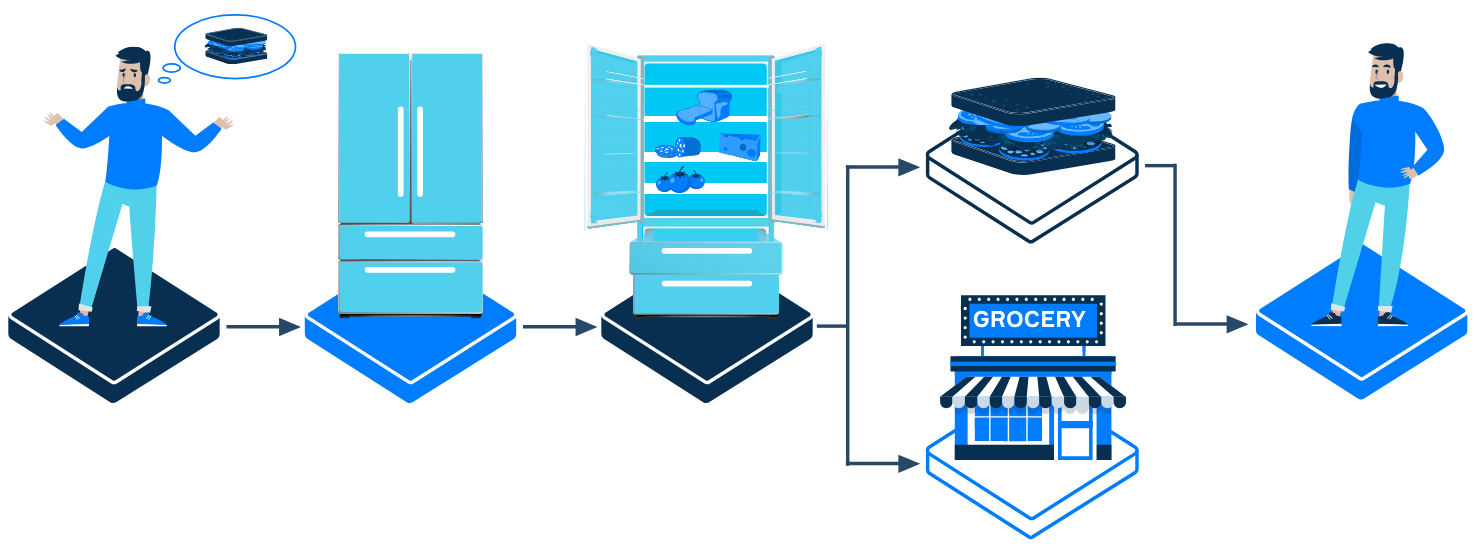
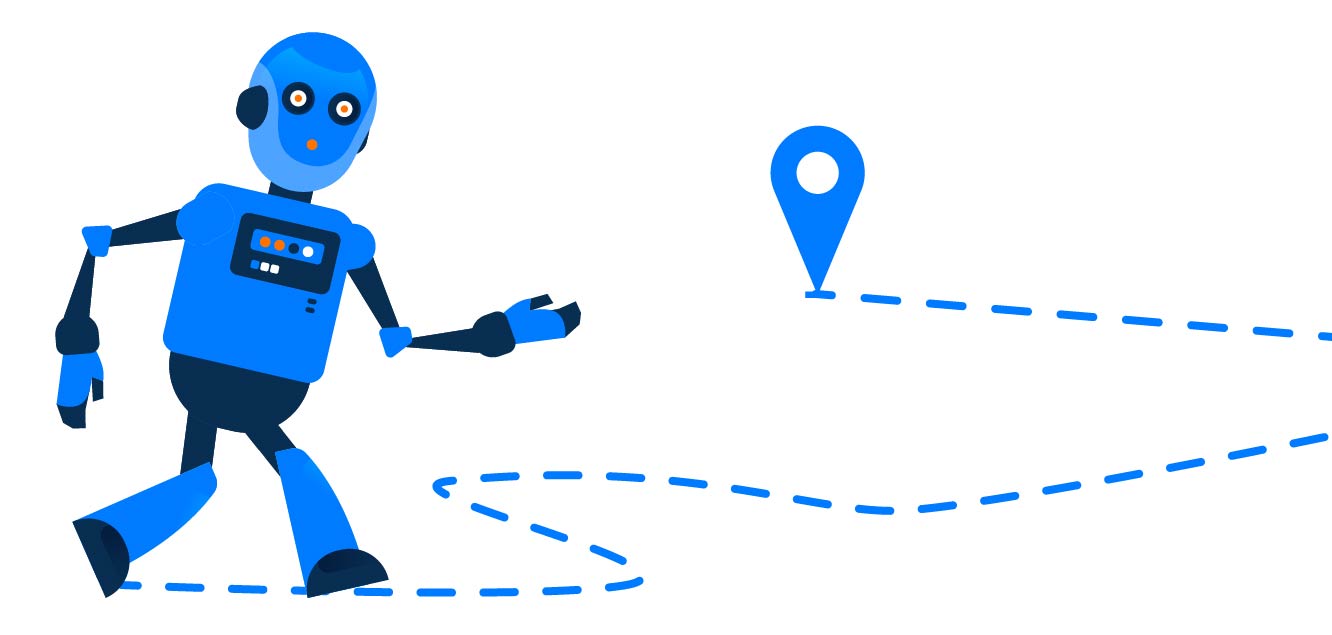
You have probably heard something about **algorithms** in real life. Simply put, an algorithm is a step-by-step sequence of actions you need to perform to achieve the desired result. It can be an algorithm for cooking a sandwich described by a recipe or an algorithm for getting dressed according to today's weather and your mood.



