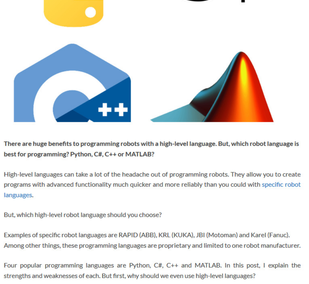
Python vs C++ vs C# vs MATLAB: Which Robot Language is Best?

July 21, 2018 · by Alex Owen-Hill · Leave a Comment



There are huge benefits to programming robots with a high-level language. But, which robot language is best for programming? Python, C#, C++ or MATLAB?

High-level languages can take a lot of the headache out of programming robots. They allow you to create programs with advanced functionality much quicker and more reliably than you could with specific robot languages.

But, which high-level robot language should you choose?

Examples of specific robot languages are RAPID (ABB), KRL (KUKA), JBI (Motoman) and Karel (Fanuc). Among other things, these programming languages are proprietary and limited to one robot manufacturer.

Four popular programming languages are Python, C#, C++ and MATLAB. In this post, I explain the strengths and weaknesses of each. But first, why should we even use high-level languages?

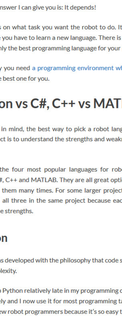
Why Use a High-Level Programming Language?

In the beginning, robots could only be programmed with low-level languages. Early robot languages like MAIL, VAL and SIGLA were introduced around 1973 and were very limited. Most languages could only control one specific robot to make basic movements, which made them very inflexible.

Years passed and programming languages became more advanced. People started to use general-purpose, high-level languages to control their robots. C++ entered the robotics landscape in 1992, Python in 1990 and MATLAB in 2012.

Although many robot languages are still single-purpose today, there are huge benefits to using high-level languages. For example:

- They allow you to add advanced functionality to a robot simply by adding an existing software library.
- You can reuse most of your code with different robots.
- Use advanced debugging tools: vendor-specific programming languages rarely provide tools for debugging.
- Solving problems is quick because many people use the languages and the community is usually helpful.



Which Programming Language is Best for Robotics?

This is one of the most common questions I hear from new roboticists. Two years ago, I wrote an article which addressed the question. It has been one of the most popular articles I've ever written.

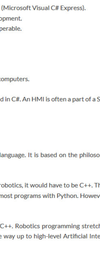
However, although it's a popular question, there is no simple answer.

The best answer I can give you is: It depends!

It depends on what task you want the robot to do. It also depends on your programming skills and how much time you have to learn a new language. There is really no "best" programming language for robotics. There is only the best programming language for your project and your programming skills.

That's why you need a programming environment which supports various languages. That way, you can choose the best one for you.

Python vs C#, C++ vs MATLAB: Which is Best?



With that in mind, the best way to pick a robot language for your project is to understand the strengths and weaknesses of each one.

Probably the four most popular languages for robotics are Python, C#, C++ and MATLAB. They are all great options and I have used them many times. For some larger projects, I have even used all three in the same project because each has its own unique strengths.

Python

Python was developed with the philosophy that code should be easy to read and that simplicity is better than complexity.

I picked up Python relatively late in my programming career. It was so easy to learn that I was able to use it immediately and I now use it for most programming tasks. Python is the language that I recommend most often to new robot programmers because it's so easy to get started with it.

Python's strengths for robot programming are:

- Easy to learn and easy to read.
- Gives access to many powerful libraries.
- Quick to write usable (and reusable) code.
- It's very popular so there's lots of help available from the Python community.

Python's weaknesses for robot programming are:

- Code can easily become messy for big projects.
- "Jack of all trades, master of none." It's good at doing lots of things but it does not excel at anything in particular.
- Sometimes hard to spot errors due to the fact it is an interpreted language, which can lead to problems.

In my opinion, Python is best for quick, small-to-medium robot programming projects. It's great if you want to access the powerful features provided by libraries and do not need real-time performance. However, if you want reliable, high-performance code, it might not be the best option.

C#

C# (pronounced C Sharp) was developed by Microsoft and released the early 2000s. Since then, C# has quickly gained popularity and is now one of the most used programming languages in the manufacturing industry.

Contrary to C++, C# is easy to learn. It's also excellent at automatically handles memory management. This is accomplished through the so-called "garbage collection" scheme.

C#'s strengths for robot programming are:

- It is easy to learn and integrate with large projects.
- There is a wide variety of libraries available.
- It has an excellent and free development environment (Microsoft Visual C# Express).
- Microsoft Visual Studio has good tools for team development.
- C# runs on the .NET Framework and it is highly interoperable.

C#'s weaknesses for robot programming are:

- Software Development is limited to Windows.
- You can't easily deploy your project to non-Windows computers.

Many HMI (Human Machine Interfaces) projects are developed in C#. An HMI is often a part of a SCADA (Supervisory Control and Data Acquisition) system.

C++

C++ is an object-oriented language which is based on the C language. It is based on the philosophy that performance is key and that code should be easy to organize.

If I was forced to choose only one programming language for robotics, it would have to be C++. This might seem a strange thing to say after having told you that I write most programs with Python. However, there is one big reason for my choice: performance.

If you are serious about robotics, I'd recommend learning C/C++. Robotics programming stretches from the lowest level (embedded motor and sensor control) all the way up to high-level Artificial Intelligence. C++ is one of the few languages which excels at all of these.

C++'s strengths for robot programming are:

- Potential for high performance (if your code is good).
- Access to lots of libraries (many Python libraries are just wrappers around C++ libraries)
- It's the lowest level programming language you can get above assembler (the level of 1's and 0's).
- Libraries for robotic hardware components are often written in C/C++.

C++'s weaknesses for robot programming are:

- Takes time to learn and even longer to learn to code properly.
- Usually requires lots of debugging.
- Writing programs takes a long time.
- Third-party libraries are often difficult to use.

In my opinion, C++ is best when you need high performance or need to interact with low-level robotic hardware. However, if you want to program quickly with the minimum of fuss, C++ is probably not the best option.

MATLAB

MATLAB is not just a programming language, it is an entire programming environment. Its name stands for "matrix laboratory" and it excels at matrix mathematics.

Matrices are a fundamental part of robotics, as we covered in the article Robot Euler Angles: The Essential Primer. MATLAB is widely used by engineers to analyze and simulate their robots. Over time, people have created interfaces to allow the software to control physical robots.

I personally have a love-hate relationship with MATLAB. I hate using it to control physical robots as the whole process often seems very convoluted. However, when it comes to data analysis there really is nothing better. This is just my opinion as I know roboticists who use it for everything.

MATLAB's strengths for robot programming are:

- A very powerful system for data and robot kinematic analysis.
- Quick to write usable code.
- Its robotics toolbox is widely used.
- Allows complex simulation.

MATLAB's weaknesses for robot programming are:

- It's not really designed to interface with robot hardware.
- As a proprietary language, it's expensive.
- Not easy to share your code as the other person also needs MATLAB.
- Not as many third-party libraries as other options.

In my opinion, MATLAB is best for data analysis and simulation tasks but not for much else. When it comes to actually programming the robot, I usually recommend another language.

In conclusion…

My top 3 takeaways are:

- Python is best if you want an easy life. It's good for smallish, quick robot projects.
- C# is better if you want a good balance between performance and quick results.
- C++ is best if you want performance.
- MATLAB is best for data analysis.

Whichever language you choose, make sure it is supported by your robot programming environment. The RoboDK API brings the benefits of your favorite high-level programming language to industrial robots.