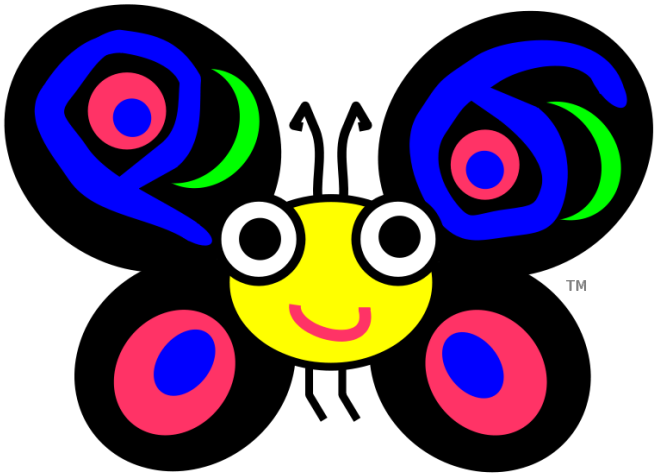


Perl 6
A Dynamic Language for Mere Mortals

Curtis "Ovid" Poe

http://allaroundtheworld.fr/

(With the unwitting assistance of Manuel Cerón http://ceronman.com/)

Copyright 2015, http://www.allaroundtheworld.fr/

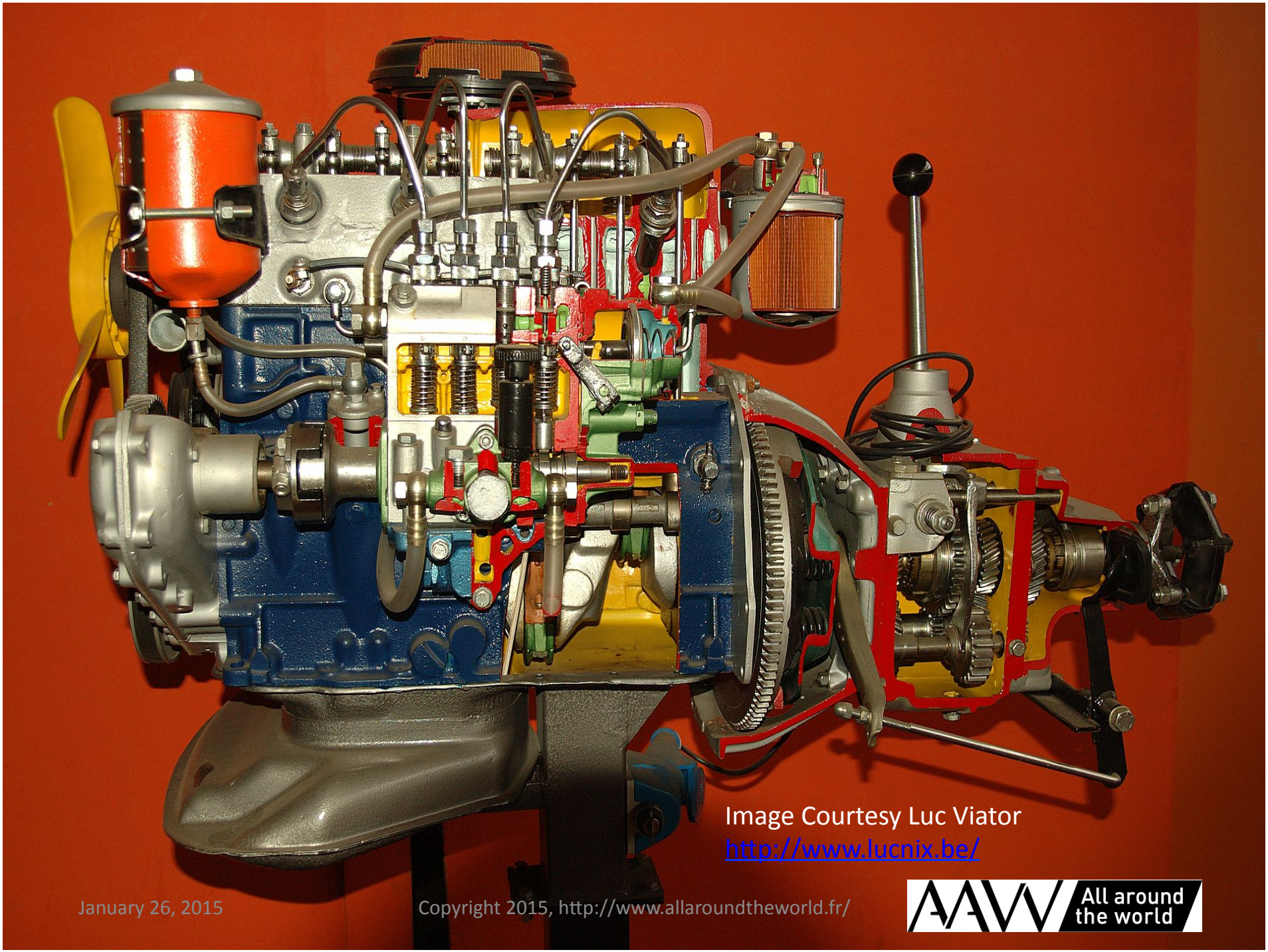


Image Courtesy Luc Viator
http://www.lucnix.be/

January 26, 2015

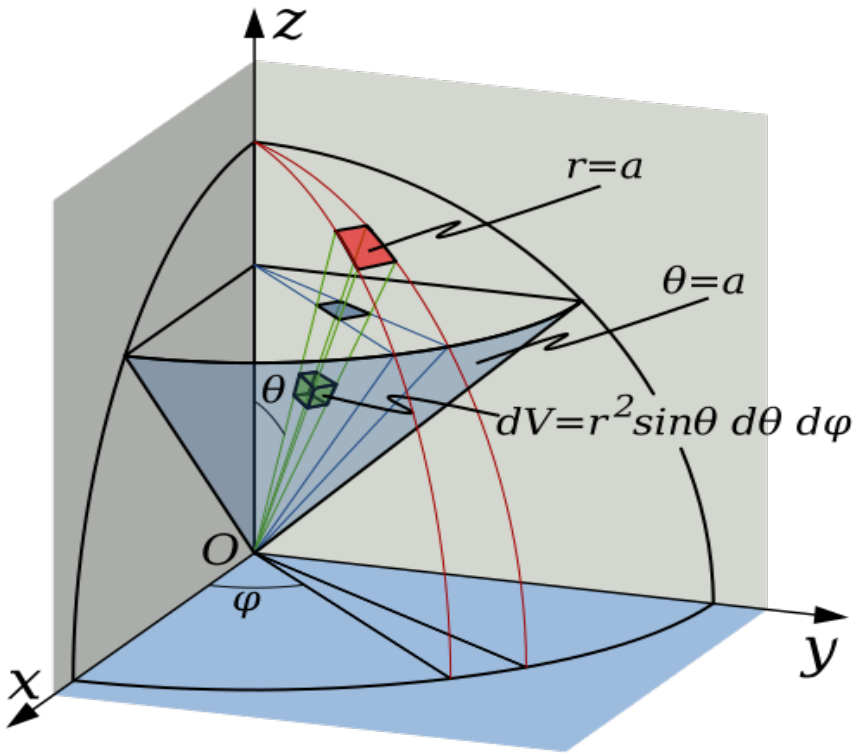
Copyright 2015, http://www.allaroundtheworld.fr/

All around the world

Basic Math

Copyright 2015, http://www.allaroundtheworld.fr/

Basic Math



Find the volume of the region bounded above by the sphere x^2+y^2+z^2 = a^2 and below by the cone z^2 sin^2(a) = (x^2+y^2)cos^2(a) where a is in the interval [0,π]

http://commons.wikimedia.org/wiki/Category:Integral_calculus#mediaviewer/File:Triple_Integral_Example_2.svg

Copyright 2015, http://www.allaroundtheworld.fr/

Solve for "x"

$$x = 7 \div 2$$

Copyright 2015, http://www.allaroundtheworld.fr/

Ruby

```
$ ruby -e 'puts 7/2'
3
```


Copyright 2015, http://www.allaroundtheworld.fr/

Python (2)

```
$ ruby -e 'puts 7/2'
3
$ python -c 'print 7/2'
3
```


Copyright 2015, http://www.allaroundtheworld.fr/

TCL

```
$ ruby -e 'puts 7/2'
3
$ python -c 'print 7/2'
3
$ echo "puts [expr 7/2]" | tclsh
3
```


