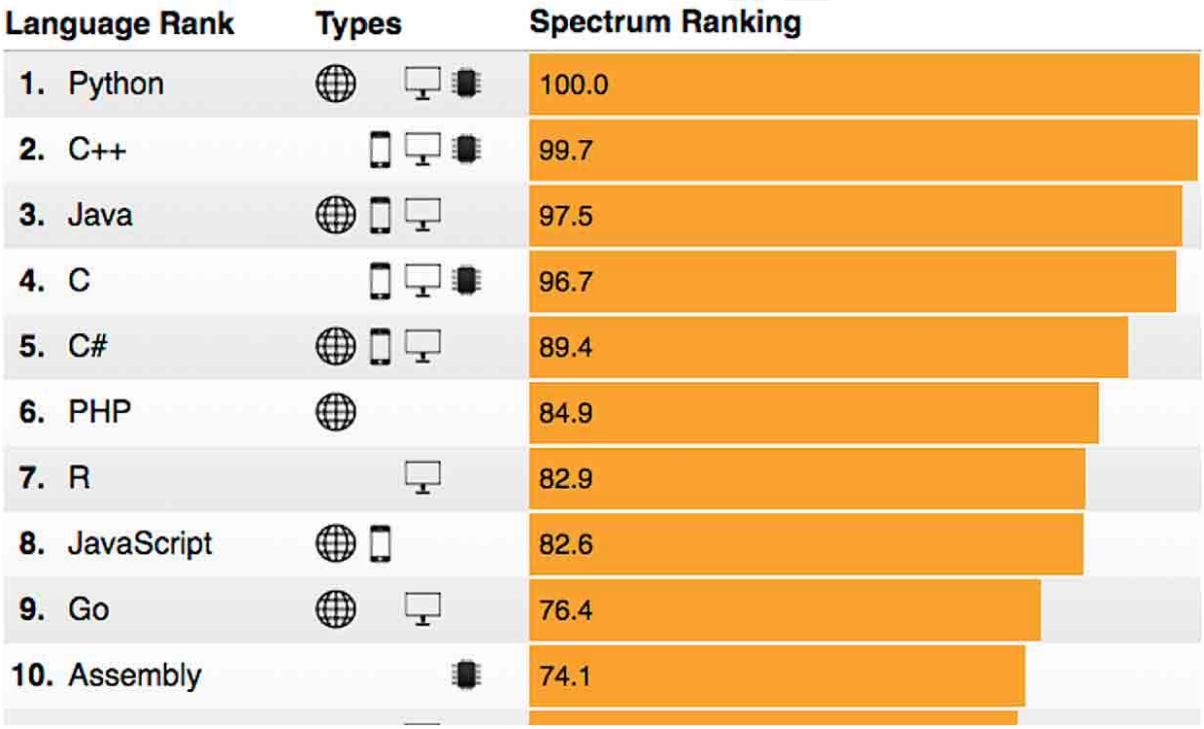
1. The most popular programming language

Python is a fairly old Programming Language (1991) compared to many other Programming Languages like C#(2000), Swift (2014), Java (1995), PHP(1995).

Python has during the last 10 years become more & more popular. Today, Python has become one of the most popular Programming Languages.

There are many different rankings regarding which programming language which is most popular. In most of these ranking, Python is in top 10. One of these rankings is the IEEE Spectrum's ranking of the top programming languages.

As we see in the below Figure they categorize the different Programming Languages into Web, Mobile, Enterprise & Embedded categories:



According to the above Figure, we see that Python can be used to program Web Applications, Enterprise Applications & Embedded Applications. So far Python is not used or not optimized for creating Mobile Applications.

In general, Python is a multipurpose programming language that can be used in many situations. But there is not one programming language which is best in all kind of situations, so it is important that you know about & have skills in different languages.

2. Summary about Python

- Python is fun to learn & use & it is also named after the British comedy group called Monty Python.
- Python has a simple & flexible code structure & the code is easy to read.
- Python is highly extendable due to its high number of free available Python Packaged & Libraries.
- Python can be used on all platforms (Windows, macOS & Linux).
- Python is multi-purpose & can be used to program Web Applications, Enterprise Applications & Embedded Applications & within Data Science & Engineering Applications.
- The popularity of Python is growing fast.
- Python is open source & free to use.
- The growing Python community makes it easy to find documentation, code examples & get help when needed.

3. Interpreter vs Compiler

Programming languages generally fall into one of two categories: Compiled or Interpreted. With a compiled language, code you enter is reduced to a set of machine-specific instructions before being saved as an executable file. Both approaches have their advantages & disadvantages.

With interpreted languages, the code is saved in the same format that you entered. Compiled programs generally run faster than interpreted ones because interpreted programs must be reduced to machine instructions at run-time. It is usually easier to develop applications in an interpreted environment because you don't have to recompile your application each time you want to test a small section.

Python is an interpreted programming language, while e.g., C/C++ are translated by running the source code through a compiler, i.e., C/C++ are compiled languages.

Interpreted languages, in contrast, must be parsed, interpreted & executed each time the program is run. Another example of an interpreted programming language is PHP, which is mainly used to create dynamic web pages & web applications.

Compiled languages are all translated by running the source code through a compiler. This results in very efficient code that can be executed any number of times. The overhead for the translation is incurred just once, when the source is compiled; thereafter, it need only be loaded & executed.

During the design of an application, you might need to decide whether to use a compiled language or an interpreted language for the application source code.

Interpreted languages, in contrast, must be parsed, interpreted & executed each time the program is run Thus, an interpreted language is generally more suited for doing "ad hoc" calculations or simulations, while compiled languages are better for permanent applications where speed is in focus.

4. Packages

In Python, instead of having all of its functionality built into its core, you need to install different packages for different topics. You just need to install these packages & then later import these modules in your code.

This is also typical approach for open source software, because everybody can create their own Python packages & distribute them. In that way you also find Python packages for almost everything, from Scientific Computing to Web Development.

Some reference links for packages are,

- https://www.python.org/about/apps/
- https://pypi.org

4.1 Science & Computation Packages

Some important Python Packages for Science & Numerical Computations are:

- **NumPy** → NumPy is the fundamental package for scientific computing.
- **SciPy** → SciPy is a free & open-source Python library used for scientific computing & technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal & image processing, ODE solvers & other tasks common in science & engineering.
- **Matplotlib** → Matplotlib is a Python 2D plotting library.
- **Pandas** → Pandas Python Data Analysis Library.

4.2 Distribution Package - Anaconda

Anaconda is a distribution package, where you get Python compiler, Python packages & the Spyder editor, all in one package. Anaconda includes Python, the Jupyter Notebook & other commonly used packages for scientific computing & data science.