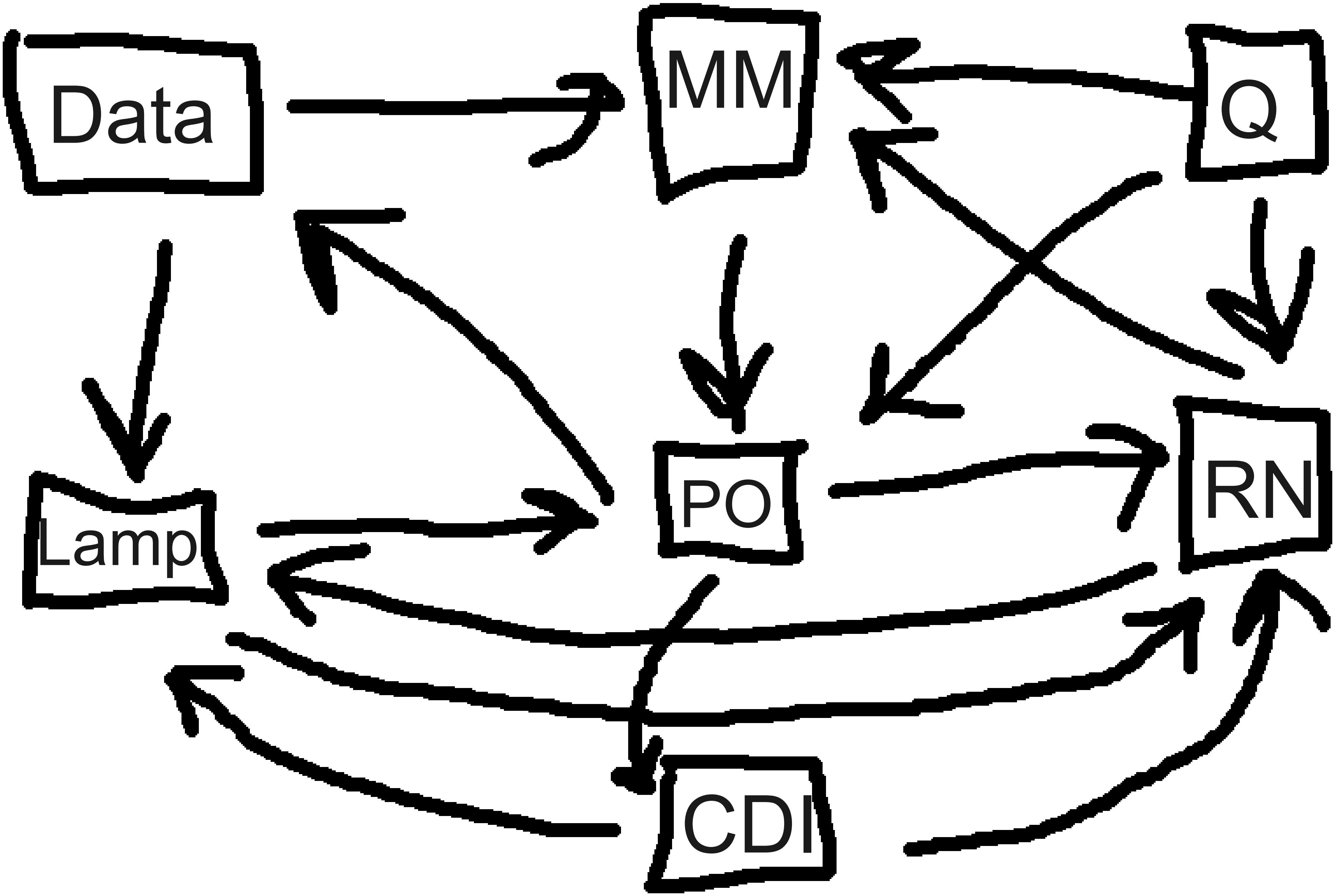
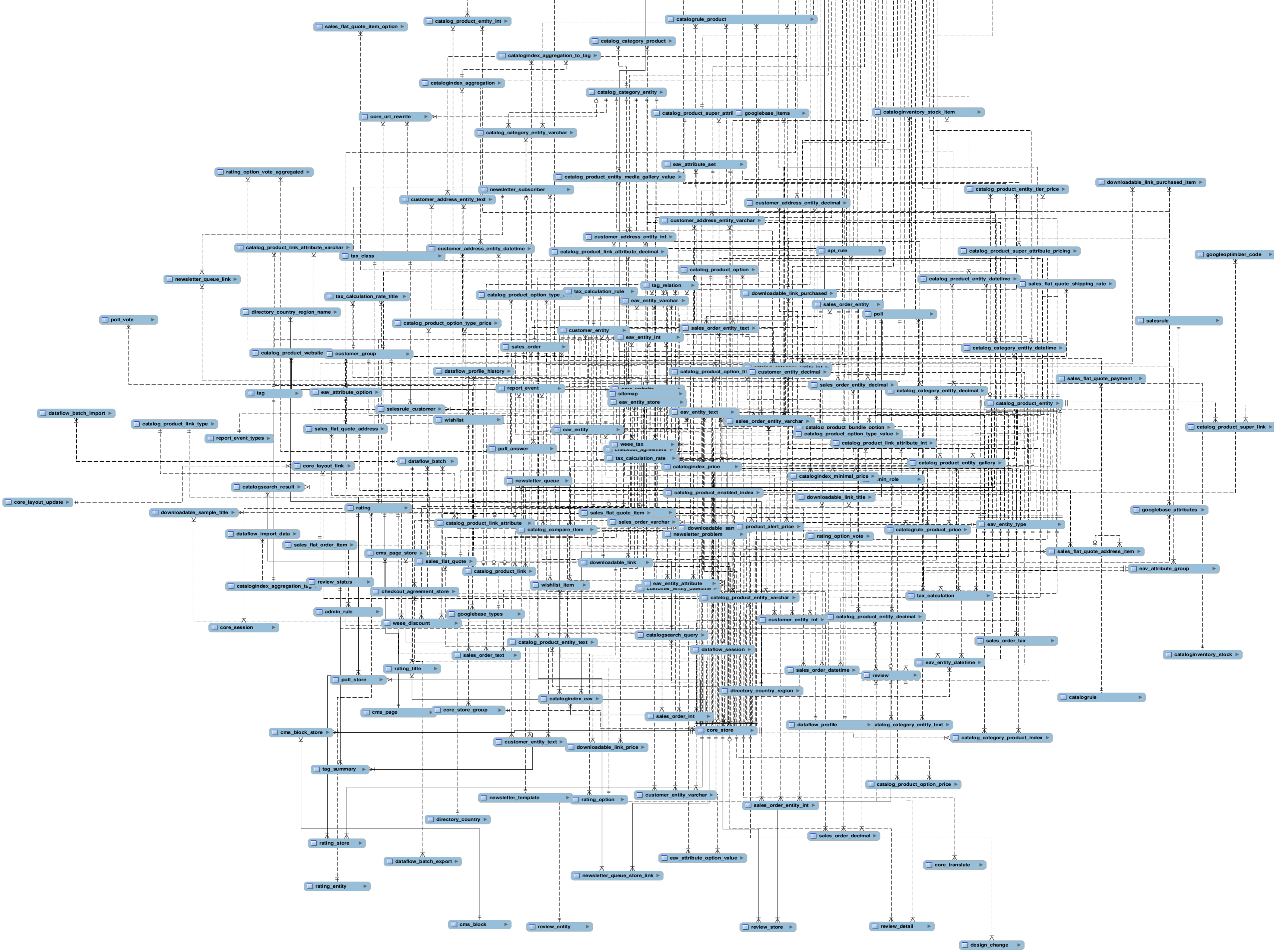
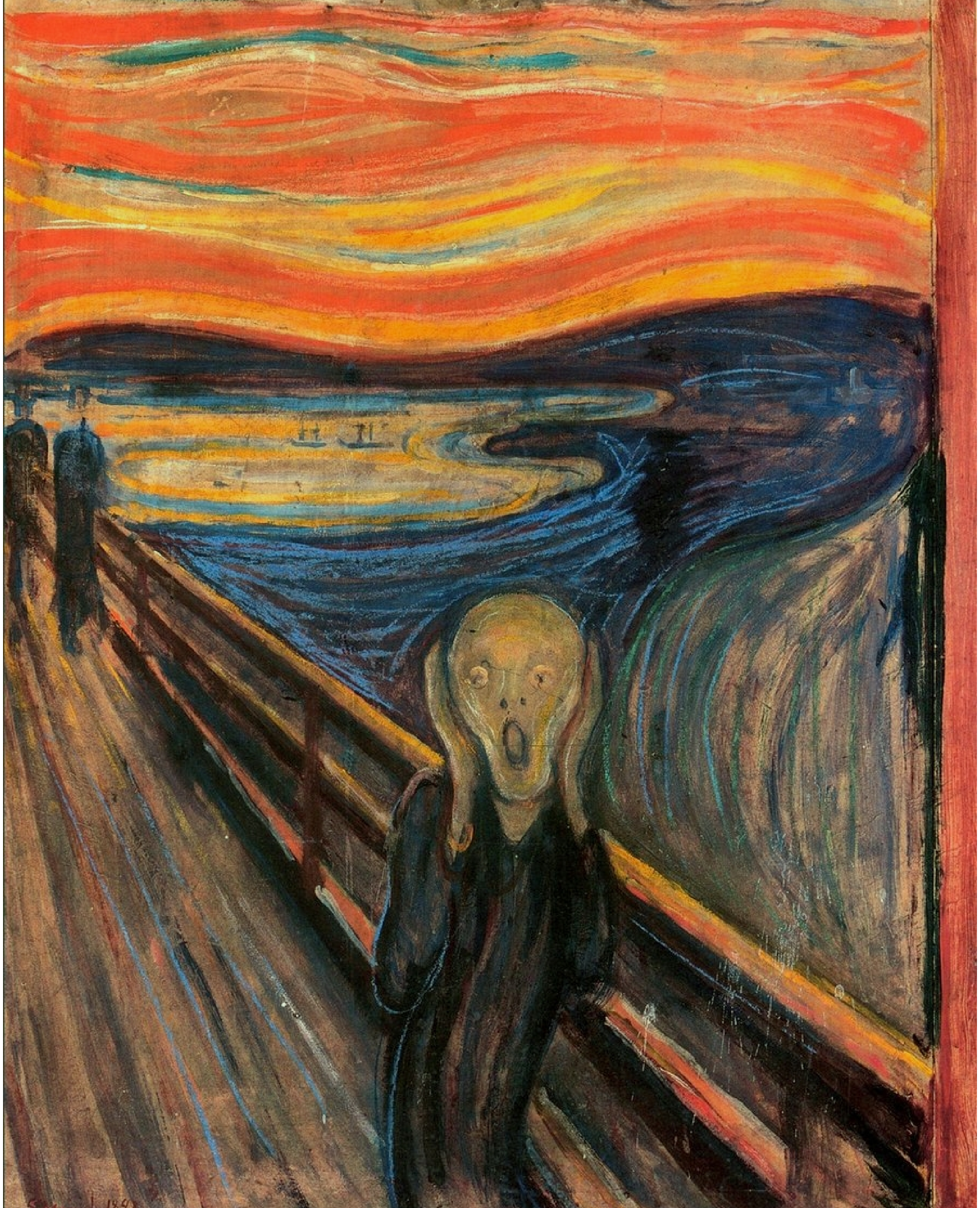
An Investigation of Lesser-Known Programming Languages