Among all algorithms, there is one special group called **computer algorithms.**They are usually created for and utilized by computers. In this topic, we will discuss in detail what computer algorithms are, as well as explain why it is important to learn them.

**Computer algorithms**

Computer algorithms are everywhere around us. Your smartphone is able to guide you through a city from one point to another using a certain algorithm. Other algorithms can control the behavior of your enemies in a computer game. Services like Google or Yahoo apply sophisticated algorithms to provide you with the most relevant results when you use them to search for information on the Web. Algorithms are also used to calculate the trajectory of rockets. And they even help doctors to determine diagnoses correctly!



An important difference between real-life and computer algorithms is that a computer cannot guess what we intend to do. If something goes wrong or an algorithm is not clear, a human can adjust the algorithm based on their experience. Computers cannot do the same. Let's get back to our sandwich algorithm. If we realize that we are out of bread, a human would think of going to the store and buying some more. On the other hand, a computer would report the absence of such an ingredient or proceed with its work without even noticing that, which would result in incorrect output. Thus, it is our responsibility to describe a computer algorithm precisely and unambiguously.

**Programs and algorithms**

As you may know, a program is a sequence of instructions to perform some tasks on a computer. The difference between programs and algorithms is that programs are written using a specific programming language while algorithms are usually described at a higher level than programming language statements. In other words, an algorithm is like an abstract schema, and a program can be its implementation.

All this also means that algorithms are language-agnostic: you can implement one algorithm using different programming languages. For example, you may use Java, Python, Kotlin, or other languages to implement the same algorithm.

Programming languages usually contain implementations of some basic algorithms for solving typical problems. These algorithms are provided in standard libraries, and software developers can reuse them instead of implementing a new one each time. However, to be able to use such algorithms correctly and efficiently and to understand how other developers use them, it is essential to learn these basic algorithms and get familiar with how they work under the hood. Moreover, the methods and tricks used in these approaches may come in handy while building your own algorithms as well. It is similar to solving exercises at school: these here are the examples you solve with your teacher during the class, and at home you have to solve other tasks using the methods discussed during the lesson.

