A First Peek at Perl

Ross Dickson, Computational Research Consultant

What is Perl?

- Practical Extraction and Reporting Language
- General-purpose interpreted language
- Very popular, many applications
    - Web scripting
    - Systems administration and automation
    - ...many, many more
- Influenced by natural languages
- Greatest strength is text manipulation

Invocation

```
$ perl first.pl
...
$ head -1 first.pl
#!/usr/bin/env perl
$ chmod +x first.pl
$ ./first.pl
...
```

first.pl

```
#!/usr/bin/env perl
use strict;
use warnings;

my $string = 'abcd';
my $number = 42;
print "String is $string, number is $number\n";
print 'but single quotes do not interpolate: $string, $number',
"\n";
```

```
String is abcd, number is 42
but single quotes do not interpolate: $string, $number
```

- Singular (“scalar”) variables prefixed with $
  - strings, numbers, ...
- Declare variables with “my”
- Double and single quotes; interpolation
- “\n” end-of-line

first.pl (continued)

```
my @list = ('a','b','c','d','e');
print "list is @list \n";
print "element 0 is $list[0], $list[4] is element 4 \n";
my @seq = `seq 1 3`;   # backticks to call out to shell
print @seq;
chomp(@seq);              # removes newline characters
print "chomped seq: @seq\n";
```

```
list is a b c d e
element 0 is a, e is element 4
1
2
3
chomped seq: 1 2 3
```

- Array vars prefixed with @
- Elements are scalars ($), indexed with [ ]
- Backticks
- chomp

hash1.pl

```
my %table = (
    "I" => 1,
    "V" => 5,
    "X" => 10,
    "L" => 50,
    "C" => 100,
);
print "V means $table{'V'}\n";
print "C means $table{C}\n";     # bareword, bad style!
```

```
V means 5
C means 100
```

- Hash variables prefixed with %
- also known as “associative arrays”
- Elements are scalars ($) indexed with { }
- Subscripts are strings

math.pl

```
print "square root of 2:  ",sqrt(2), "\n";
use Math::Trig;
print "constant pi:       ", pi, "\n";  # no $ prefix!
my $pi = 22/7;
print "my approximate pi: $pi\n";
my $i = int($pi);
print "integer part:      $i \n";
print "cos(pi/2):         ",cos(pi/2), "\n";
```

```
square root of 2:  1.4142135623731
constant pi:       3.14159265358979
my approximate pi: 3.14285714285714
integer part:      3
cos(pi/2):         6.12323399573677e-17
```

- use Pkg::Pkg;
- no distinction between integer and floating point arithmetic
- limitations of floating-point arithmetic *(not just Perl!)*

passthru1.pl

```
#!/usr/bin/env perl
use strict;
use warnings;

print "filename: $ARGV[0] \n";
open (INFILE, $ARGV[0]);
while (<INFILE>) {
   print $_;    # default variable for many things, aka $ARG
}
```

```
$ ./passthru1.pl one
filename: one
One file
One file
One file
```

- @ARGV
- filehandles
- 'while (<FILE>)' idiom
- $_

passthru2.pl

```
#!/usr/bin/perl -w
use strict;
use warnings;

while (<>) {     # Read *all* argument files, or stdin if none
   print $_;
}
```

```
$ ./passthru2.pl one another
One file
One file
One file
Another file
Another file
$ echo 'hi there' | ./passthru2.pl
hi there
```

- 'while (<>)' idiom

uc.pl

```
#!/usr/bin/perl -w
use strict;
use warnings;

while (<>) {
   print uc;     # uppercase function; if no argument, works on $_
}
```

```
$ echo 'hi there' | ./uc.pl
HI THERE
```

- Standard Perl library functions
  - there are lots of them!
  - often take $_ as default argument

Loops

- while (condition) { statements; }
- until (condition) { statements; }
- for (expr; expr; expr) { statements; }
- foreach var (in_list) { statements; }
- do { statements; } while (condition)
- do { statements; } until (condition)

Conditions

- Anything 0 or empty or undefined is false,
- anything else is true
    - ...more or less.
    - For example, “0” is false, as is 0.0,
      but string “0.0” is true, as is “ ”.
    - Usually works out the way you want.
    - For more details, see the documentation.

