my "perfect language" quest
a little tour

ariel faigon

Language Perfect

Obligatory comment

Comparing languages/technologies leads to religious arguments:

"My language is better than yours!"

Q: So who's right?

A: Unbiased metrics: speed, brevity, etc.

What's a good language?

Good language:

- *Does what we say (human → machine)*
- *Easy to write (human productivity)*
- *Easy to read (among humans)*

Some languages are better than others along these dimensions.

They make you more productive,

and they're more maintainable.

Is there really a difference?

Most programming language have:

- **Loops: for, while ...**
- **Conditions: if, else ...**
- **Function calls: f(x)**

Some have more libraries & built-in functions

But this is not a "library size" contest

Key: which language requires less "fluff"?

Is there a "perfect" language

In the minimalist sense:

The code does what you say

With nothing beyond the minimal, essential syntax to achieve the goal

Wikipedia: boilerplate-code

In object-oriented programming [edit]

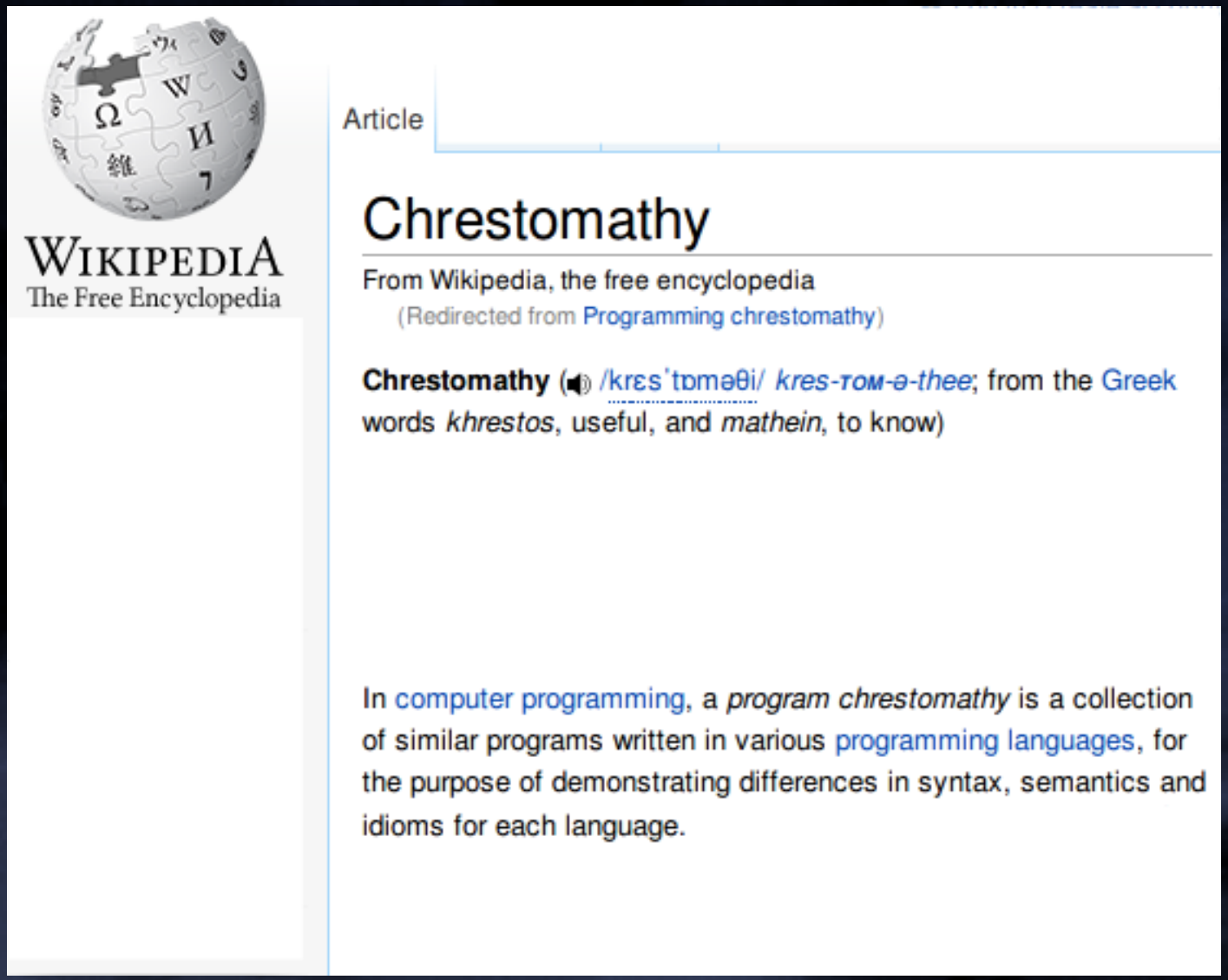
Java:

```
public class Pet {
        private String name;
        private Person owner;

        public Pet(final String name, final Person owner) {
                this.name = name;
                this.owner = owner;
        }

        public String getName() {
                return name;
        }
        public void setName(final String name) {
                this.name = name;
        }

        public Person getOwner() {
                return owner;
        }

        public void setOwner(final Person owner) {
                this.owner = owner;
        }
}
```