David Colgan
Dr. Nurkkala advising
Taylor University

Programming is HARD


Data
MM
Q
Lamp
PO
RN
CDI


sales_flat_quote_item_option
catalog_product_entity_int
catalogrule_product
catalog_category_product
catalogindex_aggregation_to_tag
catalogindex_aggregation
catalog_category_entity
catalog_product_super_attri
googlebase_items
cataloginventory_stock_item
core_url_rewrite
catalog_category_entity_varchar
eav_attribute_set
rating_option_vote_aggregated
catalog_product_entity_media_gallery_value
downloadable_link_purchased_item
newsletter_subscriber
catalog_product_entity_tier_price
customer_address_entity_text
customer_address_entity_decimal
customer_address_entity_varchar
customer_address_entity_int
catalog_product_link_attribute_varchar
customer_address_entity_datetime
catalog_product_link_attribute_decimal
api_rule
catalog_product_super_attribute_pricing
tax_class
googleoptimizer_code
catalog_product_option
newsletter_queue_link
catalog_product_entity_datetime
tag_relation
sales_flat_quote_shipping_rate
tax_calculation_rule
downloadable_link_purchased
tax_calculation_rate_title
catalog_product_option_type_
eav_entity_varchar
sales_order_entity
directory_country_region_name
poll
poll_vote
salesrule
catalog_product_option_type_price
customer_entity
sales_order_entity_text
eav_entity_int
sales_order
catalog_category_entity_datetime
catalog_product_website
customer_group
dataflow_profile_history
catalog_product_option_ti
customer_entity_decimal
sales_flat_quote_payment
sales_order_entity_decimal
report_event
catalog_category_entity_decimal
tag
eav_attribute_option
sitemap
eav_entity_store
catalog_product_entity
salesrule_customer
eav_entity_text
dataflow_batch_import
wishlist
sales_order_entity_varchar
catalog_product_link_type
catalog_product_bundle_option
catalog_product_super_link
sales_flat_quote_address
eav_entity
catalog_product_option_type_value
report_event_types
weee_tax
checkout_agreement
catalog_product_link_attribute_int
poll_answer
dataflow_batch
tax_calculation_rate
catalog_product_entity_gallery
core_layout_link
catalogindex_price
catalogindex_minimal_price
min_role
newsletter_queue
catalogsearch_result
catalog_product_enabled_index
downloadable_link_title
core_layout_update
rating
downloadable_sample_title
sales_flat_quote_item
googlebase_attributes
sales_order_varchar
catalog_product_link_attribute
eav_entity_type
downloadable_sam
product_alert_price
catalog_compare_item
catalogrule_product_price
newsletter_problem
dataflow_import_data
rating_option_vote
sales_flat_order_item
cms_page_store
sales_flat_quote_address_item
sales_flat_quote
downloadable_link
catalog_product_link
eav_attribute_group
review_status
catalogindex_aggregation_tag
wishlist_item
eav_entity_attribute
checkout_agreement_store
catalog_product_entity_varchar
tax_calculation
admin_rule
googlebase_types
weee_discount
core_session
customer_entity_int
catalog_product_entity_decimal
catalogsearch_query
sales_order_tax
catalog_product_entity_text
dataflow_session
cataloginventory_stock
sales_order_text
eav_entity_datetime
rating_title
sales_order_datetime
review
poll_store
directory_country_region
catalogrule
catalogindex_eav
core_store_group
cms_page
sales_order_int
dataflow_profile
atalog_category_entity_text
cms_block_store
core_store
catalog_category_product_index
customer_entity_text
downloadable_link_price
tag_summary
catalog_product_option_price
newsletter_template
rating_option
customer_entity_varchar
sales_order_entity_int
directory_country
sales_order_decimal
rating_store
eav_attribute_option_value
dataflow_batch_export
core_translate
newsletter_queue_store_link
rating_entity
cms_block
review_entity
review_store
review_detail
design_change



When Building A Huge Thing,

Why Not Use Power Tools?

