Pine Script® v6 Documentation

1 Welcome and Primer

TradingView Pine Script Programming Language

Table of Contents

Welcome to Pine Script® v6

Pine Script® v6 Documentation

User Manual / Welcome to Pine Script® v6

Welcome to Pine Script® v6

Pine Script® is TradingView's programming language. It allows traders to create their own trading tools and run them on our servers. We designed Pine Script as a lightweight, yet powerful, language for developing indicators and strategies that you can then backtest. Most of TradingView's built-in indicators are written in Pine Script, and our thriving community of Pine Script programmers has published more than 150,000 Community Scripts, half of which are open-source.

Requirements

It's our explicit goal to keep Pine Script accessible and easy to understand for the broadest possible audience. Pine Script is cloud-based and therefore different from client-side programming languages. While we likely won't develop Pine Script into a full-fledged language, we do constantly improve it and are always happy to consider requests for new features.

Because each script uses computational resources in the cloud, we must impose limits in order to share these resources fairly among our users. We strive to set as few limits as possible, but will of course have to implement as many as needed for the platform to run smoothly. Limitations apply to the amount of data requested from additional symbols, execution time, memory usage and script size.

Next

Redirect

Pine Script® v6 Documentation

First steps

Pine Script® v6 Documentation

User Manual / Pine Script® primer / First steps

First steps

Introduction

Welcome to the Pine Script® v6 User Manual, which will accompany you in your journey to learn to program your own trading tools in Pine Script. Welcome also to the very active community of Pine Script programmers on TradingView.

On this page, we present a step-by-step approach that you can follow to gradually become more familiar with indicators and strategies (also called *scripts*) written in the Pine Script programming language on TradingView. We will get you started on your journey to:

1. **Use** some of the tens of thousands of existing scripts on the platform.
2. **Read** the Pine Script code of existing scripts.
3. **Write** Pine scripts.

If you are already familiar with the use of Pine scripts on TradingView and are now ready to learn how to write your own, then jump to the Writing scripts section of this page.

If you are new to our platform, then please read on!

Using scripts

If you are interested in using technical indicators or strategies on TradingView, you can first start exploring the thousands of indicators already available on our platform. You can access existing indicators on the platform in two different ways:

- By using the chart’s “Indicators, metrics, and strategies” button.
- By browsing TradingView’s Community scripts, the largest repository of trading scripts in the world, with more than 150,000 scripts, half of which are free and *open-source*, which means you can see their Pine Script code.

If you can find the tools you need already written for you, it can be a good way to get started and gradually become proficient as a script user, until you are ready to start your programming journey in Pine Script.

Loading scripts from the chart

To explore and load scripts from your chart, click the “Indicators, metrics, and strategies” button, or use the forward slash / keyboard shortcut:

image

The dialog box that appears presents different categories of scripts in its left pane:

- **“Favorites”** lists the scripts you have “favorited” by clicking on the star that appears to the left of the script name when you hover over it.
- **“Personal”** displays the scripts you have written and saved in the Pine Editor. They are saved on TradingView’s servers.
- **“Technicals”** groups most TradingView built-in scripts, organized in four categories: “Indicators”, “Strategies”, “Profiles”, and “Patterns”. Most are written in Pine Script and available for free.
- **“Financials”** contains all built-in indicators that display financial metrics. The contents of that tab and the subcategories they are grouped into depend on the symbol currently open on the chart.
- **“Community”** is where you can search from the more than 150,000 published scripts written by TradingView users. The scripts can be sorted by one of the three different filters —

"Editors' picks" only shows open-source scripts hand-picked by our script moderators, "Top" shows the most popular scripts of all time, and "Trending" displays the most popular scripts that were published recently.

- **"Invite-only"** contains the list of the invite-only scripts you have been granted access to by their authors.