Algorithms from standard libraries cannot cover all possible problems developers can encounter, so you will usually have to construct and implement your own algorithms for your problems. Classic algorithms serve as nice examples for learning the basic principles of algorithm construction.

Furthermore, there are so many well-known problems and their efficient solutions that standard libraries typically implement only the most prominent ones. Thus, sometimes you will need to implement a classic solution for a problem yourself, from scratch. This is yet another reason why it is essential to learn algorithms: you need to know which one and when to apply and how to implement it efficiently.

**Summary**

An algorithm is a sequence of actions you need to perform to achieve a certain result. An important group of algorithms is computer algorithms: the ones created for computers and utilized by them. There are several reasons why it is essential to learn computer algorithms:

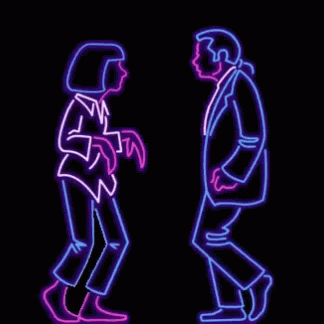
* Software developers often encounter tasks of the same type while working on different projects. For such typical tasks, programming languages provide ready-to-use algorithms in standard libraries. To utilize these algorithms efficiently, you need to understand how they work under the hood;
* There are numerous well-known algorithms that cannot be found as ready-to-use functions from libraries. Sometimes you may even encounter problems that are impossible to solve using only well-known algorithms. In both cases, you need to implement a suitable algorithm yourself. To be able to do that, you need to know basic algorithmic approaches, their pros and cons, and which one to apply in a particular case;
* Often you need not only to write the code yourself, but also to read the code written by other developers. If you want to understand the algorithms they might use, you need to know basic algorithms and algorithmic approaches as well;
* Implementing algorithms will help you improve your programming skills.

We believe that there are other reasons why learning algorithms is worth it. If you have some ideas, don't hesitate to write them in the comments.

Good luck with learning algorithms!

Read more on this topic in [Demystifying Data Science: Exploring the Top Languages](https://hyperskill.org/blog/post/demystifying-data-science-exploring-the-top-languages) on Hyperskill Blog.

# Demystifying Data Science: Exploring the Top Languages



**[by](https://hyperskill.org/profile/1254124" \t "_blank)**

**[Vsevolod Kuznetsov](https://hyperskill.org/profile/1254124" \t "_blank)**

Posted on

Apr 24, 2023

Data science is a rapidly growing and evolving field, and with it comes a demand for skilled data scientists. Dealing with this field requires programming skills among your statistical and mathematical skills. You can’t process tons of raw data without some specialized tools.

Luckily, we have a variety of high-level programming languages to choose from. They allow us to carry out numerical analysis, visualize the data, and obtain the insights we need, whether we work at a company, on a pet project, or participate in a Kaggle competition. Of course, analyzing the data is one of many things they can help you with. Most of them have libraries that are suitable for Deep Learning Algorithms too. Unfortunately, we can’t just stick to one language and use it mindlessly for the rest of our data science careers. Instead, start with one language and expand your knowledge with other languages and tools.

‍

All programming languages have their advantages and disadvantages. Some of them are slow, and some are fast. Some are low-level languages, and some are not. Some have easy-to-read syntax. Some have libraries that are more suitable to work with statistics, and some are a must to work with tabular data. It doesn’t matter what language you use; libraries will decrease your boilerplate work dramatically. The relatively new and promising languages like Julia compete with the titans of the field like Python and R by offering exciting and useful packages like Pluto.jl, aiming to be the only language you ever need.

Choosing a programming language that fits you can be daunting, especially if this is your first time working with any. This article will discuss the different languages suitable for data science, their strengths, and weaknesses. We will also give you tips on how to pick the right language for your needs. Without further ado, let’s talk about our first and most beloved language out there — Python.