Copyright 2015, http://www.allaroundtheworld.fr/

bc

```
$ ruby -e 'puts 7/2'
3
$ python -c 'print 7/2'
3
$ echo "puts [expr 7/2]" | tclsh
3
$ echo "7/2" | bc
3
```


Copyright 2015, http://www.allaroundtheworld.fr/

C

```
$ echo 'int printf(const char * restrict
  format, ... ); int main(void)
  { printf("\n%f\n", 7/2); return 0; }' \
> | gcc -xc - && ./a.out

<stdin>:1:82: warning: format specifies
  type 'double' but the argument has type
  'int' [-Wformat]
1 warning generated.

0.000000
```


Copyright 2015, http://www.allaroundtheworld.fr/

C

```
$ echo 'int printf(const char * restrict
  format, ... ); int main(void)
  { printf("\n%d\n", 7/2); return 0; }' \
> | gcc -xc - && ./a.out

3
```


Copyright 2015, http://www.allaroundtheworld.fr/

The purpose of software is to help humans, not computers.

Copyright 2015, http://www.allaroundtheworld.fr/

Perl 5

```
$ perl -E 'say 7/2'
3.5
```


Copyright 2015, http://www.allaroundtheworld.fr/

Solve for "x"

$$x = -7 \div 2$$

Copyright 2015, http://www.allaroundtheworld.fr/

C

```
$ echo 'int printf(const char * restrict
  format, ... ); int main(void)
  { printf("\n%d\n", -7/2); return 0; }' \
> | gcc -xc - && ./a.out

-3
```


Copyright 2015, http://www.allaroundtheworld.fr/

-7/2

```
$ ruby -e 'puts -7/2'
?
$ python -c 'print -7/2'
?
$ echo "puts [expr -7/2]" | tclsh
?
$ echo "-7/2" | bc
?
$ perl -E 'say -7/2'
-3.5
```


Copyright 2015, http://www.allaroundtheworld.fr/

bc

```
$ ruby -e 'puts -7/2'
-4
$ python -c 'print -7/2'
-4
$ echo "puts [expr -7/2]" | tclsh
-4
$ echo "-7/2" | bc
-3
$ perl -E 'say -7/2'
-3.5
```


Copyright 2015, http://www.allaroundtheworld.fr/

C89 — ANSI-C (3.3.5)

When integers are divided and the division is inexact …

if either operand is negative …

the result …

is implementation-defined.

Copyright 2015, http://www.allaroundtheworld.fr/

Solve for "x"

$$x = .1 + .2 - .3$$

Copyright 2015, http://www.allaroundtheworld.fr/

Solve for "x"

$$x = .1 + .2 - .3$$

Copyright 2015, http://www.allaroundtheworld.fr/

Solve for "x"

$$x = .3 - .3$$

Copyright 2015, http://www.allaroundtheworld.fr/

Solve for "x"

$$x = 0$$

Copyright 2015, http://www.allaroundtheworld.fr/

Solve for "x"

$$0 = .1 + .2 - .3$$

Copyright 2015, http://www.allaroundtheworld.fr/

Perl's Idea of Zero

```
$ perl -E 'say .1 + .2 - .3'
5.55111512312578e-17

# 0.000000000000000055511512312578
```


Copyright 2015, http://www.allaroundtheworld.fr/

One Divided by Zero

```
$ perl -E 'say 1/(.1 + .2 - .3)'
1.80143985094819e+16

# 18,014,398,509,482,000
# or roughly 18 quadrillian
```


Copyright 2015, http://www.allaroundtheworld.fr/

Mass of the Sun Multiplied by Zero

```
$ perl -E 'say 1.99E30 * (.1+.2-.3)'
110467190950203

# 110,467,190,950,203
# Roughly 110 trillion kilograms
```


Copyright 2015, http://www.allaroundtheworld.fr/

http://en.wikipedia.org/wiki/Mount_Everest#mediaviewer/File:Everest_North_Face_toward_Base_Camp_Tibet_Luca_Galuzzi_2006.jpg