Here, we selected the "Technicals" tab to see the TradingView built-in indicators:

image

Clicking on one of the listed indicators or strategies loads the script on your chart. Strategy scripts are distinguished from indicators by a special symbol that appears to the right of the script name.

Browsing community scripts

To access the Community scripts feed from TradingView's homepage, select "Indicators and strategies" from the "Community" menu:

image

You can also search for scripts using the homepage's "Search" field, and filter scripts using different criteria. See this Help Center page explaining the different types of scripts that are available.

The scripts feed generates *script widgets*, which show the title and author of each publication with a preview of the published chart and description. Clicking on a widget opens the *script page*, which shows the publication's complete description, an enlarged chart, and any additional release notes. Users can boost, favorite, share, and comment on publications. If it is an open-source script, the source code is also available on the script page.

image

When you find an interesting script in the Community scripts, follow the instructions in the Help Center to load it on your chart.

Changing script settings

Once a script is loaded on the chart, you can double-click on its name or hover over the name and press the “Settings” button to bring up its “Settings/Inputs” tab:

image

The “Inputs” tab allows you to change the settings which the script’s author has decided to make editable. You can configure some of the script’s visuals using the “Style” tab of the same dialog box, and which timeframes the script should appear on using the “Visibility” tab.

Other settings are available to all scripts from the buttons that appear to the right of its name when you mouse over it, and from the “More” menu (the three dots):

image

Reading scripts

Reading code written by **good** programmers is the best way to develop your understanding of the language. This is as true for Pine Script as it is for all other programming languages. Finding good open-source Pine Script code is relatively easy. These are reliable sources of code written by good programmers on TradingView:

- The TradingView built-in indicators
- Scripts selected as Editors’ Picks
- Scripts by the authors the PineCoders account follows
- Many scripts by authors with high reputations and open-source publications

Reading code from Community scripts is easy; if there is no grey or red “lock” icon in the upper-right corner of the script widget, then the script is open-source. By opening the script page, you can read its full source code.

To see the code of a TradingView built-in indicator, load the indicator on your chart, then hover over its name and select the “Source code” curly braces icon (if you don’t see it, it’s because the indicator’s source is unavailable). When you click on the {} icon, the Pine Editor opens below the chart and displays the script’s code. If you want to edit the script, you must first select the “Create a working copy” button. You will then be able to modify and save the code. Because the working copy is a different version of the script, you need to use the Editor’s “Add to chart” button to add that new copy to the chart.

For example, this image shows the Pine Editor, where we selected to view the source code from the “Bollinger Bands” indicator on our chart. Initially, the script is *read-only*, as indicated by the orange warning text:

image

You can also open editable versions of the TradingView built-in scripts from the Pine Editor by using the “Create new” > “Built-in…” menu selection:

image

Writing scripts

We have built Pine Script to empower both new and experienced traders to create their own trading tools. Although learning a first programming language, like trading, is rarely **very** easy for anyone, we have designed Pine Script so it is relatively easy to learn for first-time programmers, yet powerful enough for knowledgeable programmers to build tools of moderate complexity.

Pine Script allows you to write three types of scripts:

- **Indicators**, like RSI, MACD, etc.
- **Strategies**, which include logic to issue trading orders and can be backtested and forward-tested.
- **Libraries**, which are used by more advanced programmers to package often-used functions that can be reused by other scripts.

The next step we recommend is to write your first indicator.

Next

First indicator

First indicator

The Pine Editor

The Pine Editor is where you will be working on your scripts. While you can use any text editor you want to write your Pine scripts, using the Pine Editor has many advantages:

- It highlights your code following Pine Script® syntax.
- It pops up syntax reminders when you hover over language constructs.
- It provides quick access to the Pine Script Reference Manual popup when you select `Ctrl` or `Cmd` and a built-in Pine Script construct, and opens the library publication page when doing the same with code imported from libraries.
- It provides an auto-complete feature that you can activate by selecting `Ctrl+Space` or `Cmd+I`, depending on your operating system.
- It makes the write/compile/run cycle more efficient because saving a new version of a script already loaded on the chart automatically compiles and executes it.