Why Not Use Power Tools?

VS

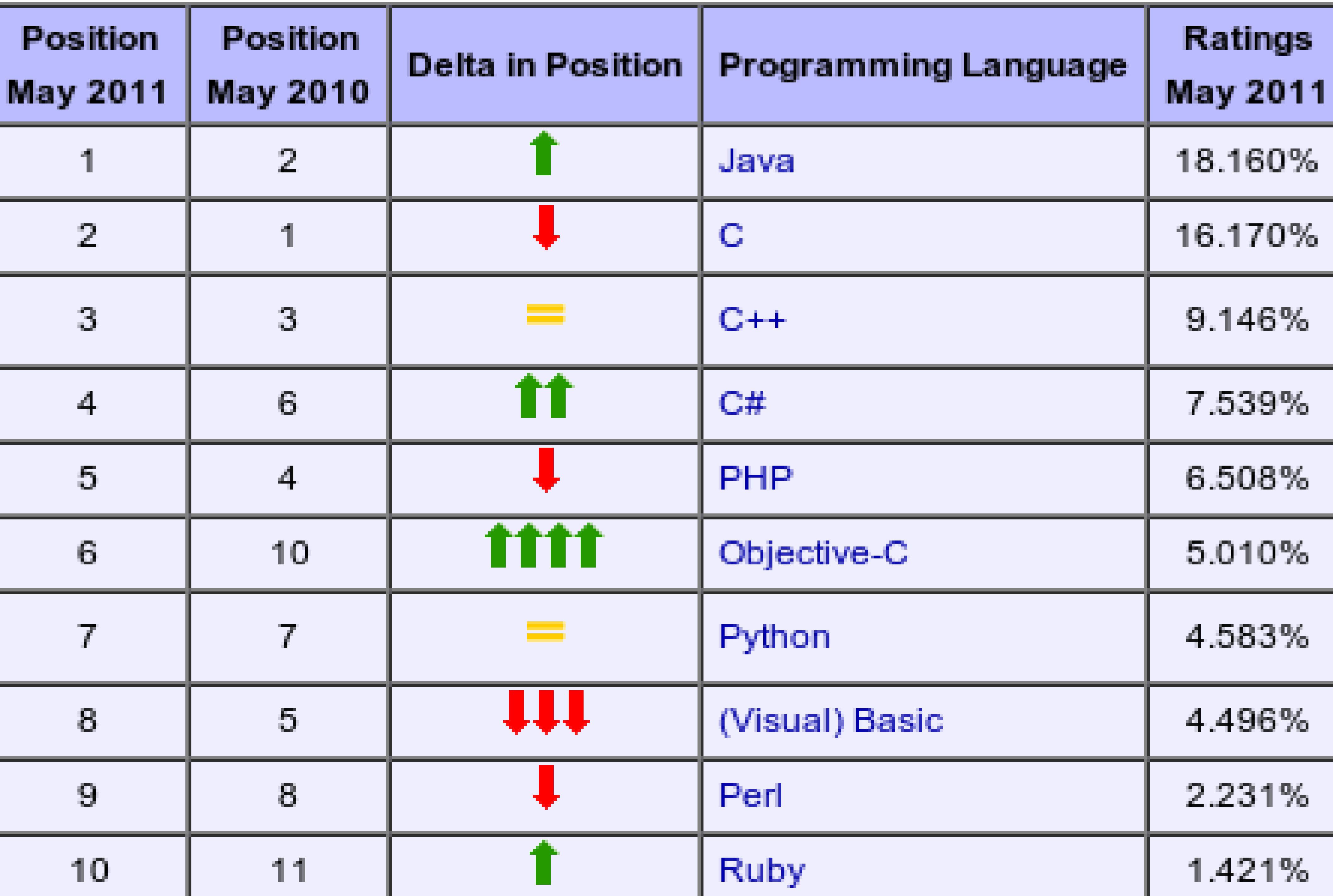
Why Not Use Power Tools?

VS

Why Not Use Power Tools?

VS

TIOBE Index



Position May 2011	Position May 2010	Delta in Position	Programming Language	Ratings May 2011
1	2	↑	Java	18.160%
2	1	↓	C	16.170%
3	3	=	C++	9.146%
4	6	↑↑	C#	7.539%
5	4	↓	PHP	6.508%
6	10	↑↑↑↑	Objective-C	5.010%
7	7	=	Python	4.583%
8	5	↓↓↓	(Visual) Basic	4.496%
9	8	↓	Perl	2.231%
10	11	↑	Ruby	1.421%

Assembly, C, C++, Java, PHP, Visual Basic, Python, Perl, Ruby, Lua, Common Lisp, Clojure, Scheme, Haskell, Erlang, Objective Caml, Factor, Actionscript, Javascript, Bash, Prolog, Smalltalk, J

What makes a language powerful?

Minimized programmer exertion
Expressiveness
Readability
Performance

Meet the Languages:

