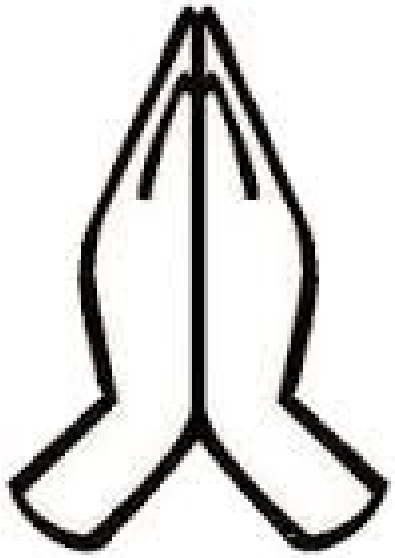


namaskaaram!

Sameer Deshmukh


github.com/v0dro


@v0dro

City of Pune.

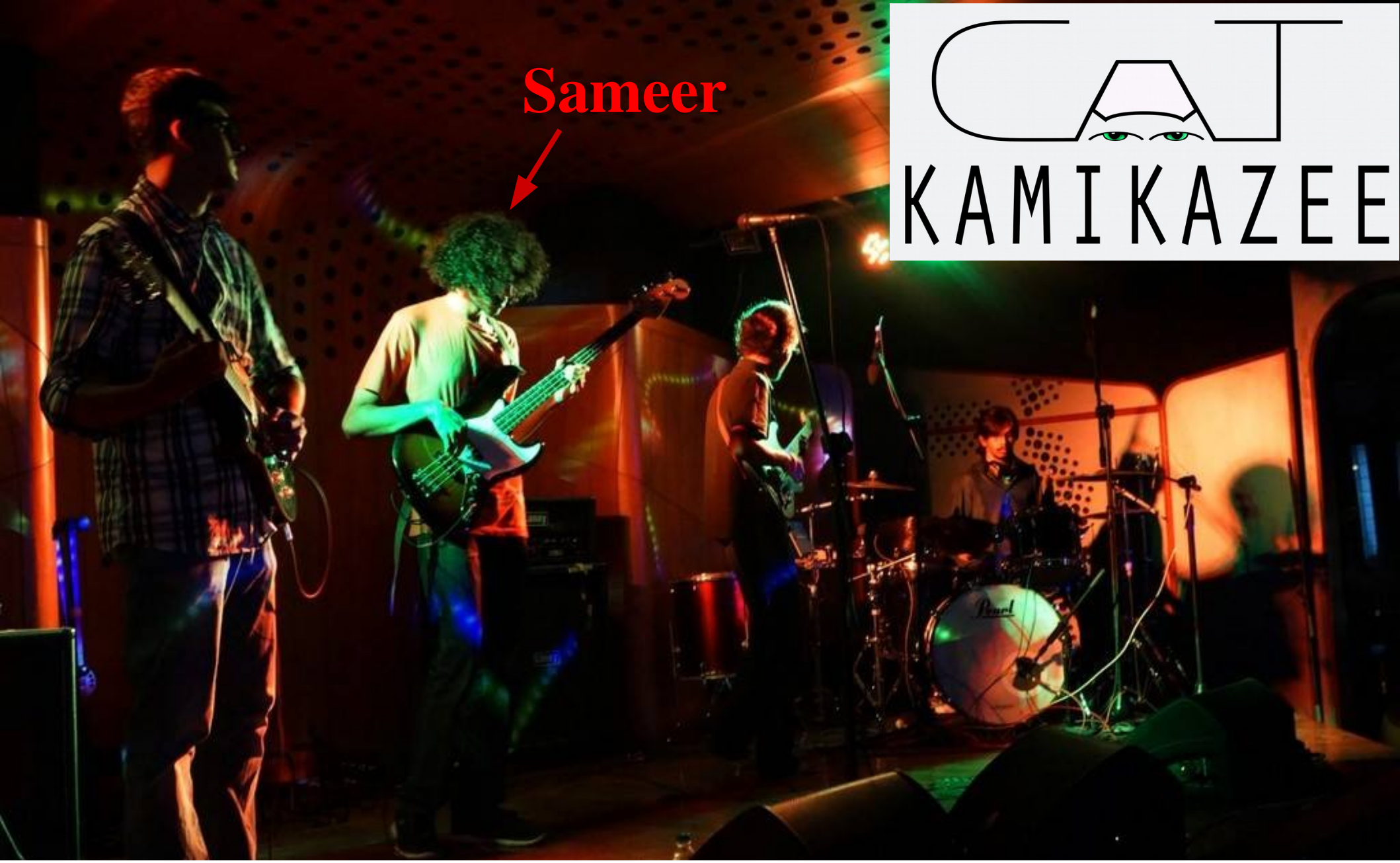
Population: 6 million.
Oxford of the East.