To open the Pine Editor, select the “Pine Editor” tab at the bottom of the TradingView chart.

First version

Let’s create our first working Pine script, an implementation of the MACD indicator:

1. Open the Pine Editor’s dropdown menu (the arrow at the top-left corner of the Pine Editor pane, beside the script name) and select “Create new/Indicator”.
2. Copy the example script code below by clicking the button on the top-right of the code widget.
3. Select all the code already in the editor and replace it with the example code.
4. Save the script by selecting the script name or using the keyboard shortcut `Ctrl+S`. Choose a name for the script (e.g., “MACD #1”). The script is saved in TradingView’s cloud servers, and is local to your account, meaning only you can see and use this version.
5. Select “Add to chart” in the Pine Editor’s menu bar. The MACD indicator appears in a *separate pane* under the chart.

```
//@version=6 indicator("MACD #1") fast = 12 slow = 26 fastMA = ta.ema(close, fast) slowMA = ta.ema(close, slow) macd = fastMA - slowMA signal = ta.ema(macd, 9) plot(macd, color = color.blue) plot(signal, color = color.orange)
```

Our first Pine script is now running on the chart, which should look like this:

image

Let’s look at our script’s code, line by line:

Line 1: `//@version=6`

This is a compiler annotation telling the compiler the script uses version 6 of Pine Script.

Line 2: `indicator("MACD #1")`

Declares this script as an indicator, and defines the title of the script that appears on the chart as “MACD #1”.

Line 3: `fast = 12`

Defines an integer variable `fast` as the length of the fast moving average.

Line 4: `slow = 26`

Defines an integer variable `slow` as the length of the slow moving average.

Line 5: `fastMA = ta.ema(close, fast)`

Defines the variable `fastMA`, which holds the result of the *EMA* (Exponential Moving Average) calculated on the close series, i.e., the closing price of bars, with a length equal to `fast` (12).

Line 6: `slowMA = ta.ema(close, slow)`

Defines the variable `slowMA`, which holds the result of the EMA calculated on the close series with a length equal to `slow` (26).

Line 7: `macd = fastMA - slowMA`

Defines the variable `macd` as the difference between the two EMAs.

Line 8: `signal = ta.ema(macd, 9)`

Defines the variable `signal` as a smoothed value of `macd` using the EMA algorithm with a length of 9.

Line 9: `plot(macd, color = color.blue)`

Calls the plot() function to output the variable `macd` using a blue line.

Line 10: `plot(signal, color = color.orange)`

Calls the plot() function to output the variable `signal` using an orange line.

Second version

The first version of our script calculated the MACD using multiple steps, but because Pine Script is specially designed to write indicators and strategies, built-in functions exist for many common indicators, including one for MACD: ta.macd().

Therefore, we can write a second version of our script that takes advantage of Pine’s available built-in functions:

```
//@version=6 indicator("MACD #2") fastInput = input(12, "Fast length")
slowInput = input(26, "Slow length") [macdLine, signalLine, histLine] =
ta.macd(close, fastInput, slowInput, 9) plot(macdLine, color =
color.blue) plot(signalLine, color = color.orange)
```

Note that:

- We add inputs so we can change the lengths of the moving averages from the script’s settings.
- We now use the ta.macd() built-in function to calculate our MACD directly, which replaces three lines of calculations and makes our code easier to read.

Let’s repeat the same process as before to create our new indicator:

1. Open the Pine Editor’s dropdown menu (the arrow at the top-left corner of the Pine Editor pane, beside the script name) and select “Create new/Indicator”.
2. Copy the example script code below. The button on the top-right of the code widget allows you to copy it with a single click.
3. Select all the code already in the editor and replace it with the example code.
4. Save the script by selecting the script name or using the keyboard shortcut `Ctrl+S`. Choose a name for your script that is different from the previous one (e.g., “MACD #2”).
5. Select “Add to chart” in the Pine Editor’s menu bar. The “MACD #2” indicator appears in a *separate pane* under the “MACD #1” indicator.

