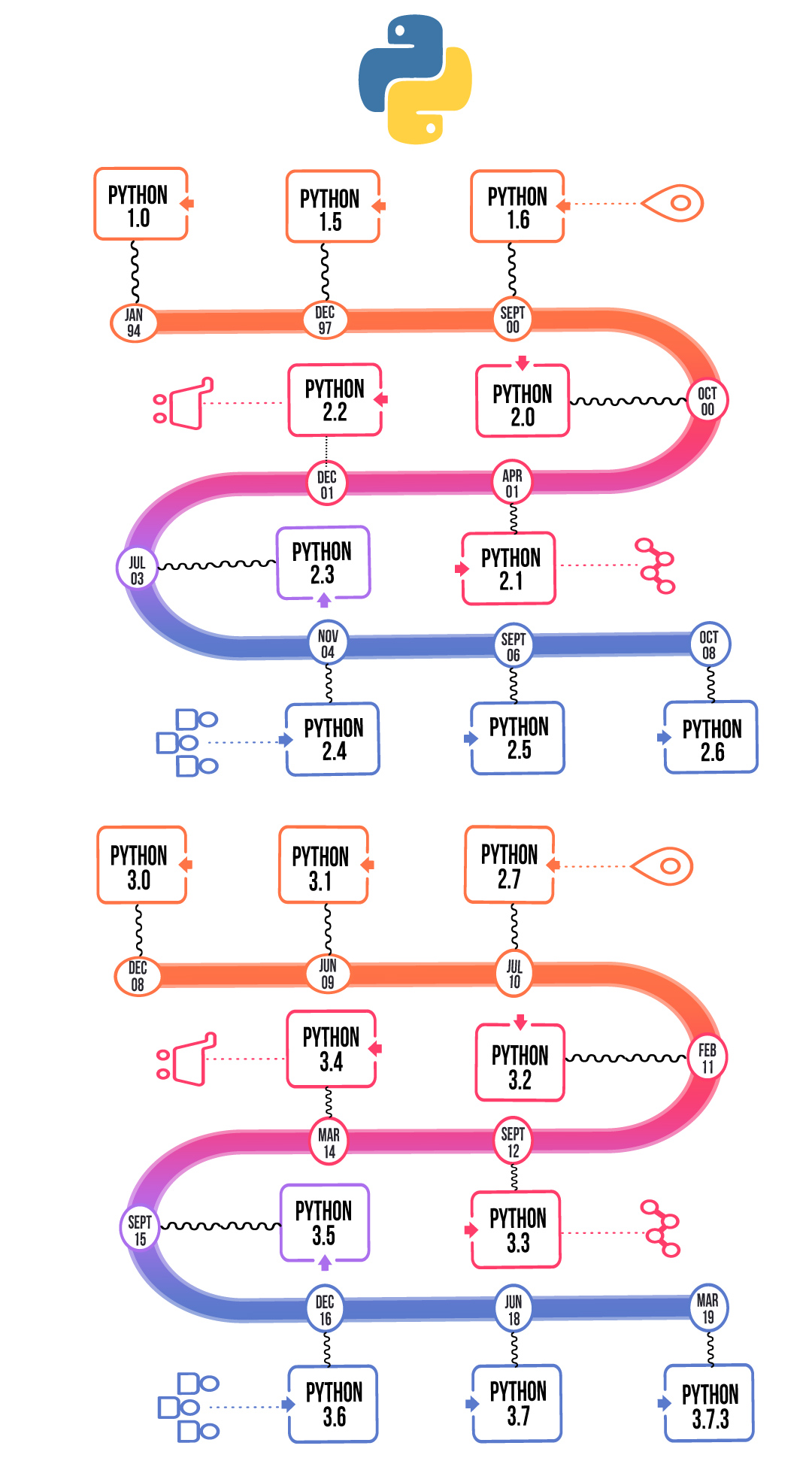
**Python History**

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
* In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
* In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
* Python 2.0 added new features like: list comprehensions, garbage collection system.
* On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
* *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by following programming languages:
  + ABC language.
  + Modula-3

**Python Version List**

* Python programming language is being updated regularly with new features and supports. There are lots of updations in python versions, started from 1994 to current release.
* A list of python versions with its released date is given below.

|  |  |
| --- | --- |
| Python Version | Released Date |
| 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 |
| Python 3.7 | June 27, 2018 |



**When, How and Why was python developed**

**When Python?**

Python was conceived in the late 1980s by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatics](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) (CWI) in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to the [ABC language](https://en.wikipedia.org/wiki/ABC_(programming_language)) (itself inspired by [SETL](https://en.wikipedia.org/wiki/SETL)), capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)) operating system.[[8]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-faq-created-8) 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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Cycle_detection) [garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and support for [Unicode](https://en.wikipedia.org/wiki/Unicode).

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility). Many of its major features were [backported](https://en.wikipedia.org/wiki/Backporting) 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](https://en.wikipedia.org/wiki/End-of-life_(product)) 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.

**Why Python?**

What makes Python so special? How does it happen that programmers, young and old, experienced and novice, want to use it? How did it happen that large companies adopted Python and implemented their flagship products using it?