Dr. Gopal Deshmukh

Dr. Hemchandra Deshmukh

Dr. Satish Deshmukh

Sameer Desmukh (not a doctor)



catkamikazee.bandcamp.com

Pune Ruby Users Group

@punerb

@deccanrubyconf

www.punerb.org

www.deccanrubyconf.org

Ruby Science Foundation

@sciruby

@sciruby

www.sciruby.com


Google
SUMMER
of
CODE
2015

Daru -

Data Analysis in RUby

A Ruby gem for analysis, plotting and cleaning of data.

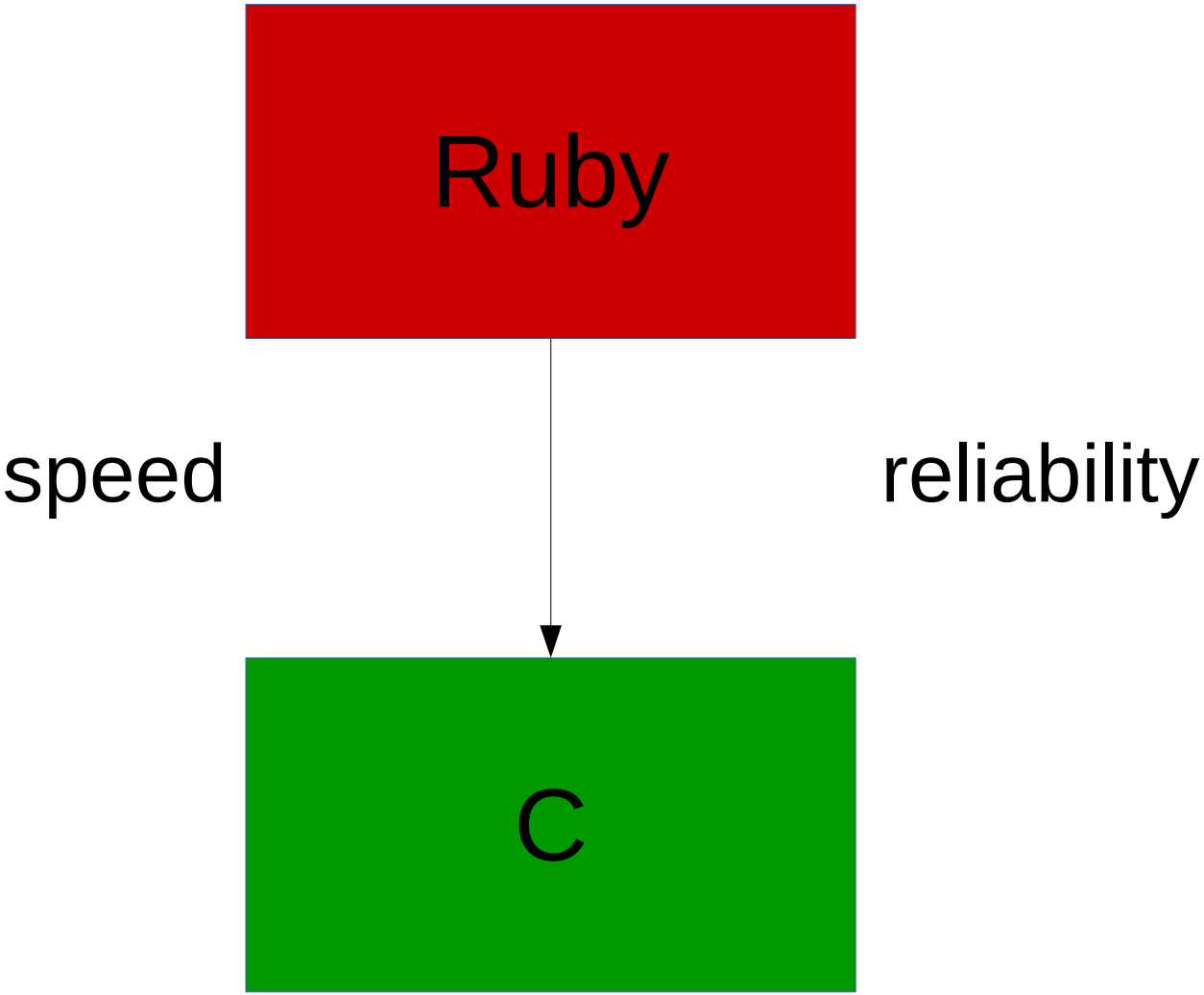
https://github.com/SciRuby/daru

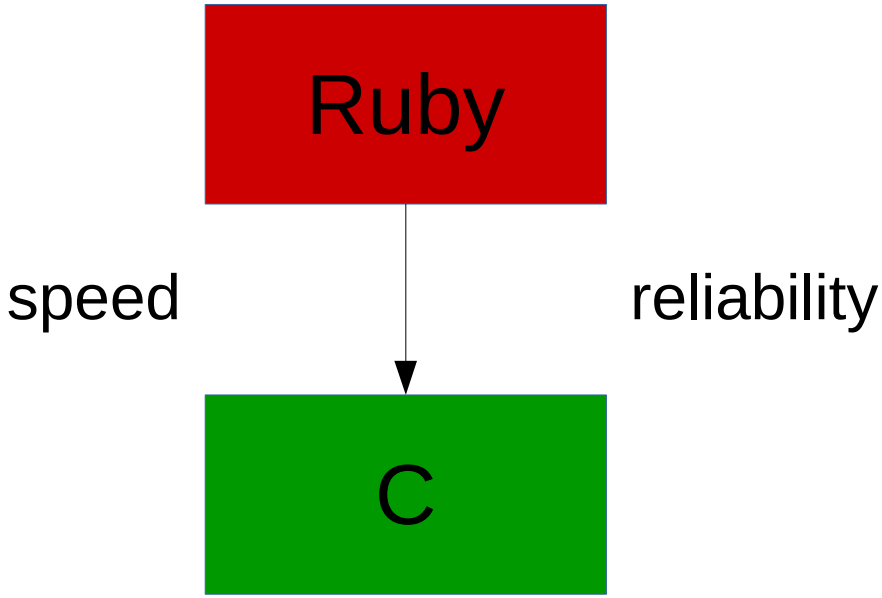

KERALA STATE
BEVERAGES (M&M) CORPORATION
SASTHAMANGALAM. THIRUVANANTHAPURAM
Pin. 695010. Ph. 23249 70.
(A KERALA GOVT. UNDERTAKING)
BRANCH. VVTTILA

