What is Perl?

Perl is a programming language. Perl stands for Practical Report and Extraction Language. You'll notice people refer to 'perl' and "Perl". "Perl" is the programming language as a whole whereas 'perl' is the name of the core executable. There is no language called "Perl5" -- that just means "Perl version 5". Versions of Perl prior to 5 are very old and very unsupported.

Some of Perl's many strengths are:

- **Speed of development.** You edit a text file, and just run it. You can develop programs very quickly like this. No separate compiler needed. I find Perl runs a program quicker than Java, let alone compare the complete modify-compile-run-oh no-forgot-that-semicolon sequence.
- **Power.** Perl's regular expressions are some of the best available. You can work with objects, sockets...everything a systems administrator could want. And that's just the standard distribution. Add the wealth of modules available on CPAN and you have it all. Don't equate scripting languages with toy languages.
- **Usuability.** All that power and capability can be learnt in easy stages. If you can write a batch file you can program Perl. You don't have to learn object oriented programming, but you can write OO programs in Perl. If autoincrementing non-existent variables scares you, make perl refuse to let you.There is always more than one way to do it in Perl. You decide your style of programming, and Perl will accommodate you.
- **Portability.** On the Superhighway to the Portability Panacea, Perl's Porsche powers past Java's jaded jalopy. Many people develop Perl scripts on NT, or Win95, then just FTP them to a Unix server where they run. No modification necessary.
- **Editing tools** You don't need the latest Integrated Development Environment for Perl. You can develop Perl scripts with any text editor. Notepad, vi, MS Word 97, or even direct off the console. Of course, you can make things easy and use one of the many freeware or shareware programmers file editors.
- **Price.** Yes, 0 guilders, pounds, dmarks, dollars or whatever. And the peer to peer support is also free, and often far better than you'd ever get by paying some company to answer the phone and tell you to do what you just tried several times already, then look up the same reference books you already own.

What can I do with Perl ?

Just two popular examples :

The Internet

Go surf. Notice how many websites have dynamic pages with .pl or similar as the filename extension? That's Perl. It is the most popular language for CGI programming for many reasons, most of which are mentioned above. In fact, there are a great many more dynamic pages written with perl that may not have a *.pl* extension. If you code in Active Server Pages, then you should try using ActiveState's PerlScript. Quite frankly, coding in PerlScript rather than VBScript or JScript is like driving a car as opposed to riding a bicycle. Perl powers a good deal of the Internet.

Systems Administration

If you are a Unix sysadmin you'll know about sed, awk and shell scripts. Perl can do everything they can do and far more besides. Furthermore, Perl does it much more efficiently and portably. Don't take my word for it, ask around.
If you are an NT sysadmin, chances are you aren't used to programming. In which case, the advantages of Perl may not be clear. Do you need it? Is it worth it?
A few examples of how I use Perl to ease NT sysadmin life:

- **User account creation**. If you have a text file with the user's names in it, that is all you need. Create usernames automatically, generate a unique password for each one

and create the account, plus create and share the home directory, and set the permissions.

- **Event log munging.** NT has great Event Logging. Not so great Event Reading. You can use Perl to create reports on the event logs from multiple NT servers.
- **Anything else** that you would have used a batch file for, or wished that you could automate somehow. Now you *can*.

Perl Backgrounder: Versions and Naming Conventions

Variables

- single word: atom, chain
- multi word: atomName, centralAtomName
- constants: CALPHA_ATOM_NAME or ATOM_NAME_CALPHA, ATOM_NAME_CBETA

Perl is not type-safe and this can cause confusion and errors. Use a limited prefix notation for such common basic types as array, hash, FileHandle.

- array refs ('a' prefix): aAtoms, aChains
- hash refs ('h' prefix): hNames2Places, hChains
- FileHandle objects ('fh' prefix): fhIn, fhOut, fhPdb
- or ("ist"=input stream, "ost"=output stream): ostPdb, istMsa

Functions

- single word: Trim()
- multi word: OpenFilesForReading()

Modules (packages that are not classes)

- single word: Assert
- multi word: FileIoHelper

Classes

As for modules but with 'C' prefix: CStopwatch, CWindowPanel, Pdb::CResidue

Instance methods

- public method: plot(), getColour(), classifyHetGroups()
- private method: _plot(), _getColour(), _classifyHetGroups()
- accessor methods same as JavaBeans: getProperty(), setProperty(), isProperty()

Perl, perl or PeRl?