Branch statements

- if (condition)
  { statements; }
  elsif (condition)
  { statements; }
  else
  { statements; }
- unless (condition) { statements; } . . .
- do { statements; } if (condition)
- do { statements; } unless (condition)

hash2.pl

```
my %table = (
    "I" => 1,
    "V" => 5,
    "X" => 10,
    "L" => 50,
    "C" => 100,
);
foreach my $key (sort keys %table) {
    print "$key\t$table{$key}\n"; }
```

```
C       100
I       1
L       50
V       5
X       10
```

- foreach
- sort( )
- keys( )

grab_geom.pl

Extract a molecular geometry from an output file

- “do X or die”
- =~ /regular_expression/
- split

```
           CH            0.07917   1.40589
           CCH           0.16996  -0.09285   1.51787
     Eigenvalues ---     0.50396   1.37933   1.58298
 Angle between quadratic step and forces=  28.04 degrees.
 Linear search not attempted -- first point.

Variable       Old X    -DE/DX   Delta X   Delta X   Delta X     New X
                                 (Linear)    (Quad)   (Total)
    CC        2.63608   0.00000   0.00000   0.00001   0.00001   2.63609
    CH        2.04284   0.00000   0.00000   0.00000   0.00000   2.04283
   CCH        2.08829   0.00000   0.00000   0.00000   0.00000   2.08829
         Item               Value     Threshold  Converged?
 Maximum Force            0.000004     0.000450     YES
 RMS     Force            0.000003     0.000300     YES
 Maximum Displacement     0.000006     0.001800     YES
 RMS     Displacement     0.000004     0.001200     YES
 Predicted change in Energy=-1.395198D-11
 Optimization completed.
    -- Stationary point found.
                          ----------------------------
                          !   Optimized Parameters   !
                          ! (Angstroms and Degrees)  !
 ----------------------                                 ----------------------
 !      Name           Value   Derivative information (Atomic Units)     !
 ---------------------------------------------------------------------------
 !       CC            1.395   -DE/DX =    0.0                          !
 !       CH            1.081   -DE/DX =    0.0                          !
 !       CCH         119.6501  -DE/DX =    0.0                          !
 ---------------------------------------------------------------------------
 GradGradGradGradGradGradGradGradGradGradGradGradGradGradGradGradGradGrad

 Leave Link  103 at Thu Apr  9 13:17:32 2009, MaxMem=    6291456 cpu:  0.1
```

grab_geom.pl

```
open IFILE, "<$ARGV[0]"
    or die "Unable to open $ARGV[0] for reading: $! \n";

while (my $line = <IFILE>) {
    if ($line =~ /Optimized Parameters/) {
        # skip next four lines
        for (my $i=0; $i<4; $i++) { my $junkline = <IFILE>; }
        $line = <IFILE>;
        while ($line =~ /DE\/DX/) {
            chomp($line);
            my @words = split ' ', $line;
            my $coord = $words[1];
            my $value = $words[2];
            print STDOUT "$coord $value\n";
            $line = <IFILE>;
        }
        print STDOUT "\n";
    }
}
close IFILE;
```