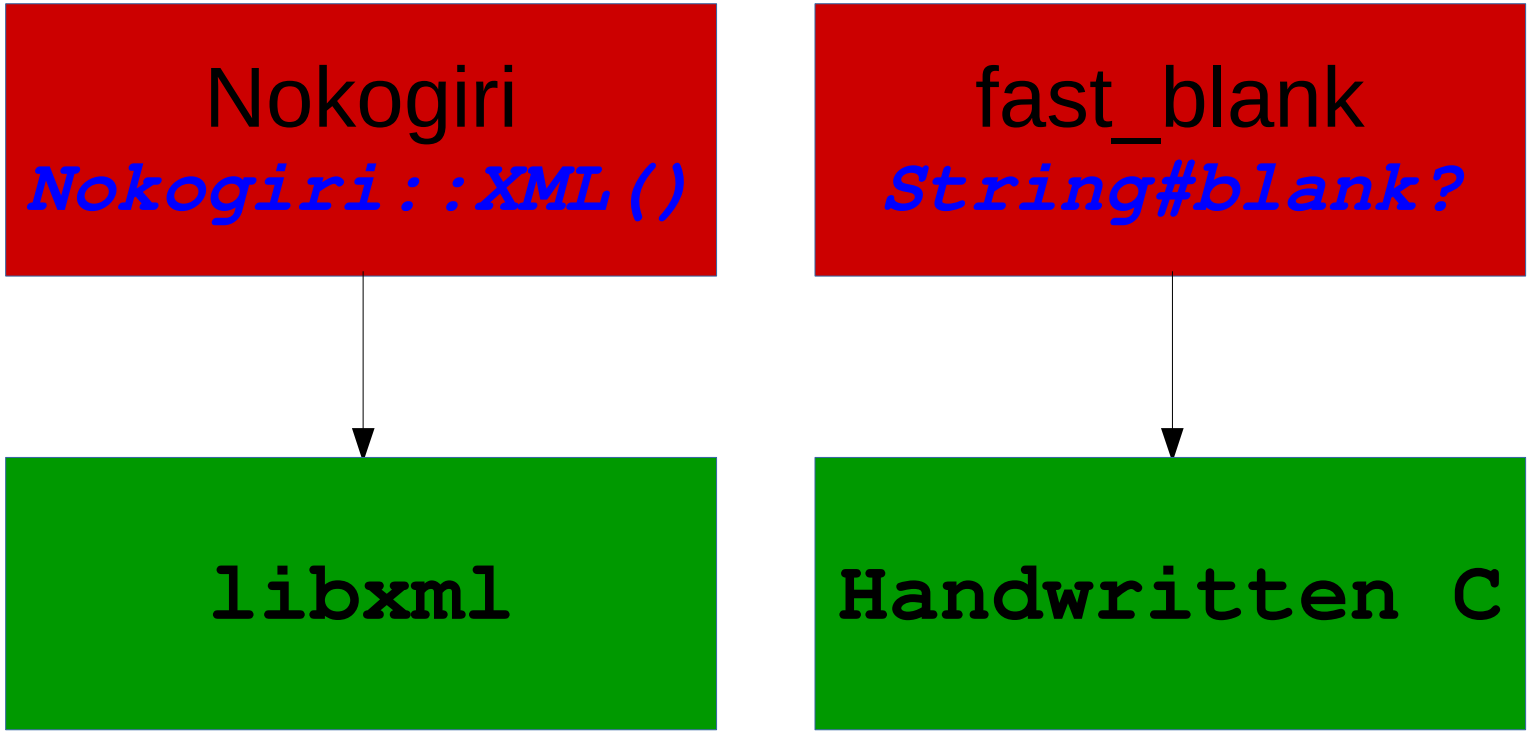
Rubex:

A better way to write Ruby C extensions.

What is a C extension?


Ruby
speed
reliability
C


Ruby
speed
reliability
C


Nokogiri
Nokogiri::XML()
fast_blank
String#blank?
libxml
Handwritten C

```ruby
# In test.rb
require 'fast_blank'
a = "hello"
a.blank?
```


```c
/* In fast_blank.c: */
VALUE rb_str_blank(VALUE str)
```

BIG Problems

- Difficult and irritating to write.
- Debugging is time consuming.
- Tough to trace memory leaks.
- Change mindset from high level to low level language.
- Remember the CRuby C API.
- Need to care about small things.™*