## ****Python, go-to for data science****

Let’s start with Python, one of today's most popular programming languages, for data science, machine learning, and almost any other application. Python is a general-purpose language, as you might have guessed already. It’s an open-source language, making it easy for personal and enterprise projects. It also has well-written documentation where you can learn anything about Python anytime. All this builds another advantage — the vast community of Python enthusiasts. And they don’t waste any time. Thousands of libraries that hobbyists and big companies build are shared via PyPI (The Python Package Index with about 450,000 projects), which anyone can download and use for free. It’s the official repository, so most people upload their projects there to reach a wider audience. This is not the only repository of Python packages that you can use. There are also others, such as the Anaconda repository or GIT-like repositories. Pandas, NumPy, scikit-learn, matplotlib, and SciPy are some compelling and valuable libraries you can use with your projects. But before we dive into these, let’s get familiar with Python syntax and its standard library. Let’s start with a basic example.

Suppose that we have a .csv file with Netflix shows named[netflix\_titles.csv](https://www.kaggle.com/datasets/shivamb/netflix-shows/download?datasetVersionNumber=5). You can read it by using the built-in functionality.

import csv

from pprint import pprint

with open('netflix\_titles.csv', encoding='U8') as f:

csv\_data = list(csv.reader(f))

pprint(csv\_data[:2], width=60, compact=True)

It will print the following:

[['show\_id', 'type', 'title', 'director', 'cast', 'country',

'date\_added', 'release\_year', 'rating', 'duration',

'listed\_in', 'description'],

['s1', 'Movie', 'Dick Johnson Is Dead', 'Kirsten Johnson',

'', 'United States', 'September 25, 2021', '2020',

'PG-13', '90 min', 'Documentaries',

'As her father nears the end of his life, filmmaker '

'Kirsten Johnson stages his death in inventive and '

'comical ways to help them both face the inevitable.']]

As you can see, the variable csv\_data is a 2D array where the first row contains the names of the columns, and the rest is the actual data. What can we do with this? We can do anything, but implementing it may take much work. Even a simple table analysis can take ages. Wouldn’t it be better to have some functionality out of the box? Yes! Here comes the Pandas package. To install Pandas, write pip **install Pandas tabulate** in your terminal. It will install Pandas and all dependencies (e.g., NumPy, tabulate). Let’s try to use pandas:

import pandas

df = pandas.read\_csv('netflix\_titles.csv', encoding='U8')

filtered\_df = df[['type', 'title', 'listed\_in', 'description']]

print(filtered\_df[df['cast'].str.contains('Jenna Ortega',na=False)].to\_markdown())

As you can see, reading a .csv is easy. We also filter the columns we want to see in the final result. We are retrieving only the shows with ‘Jenna Ortega’ in the cast. The output is lovely.

| | type | title | listed\_in | description |

|-----:|:--------|:---------------------------------------------|:-----------------------------------------------------|:----------------------------------------------------------------------------------------------------------------------------------------------------------|

| 530 | TV Show | Richie Rich | Kids' TV, TV Comedies | After turning his veggies into green energy, Richie Rich earns a trillion dollars overnight and a new life filled with nonstop fun and adventure. |

| 855 | TV Show | Jurassic World Camp Cretaceous | Kids' TV, TV Action & Adventure, TV Sci-Fi & Fantasy | Six teens invited to attend a state-of-the-art adventure camp on Isla Nublar must band together to survive when the dinosaurs break out of captivity. |

| 1218 | Movie | YES DAY | Children & Family Movies, Comedies | A mom and dad who usually say no decide to say yes to their kids' wildest requests - with a few ground rules - on a whirlwind day of fun and adventure. |

| 2029 | Movie | The Babysitter: Killer Queen | Comedies, Horror Movies | Two years after Cole survived a satanic blood cult, he's living another nightmare: high school. And the demons from his past? Still making his life hell. |

| 3324 | Movie | The Massively Mixed-Up Middle School Mystery | Children & Family Movies, Comedies | After the school principal gets pranked, a curious crew of preteen super sleuths tests their detective skills to solve an underwater ruse.

