

1.1.2.8.6

Intermediate Perl – Session 5

- modules
- CPAN



modules

- modules breathe life into Perl
 - limitlessly expand built-in functionality
 - allow anyone to contribute to Perl
 - centralized and mirrored module repository
 - comprehensive Perl archive network (CPAN)
 - limit code repetition
 - written in Perl, C/C++ or a combination
 - C modules add the raw speed of C to Perl
- modules typically contain
 - functions
 - many per module
 - classes (code designed to be used in an object oriented fashion)
 - usually one per module
- modules are available
 - as bundles – multiple modules which are related (e.g. Bundle::LWP, bioperl)
 - as individual classes – modules which perform a single task (e.g. Math::Round)

@INC - location of modules

- perl binary will look in the list of paths stored in @INC for modules
- by default, @INC is populated with several paths that are dependent on
 - version of perl
 - architecture/OS of the machine that compiled perl
- you can see the default @INC contents by checking compile settings with perl -V

```
> perl -V
...
@INC:
/home/martink/perl/5.8.7/lib/5.8.7/i686-linux-ld
/home/martink/perl/5.8.7/lib/5.8.7
/home/martink/perl/5.8.7/lib/site_perl/5.8.7/i686-linux-ld
/home/martink/perl/5.8.7/lib/site_perl/5.8.7
/home/martink/perl/5.8.7/lib/site_perl
.
```

modifying @INC

- set **PERL5LIB** environment variable

```
> export PERL5LIB=/home/martink/modules
> perl -V
...
%ENV:
    PERL5LIB="/home/martink/modules"
@INC:
    /home/martink/perl/5.8.7/lib/5.8.7/i686-linux-ld
    /home/martink/perl/5.8.7/lib/5.8.7
    /home/martink/perl/5.8.7/lib/site_perl/5.8.7/i686-linux-ld
    /home/martink/perl/5.8.7/lib/site_perl/5.8.7
    /home/martink/perl/5.8.7/lib/site_perl
    .
```

- use **-Idir** for each directory at runtime

```
> perl -I/home/martink/modules -I/home/martink/othermodules script.pl
```

modifying @INC

- employ the **lib** pragma in your script

```
#!/usr/bin/perl

use lib "/home/martink/modules";
...
```

- typically, you do not have permission to write into the default module locations
 - e.g. **/usr/lib** or **/usr/local/lib**
- install to a location of your choice and then modify **@INC** to include that location in the search path

module compatibility

- each perl binary has its own @INC
- depending on the module
 - cross-version compatibility (e.g. perl 5.6 vs 5.10) is not assured
 - cross-platform compatibility (e.g. systems with different versions of libc) is not assured
- modules written entirely in Perl tend to be relatively compatible
 - if you download and compile a [module written in Perl](#) and install it into [/your/module/directory](#), chances are good it will work with 5.6, 5.8 and 5.10, unless you are using features specific to a version (e.g. regex engine features)
 - modules with components [written in C](#) are less compatible and it is safe to assume that [they will not work](#) across multiple versions

module compatibility

- when you compile a module that contains components written in C, system libraries may be dynamically linked into the module and the module may not work on a Linux install significantly older/newer than the system which compiled the module
- multiple versions of the same module are not natively supported
 - you can use `only.pm` module from CPAN for this

creating a basic module

- a module is typically a package (namespace) with function and/or object definitions
- the most basic module is one with only variable and/or function definitions
- below is a module that defines a single function, **square**

```
package MyModule;

sub square {
    my $num = shift;
    return $num**2;
}

1;
```

- the module file should be the same as the package name with **.pm** suffix
 - **MyModule.pm**

creating a basic module

- place the module file in your module directory

```
> cp MyModule.pm /home/martink/modules/.
```

