# Installing and getting started with Perl

strict warnings say print chomp scalar $ STDIN

Prev Next

This is the first part of the Perl tutorial.

In this part you will learn how to install Perl on Microsoft Windows and how to start using it on Windows, Linux or on a Mac.

You'll get directions to set up your development environment, or in less grandiose words: which editor or IDE to use for writing Perl?

We will also see the standard "Hello World" example.

You can also watch the related video.

## Windows

For Windows we are going to use DWIM Perl. It is a package that contains the Perl compiler/interpreter, Padre, the Perl IDE and a number of extensions from CPAN.

In order to get started visit the website of DWIM Perl and follow the link to download **DWIM Perl for Windows**.

Go ahead, download the exe file and install it on your system. Before doing so, please make sure you don't have any other Perl installed.

They could work together but that would require some more explanations. For now let's have one single version of Perl installed on your system.

## Linux

Most Linux modern Linux distributions come with a recent version of Perl. For now we are going to use that version of Perl. For editor, you can install Padre - most Linux distribution offer it from their official package management system. Otherwise you can pick any regular text editor. If you are familiar with vim or Emacs, use the one you like. Otherwise Gedit might be a good simple editor.

## Apple

I believe Macs also come with Perl or you can easily install it via the standard installation tools.

# Editor and IDE

Even though it is recommended, you don't have to use the Padre IDE to write Perl code. In the next part I'll list a couple of editors and IDEs you can use for your Perl programming. Even if you select another editor I'd recommend - for the Windows users - to install the above mentioned DWIM Perl package.

It has lots of Perl extensions bundled so it will save you a lot of time down the road.

# Video

If you prefer, you can also watch the Hello world with Perl video I uploaded to YouTube.

In that case you might also want to check out the Beginner Perl Maven video course.

# First program

Your first program will look like this:

```
1. use 5.010;
   use strict;
   use warnings;

5. say "Hello World";
```

Let me explain it step-by-step.

# Hello world

Once you installed DWIM Perl you can click on "Start -> All programs -> DWIM Perl -> Padre" which will open the editor with and empty file.

Type in

```
1. print "Hello World\n";
```

As you can see statements in perl end with a semi-colon `;`. The `\n` is the sign we used to denote a newline. Strings are enclosed in double-quotes `"`. The `print` function prints to the screen. When this is executed perl will print the text and at the end it will print a newline.

Save the file as hello.pl and then you can run the code by selecting "Run -> Run Script" You will see a separate window showing up with the output.

That's it, you wrote your first perl script.

Let's enhance it a bit.

# Perl on the command line for non-Padre users

If you are not using Padre or one of the other IDEs you won't be able to run your script from the editor itself. At least not by default. You will need to open a shell (or cmd in Windows), change to the directory where you saved the hello.pl file and type in:

`perl hello.pl`

That's how you can run your script on the command line.

# say() instead of print()

Let's improve our one-line Perl script a bit:

First of all let's state the minimum version of Perl we would like to use:

```
1. use 5.010;
   print "Hello World\n";
```

Once you typed this in, you can run the script again by selecting "Run -> Run Script" or by pressing **F5**. That will automatically save the file before running it.

It is generally a good practice to tell what is the minimum version of perl your code requires.

In this case it also adds a few new features to perl including the `say` keyword. `say` is like `print` but it is shorter and it automatically adds a newline at the end.

You can change your code like this:

```
1. use 5.010;
   say "Hello World";
```

We replaced `print` by `say` and remove the `\n` from the end of the string.

The current installation you are using is probably version 5.12.3 or 5.14. Most modern Linux distributions come with version 5.10 or newer.

Unfortunately there are still places using older versions of perl. Those won't be able to use the `say()` keyword and might need some adjustments to the examples later. I'll point out when I am actually using features that require version 5.10.

# Safety net

In addition in every script I'd strongly recommend to make some further modifications to the behavior of Perl. For this we add 2, so called pragmatas, that are very similar to compiler flags in other languages:

```
1. use 5.010;
   use strict;
   use warnings;

5. say "Hello World";
```

In this case the `use` keyword tells perl to load and enable each pragma.

`strict` and `warnings` will help you catch some common bugs in your code or sometimes even prevent you from making them in the first place. They are very handy.

# User Input

Now let's improve our example by asking the user her name and including it in the response.

```
1. use 5.010;
   use strict;
   use warnings;

5. say "What is your name? ";
   my $name = <STDIN>;
   say "Hello $name, how are you?";
```

`$name` is called a scalar variable.

Variables are declared using the **my** keyword. (actually that's one of the requirements `strict` adds.)

Scalar variables always start with a `$` sign. The <STDIN> is the tool to read a line from the keyboard.

Type in the above and run it by pressing F5.

It will ask for your name. Type in your name and press ENTER to let perl know you have finished typing in your name.

You will notice that the output is a bit broken: The comma after the name appears on a newline. That's because the ENTER you pressed, when typing in your name, got into the `$name` variable.

# Getting rid of newlines

```
1. use 5.010;
   use strict;
   use warnings;

5. say "What is your name? ";
   my $name = <STDIN>;
   chomp $name;
   say "Hello $name, how are you?";
```

It is such a common task in Perl, that there is a special function called `chomp` to remove the trailing newlines from strings.

# Conclusion

In every script you write you should **always** add `use strict;` and `use warnings;` as the first two statements. It is also very recommended to add `use 5.010;`.

# Exercises

I promised exercises.

Try the following script:

```
1. use strict;
   use warnings;
   use 5.010;

5. say "Hello ";
```

```
    say "World";
```

It did not show on one line. Why? How to fix it?

# Exercise 2

Write a script that asks the user for two numbers, one after the other. Then prints out the sum of the two numbers.

# What's next?

The next part of the tutorial is about editors, IDEs and development environment for Perl.