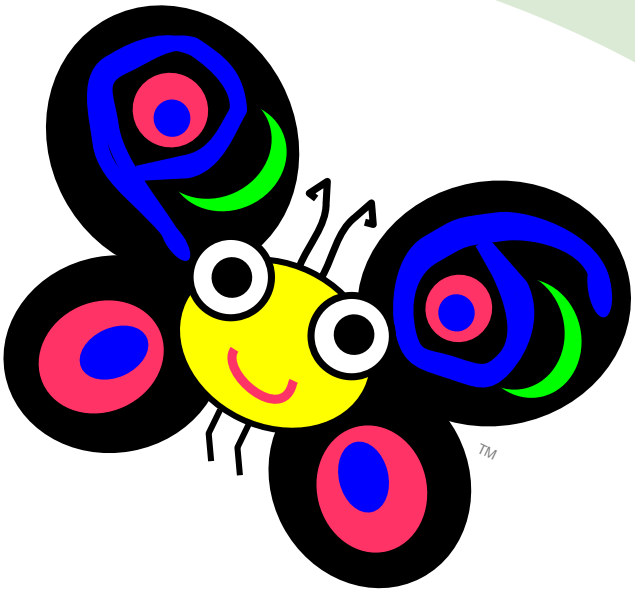
Perl 6

Rakudo Perl 6 on MoarVM (see Rakudo Star, rakudo.org) is currently the most advanced implementation of Perl 6. Any implementation that passes the official test suite can call itself "Perl 6". For other implementations (Niecza, Perlito, Pugs) at various levels of maturity and completeness, see perl6.org/compilers/

Perl 6 now runs on two virtual machines: MoarVM and JVM. MoarVM offers a wider support for specified features than Perl 6 on JVM (work is in progress to add JavaScript as a virtual machine). MoarVM is solely developed for Perl 6 and not (yet) used for other programming languages.



Why the '6' in Perl 6?
Perl 6 developers don't need to keep Perl 6 backward compatible to Perl 5, and have changed and extended the language, and added new syntactic and semantic features that enable more power and expressiveness. This means a major break in syntactic and semantic compatibility from Perl 5, thus the increase from 5 to 6. However, this does not mean that Perl 5 is going away anytime soon. In fact, quite the opposite. Both Perl 5 and Perl 6 have very active developer communities which mold the languages.

There are many reasons why you would be interested in Perl 6

General

- Perl 6 is a clean, modern, multi-paradigm language; it offers procedural, object-oriented AND functional programming methodologies.
- Easy to use consistent syntax, using invariable sigils for data-structures.
- Perl 6 is a very mutable language (define your own functions, operators, traits and data-types, which modify the parser for you).
- Adding a custom operator or adding a trait is as simple as writing a subroutine.
- Advanced error reporting based on introspection of the compiler/ runtime state. This means more useful, more precise error messages.
- Multiple versions of a module can be installed and loaded simultaneously.
- System administration is simplified due to simpler update/upgrade policies.
- Runtime optimization of hot code paths during execution (JIT), by inlining small subroutines and methods.
- Runs on small (e.g. Raspberry Pi) and large multi-processor hardware.
- Garbage collection based: no timely destruction, so no ref-counting necessary. Use phasers for timely actions.
- Fewer lines of code allow for more compact program creation. Huffman-coding of names allows for better readability.

Text-processing

- Full grapheme based Unicode support, including Annex #29, meaning almost unparalleled excellent Unicode support.
- Regular expressions are cleaned up, made more readable, taken to the next level of usability, with a lot more functionality. Named regular expressions are made possible for ease of use.
- Extensible grammars for parsing data or code (which Perl 6 uses to parse itself).
- Execute code at any time during parsing of a grammar, or when a certain match occurred.

Scoping

- Dynamic variables provide a lexically scoped alternative to global variables.
- Emphasis on composability and lexical scoping to prevent "action at a distance". For example, imports are always lexically scoped.
- Easy to understand consistent scoping rules and closures.
- Phasers (like `BEGIN` / `END`) allow code to be executed at scope entry / exit, loop first / last / next and many more special contexts.

Object-Oriented Programming

- Powerful object orientation, with classes and roles (everything can be seen as an object). Inheritance. Subtyping. Code-reuse.
- Introspection into objects and meta-objects (turtles all the way down).
- Meta Object Protocol allowing for meta-programming without needing to generate / parse code.
- Subroutine and method signatures for easy unpacking of positional and named parameters, and data structures.
- Methods can be mixed into any instantiated object at runtime, e.g. to allow adding out-of-band data.

Typing

- Multi dispatch on identically named subroutines/methods with different signatures, based on arity, types and optional additional code.
- Compile time error reporting on unknown subroutines / impossible dispatch.
- Optional gradual type-checking at no additional runtime cost. With optional type annotations.
- Easy command-line interface accessible by `MAIN` subroutine with multiple dispatch and automated usage message generation.

Concurrency, Parallelism, Asynchrony

- High level concurrency model, both for implicit as well as explicit multi-processing, which goes way beyond primitive threads and locks. Perl 6's concurrency offers a rich set of (composable) tools.
- Multiple-core computers are getting used more and more, and with Perl 6 these can be used thanks to parallelism, both implicit (e.g. with the `>>.` method) and explicit (`start { code }`). This is important, because Moore's Law is ending.
- Structured language support is provided to enable programming for asynchronous execution of code.
- Supplies allow code to be executed when something happens (like a timer, or a signal, or a file-system event, or gui events).
- The keywords react / whenever / supply allow easy construction of interactive, event driven applications.