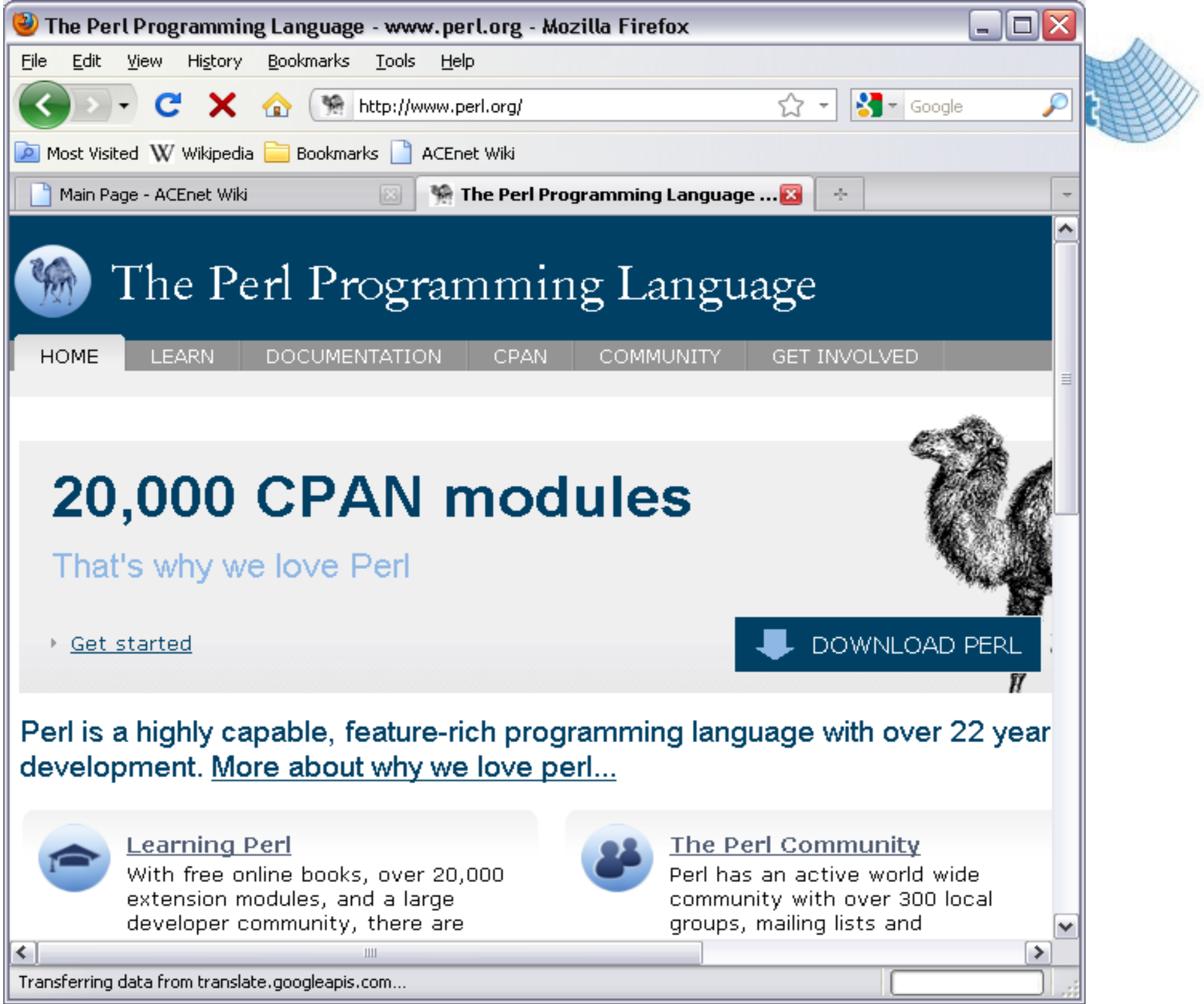
grab_geom.pl

```
$ ./grab_geom.pl test156.gof
CC 1.395
CH 1.081
CCH 119.6501

CC 1.395
CH 1.081
CCH 119.6501

CC 1.395
CH 1.081
CCH 119.6501

CC 1.395
CH 1.081
CCH 119.6501

CC 1.395
CH 1.081
CCH 119.6501

CC 1.395
CH 1.081
CCH 119.6501
```