As you can see, working with a 3rd party library is simple, and you don’t need to think about lots of mind-boggling stuff. For example, this output is properly formatted in a table-suited markdown syntax!

Let’s look at some of the most used libraries:

* **NumPy** — the library that lets you work with huge arrays of the same type efficiently. It’s written in one of the low-level programming languages (C) and is well-optimized! It has many mathematical functions, random generators, linear algebra routines, Fourier transforms, and more. NumPy is the backbone of most scientific data science and machine learning libraries.
* **Pandas** — the library to work with all kinds of tabular data. It’s not a coincidence that we touched it during our example. Along with all that, filtering, grouping, reading, and writing pandas is good with dates! Series is the second most used class in pandas.
* **Matplotlib** — the library to create all types of visualization. If you need animated, interactive, or static plots, matplotlib is the way to go.
* **SciPy**— the library that provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics, and many other classes of problems.
* **PyTorch** — the library for deep learning models that supports computer vision, NLP, and more
* **TensorFlow** — if you are interested in machine learning applications, this library is for you
* **scikit-learn** — machine learning, data mining. It has a bunch of analytics algorithms for things like finding patterns, making predictions, grouping things, reducing complexity, picking features, and more.

Let’s summarize:

* Python is easy to learn and use. The syntax is clear and readable.
* Python uses a lot of memory and may require much additional work to reduce it.
* Python is popular in machine learning and deep learning, with libraries such as TensorFlow, PyTorch, and scikit-learn.
* Python has powerful data manipulation capabilities, thanks to libraries such as NumPy and Pandas.
* Python is an interpreted language, meaning it is slower than compiled languages.
* Python has a huge and active community of developers and data scientists who provide many libraries for data analysis.
* Python is a general-purpose language that can be used for various tasks.
* Python can be used within a Jupyter Notebook. They can be run locally or by using Google Colab. This fact makes your work much simpler to share, collaborate with others and test everything on the fly!

## ****Julia, respected by scientific community****

Julia is the next language on our list, explicitly created for scientific and numerical computing. It is a versatile and compiled language that doesn't require extra layers to work with arrays, making it fast and efficient. It can achieve performance levels comparable to those of C! Additionally, Julia is significantly faster than Python, although it's worth noting that Python uses C libraries to keep up. Julia's syntax is straightforward, expressive, and more user-friendly than Python's, which makes it an excellent choice for beginners who want to focus solely on science.  
Like Python, Julia is a general-purpose language for scientific computing, numerical optimization, machine learning, data science, visualizations, etc. In addition, a growing and enthusiastic community also works on creating custom and helpful libraries. Julia has several extensive libraries for machine learning and data science. Besides of ScikitLearn.jl and TensorFlow.jl, it’s including Flux.jl for neural networks, DataFrames.jl for tabular data, MLJ.jl for building and evaluating machine learning models, and Pluto.jl as a reactive notebook! Let's put some code into practice to see if it’s worth it.

# first, let's load some packages.

using RDatasets

using DataFrames

using Plots

# let's load our dataset next

iris = dataset("datasets", "iris")

# finally, let's make a plot!

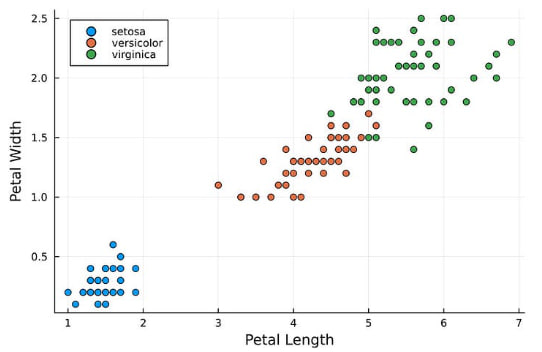
scatter(iris.PetalLength, iris.PetalWidth,

group = iris.Species,

xlabel = "Petal Length",

ylabel = "Petal Width")

The output is:



Output graph

As you can see, this is a classic iris dataset plotted using Julia within five lines. Another point for Julia!