- import and use the module in your script

```
use lib "/home/martink/modules";
use MyModule;

print MyModule::square(5);
```

creating a basic module

- to export the module's square function into the current namespace use `Exporter` and define the list of symbols to be automatically exported in `@EXPORT`

```
package MyModule;  
@ISA=qw(Exporter);  
@EXPORT=qw(square);  
  
sub square {  
    my $num = shift;  
    return $num**2;  
}  
  
1;
```

- whenever you use the module, `square` is exported into the main namespace and available directly

```
use lib "/home/martink/modules";  
use MyModule;  
  
print square(5);
```

creating a basic module

- to define symbols that can be exported explicitly set `@EXPORT_OK`

```
package MyModule;
@ISA=qw(Exporter);
@EXPORT=();          # nothing is exported by default
@EXPORT_OK=qw(square); # symbols that can be exported by request

sub square {
    my $num = shift;
    return $num**2;
}

1;
```

- now you must ask to export `square`

```
use lib "/home/martink/modules";
use MyModule qw(square);

print square(5);
```

creating an object module

- modules provide encapsulation and interfaces
 - **encapsulation** – hide implementation specifics from user
 - **interface** – provide a stable set of functions to create and manipulate data
- when perl modules are described, object-oriented terminology is frequently used

object-oriented	perl
class	package
object	reference blessed into a package
method	subroutine in a class
class method	method designed to be called using a class
object method	method that expects to be called using a class
constructor	class method that returns a new object

creating an object module

- let's create a counter object which implements the following methods
 - new – creates a counter and sets it to 0
 - increment – adds 1 to a counter, and returns new value
 - decrement – subtracts 1 from a counter, and returns new value
 - reset – resets the counter value to zero
 - value – returns current value

```
use MyCounter;

my $counter = MyCounter->new();
$counter->increment;
print $counter->value;          # 1
$counter->reset;
$counter->decrement;
$counter->decrement;
print $counter->value;          # -2
```

creating an object module

```
package MyCounter;

sub new {
    my ($pkg) = @_;
    my $self = {value=>0}; # this is the object - commonly implemented as a hash reference
    bless ($self,$pkg); # bless tells perl that $self is an object in the package $pkg
    return $self;
}
sub increment { # when $counter->increment is called, first argument to increment() is $counter
    my ($self) = @_;
    return $self->{value}++;
}
sub decrement {
    my ($self) = @_;
    return $self->{value}--;
}
sub reset {
    my ($self) = @_;
    return $self->{value}=0;
}
sub value {
    my ($self) = @_;
    return $self->{value};
}

1;
```

extending the object module

- in addition to previous functionality,
 - initial counter value can be set
 - increase/decrease step can be set and changed

```

package MyCounter;

sub new {
    my ($pkg,$value,$step) = @_;
    my $self = {value=>$value,step=>$step||1};
    bless ($self,$pkg);
    return $self;
}

sub increment {
    my ($self) = @_;
    return $self->{value} += $self->{step};
}

sub decrement {
    my ($self) = @_;
    return $self->{value} -= $self->{step};
}

sub reset {
    my ($self) = @_;
    return $self->{value}=0;
}

# set a new step value, if the argument is provided
sub step {
    my ($self,$step) = @_;
    if(defined $step) {
        $self->{step} = $step;
    }
    return $self->{step};
}

sub value {
    my ($self) = @_;
    return $self->{value};
}

1;

```

extending the object module

- now the counter has more functionality

```
use MyCounter;

my $c = MyCounter->new(10);

$c->increment();
print $c->value,$c->step;      # 11 1
$c->reset();
$c->step(5);
print $c->value,$c->step;      # 0 5
$c->decrement();
$c->decrement();
print $c->value;              # -10

print ref($c);                # MyModule
```

- if the module is in the same directory as the script, it will be found
- if not, then `use lib "/path/to/module/directory"`

www.cpan.org

- the first and last stop for modules is CPAN
 - thousands of modules by thousands of authors
 - code is available, authors can be contacted, community is open



