

Data Science

Software Engineering Perspectives

Expert Contributors

Pseudocode: What It Is and How to Write It

Pseudocode describes the distinct steps of an algorithm in a way that anyone with basic programming skills can understand. Here's how to write your own.



Written by [Sara A. Metwalli](#)



Image: Shutterstock / Built in

UPDATED BY

[Matthew Urwin](#) | Sep 03, 2024

REVIEWED BY

[Artem Oppermann](#)

As developers and [data scientists](#), we go through many stages, from getting an idea to reaching a valid, working implementation of it. We need to design and validate an [algorithm](#), apply it to the problem at hand and then test it for various input datasets.

What Is Pseudocode?

Pseudocode in data science or web development is a technique used to describe the distinct steps of an algorithm in a way that's easy for anyone with basic programming knowledge to understand.

In the initial state of [solving a problem](#), it helps to eliminate the limitations offered by a specific [programming language](#)'s syntax rules when we design or validate an algorithm. By doing this, we can focus our attention on the thought process behind the algorithm and how it will (or won't) work instead of focusing on our syntax.

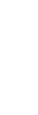
Here's where pseudocode comes to the rescue.

What Is Pseudocode?

We use pseudocode in various fields of programming, whether it be app development, [data science](#) or web development. Pseudocode is a technique used to describe the distinct steps of an algorithm in a manner that's easy to understand for anyone with [basic programming knowledge](#). Although pseudocode is a syntax-free description of an algorithm, it must provide a full description of the algorithm's logic so that moving from pseudocode to implementation is merely a task of translating each line into code using the syntax of any given [programming language](#).

CLARIFY YOUR CODE

[5 Ways to Write More Pythonic Code](#)



The Main Constructs of Pseudocode

At its core, pseudocode is the ability to represent six programming constructs (always written in uppercase): SEQUENCE, CASE, WHILE, REPEAT-UNTIL, FOR, and IF-THEN-ELSE.

These constructs — also called keywords — are used to describe the control flow of the algorithm.

1. **SEQUENCE** represents linear tasks sequentially performed one after the other.
2. **WHILE** is a loop with a condition at its beginning.
3. **REPEAT-UNTIL** is a loop with a condition at the bottom.
4. **FOR** is another way of looping.
5. **IF-THEN-ELSE** is a [conditional statement](#) changing the flow of the algorithm.
6. **CASE** is the generalization form of IF-THEN-ELSE.

PSEUDOCODE CONSTRUCTS

SEQUENCE
Input: READ, OBTAIN, GET
Output: PRINT, DISPLAY, SHOW
Compute: COMPUTE,
CALCULATE, DETERMINE
Initialize: SET, INIT
Add: INCREMENT, BUMP
Sub: DECREMENT

FOR
FOR iteration bounds
sequence
ENDFOR

WHILE
WHILE condition
sequence
ENDWHILE

CASE
CASE expression OF
condition 1: sequence 1
condition 2: sequence 2
...
condition n: sequence n
OTHERS:
default sequence
ENDCASE

REPEAT-UNTIL
REPEAT
sequence
UNTIL condition

IF-THEN-ELSE
IF condition THEN
sequence 1
ELSE
sequence 2
ENDIF

Although these six constructs are the ones we use most often you can theoretically use them to implement any algorithm. You might find yourself needing more based on your specific application. Perhaps the two most needed commands are:

1. Invoking classes or [calling functions](#) (using the CALL keyword).
2. Handling [exceptions](#) (using EXCEPTION, 5WHEN keywords).

EXTRA PSEUDOCODE CONSTRUCTS

CALLING CLASSES/ FUNCTIONS
CALL AvgAge with StudentAges
CALL Swap with CurrentItem and TargetItem
CALL getBalance RETURNING aBalance
CALL SquareRoot with orbitHeight RETURNING nominalOrbit

EXCEPTION HANDLING
BEGIN
statements
EXCEPTION
WHEN exception
statements to handle the exception
WHEN another exception
statements to handle the exception
END

Of course, based on the field you're working in, you might add more constructs (keywords) to your pseudocode glossary as long as you never use these keywords as variable names and ensure they're well-known within your field or company.

MORE FROM SARA A. METWALLI

[5 Git Commands That Don't Get Enough Hype](#)



How to Write Pseudocode

When writing pseudocode, everyone has their own style of presenting since humans are reading it and not a computer; pseudocode's rules are less rigorous than those of a [programming language](#). However, there are some simple rules that help make pseudocode more universally understood.

How to Write Pseudocode

1. Always capitalize the initial word (often one of the main six constructs).
2. Make only one statement per line.
3. Indent to show hierarchy, improve readability and show nested constructs.
4. Always end multi-line sections using any of the END keywords (ENDIF, ENDWHILE, etc.).
5. Keep your statements programming language independent.
6. Use the naming domain of the problem, not that of the implementation. For instance: "Append the last name to the first name" instead of "name = first+last."
7. Keep it simple, concise and readable.