Another exciting feature of the language is the **multiple dispatch**, which is the ability to pick the correct function based on the provided arguments.

function foobar(x::Array)

println("foobar(x::Array)")

end

function foobar(x::Tuple)

println("foobar(x::Array)")

end

function foobar(x::Tuple, y::Int)

println("foobar(x::Tuple, y::Int)")

end

function foobar(x::Tuple, y::Float64)

println("foobar(x::Tuple, y::Double)")

end

foobar((1, 2)) # foobar(x::Array)

foobar([1, 2]) # foobar(x::Array)

foobar([1, 2], 5) # MethodError

foobar((1, 2), 2.0) # foobar(x::Tuple, y::Double)

foobar((1, 2), 0) # foobar(x::Tuple, y::Int)

Multiple dispatch can handle different kinds of data and operations, which in the end, will make your code cleaner and simple to work with.

Let’s summarize:

* Julia is a fast and efficient language that can handle large-scale data processing and machine-learning tasks.
* Julia is a general-purpose and multi-paradigm language that allows you to write code in any way you want: OOP, functional, or declarative.
* Julia has a multiple dispatch system which makes your code more efficient and easy to maintain.
* Julia has a growing and enthusiastic community with a constantly growing number of high-quality libraries.
* A reproducible environment makes sharing and testing your code and collaborating with others as easy as using Python.
* There is little to say about the cons for Julia. It’s a fast-growing language that can fix any of these in no time!

## ‍****R, best for statistical modeling****

**R** is a system used for statistical analysis, computation, and graphics. It includes a language, a run-time environment with graphics, a debugger, access to specific system functions, and the ability to run programs stored in script files. It is important to note that **R** is not a general-purpose programming language but is explicitly designed for statistical computations and analysis. Let's dive into it further.

**R** is a data manipulation language. Like the two other languages we’ve discussed, you can read, clear, and transform data from various file formats, such as CSV, JSON, XML, SQL, and so on. You can use **R** to perform any kind of EDA, such as summary statistics, finding outliers, and handling missing values. Machine learning tasks are also possible—regression, classification, clustering, and natural language processing. You can also evaluate and compare different statistical models, tune, and deploy your models. Statistical computing and analysis, such as hypothesis testing, ANOVA, linear models, time series analysis, and Bayesian inference, are also possible with **R**.

You can also use **R** to simulate data and perform Monte Carlo methods. Data visualization is one of the strongest sides of the R language. You can create charts, graphs, and many other interesting and unusual visualization types by using libraries such as ggplot2, plotly, shiny, or without any! Let’s find out if we can plot anything without 3rd party libraries.

x <- seq(-pi, pi, 0.1)

plot(x, cos(x),

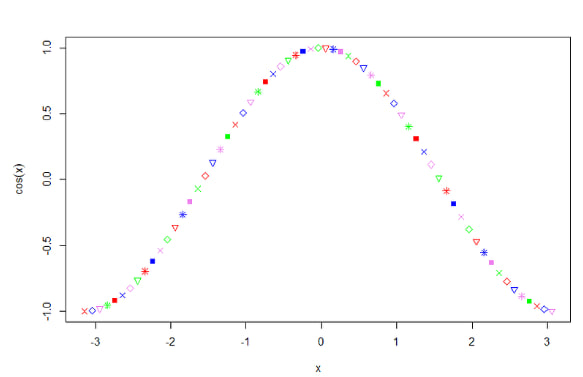
pch = c(4, 5, 6, 8, 15),

col = c("red", "blue", "violet", "green"),

xlab = expression(x),

ylab = expression(cos(x)))

And the output is:



Cos(x) output graph

You read that right! You can plot without importing any libraries!