C

Python

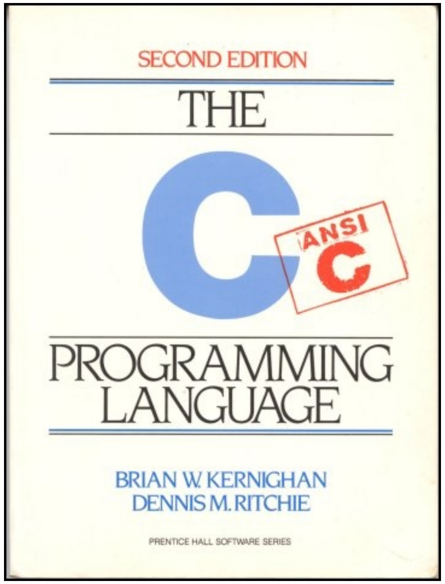
Clojure

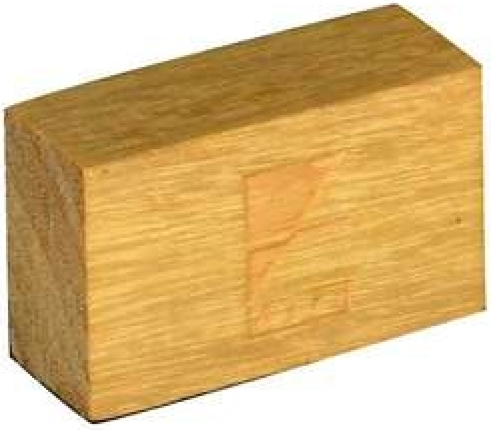
Haskell

Factor

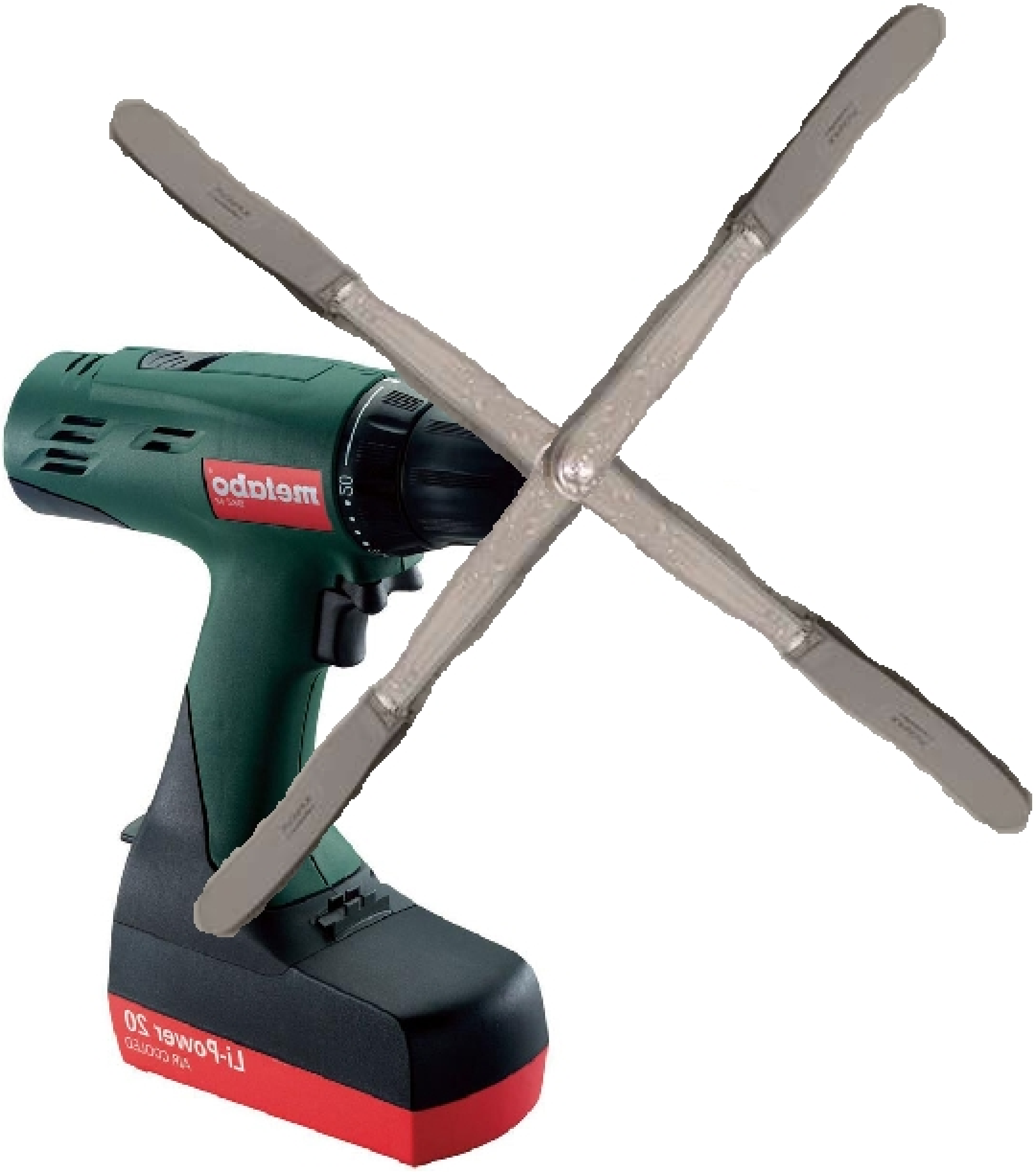
C – The Venerable Classic

```
int remove_die(int* dice, int die)
{
    int i;
    for (i=0; i<6; i++){
        if(dice[i] == die){
            for (; i<5; i++){
                dice[i] = dice[i+1];
            }
            dice[5] = E;
            return 1;
        }
    }
    return 0;
}
```