Copyright 2015, http://www.allaroundtheworld.fr/

All around the world

Other Languages

```
$ ruby -e 'puts 0.1 + 0.2 - 0.3'
5.551115123125783e-17
$ python -c 'print .1 + .2 - .3'
5.55111512313e-17
$ echo "puts [expr .1+.2-.3]"|tclsh
5.551115123125783e-17
$ echo ".1 + .2 - .3" | bc
0
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6's Idea of Zero

```
$ perl6 -e 'say .1 + .2 - .3'
0
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6's Idea of Zero

```
$ perl6 -e 'say 1/(.1 + .2 - .3)'
# Divide by zero in method
  Numeric at …
```


Copyright 2015, http://www.allaroundtheworld.fr/

say WHAT?

```
$ perl6
> say .3.WHAT
(Rat)
> say .3.numerator
3
> say .3.denominator
10
> say .3.nude.perl
(3, 10)
> say 3.1415927.nude.perl
(31415927, 10000000)
```


Copyright 2015, http://www.allaroundtheworld.fr/

But you can forget about that

```
$ perl6
> say .1 + .2 - .3
0
```


Copyright 2015, http://www.allaroundtheworld.fr/

Functions

```
sub reciprocal {
   return 1 / shift;
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

TIMTOWTE mbarrass Y ourself

```
sub reciprocal {
  return 1 / shift;
}
sub reciprocal {
  1/$_[0];
}
sub reciprocal {
  my $num = shift;
  return 1/$num;
}
sub reciprocal {
  my ($num) = @_;
  return 1/$num;
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Screw This

```
sub reciprocal {
  return 1 / shift;
}
sub reciprocal {
  1/$_[0];
}
sub reciprocal {
  my $num = shift;
  return 1/$num;
}
sub reciprocal {
  my ($num) = @_;
  return 1/$num;
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Fibonacci

1. Start with the list **0 , 1**
2. Add last two elements
3. Append result to end of list
4. Go to step 2

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

I'd credit the original source if I could find it.

Copyright 2015, http://www.allaroundtheworld.fr/

Fibonacci – Hate Me If You Must

0, 1

0, 1 → 1

0, 1, 1 → 2

0, 1, 1, 2 → 3

0, 1, 1, 2, 3 → 5

0, 1, 1, 2, 3, 5 → 8

0, 1, 1, 2, 3, 5, 8 → 13

…

Copyright 2015, http://www.allaroundtheworld.fr/

Basic Function Signatures

```
sub fib($nth) {
  given $nth {
    when 0  { 0 }
    when 1  { 1 }
    default {fib($nth-1)+fib($nth-2)}
  }
}
say fib(8);
# 21
```


Copyright 2015, http://www.allaroundtheworld.fr/

Infinite Loop

```
sub fib($nth) {
  given $nth {
    when 0  { 0 }
    when 1  { 1 }
    default {fib($nth-1)+fib($nth-2)}
  }
}
say fib(3.7);
# …
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 5

```
sub fib {
  die unless $_[0] =~ /^\d+$/;
  …

sub fib {
  die unless $_[0] == int($_[0]);
  …

use Regexp::Common;
sub fib {
  die unless $_[0] =~ $RE{num}{int};
  …
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 – Optional Type Checking

```
sub fib(Int $nth) {
  given $nth {
    when 0  { 0 }
    when 1  { 1 }
    default {fib($nth-1)+fib($nth-2)}
  }
}
say fib(3.7);
# Type check failed in binding $nth;
  expected 'Int' but …
```


Copyright 2015, http://www.allaroundtheworld.fr/

Infinite Loop

```
sub fib(Int $nth) {
  given $nth {
    when 0  { 0 }
    when 1  { 1 }
    default {fib($nth-1)+fib($nth-2)}
  }
}
say fib(-3);
# …
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 — Constraints

```
sub fib(Int $nth where * >= 0) {
  given $nth {
    when 0  { 0 }
    when 1  { 1 }
    default {fib($nth-1)+fib($nth-2)}
  }
}
say fib(-3);
# Constraint type check failed for
  parameter '$nth' …
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 — Subsets