Our second Pine script is now running on the chart. If we double-click on the indicator’s name on the chart, it displays the script’s “Settings/Inputs” tab, where we can now change the fast and slow lengths used in the MACD calculation:

image

Let’s look at the lines that have changed in the second version of our script:

Line 2: `indicator("MACD #2")`

We have changed #1 to #2 so the second version of our indicator displays a different name on the chart.

Line 3: `fastInput = input(12, "Fast length")`

Instead of assigning a constant value to the variable, we used the input() function so we can change the length value from the script’s “Settings/Inputs” tab. The default value is 12 and the input field’s label is `"Fast length"`. When we change the value in the “Inputs” tab, the `fastInput` variable updates to contain the new length and the script re-executes on the chart with that new value. Note that, as our Pine Script Style guide recommends, we add `Input` to the end of the variable’s name to remind us, later in the script, that its value comes from a user input.

Line 4: `slowInput = input(26, "Slow length")`

As with `fastInput` in the previous line, we do the same for the slow length, taking care to use a different variable name, default value, and title string for the input field’s label.

Line 5: `[macdLine, signalLine, histLine] = ta.macd(close, fastInput, slowInput, 9)`

This is where we call the ta.macd() built-in function to perform all the first version’s calculations in only one line. The function requires four *parameters* (the values enclosed in parentheses after the function name). It returns *three* values, unlike the other functions we’ve used so far that only returned one. For this reason, we need to enclose the list of three variables receiving the function’s result in square brackets (to form a tuple) to the left of the = sign. Note that two of the values we pass to the function are the “input” variables containing the fast and slow lengths (`fastInput` and `slowInput`).

Lines 6 and 7: `plot(macdLine, color = color.blue)` and `plot(signalLine, color = color.orange)`

The variable names we are plotting here have changed, but the lines still behave the same as in our first version.

Our second version of the script performs the same calculations as our first, but we've made the indicator more efficient as it now leverages Pine's built-in capabilities and easily supports variable lengths for the MACD calculation. Therefore, we have successfully improved our Pine script.

Next

We now recommend you go to the Next Steps page.

Previous

Next

Next steps

Next steps

After your first steps and your first indicator, let us explore a bit more of the Pine Script® landscape by sharing some pointers to guide you in your journey to learn Pine Script.

”indicators” vs “strategies”

Pine Script strategies are used to backtest on historical data and forward test on open markets. In addition to indicator calculations, they contain `strategy.*()` calls to send trade orders to Pine Script’s broker emulator, which can then simulate their execution. Strategies display backtest results in the “Strategy Tester” tab at the bottom of the chart, next to the “Pine Editor” tab.

Pine Script indicators also contain calculations, but cannot be used in backtesting. Because they do not require the broker emulator, they use less resources and will run faster. It is thus advantageous to use indicators whenever you can.

Both indicators and strategies can run in either overlay mode (over the chart’s bars) or pane mode (in a separate section below or above the chart). Both can also plot information in their respective space, and both can generate alert events.

How scripts are executed

A Pine script is **not** like programs in many programming languages that execute once and then stop. In the Pine Script *runtime* environment, a script runs in the equivalent of an invisible loop where it is executed once on each bar of whatever chart you are on, from left to right. Chart bars that have already closed when the script executes on them are called *historical bars*. When execution reaches the chart's last bar and the market is open, it is on the *realtime bar*. The script then executes once every time a price or volume change is detected, and one last time for that realtime bar when it closes. That realtime bar then becomes an *elapsed realtime bar*. Note that when the script executes in realtime, it does not recalculate on all the chart's historical bars on every price/volume update. It has already calculated once on those bars, so it does not need to recalculate them on every chart tick. See the Execution model page for more information.