The screenshot shows the CPAN homepage. At the top is a large logo with the letters 'CPAN' where the 'A' is stylized as a stack of books. Below the logo is a dark blue navigation bar with links for Home, Authors, Recent, News, Mirrors, FAQ, and Feedback. Underneath the navigation bar is a search bar containing a text input field, a dropdown menu set to 'All', and a button labeled 'CPAN Search'. Below the search bar is a grid of category links. The categories are arranged in two columns. The left column includes: Archiving, Compression, Conversion, Bundles (and SDKs), Commercial Software Interfaces, Control Flow Utilities, Data and Data Types, Database Interfaces, Development Support, Documentation, and File Handle Input/Output. The right column includes: File Name Systems, Locking, Option Parameter Config Processing, Graphics, Internationalization Locale, Pragmas, Language Extensions, Language Interfaces, Mail and Usenet News, Miscellaneous, Networking Devices IPC, Operating System Interfaces, Perl6, Security, Server Daemon Utilities, String Language Text Processing, User Interfaces, and World Wide Web.

Archiving	Compression	Conversion	File Name Systems	Locking	Option Parameter Config Processing
Bundles (and SDKs)			Graphics	Perl6	
Commercial Software Interfaces			Internationalization Locale	Pragmas	
Control Flow Utilities			Language Extensions	Security	
Data and Data Types			Language Interfaces	Server Daemon Utilities	
Database Interfaces			Mail and Usenet News	String Language Text Processing	
Development Support			Miscellaneous	User Interfaces	
Documentation			Networking Devices IPC	World Wide Web	
File Handle Input/Output			Operating System Interfaces		

CPAN: String Language Text Processing

- clicking on String Language Text Processing

CPAN			
Home Authors Recent News Mirrors FAQ Feedback			
<input type="text" value="in All"/> CPAN Search			
Top > String Language Text Processing			
Lingua::PDE::Parse::PostScript	RDF::String::Text	XML::Parse::Text	
Acme::Wabby	Generate random sentences based on seed text	POZNICK	JUCKER
AnyData	Read hash interface to data in many formats	AMBIS	BERJON
Biblio::Thesaurus	A Multi-Lingue Thesaurus tool	JNOLAN	
CSS::SAC	Perl implementation of the Simple API to CSS	AUTRULUS	AUTRULUS
Chatbot::Eliza	Eliza algorithm encapsulated in an object	PAUL	PHOENIX
Convert::CharMap	Conversion between Unicode Character Maps	MIYAGAWA	GASS
Convert::GeekCode	Convert and generate geek code sequences	MHOSEN	MHOSEN
DMOZ::ParseRDF	Parse the DMOZ RDF file into smaller chunks	JANPAZ	
ERG	An extensible report generator framework	PEASE	PEASE
Email::Find	Find RFC 822 email addresses in plain text	PEASE	PEASE
Font::AFM	Parse Adobe Font Metric files	KMACLEOD	EPIET
Font::Frat	Fret - Font REporting Tool		
Font::TFM	Read info from TeX font metric files		
Font::TTF	TrueType font manipulation module		
FrameMaker	Top level FrameMaker interface		
FrameMaker::Control	Control a FrameMaker session		
FrameMaker::FDK	Interface to Adobe FDK		
FrameMaker::MF	Parse and Manipulate FrameMaker MIF files		
Frontier::RPC	Performs Remote Procedure Calls using XML		
Language::VBParser	Visual Basic 6 source parser		
MMOS			
MP3::MSU::Parser			
Marpa			
Number::Encode			
Number::Format			
Number::Phone::US			
OurNet::FuzzyIndex			
PDF::Text			
MIMIDOS			
MP3::M3U::Parser	Mp_dhp	MIMIMAN Markup Document System	JV
Marpa	Rp_d0p	Perl extension for parsing m3u mp3 lists.	BURAK
Number::Encode	c+d0	Context Free Parser	JKEGL
Number::Format	Rp_d0p	Encode bit strings into digit strings	LUISMUNOZ
Number::Phone::US	Rp_d0p	Package for formatting numbers for display	WRW
OurNet::FuzzyIndex	Rp_d0f	Validates several US phone number formats	KENNEDYH
PCL::Simple	Rp_mfp	Inverted search for double-byte characters	AUTRIJUS
PHP::Include	Rp_dhp	Create PCL for printing plain text files	PRL
		Include PHP files from Perl	ESUMMERS

