Standard modules

These are several thousand standard Perl modules available via **use** keyword. The module name is prefixed with :: Examples:

- use DB_File; functions for maintaining an external hash
- use Getopt::Std; functions for processing command-line flags
- use File::Find; find function like the shell's find
- use Math::BigInt; unlimited-precision arithmetic

Defining A Module - Example

```
use base 'Exporter';
our @EXPORT = qw/min max/;
use List::Util qw/reduce/;
sub sum {
return reduce {$a + $b} @;
}
sub min {
return reduce {$a < $b ? $a : $b} @ ;
# must return true to indicate loading succeeded
1;
```

Using A Module - Example

```
use Example_Module qw/max/;

# As max is in our import list
# it can be used without module name
print max(42,3,5), "\n";

# We don't import min explicitly
# so it needs the module name
print Example_Module::min(42,3,5), "\n";
```

The directory containing Example_Module.pm must be in listed environment variable PERL5LIB

PERL5LIB is colon separated list of directory equivalent to Shell PATH

Pragmas

Perl provides a way of controlling some aspects of the interpreter's behaviour (through *pragmas*) also introduced by the *use* keyword.

- use English; allow names for built-in vars, e.g., \$NF = \$.
 and \$ARG = \$_.
- use integer; truncate all arithmetic operations to integer, effective to the end of the enclosing block.
- use strict 'vars'; insist on all variables declared using my.

CPAN

Comprehensive Perl Archive Network (CPAN) is an archive of 150,000+ Perl modules.

Hundreds of mirrors, including

http://mirror.cse.unsw.edu.au/pub/CPAN/

Command line tools to quickly install modules from CPAN.