*Matz.

```
def addition a,b
  return a + b
end
```

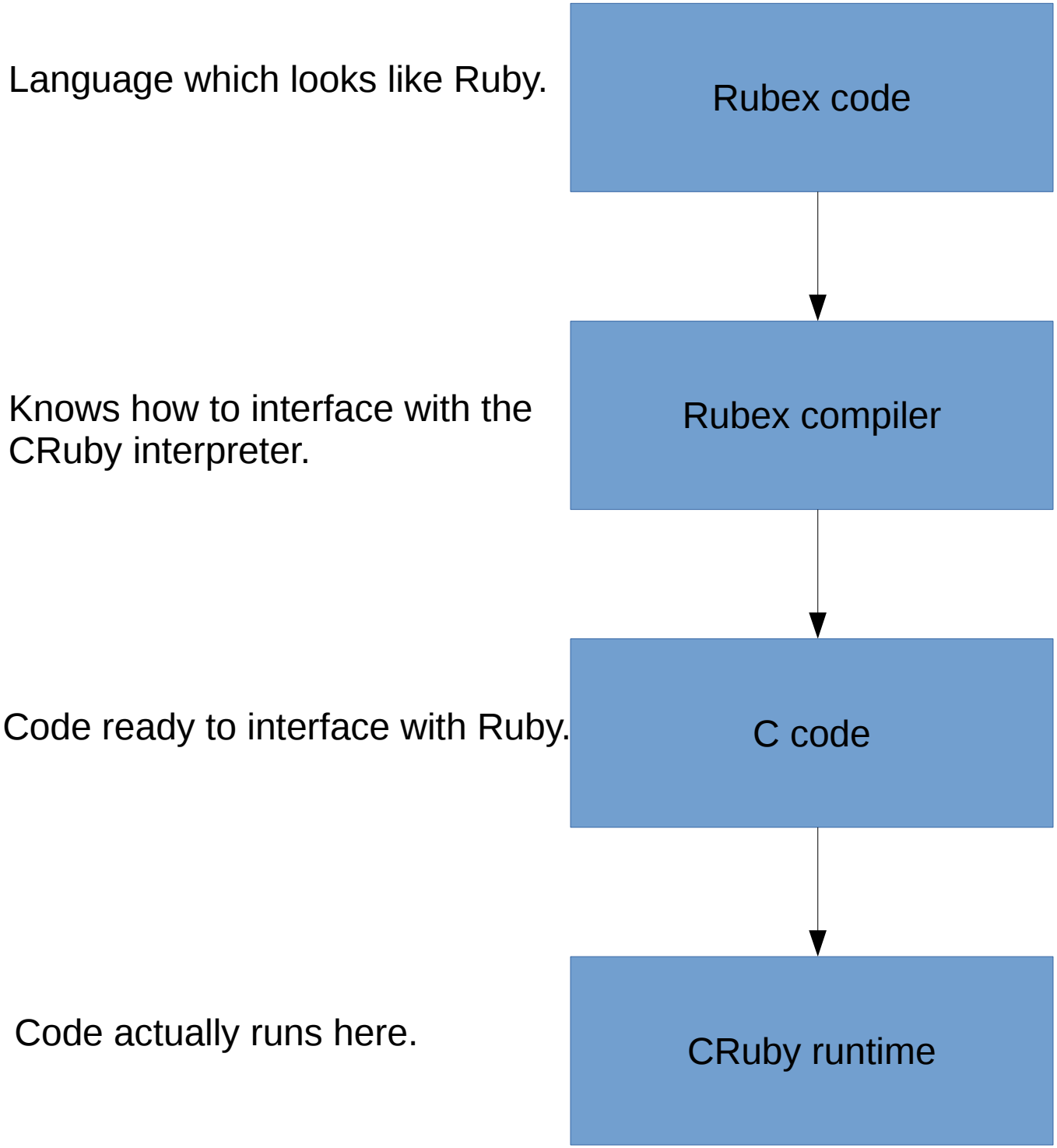
```
int
calc_addition(int a, int b)
{
  return (a + b);
}

static VALUE
caddition(VALUE self, VALUE a, VALUE b)
{
  int i = FIX2INT(a);
  int j = FIX2INT(b);
  return INT2FIX(calc_addition(i, j));
}
```

```
void Init_addition()
{
  VALUE cAdd = rb_define_class("Add",
     rb_cObject);

  rb_define_method(cAdd, "addition",
        caddition, 2);
}
```

```
require ‘addition.so’

a = Add.new
a.addition(5, 4)
```


WTF?!


Hmmm should try Rubex!