module names

- many modules contain ::
 - `File::Find`
 - `File::Find::Closures`
 - `File::Flock`
- on the file system :: corresponds to a directory delimiter
 - `File::Find` would be `File/Find.pm`
 - `File::Find::Closures` would be `File/Find/Closures.pm`
- similarity in names **does not imply** a functional relationship
 - `File::Find::Closures` may have nothing to do with `File::Find`
 - but both have something to do with finding files
 - `MyModule::Utils` does not necessarily inherit from `MyModule`

Number::Format

- documentation follows a convention
 - NAME
 - SYNOPSIS
 - DESCRIPTION
 - METHODS
 - various sections
 - HISTORY
 - BUGS
 - AUTHOR
 - SEE ALSO
- the SYNOPSIS is for the impatient
 - informative enough to start

CPAN

Home · Authors · Recent · News · Mirrors · FAQ · Feedback

in All CPAN Search

[William R Ward > Number-Format-1.45 > Number::Format](#)

Module Version: 1.45 [Source](#)

[NAME](#) 

Number::Format - Perl extension for formatting numbers

[SYNOPSIS](#) 

```

use Number::Format;
my $x = new Number::Format %args;
$formatted = $x->round($number, $precision);
$formatted = $x->format_number($number, $precision, $trailing_zeroes);
$formatted = $x->format_negative($number, $picture);
$formatted = $x->format_picture($number, $picture);
$formatted = $x->format_price($number, $precision);
$formatted = $x->format_bytes($number, $precision);
$number    = $x->unformat_number($formatted);

use Number::Format qw(:subs);
$formatted = round($number, $precision);
$formatted = format_number($number, $precision, $trailing_zeroes);
$formatted = format_negative($number, $picture);
$formatted = format_picture($number, $picture);
$formatted = format_price($number, $precision);
$formatted = format_bytes($number, $precision);
$number    = unformat_number($formatted);

```

Number::Format has both OOP and functional API

- the SYNOPSIS section for this module shows

```
use Number::Format;
my $x = new Number::Format %args;
$formatted = $x->round($number, $precision);
$formatted = $x->format_number($number, $precision, $trailing_zeroes);
$formatted = $x->format_negative($number, $picture);
$formatted = $x->format_picture($number, $picture);
$formatted = $x->format_price($number, $precision);
$formatted = $x->format_bytes($number, $precision);
$number    = $x->unformat_number($formatted);

use Number::Format qw(:subs);
$formatted = round($number, $precision);
$formatted = format_number($number, $precision, $trailing_zeroes);
$formatted = format_negative($number, $picture);
$formatted = format_picture($number, $picture);
$formatted = format_price($number, $precision);
$formatted = format_bytes($number, $precision);
$number    = unformat_number($formatted);
```

Downloading Modules

CPAN

Home Authors Recent News Mirrors FAQ Feedback

in All CPAN Search

William R Ward > Number-Format-1.45 > Number::Format

Module Version: 1.45 Source

NAME SYNOPSIS REQUIRES DESCRIPTION EXPORTS METHODS BUGS AUTHOR SEE ALSO

NAME  Number::Format - Perl extension for formatting numbers

SYNOPSIS 

```
use Number::Format;
my $x = new Number::Format 'argent';
$formatted = $x->round($number, $precision);
$formatted = $x->format_number($number, $precision);
$formatted = $x->format_separators($number, $precision);
$formatted = $x->format_picture($number, $precision);
$formatted = $x->format_price($number, $precision);
$formatted = $x->format_bytes($number, $precision);
$number = $x->unformat_number($formatted);

use Number::Format 'argent';
$formatted = $x->round($number, $precision);
$formatted = $x->format_number($number, $precision);
$formatted = $x->format_negative($number, $precision);
$formatted = $x->format_picture($number, $precision);
$formatted = $x->format_price($number, $precision);
$formatted = $x->format_bytes($number, $precision);
$number = $x->unformat_number($formatted);
```