When a script executes on a historical bar, the close built-in variable holds the value of that bar's close. When a script executes on the realtime bar, close returns the **current** price of the symbol until the bar closes.

Contrary to indicators, strategies normally execute only once on realtime bars, when they close. They can also be configured to execute on each price/volume update if that is what you need. See the page on Strategies for more information, and to understand how strategies calculate differently than indicators.

Time series

The main data structure used in Pine Script is called a time series. Time series contain one value for each bar the script executes on, so they continuously expand as the script executes on more bars. Past values of the time series can be referenced using the history-referencing operator: []. `close[1]`, for example, refers to the value of close on the bar preceding the one where the script is executing.

While this indexing mechanism may remind many programmers of arrays, a time series is different and thinking in terms of arrays will be detrimental to understanding this key Pine Script concept. A good comprehension of both the execution model and time series is essential in understanding how Pine scripts work. If you have never worked with data organized in time series before, you will need practice to put them to work for you. Once you familiarize yourself with these key concepts, you will discover that by combining the use of time series with our built-in functions specifically designed to handle them efficiently, much can be accomplished in very few lines of code.

Publishing scripts

TradingView is home to a large community of Pine Script programmers and millions of traders from all around the world. Once you become proficient enough in Pine Script, you can choose to share your scripts with other traders. Before doing so, please take the time to learn Pine Script well-enough to supply traders with an original and reliable tool. All publicly published scripts are analyzed by our team of moderators and must comply with our Script Publishing Rules, which require them to be original and well-documented.

If you want to use Pine scripts for your own use, simply write them in the Pine Editor and add them to your chart from there; you don't have to publish them to use them. If you want to share your scripts with just a few friends, you can publish them privately and send your friends the browser's link to your private publication. See the page on Publishing for more information.

Getting around the Pine Script documentation

While reading code from published scripts is no doubt useful, spending time in our documentation will be necessary to attain any degree of proficiency in Pine Script. Our two main sources of documentation on Pine Script are:

- This Pine Script v6 User Manual
- Our Pine Script v6 Reference Manual

The Pine Script v6 User Manual, which is located on its separate page and in English only.

The Pine Script v6 Reference Manual documents what each language construct does. It is an essential tool for all Pine Script programmers; your life will be miserable if you try to write scripts of any reasonable complexity without consulting it. It exists in two formats: a separate page linked above, and the popup version. Access the popup version from the Pine Editor by either selecting `Ctrl` or `Cmd` and selecting a keyword, or by using the Editor's "More/Reference Manual…" menu. The Reference Manual is translated into multiple languages.

There are six different versions of Pine Script released. Ensure the documentation you use corresponds to the Pine Script version you are coding with.

Where to go from here?

This Pine Script v6 User Manual contains numerous examples of code used to illustrate the concepts we discuss. By going through it, you will be able to both learn the foundations of Pine Script and study the example scripts. Reading about key concepts and trying them out right away with real code is a productive way to learn any programming language. As you hopefully have already done in the First indicator page, copy this documentation's examples in the Editor and play with them. Explore! You won't break anything.

This is how the Pine Script v6 User Manual you are reading is organized:

- The Language section explains the main components of the Pine Script language and how scripts execute.
- The Concepts section is more task-oriented. It explains how to do things in Pine Script.
- The Writing section explores tools and tricks that will help you write and publish scripts.
- The FAQ section answers common questions from Pine Script programmers.
- The Error messages page documents causes and fixes for the most common runtime and compiler errors.
- The Release Notes page is where you can follow the frequent updates to Pine Script.

- The Migration guides section explains how to port between different versions of Pine Script.
- The Where can I get more information page lists other useful Pine Script-related content, including where to ask questions when you are stuck on code.

We wish you a successful journey with Pine Script… and trading!

Previous