

How to use Perl Module Library (CPAN)

Spoken Tutorial Project

<http://spoken-tutorial.org>

National Mission on Education through ICT

<http://sakshat.ac.in>

Nirmala Venkat

22 October 2015



Learning Objectives

We will learn to



Learning Objectives

We will learn to

- **Use existing modules**



Learning Objectives

We will learn to

- Use existing modules
- Create new modules



System Requirements



System Requirements

- **Ubuntu Linux 12.04 OS**



System Requirements

- **Ubuntu Linux 12.04 OS**
- **Perl 5.14.2**



System Requirements

- **Ubuntu Linux 12.04 OS**
- **Perl 5.14.2**
- **gedit Text Editor**



Pre-requisites

To follow this tutorial, you should have



Pre-requisites

To follow this tutorial, you should have

- Working knowledge of Perl Programming



Pre-requisites

To follow this tutorial, you should have

- Working knowledge of Perl Programming
- For relevant Perl tutorials, visit <http://spoken-tutorial.org>



Modules



Modules

- **Are code files that contain common routines**



Modules

- Are code files that contain common routines
- That are written by different authors



Modules

- Are code files that contain common routines
- That are written by different authors
- And can be used by several programs at a time



CPAN (Comprehensive Perl Archive Network)



CPAN (Comprehensive Perl Archive Network)

- **PERL is an open source language and anyone can contribute to PERL's standard CPAN library**



CPAN (Comprehensive Perl Archive Network)

- PERL is an open source language and anyone can contribute to PERL's standard CPAN library
- CPAN has thousands of ready-to-use modules written by different authors



CPAN (Comprehensive Perl Archive Network)

- **PERL is an open source language and anyone can contribute to PERL's standard CPAN library**
- **CPAN has thousands of ready-to-use modules written by different authors**



CPAN (Comprehensive Perl Archive Network)

- PERL is an open source language and anyone can contribute to PERL's standard CPAN library
- CPAN has thousands of ready-to-use modules written by different authors

The official website of CPAN is

<http://www.cpan.org>



Modules: Example



Modules: Example

```
use List::Util
```



Modules: Example

use List::Util

- Gives access to the functions which are already written inside this module



perl-doc Installation



perl-doc Installation

- Install the **perl-doc** package



perl-doc Installation

- Install the **perl-doc** package
- Do so using **Synaptic Package Manager**



perl-doc Installation

- Install the **perl-doc** package
- Do so using **Synaptic Package Manager**
- Refer to the relevant **Linux** spoken tutorials on <http://spoken-tutorial.org>



Steps in creating a module



Steps in creating a module

- **Create a place to develop the module**



Steps in creating a module

- Create a place to develop the module
- Create skeleton files for the module



Steps in creating a module

- Create a place to develop the module
- Create skeleton files for the module
- Document the module



Steps in creating a module

- Create a place to develop the module
- Create skeleton files for the module
- Document the module
- Write Perl code



Steps in creating a module

- Create a place to develop the module
- Create skeleton files for the module
- Document the module
- Write Perl code
- Write code for testing



Steps in creating a module

- Create a place to develop the module
- Create skeleton files for the module
- Document the module
- Write Perl code
- Write code for testing
- **Distribute the module in CPAN**



Command to generate files for a new module



Command to generate files for a new module

Command: **h2xs**

Example: **h2xs -PAXn Math::Simple**



Command to generate files for a new module

Command: **h2xs**

Example: **h2xs -PAXn Math::Simple**

- **Math::Simple** specifies the module name



Command to generate files for a new module

Command: **h2xs**

Example: **h2xs -PAXn Math::Simple**

- **Math::Simple** specifies the module name
- It creates skeleton files for the module



Command to generate files for a new module

Command: **h2xs**

Example: **h2xs -PAXn Math::Simple**

- **Math::Simple** specifies the module name
- It creates skeleton files for the module
- **-PAX** are options that omit autoload and autogenerate





Change @INC from inside the script:

syntax:

use lib "path"

Example:

use lib t/Math-simple.t

(or)

**Begin {unshift(@INC,
t/Math-simple.t)}**





Change @INC from the command line:

Example:

```
perl -Ilib t/Math-simple.t
```



Distribute the Module

Run these commands from the terminal:



Distribute the Module

Run these commands from the terminal:

```
perl Makefile.PL
```

```
make
```

```
make test
```

```
make dist
```



Summary

In this tutorial we learnt to

- **Use existing modules**
- **Create new modules**



Assignment

- 1 Use the **Text::Wrap** module
- 2 Make use of the **Wrap()** function which wraps the input text to form neat paragraphs



Assignment (cont.)

- 3 **Text::Wrap** module has a variable **columns**. Set the **columns** value to 30.
- 4 Print the text to see the formatted output



About the Spoken Tutorial Project

- Watch the video available at http://spoken-tutorial.org/What_is_a_Spoken_Tutorial
- It summarises the Spoken Tutorial project
- If you do not have good bandwidth, you can download and watch it



Spoken Tutorial Workshops

The Spoken Tutorial Project Team

- Conducts workshops using spoken tutorials
- Gives certificates on passing online tests
- For more details, please write to contact@spoken-tutorial.org



Acknowledgements

- Spoken Tutorial Project is a part of the Talk to a Teacher project
- It is supported by the National Mission on Education through ICT, MHRD, Government of India
- More information on this Mission is available at

<http://spoken-tutorial.org/NMEICT-Intro>