There are many reasons – we’ve listed some of them already, but let’s enumerate them again in a more practical manner:

* it’s **easy to learn** – the time needed to learn Python is shorter than for many other languages; this means that it’s possible to start the actual programming faster;
* it’s **easy to teach** – the teaching workload is smaller than that needed by other languages; this means that the teacher can put more emphasis on general (language-independent) programming techniques, not wasting energy on exotic tricks, strange exceptions and incomprehensible rules;
* it’s **easy to use** for writing new software – it’s often possible to write code faster when using Python;
* it’s **easy to understand** – it’s also often easier to understand someone else’s code faster if it is written in Python;
* it’s **easy to obtain**, install and deploy – Python is free, open and multiplatform; not all languages can boast that.

**How Python?**

Python was created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) in 1989 as he was looking for an interesting project to keep him occupied during Christmas. First released in 1991, it design philosophy emphasises code readability (using whitespace indentation to delimit code blocks), use fewer code to express concepts (in compare with C++ or Java).

About the origin of Python, Van Rossum wrote in 1996:

Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).

**Advantages and Disadvantages of Python.**

**Advantages of Python**

Let’s see how Python dominates over other languages.

**1. Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be**extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers**more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do. It is also quite **easy to learn**,**understand**, and **code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is **freely available**. But not only can you [**download Python**](https://data-flair.training/blogs/install-python-windows/) for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to**code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

*Any doubts till now in the advantages of Python? Mention in the comment section.*

**Advantages of Python Over Other Languages**

**1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and [**machine learning**](https://data-flair.training/blogs/machine-learning-tutorials-home/), automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

### **Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

**1. Speed Limitations**

We have seen that Python code is executed line by line. But since [Python](https://www.python.org/) is interpreted, it often results in **slow execution**. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is **dynamically-typed**. This means that you don’t need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can**raise run-time errors**.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**Comparison Python vs C++**

## **Introduction**

|  |  |
| --- | --- |
| **Python** | **C++** |
| Python- python vs c++ - edureka  It is an interpreted, high-level, general-purpose programming language which helps the programmers write clear and logical code for small and large-scale projects. | C++ - python vs c++ - edureka  It is a general-purpose object-oriented programming language which allows procedural programming for intensive functions of CPU and provides control over hardware. |

## **Usage**

|  |  |
| --- | --- |
| **Python** | **C++** |
| Usage - Python vs C++ - edureka  It is easier to write a code in [Python](https://www.edureka.co/blog/python-tutorial/) as the number of lines is less comparatively. | Usage- python vs c++ - edureka    It is not easy to [write a code](https://www.edureka.co/blog/fibonacci-series-in-c%2B%2B/) in C++ in contrast to Python due to the complex syntax. |

## **Compilation**

|  |  |
| --- | --- |
| **Python** | **C++** |
| https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/Picture2-2-150x144.png  Python is an interpreted language and it runs through an interpreter during compilation. | https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/code-150x150.png  C++ is a pre-compiled programming language and doesn’t need any interpreter during compilation. |

## **Performance**

|  |  |
| --- | --- |
| **Python** | **C++** |
| python speed - python vs c++ - edureka  When it comes to Python vs C++, it is a dynamic language which reduces complexity when it comes to collaborating and optimizes programmer efficiency. | https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/pyspeed-300x158.png  C++ has the advantage of being a statically typed language. The performance crown goes to C++ for creating a more compact and faster runtime code. |

## **Scope of Variables**

|  |  |
| --- | --- |
| **Python** | **C++** |
| python scope - python vs c++ - edureka    In Python, variables are accessible even outside the [loop](https://www.edureka.co/blog/loops-in-python/). | c++ scope - python vs c++ - edureka  In C++, the scope of variables is limited within the loops. |

## **Functions**

|  |  |
| --- | --- |
| **Python** | **C++** |
| python function- python vs c++ - edureka[Python Functions](https://www.edureka.co/blog/python-functions) do not have restrictions on the type of the argument and the type of its return value. | https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/website-process-information-programming-technology_24640-23886-300x300.jpgIn C++, the [function](https://www.edureka.co/blog/function-overloading-and-overriding-in-c/) can accept and return the type of value which is already defined. |

## **Scope of Variables**

|  |  |
| --- | --- |
| **Python** | **C++** |
| python scope - python vs c++ - edureka    In Python, variables are accessible even outside the [loop](https://www.edureka.co/blog/loops-in-python/). | c++ scope - python vs c++ - edureka  In C++, the scope of variables is limited within the loops. |

## **Functions**

|  |  |
| --- | --- |
| **Python** | **C++** |
| python function- python vs c++ - edureka  [Python Functions](https://www.edureka.co/blog/python-functions) do not have restrictions on the type of the argument and the type of its return value. | https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/website-process-information-programming-technology_24640-23886-300x300.jpg In C++, the [function](https://www.edureka.co/blog/function-overloading-and-overriding-in-c/) can accept and return the type of value which is already defined. |

## **Popularity**

|  |  |
| --- | --- |
| **Python** | **C++** |
| https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/123-300x218.jpg  Python has huge community support. When it comes to popularity, beginner and novice programmers tend to turn towards Python. | https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/10-300x211.jpg  C++ also has dedicated followings online. But only the people who have some experience in the field show a lot of interest in C++. |

## **Salary**

|  |  |
| --- | --- |
| **Python** | **C++** |
| python salary- python vs c++ - edureka  Python developers can expect to make an [average salary](https://www.edureka.co/blog/python-career-opportunities-your-guide-to-a-career-in-python-programming) of $92,000 USD per year, which would roughly come out to be $7670 USD per month. | https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2019/07/c-salary-150x150.png  C++ developers are expected to earn around $95,000 USD per year, or almost $7920 USD per month. |