Let’s summarize and find out if it has some cons too:

* **R** was designed as a statistical programming language. Created by statisticians for statisticians with lots of built-ins for data analysis, modeling, testing, and visualization.
* **R** has a big community of developers creating libraries for data science tasks and machine learning.
* **R** is a good choice for data science if you are interested in statistical operations or want to create interactive dashboards or reports.
* However, it has a steep learning curve, and because of its intuitive syntax (many ways to write the same thing and no preferable coding style), it may create even more problems at the beginning.
* Like Python, it’s slow and doesn’t handle memory well.
* The lack of built-in security tools may cause some troubles with sensitive data.
* The lack of parallel computing puts us even lower in our list of languages.

## ****SQL, a query language of some sort****

SQL is not really a programming language but a powerful tool. Its purpose lies within the abbreviation — **Structured Query Language**. SQL allows you to interact with relational database management systems by writing queries. Databases are (usually) well-optimized tables that consist of rows and columns. You can retrieve, analyze, manipulate, and even combine tables by making queries. Since SQL is everywhere, we think we absolutely must mention it. With SQL, you have lots of functionality, such as sorting, grouping, joining, filtering, and more. SQL supports functional programming for calculations, such as average, sum, and others.  
SQL is easy to learn and work with. You can integrate SQL with many programming languages and work with it using code or by writing SQL via your language of choice. For example, Python has a built-in sqlite3 library that allows you to work with the SQLite dialect.

Here is a made-up query to get you familiar with the SQL syntax:

SELECT A,

SUM(B.price \* B.quantity) AS total

FROM C

JOIN G ON C.D = G.D

JOIN B ON C.E = B.E

JOIN F ON B.F = F.F

WHERE F.G BETWEEN '2023-01-01' AND '2023-12-31'

GROUP BY A

ORDER BY total DESC;

This query joins multiple tables, filters data by date, groups by A, calculates the total by multiplying the price by quantity, and finally sorts it in descending order.

Does it have any cons?

SQL is faster than other languages, but it can choke with large computations.

SQL does not have libraries.

SQL is heavily using memory without any shame.

SQL errors can be hard to deal with.

‍

## ****Conclusion****

The list of languages we have presented to you is incomplete. MATLAB, Ruby, Java, C++, JavaScript, and more are suitable for some tasks. It all depends on your interests, goals, and preferences.

## ****Let’s answer some common questions from the developer community****

‍

### What should you learn first? 🤔

After learning about different programming languages, you may wonder which one to start with. This is a valid question. Some languages, such as Python and Julia, have similarities that make choosing difficult. You may also need to decide whether to use Python's Pandas library, an SQL language, Julia's statistics libraries, or R's statistics libraries.

We suggest choosing Python and sticking with it. This language is used by many universities worldwide to teach STEM and Data Science. You can find lectures and tutorials online, making it an excellent study resource. Python has been around for a long time and is a general-purpose language for software development. It has a straightforward syntax, making it easy for beginners to learn. With the help of online resources, you can write your first Data Science project in no time.

If you have some second thoughts about Python, here are more general tips for picking a language:

* If you know a language, sticking with it for data science may be easier. For example, if you know Python, you can learn data science libraries for Python faster than a new language.
* Different languages can do other things. For example, SQL is good for working with relational databases, while R has many libraries for statistics. Consider what you want to do and pick a language that can do it well.
* Some languages have more prominent and more active communities than others. A big community can mean more learning resources and more libraries to use.
* Performance and scalability matter if you’re working with big data or need to do complex tasks. Some languages, such as Julia, are made to be fast and efficient, while others may need workarounds to make them faster.
* Python, Julia, R, and SQL are all powerful languages with their advantages and disadvantages. It’s worth trying several to find the one that suits your needs best.