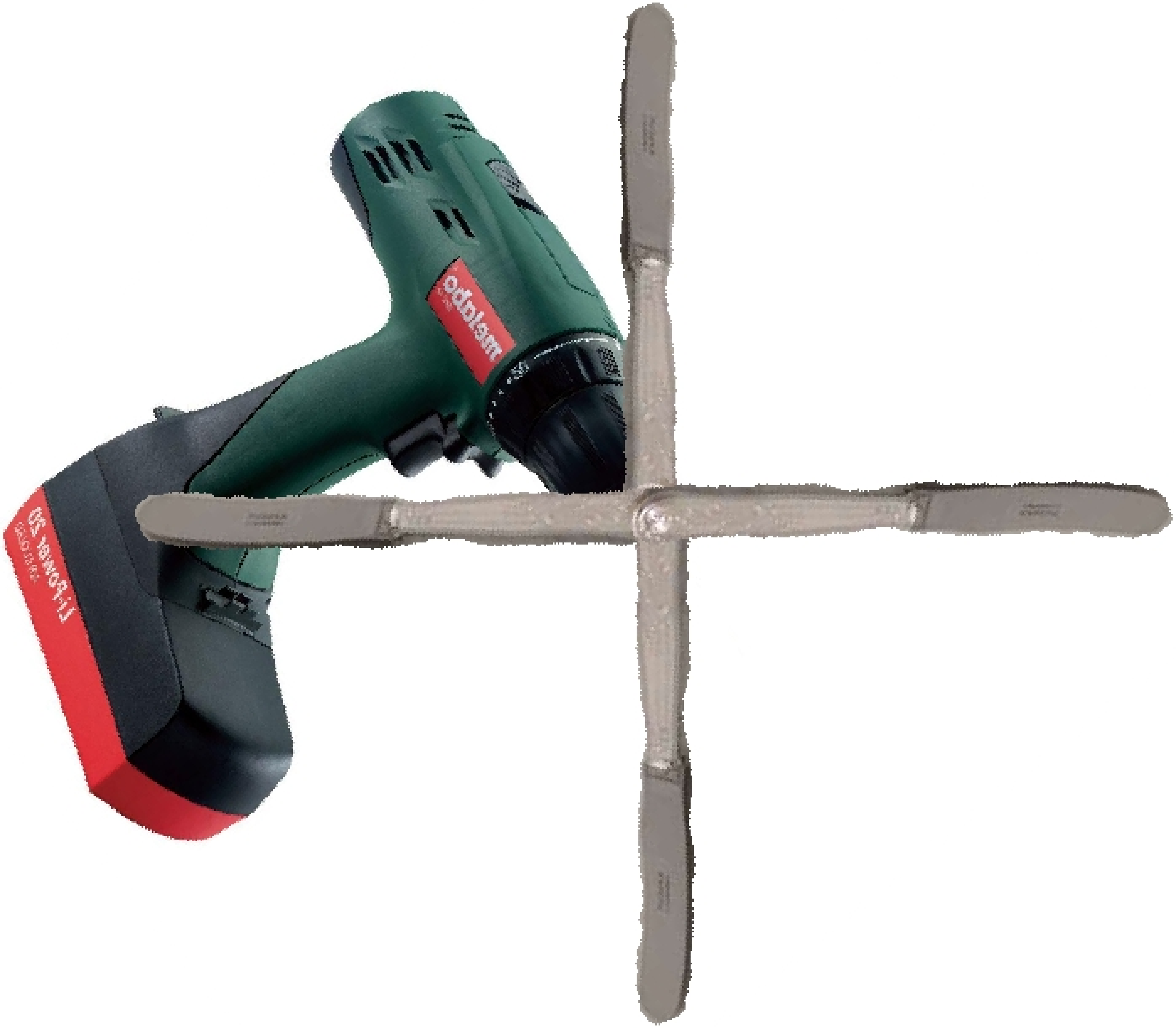




C is like



C is



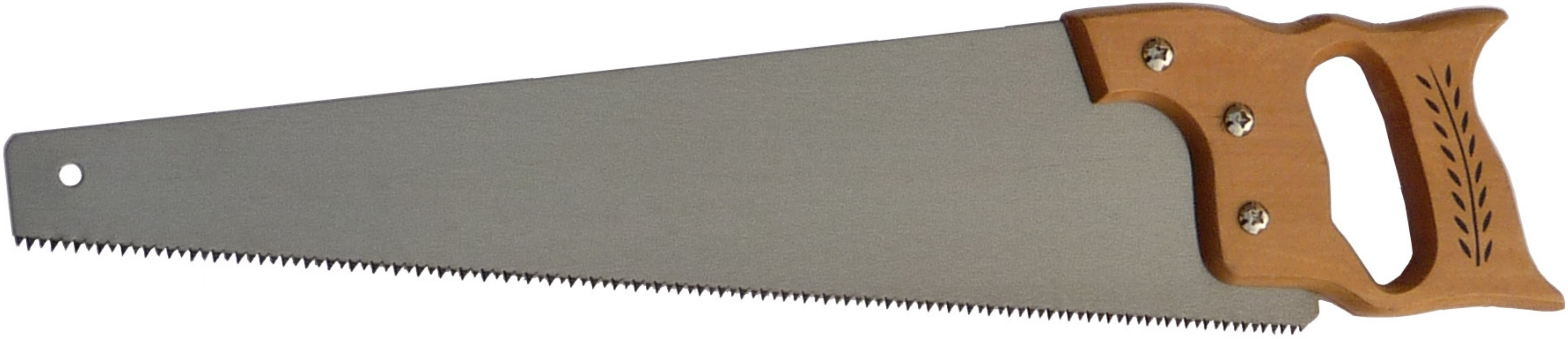
Python – A Fine Choice

"Python is a programming language that lets you **work more quickly** and integrate your systems more effectively.

You can learn to use Python and **see almost immediate gains** in **productivity** and **lower maintenance costs**."

-Python.org

Python is like



Clojure – The Next Big Thing?

Higher order functions
Immutable variables
Laziness
Derived from Lisp
Strong support for concurrency

Also claims to make programs shorter and have fewer bugs.

Clojure is like

Haskell – The Zen of Programming?

"Haskell is an advanced **purely-functional** programming language. An open-source product of more than twenty years of cutting-edge research, it allows **rapid development of robust, concise, correct software.**

Haskell makes it easier to produce **flexible, maintainable, high-quality software**."

-Haskell.org

Haskell is like

Factor – And now for something completely different

A concatenative language

Metaprogramming

Macros

Potential for very short programs

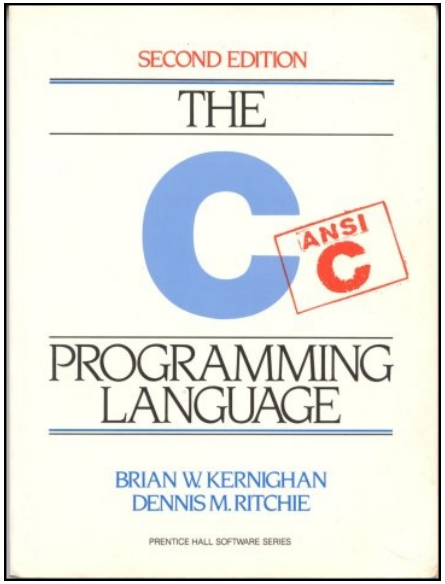
Reduces redundancy other languages can't

Interactive Environment

FACTOR

A PRACTICAL STACK LANGUAGE

Factor is like



SECOND EDITION

THE C PROGRAMMING LANGUAGE

ANSI C

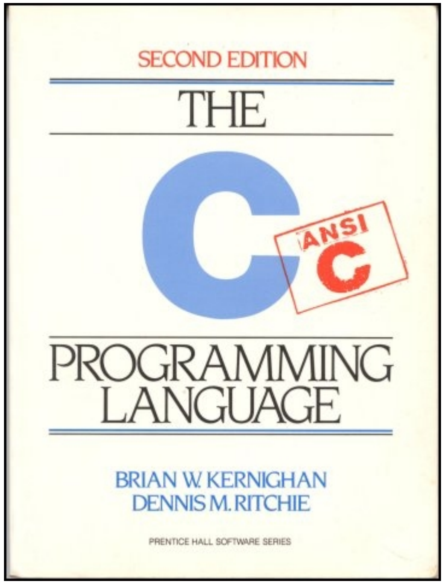
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

PRENTICE HALL SOFTWARE SERIES

A Whirlwind Tour

Imperative Average in C

```
int find_average(int* arr, int len){
    int i;
    int sum = 0;
    for(i=0; i < len; i++){
        sum += dice[i];
    }
    return sum / len;
}
```



Higher Level Average in Python

```
def find_average(lst):
    sum = 0
    for elem in lst:
        sum += elem
    return sum / len(lst)
```

Functional Average in Clojure

```clojure
(defn find-average [lst]
 (/ (reduce + 0 lst)
    (count lst)))
```

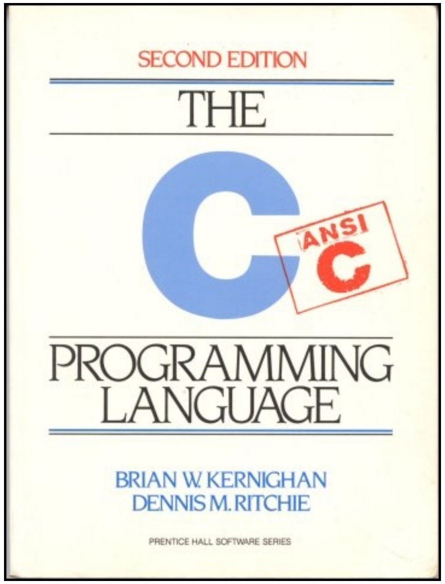
Purely Functional Average in Haskell

```haskell
findAverage :: [Int] -> Int
findAverage lst =
    (foldl (+) 0 lst) / (length lst)
```

Concatenative Average in Factor

```
find-average ( seq -- x )
    [ 0 [ + ] reduce ] [ length / ] bi ;
```

FACTOR
A PRACTICAL STACK LANGUAGE


SECOND EDITION

THE C PROGRAMMING LANGUAGE

ANSI C

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

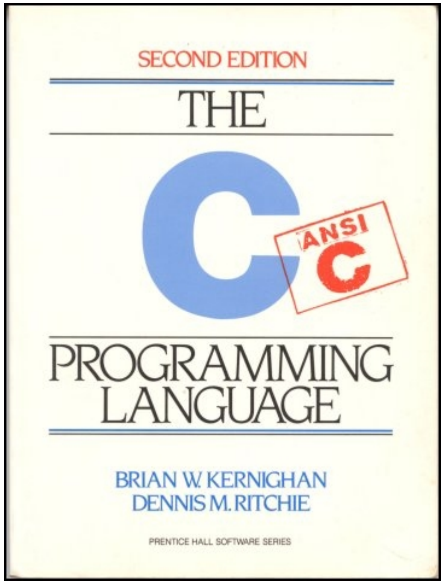
PRENTICE HALL SOFTWARE SERIES

Comparisons with Farkle

We need a program

Fun with C:
Can you spot the bug?

```
/* dice == {1, 2, 4, -1, -1, -1} */

for (i=0; i<6; i++){
    die_counts[dice[i]]++;
}
```