Pseudocode Example

Following these rules helps you generate [readable pseudocode](#). Say an employer is designing a 10-question, multiple-choice quiz for employees to take after reading content on workplace safety. Those who get at least eight questions correct pass the quiz and receive a certificate of completion. Those who don't reach this benchmark must retake the quiz. The pseudocode for an algorithm determining those who pass and fail could look like this:

- IF employee gets eight or more questions correct
 - Display message: "Congratulations on passing the quiz!"
 - Transition to printable certificate of completion page
- ELSE
 - Display message: "Let's try again!"
 - Transition back to first page of quiz

Whether you're a [computer science major](#), went to bootcamp or took a [programming class](#), you've probably heard of pseudocode before. When I teach my students pseudocode, at first, they don't see the use of it; they think it's a waste of time. As they put it, "Why write code twice?" That might be correct in the case of simple, straightforward problems. However, as the complexity and the size of the project increases, programmers come to realize how generating pseudocode makes writing the actual code much easier. Pseudocode helps you realize possible problems or design flaws in the algorithm earlier in the development stage, which saves you more time and effort on [fixing bugs](#) and avoiding [errors down the road](#).

BE A BETTER DATA SCIENTIST

[4 Data Science Portfolio Projects You Need to Create](#)



Why Use Pseudocode?

Pseudocode Is Easier to Read

Often, programmers work alongside people from other fields such as mathematicians, managers and business partners. Using pseudocode to explain the mechanics of the code makes [communicating between different specialties](#) easier and more efficient.

Pseudocode Simplifies Code Construction

When the programmer goes through the process of developing and generating pseudocode, converting that into real code written in any programming language will become much easier and faster.

Pseudocode Is a Good Middle Point Between Flowchart and Code

Moving directly from the idea to the flowchart to the code is not always a smooth ride. That's where pseudocode presents a way to make the transition between the different stages somewhat simpler.

Pseudocode Is a Helpful Starting Point for Documentation

Documentation is an essential [aspect of building a good project](#) but starting documentation is often the most difficult part of the process. Pseudocode can represent a good starting point for what the documentation should include. Sometimes, programmers even include the pseudocode as a docstring at the beginning of the code file.

Pseudocode Allows for Quick Bug Detection

Since pseudocode is written in a human-readable format, it is easier to edit and [discover bugs](#) before actually writing a single line of code. We can edit pseudocode more efficiently than [testing](#), debugging and fixing actual code.

Pseudocode is an underestimated and under-utilized tool within the programming community, but a clear, concise, straightforward pseudocode can make a big difference on the road from idea to implementation — and a much smoother ride for the programmer.

Frequently Asked Questions

What is pseudocode?

Pseudocode is a representation of code used to demonstrate the implementation of an algorithm without actually doing so. It often acts as a rough draft of coding projects, and is written in an explainable manner to be understandable by programmers at any knowledge level.

How do you write a pseudocode?

Some general guidelines for writing pseudocode include the following:

- Make only one statement per line
- Capitalize the initial word of a line (which are often pseudocode constructs)
- Use indentations to show hierarchy and nested constructs
- Use the naming domain of the problem, not of the implementation
- Keep statements simple, readable and programming language independent

At the same time, pseudocode writing style can vary for each programmer, so it may not always include these guidelines.

What is an example of pseudocode?

Consider an online store that offers discounts on orders of \$50 or more. Following the principles of writing good pseudocode, the pseudocode for an algorithm determining if a user gets a discount could go as follows:

- IF price of an order is equal to or greater than \$50
 - Apply discount
 - Display message: "Discount has been applied!"
- ELSE
 - Do not apply discount
 - Display message: "Add more items to cart for discount!"

Is pseudocode easy to learn?

Pseudocode avoids programming syntax in favor of plain English, so those with basic programming skills can easily learn to read and write pseudocode.

What is the difference between pseudocode and code?

While both pseudocode and code describe an algorithm's logic, pseudocode forgoes programming syntax in favor of plain English. Developers can add programming syntax to translate pseudocode into actual coding instructions for implementing an algorithm. As a result, pseudocode merely acts as a master plan for writing code.

Recent Data Science Articles

[Meet Thinking Machines Lab: The Mysterious AI Startup Already Worth Billions](#)

[Nvidia's Alpaymado Could Usher in a New Era of Thinking Machines](#)

[48 Top AI Apps to Know](#)

About

[Our Story](#)

[Careers](#)

[Our Staff Writers](#)

[Content Descriptions](#)

Get Involved

[Recruit With Built In](#)

[Become an Expert Contributor](#)

Resources

[Customer Support](#)

[Share Feedback](#)

[Report a Bug](#)

[Tech Job Tools + Career Resources](#)

[Browse Jobs](#)

[Tech A-Z](#)

Tech Hubs

[Our Sites](#)

Built In is the online community for startups and tech companies. Find startup jobs, tech news and events.