```
subset NonNegativeInt of Int where * >= 0;
sub fib(NonNegativeInt $nth) {
  given $nth {
    when 0  { 0 }
    when 1  { 1 }
    default {fib($nth-1)+fib($nth-2)}
  }
}
say fib(-3);
# Constraint type check failed for parameter
  '$nth' …
```


Copyright 2015, http://www.allaroundtheworld.fr/

Function Signatures

```
1.sub optional                      {…}
2.sub basic($foo)                   {…}
3.sub default($foo = 3)             {…}
4.sub named(:$name)                 {…}
5.sub typed(Bool $is_foo)           {…}
6.sub constraint(Str $name
    where *.chars > 0)              {…}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Function Signatures

```
subset NonEmptyString of Str
 where *.chars > 0;

sub foo(NonEmptyString $name){…}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Function Signatures

```
subset FirstName of Str
  where 0 < *.chars < 256;

sub FirstName(Firstname $name){…}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Speed It Up

```
sub fib(NonNegativeInt $nth) {
  state %fib_for;
  unless %fib_for{$nth}:exists {
    given $nth {
      when 0  { 0 }
      when 1  { 1 }
      default { %fib_for{$nth}
                = fib($nth-1) + fib($nth-2) }
    }
  }
  return %fib_for{$nth};
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Speed It Up

```
sub fib(NonNegativeInt $nth) {
  my ( $prev, $next ) = 0, 1;
  my $result;
  for 0 .. $nth {
    $result = $prev + $next;
    $prev   = $next;
    $next   = $result;
  }
  return $result;
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Fibonacci Sequence

$F_0 = 0$

$F_1 = 1$

$F_n = F_{n-1} + F_{n-2}$

```
when 0   { 0 }
when 1   { 1 }
default {fib($nth-1)+fib($nth-2)}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Speed It Up

```
sub fib(NonNegativeInt $nth) is cached {
   given $nth {
      when 0   { 0 }
      when 1   { 1 }
      default {fib($nth-1)+fib($nth-2)}
   }
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Multi Subs

```
sub travel_to {
  my ( $self, $location ) = @_;

  # my kingdom for MMD :(
  if ( $location->isa('StationArea') ) {
    $self->_travel_to_area($location);
  } elsif ( $location->isa('Station') ) {
    $self->_travel_to_station($location);
  } elsif ( $location->isa('Star') ) {
    $self->_travel_to_star($location);
  } else {
    croak("I don't know how to travel to $location");
  }
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

An ugly alternative

```
sub travel_to {
  my ( $self, $location ) = @_;

  if ( $location->does('Location') ) {
    # OVS instead of SVO
    $location->travel_to($character);
  } else {
    croak("I don't know how to travel to $location");
  }
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

If I Used Perl 6

```
multi method travel_to(StationArea $station_area) { ... }
multi method travel_to(Station     $station)      { ... }
multi method travel_to(Star        $star)         { ... }
```


Copyright 2015, http://www.allaroundtheworld.fr/

And the Fibonacci Numbers Again

$F_0 = 0$

$F_1 = 1$

$F_n = F_{n-1} + F_{n-2}$

```
multi fib(0) {0}
multi fib(1) {1}
multi fib($n where * > 1) { fib($n-1) + fib($n-2) }
```


Copyright 2015, http://www.allaroundtheworld.fr/

Asserting Return Types

```
sub foo() returns Bool {
  return some-other-func(); # what does this return?
}

say "Yes" if foo();
# Yes
```


Copyright 2015, http://www.allaroundtheworld.fr/

Asserting Return Types

```
sub foo() returns Bool {
  return "Foo";
}

say "Yes" if foo();
# Type check failed for return value;
# expected 'Bool' but got 'Str'
```


Copyright 2015, http://www.allaroundtheworld.fr/

Asserting Return Types

```
sub foo() returns NonNegativeInteger {
  return some-other-func();
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Classes

```
class Point {
    has $.x = 0;
    has $.y = 0;

    method Str { "[$.x,$.y]" }
}
my $point = Point.new( x => 5, y => 3 );
say $point.x;                                   # 5
say $point.y;                                   # 3
say "$point";                                   # [5,3]
```


Copyright 2015, http://www.allaroundtheworld.fr/

Classes