SECOND EDITION

THE

C

ANSI C

PROGRAMMING LANGUAGE

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

PRENTICE HALL SOFTWARE SERIES

C is like

Conciseness Equals
Fewer Points of Failure

```c
int find_average(int* arr, int len){
    int i;
    int sum = 0;
    for(i=0; i < len; i++){
        sum += dice[i];
    }
    return sum / count;
}
```

```python
def find_average(lst):
    sum = 0
    for elem in lst:
        sum += elem
    return sum / len(lst)
```

For Loops on One Line!

```
dice.values =
    [random.randint(1, 6) for die in range(count)]
```

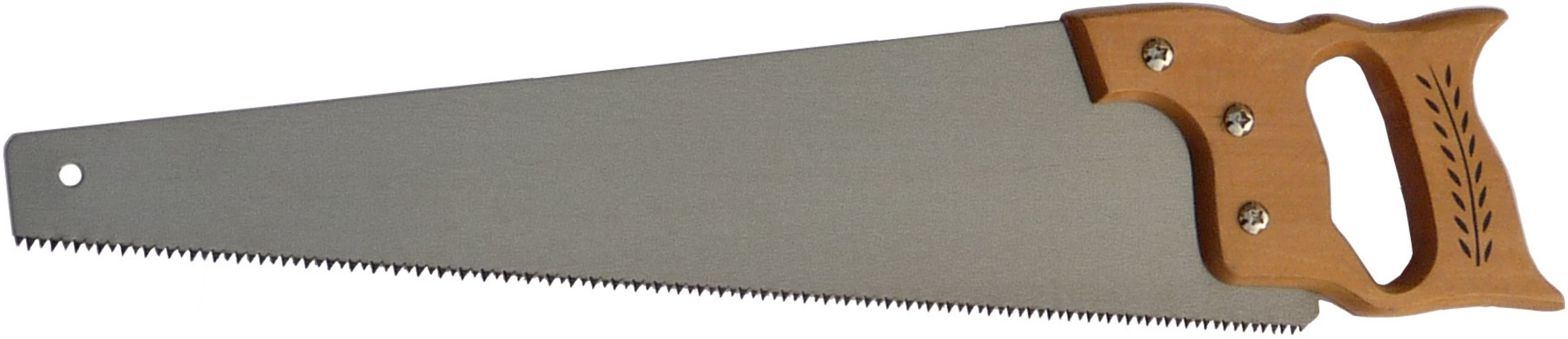
Memory Management

```
int* dice = (int*) malloc(sizeof(int) * 6);
...
free(dice);
```

VS

```
dice = []
```

Python is like



Now For the Crazy Stuff

Clojure and Haskell's Crazy Idea

No Mutable Variables

Clojure and Haskell's Crazy Idea

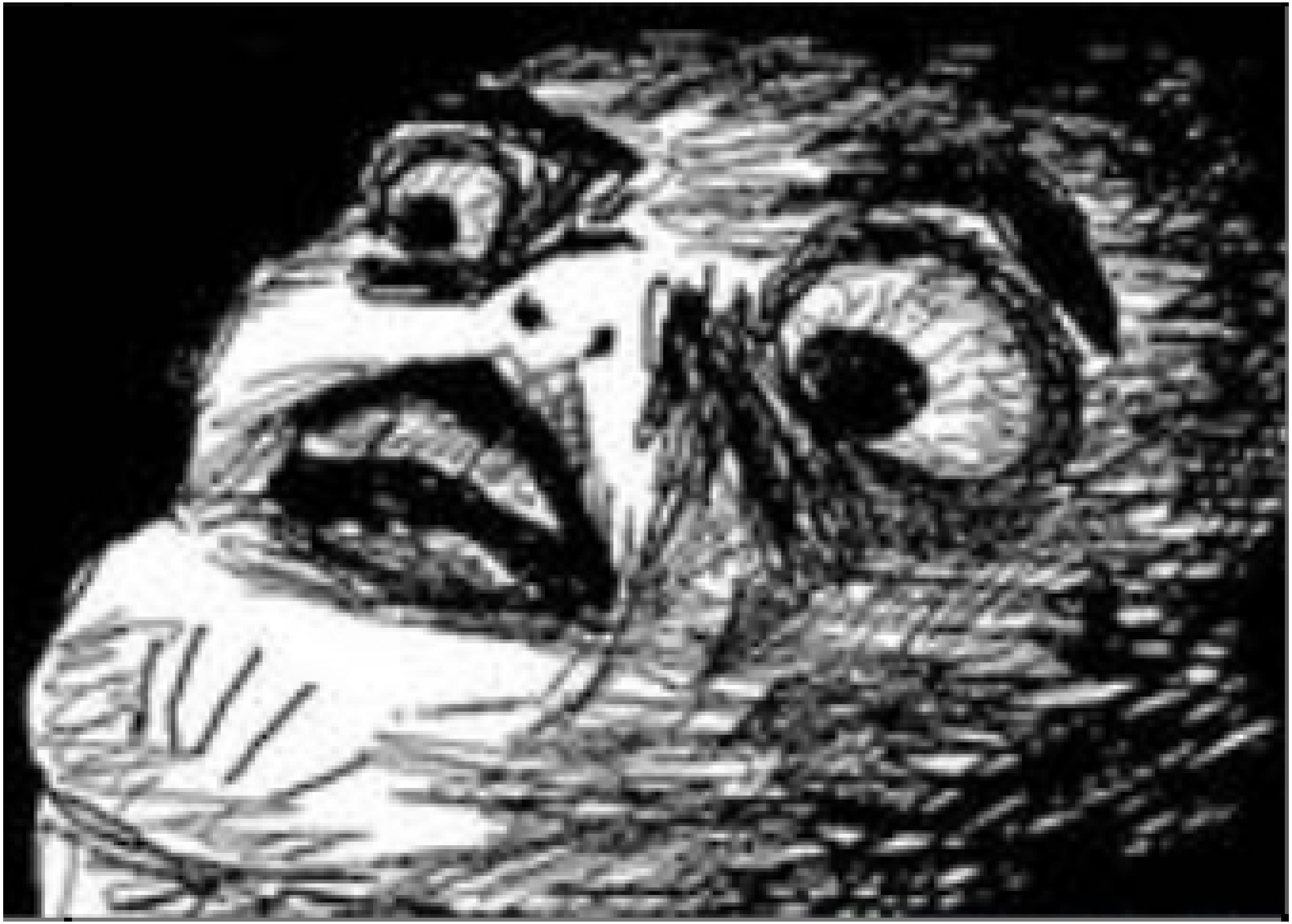
No Mutable Variables?

Clojure and Haskell's Crazy Idea

No?

```
x+=4
```

```
i++
```



How will we ever do work?

No for loops!