Language which looks like Ruby.
Rubex code
Knows how to interface with the CRuby interpreter.
Rubex compiler
Code ready to interface with Ruby.
C code
Code actually runs here.
CRuby runtime

```
# In file addition.rubex:

def addition(int a, int b)
  return a + b
end
```

```
# In file test.rb:
require ‘addition.so’

print addition(4,5)
```

```
static VALUE
rb_str_blank(VALUE str)
{
  // lots of unicode handling code omitted
  s = RSTRING_PTR(str);
  e = RSTRING_END(str);
  while (s < e) {
    //  cc = current character
    if (!rb_isspace(cc) && cc != 0)
      return Qfalse;
  }

  return Qtrue;
}
```

```
def blank?(string)
  i32 i = 0
  char *s = string
  i32 length = string.size

  while i < length do
    return false if s[i] != ' '
    i += 1
  end

  return true
end
```

```
def blank?(string)
  i32 i = 0
  char *s = string
  i32 length = string.size

  while i < length do
    return false if s[i] != ' '
    i += 1
  end

  return true
end
```

```
def blank?(string)
  i32 i = 0
  char *s = string
  i32 length = string.size

  while i < length do
    return false if s[i] != ' '
    i += 1
  end

  return true
end
```

```
def blank?(string)
  i32 i = 0
  char *s = string
  i32 length = string.size

  while i < length do
    return false if s[i] != ' '
    i += 1
  end

  return true
end
```

Benchmarks

- **Comparison :** fast_blank's `String#blank?` vs. `blank?` implemented in Rubex.
- **Data :** A Ruby String with 2500 spaces in the beginning and three ASCII letters at the end. Data taken so that non-trivial time will be spent on iterations to search for a white space.

```
str = " "*2500 + "dff"
```

- **Result:** This is new stuff is good.

Benchmark-ips results

```
Warming up --------------------------------------
        fast_blank     3.401k i/100ms
            blank?    57.041k i/100ms
Calculating -------------------------------------
        fast_blank     35.068k (± 0.4%) i/s -    176.852k in 5.043263s
            blank?    671.289k (± 1.1%) i/s -      3.365M in 5.014016s

Comparison:
            blank?:   671289.0 i/s
        fast_blank:    35067.6 i/s - 19.14x  slower
```

Conclusion of benchmarks:

- fast_blank is not that fast for ASCII strings.
- Now anybody can write C extensions with Rubex.

Most important use case of Rubex

- Not simply for abstracting away C code.
- SciRuby works with many highly optimized C libraries like ATLAS, BLAS, FFTW & GSL.
- These C libraries use complex API calls that need to be interfaced with Ruby with a lot of ‘glue’ code.
- Glue code is a pain to write/debug.

Interfacing external C libraries

- **Example :** `BLAS::gemm()` method for multiplying two square matrices.

```
gemm(
  const enum CBLAS_ORDER, const enum CBLAS_TRANSPOSE,
  const enum CBLAS_TRANSPOSE,
  const int, const int, const int, const double*,
  const double*, const int, const double*, const int,
  const double*, double*, const int
)
```

Interface `math.h` header file with Ruby using Rubex

```
# In file maths.rubex
lib math do
  double pow (double, double)
  double cos (double)
end

def maths(double power)
  double p = cos(0.5)
  return pow(p, power)
end
```

```
# In file maths.rubex
lib math do
  double pow (double, double)
  double cos (double)
end

def maths(double power)
  double p = cos(0.5)
  return pow(p, power)
end
```

```
# In file maths.rubex
lib math do
  double pow (double, double)
  double cos (double)
end

def maths(double power)
  double p = cos(0.5)
  return pow(p, power)
end
```

```
# In file maths.rubex
lib math do
  double pow (double, double)
  double cos (double)
end

def maths(double power)
  double p = cos(0.5)
  return pow(p, power)
end
```

```
# In file maths.rubex
lib math do
  double pow (double, double)
  double cos (double)
end

def maths(double power)
  double p = cos(0.5)
  return pow(p, power)
end
```

```
# In file test.rb
require ‘maths.so’

print maths(5.6)
```

Salient Features

Rubex is meant to be a super set of Ruby and is as a companion of Ruby.

It does not replace Ruby.

Everything in Rubex is NOT an object.

There can be both primitive C data types and Ruby objects co-existing in a single Rubex program.

You can declare

Abstract C Data Types

like Structs, Unions and Enums

using Rubex and pass them to

arbitrary C functions.

Future Roadmap

- Support both native C and Ruby functions.
- Ability to encapsulate methods in classes.
- Introduce advanced heuristics to convert between C and Ruby data types.
- Ability to release the Global Interpreter Lock and perform operations on native threads.

https://github.com/v0dro/rubex

Ruby Association

I haz stickers!

^_^

Thank you Kochi!

Thank you

Ruby Conf India!