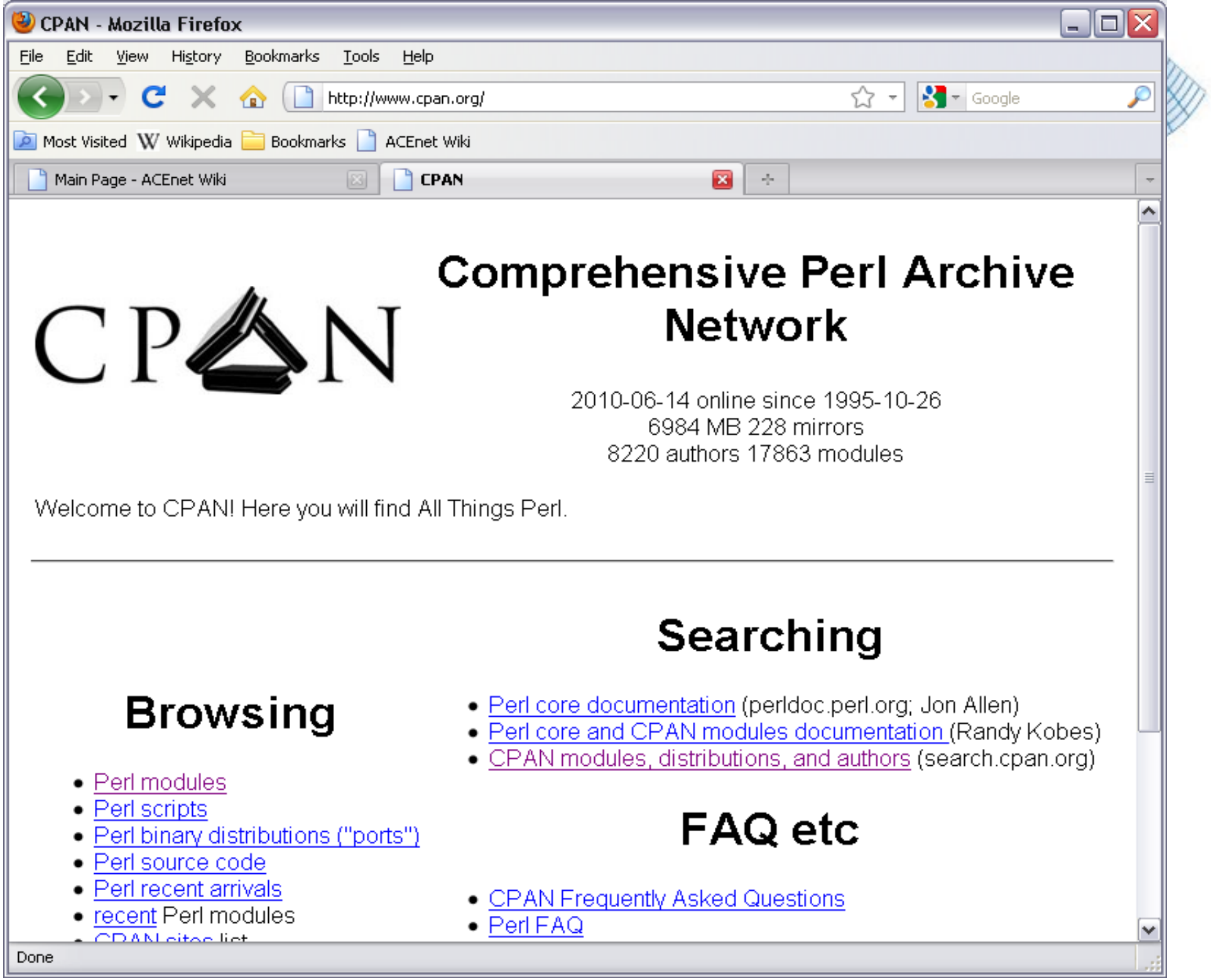
Not to mention...