Recursion and Higher Order Functions

```
(defn reduce [fn acc lst]
  (if (empty? lst)
    acc
    (reduce fn
            (fn acc (head lst))
            (tail lst))))
```

Higher Order Functions?

```
(map fn list)

(reduce fn init list)

(filter pred list)

(sum lst)

(every? pred list)

(all? pred list)
```

Higher Order Example

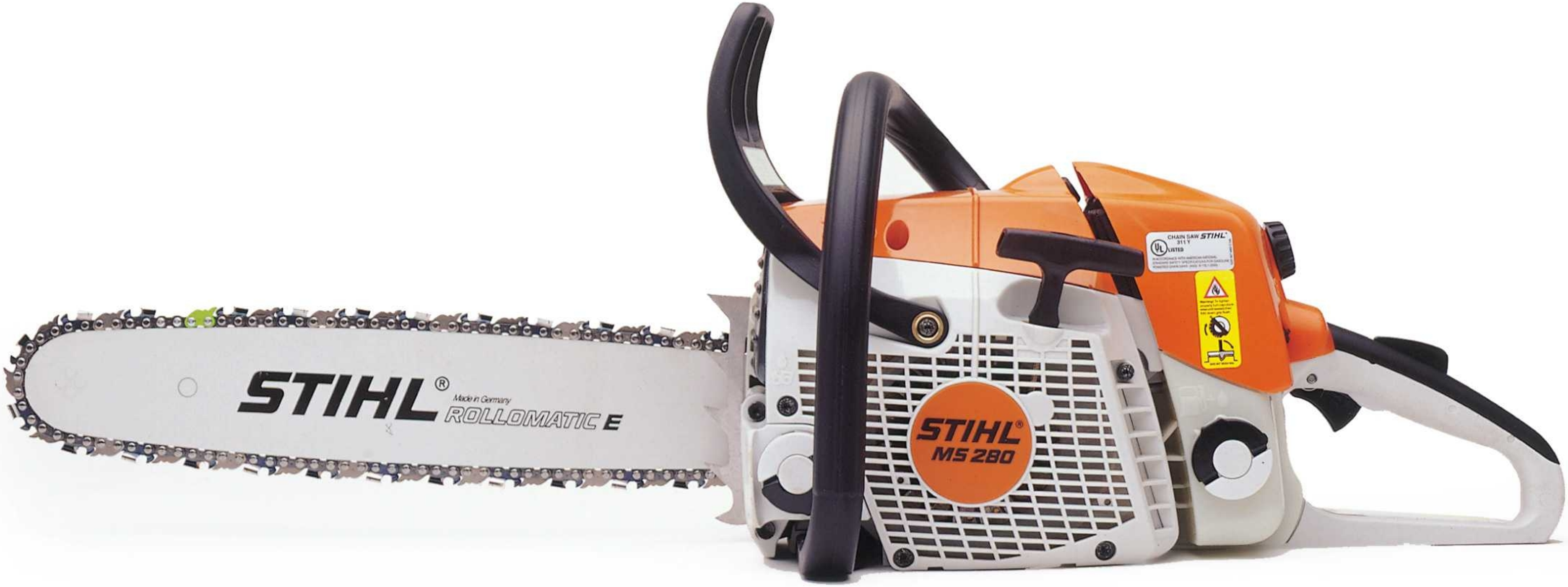
```
(defn roll-has-nonscoring-dice [dice]
  (let [die-map (frequencies dice)]
    (not
      (every? true? (map #(or (or (= (val %) 0)
                                  (<= 3 (val %) 6))
                              (= (key %) 1)
                              (= (key %) 5))
                         die-map)))))
```

Clojure

Why Immutability?

Fewer bugs!
Shorter code!
Reduce duplication!
Referential transparency!
Faster performance!
(Almost) free parallelization!

Clojure is like



Fun with Haskell:
Strict Type Safety

Clojure was mostly functional
Haskell is **purely** functional

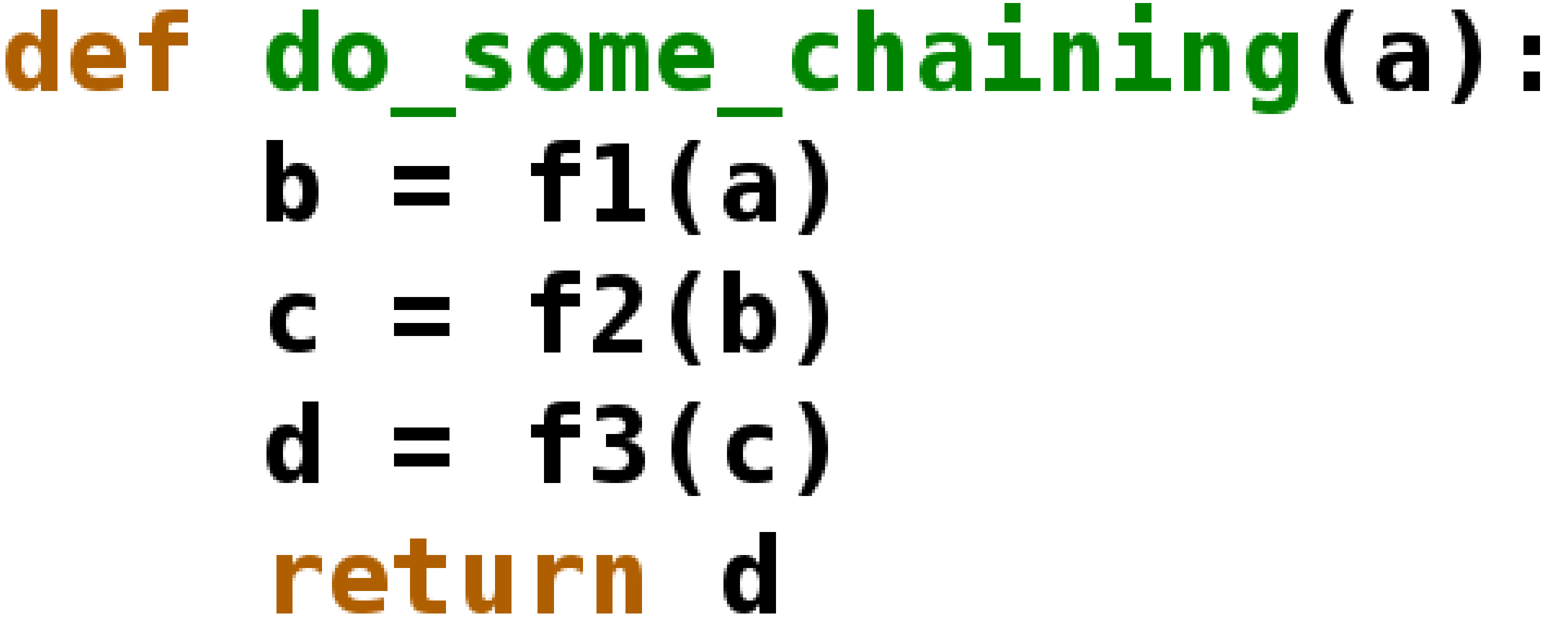
Type System

```
length :: [a] -> Int

sum :: (Num a) => [a] -> a

foldl :: (a -> b -> a) -> a -> [b] -> a

all :: (a -> Bool) -> [a] -> Bool
```

What is a Monad?