Data-structures

- Junctions allowing easy checking of multiple possibilities, e.g. `$a == 1|3|42` (meaning is $a equal to `1` or `3` or `42`).
- Lazy evaluation when possible, eager evaluation when wanted or necessary. This means, for example, lazy lists, and even infinite lazy lists, like the Fibonacci sequence, or all prime numbers.
- Lazy lists defined with a simple iterator interface, which any class can supply by minimally supplying a single method.
- Native data types for faster, closer to the metal, processing.
- Floating point math without precision loss because of Rats (rational numbers).
- Large selection of data-types, plus the possibility to create your own types.
- Multi-dimensional shaped and/or native arrays with proper bounds checking.
- Automatic generation of hyper-operators on any operator (system or custom added).

Interoperability

- Interfacing to external libraries in C / C++ is trivially simple with NativeCall.
- Interfacing with Perl 5 (CPAN) / Python modules is trivially simple with `Inline::Perl5` and `Inline::Python`.
- Perl 6 runs on a variety of back-ends. Currently MoarVM & JVM, JavaScript is in development, more may follow.

Download, use

perl6.org/getting-started/	Getting started
rakudo.org/how-to-get-rakudo/	For most people, download and installation instructions can be found here
github.com/tadzik/rakudobrew/	The simplest way to install Perl 6 is to use Rakudobrew
moarvm.org	MoarVM

Useful places to start learning Perl 6

perl6intro.com	Perl 6 Introduction
learnxinyminutes.com/docs/perl6/	Perl 6 version of Learn x in y minutes
rosettacode.org/wiki/Category:Perl_6	Perl 6 section of solving problems with different programming languages
perl6advent.wordpress.com	Perl 6 Advent calendar: every day leading up to Advent, a new interesting Perl 6 example
p6weekly.wordpress.com	Perl 6 Weekly, with the latest developments in Perl 6
6guts.wordpress.com	Perl 6 Guts, the newest innards of Perl 6 explained by Jonathan Worthington
strangelyconsistent.org	Strangely Consistent, an insightful blog by Carl Masak
pl6anet.org	Perl 6 Planet, a collection of blogs and articles
szabgab.com/perl6.html	Screencasts about Perl 6, by Gabor Szabo
blogs.perl.org/users/damian_conway/	If you ever have the chance to attend a presentation by Damian, enjoy!
design.perl6.org	Historical Perl 6 Design Documents: This is the core of what is Perl 6

Good luck and—as the Perl 6 community often says—have fun!

Run Rakudo Perl 6

To run a Perl 6 program with Rakudo Perl 6, include the installation directory in your system `PATH` variable and issue a command like:

```
$ perl6 hello.pl
```

(this must look familiar to any Perl developer).

If you invoke Rakudo Perl 6 without an explicit script to run, it enters a small interactive mode that allows the execution of Perl 6 statements from the command line.

Modules for Rakudo Perl 6

A growing list of modules is published on modules.perl6.org and at the moment of writing this, the number of modules exceeded 650. Of course, this can not (yet) be compared with the immense number of Perl 5 modules on CPAN. Nevertheless, the number of modules grows every week and they make Perl 6 more useful.

Especially useful is `Inline::Perl5`, enabling the use of virtually all Perl 5 code within your Perl 6 code. The other way is also catered for: with `Inline::Perl6` virtually all Perl 6 code can be used within your Perl 5 code.

Many important topics have been covered and more will be added. The module management tools for Rakudo Perl 6, `panda` or `zef`, makes your life as a module user easier.

In 2014 it has been made possible to upload and download Perl 6 modules to CPAN. The work is not complete yet to make it possible to install from CPAN, but that will be done soon.

```
# Words from file
for 'dict.txt'.IO.words -> $word {
  say "$word probably rhymes with Perl"
    if $word ~~ /[ea?|u|i] rl $/;

  say "$word is a palindrome"
    if $word eq $word.flip;
}
```

```
# Infinite Lazy Lists
my @fib = 0, 1, *+* ... *;
say "Fibonacci number #8 is @fib[7]";
```

```
# Custom operators
sub postfix:<!> (Int $n) {
    fail "Not a Natural Number in Factorial"
        if $n < 0;
    [*] 2..$n
}

use Test;
isa-ok (-1)!, Failure, "Factorial for -1 fails";
ok 0! == 1, "Factorial for 0";
ok 1! == 1, "Factorial for 1";
ok 5! == 120, "Factorial for a larger integer";
```

Getting involved

If you are inspired now and want to contribute to the Perl 6 community, there are some resources available to you.

World Wide Web: the Perl 6 homepage at perl6.org links to many useful resources.

IRC: the channel #perl6 on irc.freenode.net discusses all things Perl 6. The people are very friendly and very busy developing Perl 6. Keep an eye on this to stay up-to-date. The channel is logged, and you can read back to see what has been discussed: irclog.perlgeek.de/perl6/today

Mailing lists: send an email with subject '`subscribe`' to:

perl6-announce-subscribe@perl.org	Announcements and news. Low traffic.
perl6-users-subscribe@perl.org	User questions and discussions regarding the Perl 6 language and compilers.
perl6-language-subscribe@perl.org	For issues regarding the Perl 6 language specification.
perl6-compiler-subscribe@perl.org	For issues regarding various Perl 6 compilers

Get ready to party!

The official full first stable version, Perl 6 version 6.c, was released on Christmas of 2015. The whole world can start using it now and see how awesome programming can be!

Perl 6's mottos remain the same as they have been for Perl all along: *"Perl is different. In a nutshell, Perl is designed to make the easy jobs easy, without making the hard jobs impossible."* and *"There Is More Than One Way To Do It"*. Now with even more `-Ofun` added.

Perl 6 is here. It exists. It is ready to be used. Enjoy.