- subroutines and shift( )
- references
- regular expressions and pattern matching operators
- special variables besides **`$_, @ARGV, $!`**
- logical “short-circuit” operators
- operators: arithmetic, string, bitwise, file test
- scalar context vs. list context
- local scope vs. global scope
-
-

http://www.perl.org



http://www.cpan.org



http://www.oreilly.com, or Amazon, or Chapters

O'Reilly Search: Perl - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://search.oreilly.com/?q=Perl

Google

Most Visited Wikipedia Bookmarks ACEnet Wiki

Main Page - ACEnet Wiki | O'Reilly Search: Perl

Displaying 1-12 of 1,519 results for: Perl

perl on O'Reilly

A compilation of all perl information produced or published by O'Reilly Media.

Learning Perl, 5th Edition

By Randal L. Schwartz, Tom Phoenix, brian d foy

Learning Perl, popularly known as the Llama, is the book most programmers rely on to get started with Perl. The bestselling Perl tutorial since it was first published in 1993, this new fifth edition covers recent changes to the language up to Perl 5....

Format: Print, Ebook, Safari Books Online

Publish Date: June 2008

Other Editions: 1st Edition, 2nd Edition, 3rd Edition, 4th Edition

Programming Perl, 3rd Edition

By Larry Wall, Tom Christiansen, Jon Orwant

Programming Perl is not just a book about Perl; it is also a unique introduction to the language and its culture, as one might expect only from its authors. This third edition has been expanded to cover Version 5.6 of Perl. New topics include threadi...

Format: Print, Ebook, Safari Books Online

Publish Date: July 2000

Other Editions: perl Edition, 2nd Edition

Done

Perl Data Language - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://pdl.perl.org/

Perl Data Language

Most Visited Wikipedia Bookmarks ACEnet Wiki

Perl Data Language - ACEnet Wiki | Perl Data Language

PDL $\sum a_n \cos(nx) + b_n \sin(nx)$

```
perldl> p sequence 5,5
[
 [ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]
]
```