There is also a certain amount of confusion regarding the capitalization of Perl. Should it be written Perl or perl? Larry Wall now uses —Perl|| to signify the language proper and —perl|| to signify the implementation of the language.

Perl History Version	Date	Version Details
Perl 0		Introduced Perl to Larry Wall's office associates
Perl 1	Jan 1988	Introduced Perl to the world
Perl 2	Jun 1988	Introduced Harry Spencer's regular expression package
Perl 3	Oct 1989	Introduced the ability to handle binary data
Perl 4	Mar 1991	Introduced the first —Camel|| book (Programming Perl, by Larry Wall, Tom Christiansen, and Randal L Schwartz; O'Reilly & Associates). The book drove the name change, just so it could refer to Perl 4, instead of Perl 3.
Feb 1993	Feb 1993	The last stable release of Perl 4
Perl 5	Oct 1994	The first stable release of Perl 5, which introduced a number of new features and a complete rewrite.
Perl .005_02	Aug 1998	The next major stable release
Perl .005_03	Mar 1999	The last stable release before 5.6

Main Perl Features:

1. **Perl Is Free:**

Perl's source code is open and free anybody can download the C source that constitutes a Perl interpreter. Furthermore, you can easily extend the core functionality of Perl both within the realms of the interpreted language and by modifying the Perl source code.

2. Perl Is Simple to Learn, Concise, and Easy to Read:

It has a syntax similar to C and shell script, among others, but with a less restrictive format. Most programs are quicker to write in Perl because of its use of built-in functions and a huge standard and contributed library. Most programs are also quicker to execute than other languages because of Perl’s internal architecture.

3. Perl Is Fast

Compared to most scripting languages, this makes execution almost as fast as compiled C code. But, because the code is still interpreted, there is no compilation process, and applications can be written and edited much faster than with other languages, without any of the performance problems normally associated with an interpreted language.

4. Perl Is Extensible

You can write Perl-based packages and modules that extend the functionality of the language. You can also call external C code directly from Perl to extend the functionality.

5. Perl Has Flexible Data Types

You can create simple variables that contain text or numbers, and Perl will treat the variable data accordingly at the time it is used.

6. Perl Is Object Oriented

Perl supports all of the object-oriented features—inheritance, polymorphism, and encapsulation.

7. **Perl Is Collaborative**

There is a huge network of Perl programmers worldwide. Most programmers supply, and use, the modules and scripts available via CPAN, the Comprehensive Perl Archive Network

Compiler or Interpreter:

a. Compiler: A program that decodes instructions written in a higher order language and produces an assembly language program.

A compiler that generates machine language for a different type of computer than the one the compiler is running in.

b. Interpreter: In computing, an interpreter is a computer program that reads the source code of another compute program and executes that program. *A program that translates and executes source language statements one line at a time.*

c. Difference between Compiler and Interpreter

A compiler first takes in the entire program, checks for errors, compiles it and then executes it. Whereas, an interpreter does this line by line, so it takes one line, checks it for errors and then executes it.

Example of Compiler – Java

Example of Interpreter – PHP

d. Perl is interpreter or compiler?

Neither, and both. Perl is a scripting language. There is a tool, called perl, intended to run programs written in the perl language.