This Release Number-Format-1.45 [Download] [Browse] 27 Aug 2002

Other Releases Number-Format-1.44 – 11 Dec 2001 Goto

Links [CPAN Testers] [View/Report Bugs] [Tools]

CPAN Testers PASS (10) FAIL (3) [View]

Rating ★★★★ (0) [Rate this distribution]

License Unknown

Special Files [CHANGES](#) [MANIFEST](#) [Makefile.PL](#) [README](#)

Modules

Number::Format Perl extension for formatting numbers 1.45

Downloading Modules

- download the module from CPAN as .tar.gz
 - usually named **ModuleName-x.xx.tar.gz**

```
> ls
-rw-r--r--    1 martink  users        14084 2004-03-18 14:22 Number-Format-1.45.tar.gz
```

- unpack the tarball
 - the module usually comes in its own directory

```
> tar xvfz Number-Format-1.45.tar.gz
Number-Format-1.45/
Number-Format-1.45/README
...
Number-Format-1.45/t/object.t
Number-Format-1.45/Makefile.PL
Number-Format-1.45/TODO
```

Installing Modules

- you now have a directory in which into which the module files have been unpacked

```
> ls
drwxr-x---    3 martink  users          4096 2002-08-27 17:16 Number-Format-1.45/
-rw-r--r--    1 martink  users          14084 2004-03-18 14:22 Number-Format-1.45.tar.gz

> cd Number-Format-1.45/

> ls
-rw-r--r--    1 martink  users          5137 2001-12-10 16:39 CHANGES
-rw-r--r--    1 martink  users         28294 2002-08-27 16:11 Format.pm
-rw-r--r--    1 martink  users          194  2001-12-10 12:35 MANIFEST
-rw-r--r--    1 martink  users          239  1997-11-08 10:58 Makefile.PL
-rw-r--r--    1 martink  users         3261  2002-08-27 17:16 README
-rw-r--r--    1 martink  users          953  2001-12-10 12:35 TODO
drwxr-x---    2 martink  users          4096  2002-08-27 17:16 t/
```

perl Makefile.pl ; make ; make test ; make install

```
# set PREFIX to specify where the module should be installed
> perl Makefile.PL PREFIX=/home/martink/lib
Checking if your kit is complete...
Looks good
Writing Makefile for Number::Format

> make
mkdir blib
mkdir blib/lib
...
Manifying blib/man3/Number::Format.3

> make test
PERL_DL_NONLAZY=1 /usr/local/bin/perl -Iblib/arch -Iblib/lib -I/usr/local/lib/perl5/5.00503/i686-
linux -I/usr/local/lib/perl5/5.00503 -e 'use Test::Harness qw(&runtests $verbose); $verbose=0;
runtests @ARGV;' t/*.t
...
All tests successful.
Files=9, Tests=64, 1 wallclock secs ( 0.40 cusr + 0.14 csys = 0.54 CPU)

> make install
```

Using the Installed Module

- once installed, cast the appropriate `use lib` call and then `use` the module

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

use lib "/home/martink/lib";
use Number::Format;

# now use the module
```

- modules may have dependencies
 - you'll need module A to install and use module B
 - if A is installed in your personal space, modify `@INC` before attempting to install B

```
> perl -I/home/martink/lib Makefile.PL PREFIX=/home/martink/lib
```

using CPAN.pm to install modules

- you can query and install from CPAN interactively

```
> perl -MCPAN -e shell
cpan shell -- CPAN exploration and modules installation (v1.83)
ReadLine support enabled
cpan>
```

- a help menu is provided with ?