Quick links

- Home
- Download
- External Libs

Support

- Wiki
- Mailing list
- Documentation
- FAQ

Usage

- Success Stories
- Screenshots
- Demos

Credits

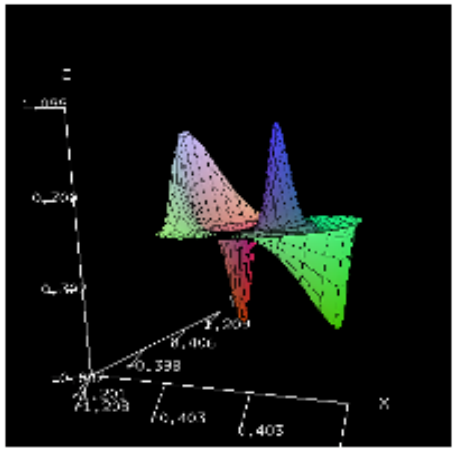
- The Team
- Citation

Development

- Get Started

Perl Data Language

Scientific computing with Perl

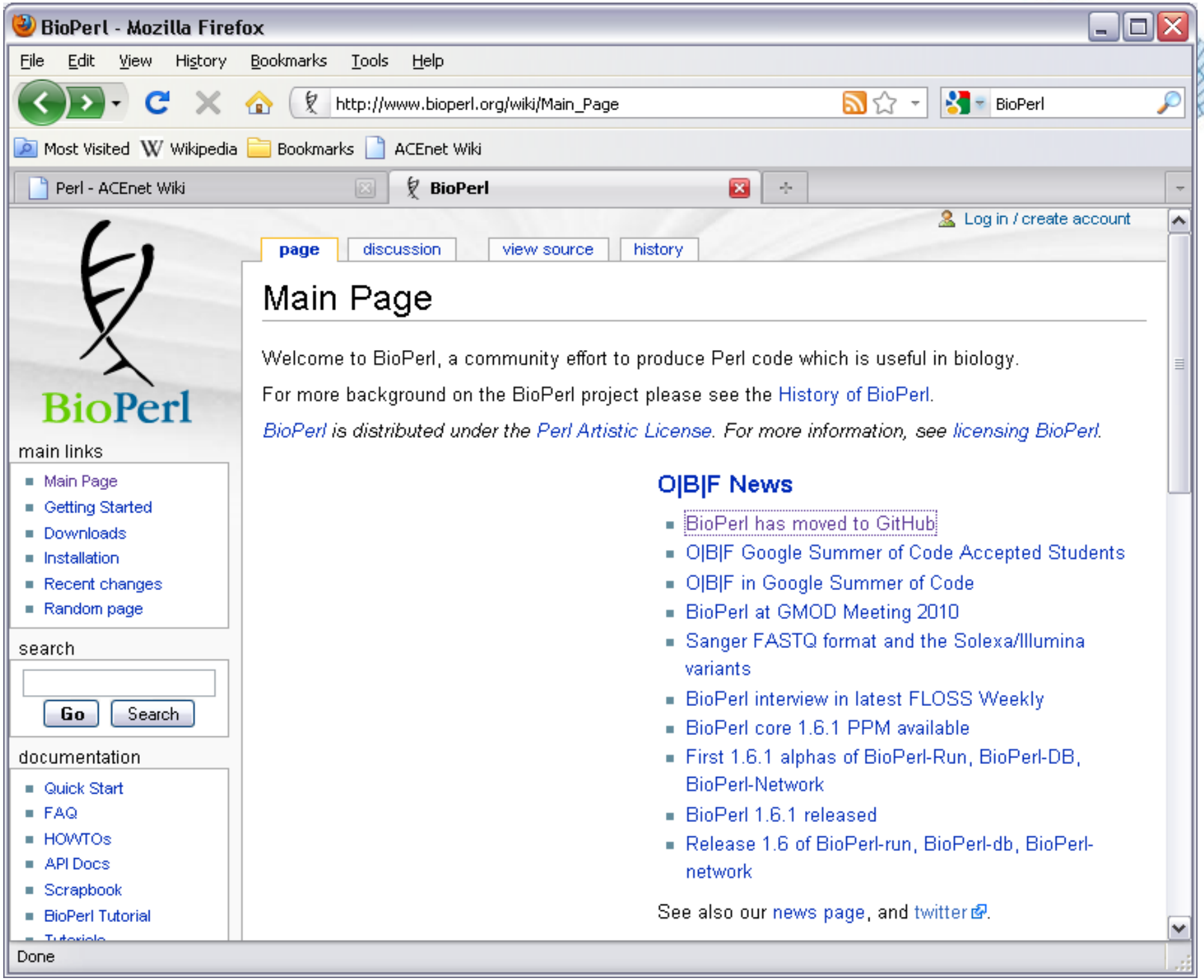


PDL ("Perl Data Language") gives standard Perl the ability to *compactly* store and *speedily* manipulate the large N-dimensional data arrays which are the bread and butter of scientific computing.

PDL turns Perl in to a free, array-oriented, numerical language similar to (but, we believe, better than) such commerical packages as IDL and MatLab. One can write simple perl expressions to manipulate entire numerical arrays all at once. For example, using PDL the Perl variable `$a` can hold a 1024x1024 floating point image, it only takes 4MB of memory to store it and expressions like `$a = sqrt($a) + 2` manipulate the whole image in a few milliseconds.

Done

http://pdl.perl.org



http://www.bioperl.org

Example code

- On ACEnet clusters...
  - brasdor
  - courtenay
  - fundy
  - glooscap
  - mahone
  - placentia
- ...in /home/rdickson/public/Perl_demo.tar
  - “tar xf /home/rdickson/public/Perl_demo.tar”