```
class Point {
    has $.x = 0;
    has $.y = 0;

    method Str { "[$.x,$.y]" }
    method set( :$x = $.x, :$y = $.y ) {
        $!x = $x; $!y = $y;
    }
}
my $point = Point.new( x => 5, y => 3 );
say "$point";                                  # [5,3]
$point.set( y => 17 );
say "$point";                                  # [5,17]
```


Copyright 2015, http://www.allaroundtheworld.fr/

And as you learn more …

```
class Point {
    has Real $.x = 0;
    has Real $.y = 0;

    method Str { "[$.x,$.y]" }
    method set( :$x = $.x, :$y = $.y ) {
        $!x = $x; $!y = $y;
    }
}
my $point = Point.new( x => 5, y => 3 );
say "$point";                                  # [5,3]
$point.set( y => "foo" );                      # Boom!
```


Copyright 2015, http://www.allaroundtheworld.fr/

And as you learn more …

```
class Point {
    has Real $.x is rw = 0;
    has Real $.y is rw = 0;

    method Str { "[$.x,$.y]" }
}
```

```
my $point = Point.new( x => 5, y => 3.0 );
say "$point";                                  # [5,3]
$point.y = 17.3;
say "$point";                                  # [5,17.3]
```


Copyright 2015, http://www.allaroundtheworld.fr/

And as you learn more …

```
class Point {
    subset PointLimit of Real where -10 <= * <= 10;
    has PointLimit $.x is rw = 0;
    has PointLimit $.y is rw = 0;

    method Str { "[$.x,$.y]" }
}


my $point = Point.new( x => 5, y => 3.0 );
say "$point";                                  # [5,3]
$point.y = 17.3;                               # Boom!
```


Copyright 2015, http://www.allaroundtheworld.fr/

The default values are silly …

```
class Point {
    subset PointLimit of Real where -10 <= * <= 10;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';

    method Str { "[$.x,$.y]" }
}


my $point = Point.new( y => 3 );        # Boom!
```


Copyright 2015, http://www.allaroundtheworld.fr/

Or …

```
class Point {
    subset PointLimit of Real where -10 .. 10;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';

    method Str { "[$.x,$.y]" }
}


my $point = Point.new( x => 5, y => 3.0 );
say "$point";                                  # [5,3]
$point.y = 17.3;                               # Boom!
```


Copyright 2015, http://www.allaroundtheworld.fr/

Guess what …

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';

    method Str { "[$.x,$.y]" }
}


my $point = Point.new( x => 5, y => 3.0 );
say "$point";                                  # [5,3]
$point.y = 17.3;                               # Boom!
```


Copyright 2015, http://www.allaroundtheworld.fr/

Simpler

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}


my $point = Point.new( x => 5, y => 3.0 );
$point.y = 7.2;
say $point.perl;      # Point.new(x => 5, y => 7.2)
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus Core Perl 5

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
package Point {
    use Carp;
    use overload '""' => \&Str, fallback => 1;
    sub _pointlimit {
        my ( $class, $num ) = @_;
        unless ( $num >= -10 && $num <= 10 ) {
            croak "…";
        }
    }
    sub new {
        my ( $class, $x, $y ) = @_;
        my $self = bless { x => undef, y => undef } => $class;
        $self->x($x);
        $self->y($y);
        return $self;
    }
    sub x {
        my ( $self, $x ) = @_;
        $self->_pointlimit ($x);
        $self->{x} = $x;
    }
    sub y {
        my ( $self, $y ) = @_;
        $self->_pointlimit ($y);
        $self->{y} = $y;
    }
    sub Str {
        my $self = shift;
        return sprintf "[%f,%f]" => $self->x, $self->y;
    }
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus Moose

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
package Point {
    use Moose;
    use overload '""' => \&Str, fallback => 1;
    use Moose::Util::TypeConstraints;
    subtype "PointLimit" => as 'Num'
      => where { $_ >= -10 && $_ <= 10 }
      => message { "$_ must be a Num between -10 and 10, inclusive" };

    has [qw/x y/] => (
      is       => 'rw',
      isa      => 'PointLimit',
      required => 1,
    );

    sub Str {
      my $self = shift;
      return sprintf "[%f,%f]" => $self->x, $self->y;
  }
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus C++