Scala:

```
class Pet(var name: String, var owner: Person)
```

comparative linguistics



WIKIPEDIA
The Free Encyclopedia

Article

Chrestomathy

From Wikipedia, the free encyclopedia
(Redirected from Programming chrestomathy)

Chrestomathy (/krɛsˈtɒməθi/ *kres-ᴛᴏᴍ-ə-thee*; from the Greek words *khrestos*, useful, and *mathein*, to know)

In computer programming, a *program chrestomathy* is a collection of similar programs written in various programming languages, for the purpose of demonstrating differences in syntax, semantics and idioms for each language.

Example

"Print the first N squares: 1, 4, 9, 16, 25, ..."

- **Easy to state & understand**
- **Has some iteration/loop in it**
- **Generic: gets a parameter, adjust result to it**
- **Does some IO**

Inspiration: a blog post by Steve Yegge

Java

This version is too long to fit on this page

So lets jump to a URL instead:

http://sites.google.com/site/steveyegge2/lisp-wins

C#

Thanks to Peter

```
int[] v = { 1, 2, 3, 4, 5 };
var squares = v.Select(x => x * x);
foreach (var x in squares)
        Console.Write(x.ToString() + " ");
```

Much more elegant, but can we do better?

Perl & python

Credit: little bro.

perl: `print join " ", map {$_ * $_} 1..5`

python: `print map(lambda n: n*n, range(1, 6))`

Semi pure/functional (like LISP), getting there...

Quiz: the above aren't equivalent. How so?

R

```
cat( (1:5) ^ 2 )
```

nirvana

Iteration + selection

One of the most common/universal programming constructs:

Select array subset based on some condition

C, C++, C#, Java, Fortran, … (all procedural languages) :

```
for each element in array[]
    If (condition on element is true)
        do something with element
```

SQL: `select (element) from table where (condition) ...`

R: iteration + selection done right

Select array subset based on some condition

array_name[logical_condition]

Example: Age[Age >= 7.5]

nirvana

R: array[other_array]

Make all "obvious" things implicit

If object is an array → iterate over it

[index] is subset selection

-- Ranges & subsets *Scores[west_coast_teams]*

-- Boolean conditions *Age[Age > 7.5]*

With no 'if's, 'for's, iterators, no fluff remains

Programs are typically ~10 times shorter and clearer

Back to our “toy” program

print natural squares up to N:

```
cat( (1:5) ^ 2 )
```

Way too trivial?

What if I want, say, a chart of the squares?

