynamic languages, all type-checking is performed at run-time by an interpreter instead of during a separate compilation step.

Python's primitive built-in data types include Booleans, numbers (machine integers, arbitrary-precision integers, and real and complex floating point numbers), and strings (8-bit and Unicode). These are all immutable types, meaning that values are represented by objects that cannot be modified after their creation. Compound built-in data types include tuples (immutable arrays), lists (resizable arrays) and dictionaries (hash tables).

## Features of Python:

### Simple:

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### Easy to Learn:

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### Free and Open Source:

Python is an example of a FLOSS (Free/LibrÃ© and Open Source Software). In simple terms, you can freely distribute copies of this software, read it's source code, make changes to it, use pieces of it in new free programs, and that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

**High-level Language:**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

**Portable:**

Due to its open-source nature, Python has been ported (i.e. changed to make it work on) to many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and even PocketPC !

**Interpreted:**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

**Object Oriented:**

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

**Extensible:**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use them from your Python program.

**Embeddable:**

You can embed Python within your C/C++ programs to give 'scripting' capabilities for your program's users.

**Extensive Libraries:**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), Tk, and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the 'Batteries Included' philosophy of Python.

**References:**

1.Introduction to Computation and Programming using Python, by John Guttag, PHI Publisher

2.Python Programming using problem solving Approach by ReemaThareja, Oxford University, Higher Education Oxford University Press; First edition (10 June 2017), ISBN-10: 0199480173

3. https://www.tutorialspoint.com/python/index.htm

4. https://www.geeksforgeeks.org/python-programming-language

# ****END OF THE LECTURE***

# ***THANK YOU***