"Compiled" languages are ones like C and C++, where you have to take the source code, compile it into an executable file, and THEN run it.
"Interpreted" languages, like Perl, PHP, and Ruby, are ones which do NOT require pre-compiling.
They are generally compiled on-the-fly (which is what the perl command-line tool does) into *opcodes*, and then run. So, Perl is an interpreted language because a tool reads the source code and immediately runs it.
Perl is a compiler because it has to compile that source code before it can be run while it's being interpreted.
Popular "Myth conceptions"
1. **It's only for the Web**

Probably the most famous of the myths is that Perl is a language used, designed, and created exclusively for developing web-based applications.
2. **It's Not Maintenance Friendly**

Any good (or bad) programmer will tell you that anybody can write unmaintainable code in any language. Many companies and individuals write maintainable programs using Perl.
3. **It's Only for Hackers**

Perl is used by a variety of companies, organizations, and individuals. Everybody from programming beginners through ―hackers‖ up to multinational corporations use Perl to solve their problems.
4. **It's a Scripting Language**

In Perl, there is no difference between a script and program. Many large programs and projects have been written entirely in Perl.
5. There's No Support

The Perl community is one of the largest on the Internet, and you should be able to find someone, somewhere, who can answer your questions or help you with your problems.
6. All Perl Programs Are Free

Although you generally write and use Perl programs in their native source form, this does not mean that everything you write is free. Perl programs are your own intellectual property and can be bought, sold, and licensed just like any other program.
7. There's No Development Environment

Perl programs are text based, you can use any source-code revision-control system. The most popular solution is CVS, or Concurrent Versioning System, which is now supported under Unix, MacOS and Windows.
8. **Perl Is a GNU Project**

While the GNU project includes Perl in its distributions, there is no such thing as ―GNU Perl.‖ Perl is not produced or maintained by GNU and the Free Software Foundation. Perl is also made available on a much more open license than the GNU Public License.

9. Perl Is Difficult to Learn

Because Perl is similar to a number of different languages, it is not only easy to learn but also easy to continue learning. Its structure and format is very similar to C, **awk**, shell script, and, to a greater or lesser extent, even BASIC.

8.2 Perl Overview:

Installing and using Perl

Perl was developed by Larry Wall. It started out as a scripting language to supplement *rn,* the USENET reader. It available on virtually every computer platform.

Perl is an interpreted language that is optimized for string manipulation, I/O, and system tasks. It has built in for most of the functions in section 2 of the UNIX manuals -- very popular with sys administrators. It incorporates syntax elements from the *Bourne shell,*

csh, awk, sed, grep, and C. It provides a quick and effective way to write interactive web applications

Writing a Perl Script

Perl scripts are just text files, so in order to actually "write" the script, all you need to do is create a text file using your favorite text editor. Once you've written the script, you tell Perl to execute the text file you created.

Under Unix, you would use

```
$ perl myscript.pl
```

and the same works under Windows:

```
C:\> perl myscript.pl
```

Under Mac OS, you need to drag and drop the file onto the *MacPerl* application. Perl scripts have a *.pl* extension, even under Mac OS and Unix.

Perl Under Unix

The easiest way to install Perl modules on Unix is to use the CPAN module. For example:

```
shell> perl -MCPAN -e shell
cpan> install DBI
cpan> install DBD::mysql
```

The DBD::mysql installation runs a number of tests. These tests attempt to connect to the local MySQL server using the default user name and password. (The default user name is your login name on Unix, and ODBC on Windows. The default password is "no password.") If you cannot connect to the server with those values (for example, if your account has a password), the tests fail. You can use force install DBD::mysql to ignore the failed tests.

DBI requires the Data::Dumper module. It may be installed; if not, you should install it before installing DBI.

Perl Under Windows

1. Log on to the Web server computer as an administrator.
2. Download the ActivePerl installer from the following ActiveState Web site: http://www.activestate.com/ (http://www.activestate.com/)
3. Double-click the **ActivePerl** installer.
4. After the installer confirms the version of ActivePerl that it is going to be installed, click **Next**.