But what if I want a chart?

just replace 'cat' with 'plot':

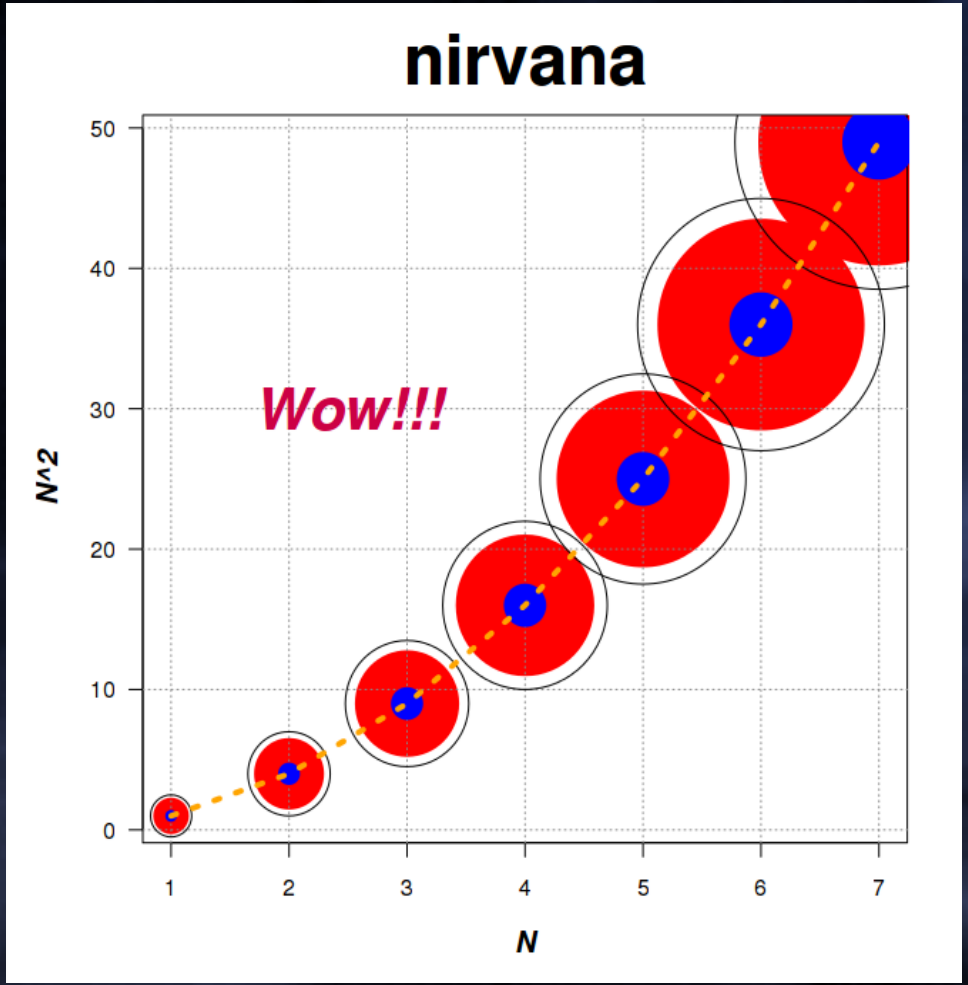
```
plot( (1:5) ^ 2 )
```

nirvana

“what if I want a …” demo

- 7 instead of 5
- data-points as cute circles
- radius growing as N
- area → as square(N)
- title and axis labels
- a grid
- fancy concentric circles
- some filled, some hollow
- a dashed line over centers
- a “Wow!!!”

All wishes come true
In just a few lines of R code
(See demo.R)



from language to platform

R is pure-functional, generic, and extensible

functions are generic/polymorphic and w/o side-effects on callers

It was a small language when it started, but it was cleanly extensible

Now it has over 3000 libraries and it keeps growing

Without breaking under the strain/complexity of additions

R is a factory

Give a man a fish – he will eat for a day

Teach a man how to fish – he can eat his whole life

Give a man tools – he can make a fishing pole...

(Guy L Steele Jr.)

I feel I found my "almost perfect" language for now

Questions?