```
cpan> ?
Display Information
  command   argument      description
  a,b,d,m WORD or /REGEXP/ about authors, bundles, distributions, modules
  i          WORD or /REGEXP/ about any of the above
  r          NONE           report updatable modules
  ls         AUTHOR or GLOB about files in the author's directory
            (with WORD being a module, bundle or author name or a distribution
             name of the form AUTHOR/DISTRIBUTION)
  ...
```

using CPAN.pm to install modules

- to list currently installed modules in @INC, use `autobundle`

```
> autobundle
```

Package namespace	installed	latest	in CPAN file
Algorithm::Cluster	1.30	1.42	MDEHOON/Algorithm-Cluster-1.42.tar.gz
AnyDBM_File	1.00	1.00	RGARCIA/perl-5.10.0.tar.gz
AnyData	0.10	0.10	JZUCKER/AnyData-0.10.tar.gz
AnyData::Format::Base	undef	undef	JZUCKER/AnyData-0.10.tar.gz
AnyData::Format::CSV	0.05	0.05	JZUCKER/AnyData-0.10.tar.gz
...			
version	0.48	0.76	JPEACOCK/version-0.76.tar.gz
version::vxs	0.48	0.76	JPEACOCK/version-0.76.tar.gz
vmsish	1.01	1.02	RGARCIA/perl-5.10.0.tar.gz
warnings	1.03	1.06	RGARCIA/perl-5.10.0.tar.gz
warnings::register	1.00	1.01	RGARCIA/perl-5.10.0.tar.gz

Wrote bundle file

/home/martink/.cpan/Bundle/Snapshot_2008_09_16_00.pm

using CPAN.pm to install modules

- once you're in the shell, you can
 - list CPAN modules, bundles and authors
 - fetch the README file from a module
 - download and install the module

```
cpan> m /number::format/
Module      Number::Format (W/WR/WRW/Number-Format-1.60.tar.gz)
Module      Number::Format::Calc (H/HO/HOLLI/Number-Format-Calc-0.01.tar.gz)
Module      Template::Plugin::Number::Format (D/DA/DARREN/Template-Plugin-Number-
1.02.tar.gz)
3 items found

cpan> m Number::Format
Module id = Number::Format
  DESCRIPTION Package for formatting numbers for display
  CPAN_USERID WRW (William R Ward <bill@wards.net>)
  CPAN_VERSION 1.60
  CPAN_FILE   W/WR/WRW/Number-Format-1.60.tar.gz
  DSLI_STATUS Rdp0 (released,developer,perl,object-oriented)
  INST_FILE   (not installed)
```

using CPAN.pm to install modules

```
> readme Number::Format
```

Number::Format - Convert numbers to strings with pretty formatting

Version: 1.60

WHAT IS IT

Number::Format is a library for formatting numbers. Functions are provided for converting numbers to strings in a variety of ways, and to convert strings that contain numbers back into numeric form. The output formats may include thousands separators - characters inserted between each group of three characters counting right to left from the decimal point. The characters used for the decimal point and the thousands separator come from the locale information or can be specified by the user.

...

using CPAN.pm to install modules

- set CPAN configuration values with `o conf`
- `install` module
 - downloads, makes, tests and installs

```
> o conf makepl_arg PREFIX=/home/martink/modules
      makepl_arg [PREFIX=~/tmp]

# this installs the latest Number::Format
> install Number::Format

# this installs some other version of Number::Format
# recall the output of m Number::Format
#   CPAN_FILE W/WR/WRW/Number-Format-1.60.tar.gz
> install WRW/Number-Format-1.52.tar.gz

# to remember changes to configuration
> o conf commit
```

1.1.2.8.6

Introduction to Perl – Session 6

- functional and object modules
- install modules manually
- using CPAN