### Is Python best for data science? 🤔

Python is one of the most used languages for Data Science. According to the PYPL and TIOBE ratings, Python is the most searched and in-demand language. Given that, the interest in this language is still on the rise. So yes, you can think of Python as not the best but one of the best choices for Data Science.

Based on its vast community, we've already recommended it to you as the first language to learn. The community is what makes it really versatile. Machine Learning, Data Analysis, and Data visualization libraries, you can do it all. You can even make a website to place your insights! All of that is because of its community.

The bigger the community, the larger your chances to find like-minded people to collaborate and share your work with.

Companies of all sizes are also using Python as their language of choice, so the search for Python programmers never ends. But, if performance issues kick in, you should make a better decision based on the previous question.

### Is Python enough for data science? 🤔

Data Science is an interdisciplinary academic field; in other words, it's a huge field of all kinds of studies.

At first, Python is enough. It's enough to get you started and get your hands dirty.

It can be enough if you are an enthusiast with some projects in mind.

However, with a wide variety of tasks, you may need help with requiring you to write lots of plain Python.

In that case, a library from another language better fits your task. Tools like SAS may save you time with tasks that Python can't do.

Your company may change or extend its tools as well. In this case, you have no choice but to learn that tool or language. 👨🏼‍🎓

Data Scientists may use some other technologies that go beyond the scope of programming languages.

Cloud Platforms for computations, storage, and processing. A simple PowerPoint presentation may come in handy to communicate your data better, and so on.

Do not hesitate, and don't be afraid of these tools. Learn them and be open to any change you might face in your DS career.

### Coding is hard in common data science programming - is it hard to learn and master? 🤔

Good news! You can learn some basics within a week. 😍

Bad news! Mastering all concepts may take months. 😨

Do you have any prior knowledge of a language? Do you have free time? ⌚ Are you taking care of some relatives? 👴🏼👵🏼

All these questions, some unrelated to learning a programming language, are a big deal.

Based on your prior knowledge, you might learn data science libraries and tools right away.

With knowledge, you can carefully learn all the concepts and their purpose.

Surprisingly, some beginners can comprehend their code after only a month. 😱 This highlights the importance of understanding and utilizing code written by oneself and others, which is a crucial skill to master.

Chasing your goals and being willing to learn new concepts can make your dreams come true.🤩

With that said, with practice and persistence, you can learn anything. 😎

‍

### How to become a data scientist? 🤔

If you are interested in data science, we suggest some next steps. A good starting point is to obtain a bachelor's degree in computer science, mathematics, statistics, or a related field from a university.

Alternatively, there are online study platforms that offer specific data science paths and provide accredited and reliable certificates without requiring a degree. One such example is Coursera.

Another popular choice is online professional certification, which enables you to showcase your skills and prove that you are qualified for the job.

It is also important to continuously work on your programming skills and build your portfolio. Participating in data science projects, competitions, internships, and meetings can help you gain new knowledge and showcase your skills. Additionally, reading and watching new papers can keep you up-to-date with the latest developments in the field. Remember to keep learning and growing! 🤪

1 Read and understand the problem.

2 Think of an algorithm that solves the problem.

‍3 Write a program (implementation).

4 Test the program.

5 Search and correct the mistakes in the program.

6 Run the code and solve the problem.

As mentioned in the topic, you can find algorithms everywhere. Which of the following can serve as analogues of algorithms?

The correct answers are:

✅ **A recipe for baking a cake**  
✅ **A list of instructions provided by a GPS navigator**  
✅ **A guide “How to wash a cat”**  
✅ **A set of assembly instructions for building a piece of furniture**

Computer algorithms are language-agnostic, which means that one such algorithm can be implemented using different programming languages.

Computer algorithms must be described precisely and unambiguously.

Programs are written using a specific programming language, while algorithms are usually described at a higher level.

An algorithm is an abstract schema, and a program can be its implementation.