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
#include <iostream>
#include <sstream>
class Point {
    double x_, y_;
  public:
    Point (double x, double y) : x_(constrain(x)), y_(constrain(y)) {}
    double x () const { return x_; }
    double y () const { return y_; }

    void x (double num) { x_ = constrain(num); }
    void y (double num) { y_ = constrain(num); }

    friend std::ostream & operator<< (std::ostream & stream, const Point & point) {
        stream << '[' << point.x() << ',' << point.y() << ']';
        return stream;
    }
  private:
    static double constrain (double num) {
        if (num < -10 || num > 10) {
            std::stringstream ss;
            ss << "'" << num << "' is not a real number between -10 and 10, inclusive";
            throw(ss.str());
        }
        return num;
    }
};

int main () {
    try {
        Point point(5, 3);
        std::cout << point << std::endl;
        point.y(17.3);
    }
    catch (const std::string & error) {
        std::cerr << error << std::endl;
    }
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus Java

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
public class Point {
    private static final double X_MIN = -10.0, X_MAX = 10.0;
    private static final double Y_MIN = -10.0, Y_MAX = 10.0;
    private double x, y;

    public Point(double x, double y) throws IllegalArgumentException {
        setX(x);
        setY(y);
    }

    public double getX() { return x; }
    public double getY() { return y; }

    public void setX(double x) throws IllegalArgumentException {
        if (x < X_MIN || x > X_MAX)
          throw new IllegalArgumentException("…");
        this.x = x;
    }
    public void setY(double y) throws IllegalArgumentException {
        if (y < Y_MIN || y > Y_MIN)
          throw new IllegalArgumentException("…");
        this.y = y;
    }

    @Override
    public String toString() {
        return String.format("[%.1f,%.1f]", x, y);
    }
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus Python 3

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
class PointLimit:
    def __init__(self, name):
        self.name =  name
    def __get__(self, point, owner):
        return point.__dict__.get(self.name)
    def __set__(self, point, value):
        if not -10 < value < 10:
            raise ValueError('Value %d is out of range' % value)
        point.__dict__[self.name] = value

class Point:
    x = PointLimit('x');
    y = PointLimit('y');

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return "[%f,%f]" % (self.x, self.y)
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus Javascript

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
function definePointLimit(point, name) {
    Object.defineProperty(point, name, {
        set: function (value) {
            if (value < -10 || value > 10)
                throw 'Value ' + value + ' is out of range';
            point['_' + name] = value;
        },
        get: function () {
            return point['_' + name];
        }
    });
}

function Point(x, y) {
    definePointLimit(this, 'x');
    definePointLimit(this, 'y');

    this.x = x;
    this.y = y;
}

Point.prototype.toString = function() {
    return '[' + this.x + ',' + this.y + ']';
}

var point = new Point(5, 5);
point.y = 100; // Boom!
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus Ruby

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
class Point
  POINT_RANGE = -10..10

  attr_constrained x: POINT_RANGE, y: POINT_RANGE

  def initialize
    @x = 0
    @y = 0
  end

  def to_s
    "[#{x},#{y}]"
  end

private
  def self.attr_constrained(attrs_and_constraints)
    attrs_and_constraints.each do |attr, constraint|
      attr_reader attr
      define_method(:"#{attr}=") do |value|
        raise "#{value} is not a valid value for #{attr}" \
            unless constraint === value
        instance_variable_set :"@#{attr}", value
      end
    end
  end
end
```


Copyright 2015, http://www.allaroundtheworld.fr/

Perl 6 Versus Go

```
class Point {
    subset PointLimit where -10.0 .. 10.0;
    has PointLimit $.x is rw = die '$.x is required';
    has PointLimit $.y is rw = die '$.y is required';
}
```

