A Quick Guide To PERL

This is a Quick reference Guide for PERL 5.8.6 programming. Perl definition is given by its creator, Larry Wall: "Perl is a language to get your job done" and he added "There is more than one way to do it!"

This guide is not exhaustive, its purpose is to give a few essential reminder to the Perl syntax, but basic knowledge of Perl programming is required.

To find help about a Perl function or keyword use perldoc:

```
perldoc -f split
perldoc -q FAQkeyword
```

For more information about Perl in general see:

http://www.perl.org

References

For more information on Perl syntax you can refer to O'Reilly’s book “Programming Perl, 3rd edition”.

Structure of a Perl script

```
#!/usr/bin/perl
```

The first line of a Perl script gives the path to the Perl executable (could be /usr/local/bin/perl)

```
`%INC`
```

contain list of directories for modules to import

```
`%ENV`
```

contain environment variables

```
`$ARGV[0]`
```

first argument

```
< > <= >= != == <=>
```

logical AND, OR and NOT

```
lt gt le ne eq cmp
```

string comparison

Example:
```
if ($var == 42) { print "$var is numeric";
```

elsif ($var eq "XLII") { print "$var is a string";
```

else { print "$var is not equal to 42";
```

Generally:
```
if (expr1) {
  statement list1
  else {print "$var is not equal to 42";}
```

and:
```
if (expr1) {
  statement list1
  else if (expr2) {
    statement list2
    else {print "$var is not equal to 42";}
```

Loops

```
while(expr) {
  statement list
```

repeat statement while expr is true

```
for($i=1;$i<=10;$i++){
  statement list
```

print $i

```
```
undef
```

end loops (while, for, etc...)

```
redo;
next;
last;
```

jump to next item in the loop

```
redo;
```

restart loop with current item

```
redo;
```

end loops (while, for, etc...)

```
redo;
```

Generally:
```
redo;
```

execute statement if expr is true (also with unless, while, until)

```
unless(expr) {
  statement list
```

execute statement unless expr is true, handle else if else (like if)

```
do {
  statement list
  } until(expr)
```

repeat statement a certain number of times

```
until(expr)
```

end loops (while, for, etc...)

```
```
Subroutines, example:
sub add_it {
  local ($a,$b)=@_;  
  $var = $a+$b;   
  return $var;
}
$result = &add_it(3,5);  
\text{call subroutine with arguments, } $result \text{ contains 8.}

\textbf{File Operators}
\begin{itemize}
  \item \texttt{open} HANDLE, filename
  \item \texttt{close} HANDLE
\end{itemize}
Example:
\begin{verbatim}
open (FH, "filename");
while (<FH>) {
  $text .= $_;
}
close(FH);
\end{verbatim}
\text{open file Handler, \texttt{$_} \text{ contains the content of file filename}\\
\text{concatenate } \$_ \text{ in } $text\\
close filehandle, $text \text{ contains the content of file filename}

\textbf{Special Handlers}
\begin{itemize}
  \item \texttt{<STDIN>} \text{read from standard input (usually keyboard)}
  \item \texttt{<STDOUT>} \text{write to standard output (usually screen)}
  \item \texttt{<STDERR>} \text{write to standard error (usually screen)}
\end{itemize}

\textbf{File Tests}
if (-e $filename) { open(READ, $filename); }
\text{Some possible tests:} 
\begin{itemize}
  \item {-} \text{readable}
  \item {-w} \text{writable}
  \item {-x} \text{executable}
  \item {-o} \text{belong to user}
  \item {-e} \text{exist}
  \item {-z} \text{zero size (file exist)}
  \item {-s} \text{nonzero size}
  \item {-f} \text{file}
  \item {-d} \text{directory}
  \item {-l} \text{symlink}
  \item {-T} \text{text file}
  \item {-A} \text{accessed in days}
\end{itemize}
@var=stat($filename);
\text{get full info on files}