5. If you agree with the terms of the license agreement, click **I accept the terms in the license agreement**, and then click **Next**. Click **Cancel** if you do not accept the license agreement. If you do so, you cannot continue the installation.
6. To install the whole ActivePerl distribution package (this step is recommended), click **Next** to continue the installation. The software is installed in the default location (typically C:\Perl).
7. To customize the individual components or to change the installation folder, follow the instructions that appears on the screen.
8. When you are prompted to confirm the addition features that you want to configure during the installation, click any of the following settings, and then click **Next**:
a. **Add Perl to the PATH environment variable**: Click this setting if you want to use Perl in a command prompt without requiring the full path to the Perl interpreter.
b. **Create Perl file extension association**: Click this setting if you want to allow Perl scripts to be automatically run when you use a file that has the Perl file name extension (.pl) as a command name.
c. **Create IIS script mapping for Perl**: Click this setting to configure IIS to identify Perl scripts as executable CGI programs according to their file name extension.
d. **Create IIS script mapping for Perl ISAPI**: Click this setting to use Perl scripts as an ISAPI filter.

Perl Components:

Variables

Perl Variables with the techniques of handling them are an important part of the Perl language. As a language-type script, Perl was designed to handle huge amounts of data text. Working with variables is fairly straightforward given that it is not necessary to define and allocate them, so no sophisticated techniques for the release of memory occupied by them.
As general information, to note that the names of Perl variables contain alphabetic characters, numbers and the underscore (_) character and are case sensitive.
A specific language feature is that variables have a non-alphabetical prefix that fashion somewhat cryptic the language.
a. scalar variables – starting with $
b. array variables – starting with @
c. hashes or associative arrays indicated by %

The $, @ and % characters actually predefine the variable type in Perl. Perl language also offers some built-in predefined variables that facilitate and shorten the programming code.

Operators

The operators work with numbers and strings and manipulate data objects called operands. We found the operators in expressions which we need to evaluate.

Statements

The statements are one of the most important topics in the Perl language, actually for any programming language. We use statements in order to process or evaluate the expressions. Perl uses the values returned by statements to evaluate or process other statements.
A Perl statement ends with the semicolon character (;) which is used to tell interpreter that the statement was complete.

Subroutines (Functions)
Definition: *Subroutine* is a block of source code which does one or some tasks with specified
purpose.
Advantages:
1. It reduces the Complexity in a program by reducing the code.
2. It also reduces the Time to run a program.In other way,It’s directly proportional to Complexity.
3. It’s easy to find-out the errors due to the blocks made as function definition outside the main function.

Modules:
A ***Perl module*** is a discrete component of software for the Perl programming language. Technically, it is a particular set of conventions for using Perl's package mechanism that has become universally adopted.
A module defines its source code to be in a *package* (much like a Java package), the Perl mechanism for defining namespaces, e.g. *CGI* or *Net::FTP* or *XML::Parser*; the file structure mirrors the namespace structure (e.g. the source code for *Net::FTP* is in *Net/FTP.pm*). A collection of modules, with accompanying documentation, build scripts, and usually a test suite, compose a ***distribution***.
8.3 Perl Parsing Rules
The Execution Process:
The execution process of *perl* contains the following steps
 It takes raw input,
 Parses each statement and converts it into a series of opcodes,
 Builds a suitable opcode tree,
Component Identity
When Perl fails to identify an item as one of the predefined operators, it treats the character sequence as a “term.” Terms are core parts of the Perl language and include variables, functions, and quotes. The term-recognition system uses these rules:
 Variables can start with a letter, number, or underscore, providing they follow a suitable variable character, such as $, @, or %.
 Variables that start with a letter or underscore can contain any further combination of letters, numbers, and underscore characters.
 Variables that start with a number can only consist of further numbers—be wary of using variable names starting with digits. The variables such as $0 through to $9 are used for group matches in regular expressions.
 Subroutines can only start with an underscore or letter, but can then contain any combination of letters, numbers, and underscore characters.
 Case is significant—$VAR, $Var, and $var are all different variables.
 Each of the three main variable types have their own name space—$var, @var, and %var are all separate variables.
 File handles should use all uppercase characters—this is only a convention, not a rule, but it is useful for identification purposes.

Operators and Precedence UNIT-V PERL 8

a) Arithmetic Operators:

The following are the arithmetic operators in Perl. **Operator**	**Description**
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Modulus operator
**	Exponentiation operator

Concentrate on Lab Programs