```
package point

import  "fmt"
import  "errors"

type Point struct {
    x, y float64
}

func NewPoint(x, y float64) (*Point, error) {
    p := &Point{x, y}
    if !IsValid(x) {
        return nil, errors.New("Point x out of range!")
    }
    if !IsValid(y) {
        return nil, errors.New("Point y out of range!")
    }
    return p, nil
}
func IsValid(p float64) (result bool) {
    if p >= -10 && p <= 10 {
        result = true
    }
    return
}
func (p *Point) String() string {
    return fmt.Sprintf("[%v,%v]", p.x, p.y)
}
func (p *Point) SetX(x float64) (float64, error) {
    ox := p.x
    if !IsValid(x) {
        return ox, errors.New("Point out of range!")
    }
    return ox, nil
}
func (p *Point) SetY(y float64) (float64, error) {
    oy := p.y
    if !IsValid(y) {
        return oy, errors.New("Point out of range!")
    }
    return oy, nil
}
func (p *Point) GetX() float64 {
    return p.x
}
func (p *Point) GetY() float64 {
    return p.y
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

For day-to-day code,
Perl 6 is *more correct* than Perl 5
Perl 6 is *safer* than Perl 5

Copyright 2015, http://www.allaroundtheworld.fr/

Review

- Math that *works*
- Function signatures
- Subsets
- Proper classes

Copyright 2015, http://www.allaroundtheworld.fr/

Summary

- Perl 6 is large
- So is Perl 5
- Common features "baked" into Perl 6
- Proper OO
- Easy to read
- Safer
- Very powerful

Copyright 2015, http://www.allaroundtheworld.fr/

Resources

- http://www.perl6.org/
- http://doc.perl6.org/
- http://design.perl6.org/
- http://learnxinyminutes.com/docs/perl6/
- irc.freenode.net #perl6

Copyright 2015, http://www.allaroundtheworld.fr/

Thank You!

Copyright 2015, http://www.allaroundtheworld.fr/

Bonus Slides?

Copyright 2015, http://www.allaroundtheworld.fr/

Infinite Lazy Lists

```
constant @primes = grep { .is-prime }, 1 .. *;
say @primes[^10];
# 2 3 5 7 11 13 17 19 23 29
```


Copyright 2015, http://www.allaroundtheworld.fr/

Infinite Lazy Lists

```
constant primes = grep { .is-prime }, 1 .. *;
say primes[^10];
# 2 3 5 7 11 13 17 19 23 29
```


Copyright 2015, http://www.allaroundtheworld.fr/

But it worked when I tested it!

```
open my $fh, '<', $file
  or die "Can't read $file: $!";

foreach (<$fh>) {
  print if /abcd/;
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

This is lazy

```
my $fh = open($file);

for $fh.lines -> $line {
   say $line if $line ~~ /abcd/;
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

So is this

```
my $fh = open($file);

for $fh.lines.grep({/abcd/}) ->$line {
  say $line;
}
```


Copyright 2015, http://www.allaroundtheworld.fr/

Lazy Fibonacci Numbers

```
my @fib = 0, 1, * + * ... *;
say @fib[^$num];
```


Copyright 2015, http://www.allaroundtheworld.fr/

Lazy Fibonacci Numbers

```
my @fib = 0, 1, * + * ... *;
say @fib[^$num];
```


Copyright 2015, http://www.allaroundtheworld.fr/

Arithmetic Sequences

```
my @sequence = 0, 2 ... *;
say @sequence[^10];
# 0 2 4 6 8 10 12 14 16 18
```


Copyright 2015, http://www.allaroundtheworld.fr/

Geometric Sequences

```
my @sequence = 2, 4, 8 ... *;
say @sequence[^10];
# 2 4 8 16 32 64 128 256 512 1024
```


Copyright 2015, http://www.allaroundtheworld.fr/

Complex Sequences

```
my @f = 0, 1, -> $a,$b {$a+$b} ... *;
say @f[^10];
# 0 1 1 2 3 5 8 13 21 34
```


Copyright 2015, http://www.allaroundtheworld.fr/

Complex Sequences

```
my @f = 0, 1, * + * ... *;
say @f[^10];
# 0 1 1 2 3 5 8 13 21 34
```


Copyright 2015, http://www.allaroundtheworld.fr/

Thank You!

(for real, this time)

Copyright 2015, http://www.allaroundtheworld.fr/