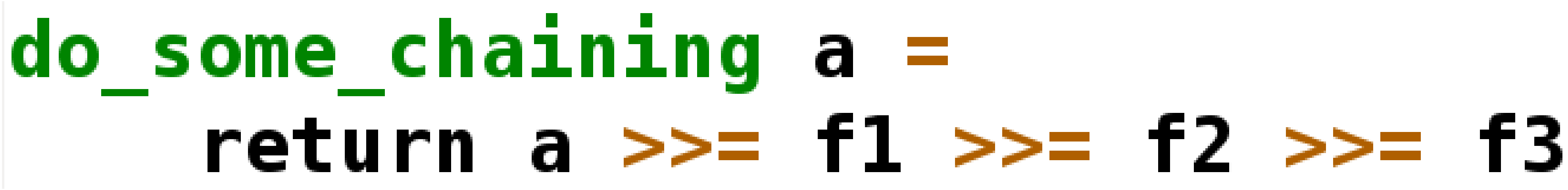
A strategy for combining computations into more complex computations.

Monads!



```python
def do_some_chaining(a):
    b = f1(a)
    c = f2(b)
    d = f3(c)
    return d
```

Monads!

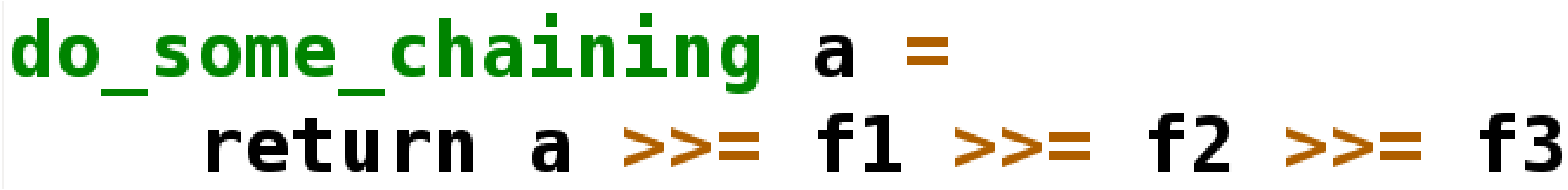


```
do_some_chaining a =
    return a >>= f1 >>= f2 >>= f3
```

Monads 2

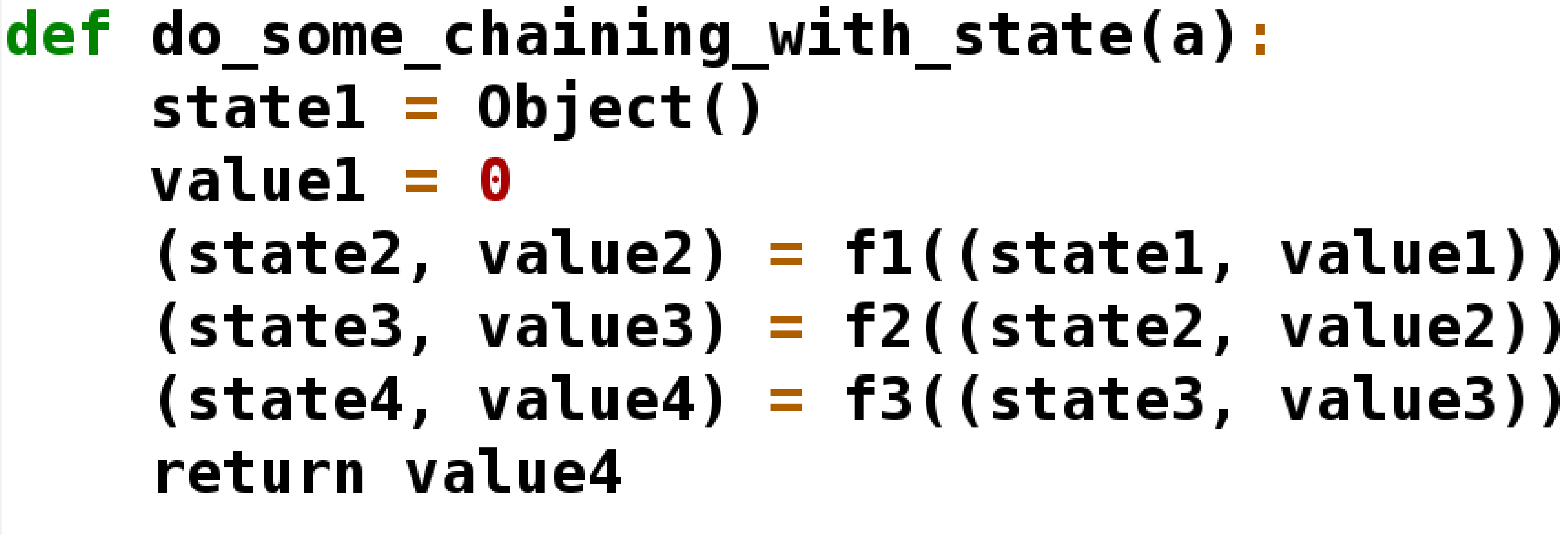
```
def do_some_chaining_with_failing(a):
    b = f1(a)
    if b == None:
        return None
    else:
        c = f2(b)
        if c == None:
            return None
        else:
            d = f3(c)
            return d
```

Monads 2



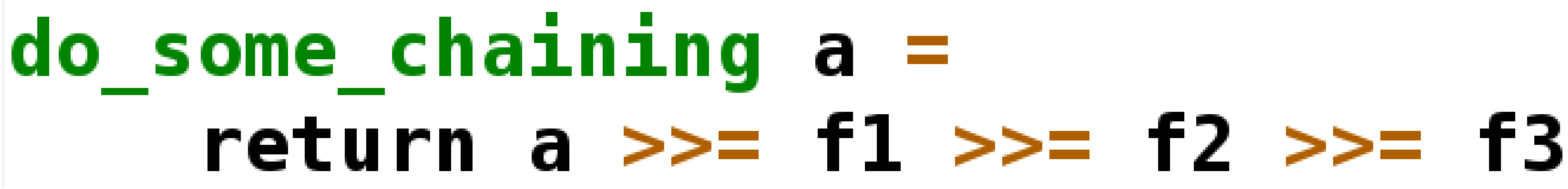
```
do_some_chaining a =
    return a >>= f1 >>= f2 >>= f3
```

Monads 3



```
def do_some_chaining_with_state(a):
    state1 = Object()
    value1 = 0
    (state2, value2) = f1((state1, value1))
    (state3, value3) = f2((state2, value2))
    (state4, value4) = f3((state3, value3))
    return value4
```

Monads 3



```
do_some_chaining a =
    return a >>= f1 >>= f2 >>= f3
```

IO Monad

```haskell
queryStop :: Int -> IO Bool
queryStop turnScore = do
    putStrLn $ "Score " ++ (show turnScore)
    putStrLn $ "Hit enter to continue rolling,
                or type 'stop' to end your turn.\n"
    choice <- getLine
    if choice == "stop" then do
        return True
    else do
        return False
```

Haskell is like

Fun with Factor:
Eliminate ALL Redundancy?

```
x = 4;
fn1(x);
fn2(x);
```

VS

```
4 [ fn1 ] [ fn2 ] bi
```

FACTOR

A PRACTICAL STACK LANGUAGE

Very Short Words

```
: roll-dice ( x -- seq )
    [ 6 random 1 + ] replicate ;

: count ( elt seq -- cnt )
    swap '[ _ = ] filter length ;
```

FACTOR
A PRACTICAL STACK LANGUAGE

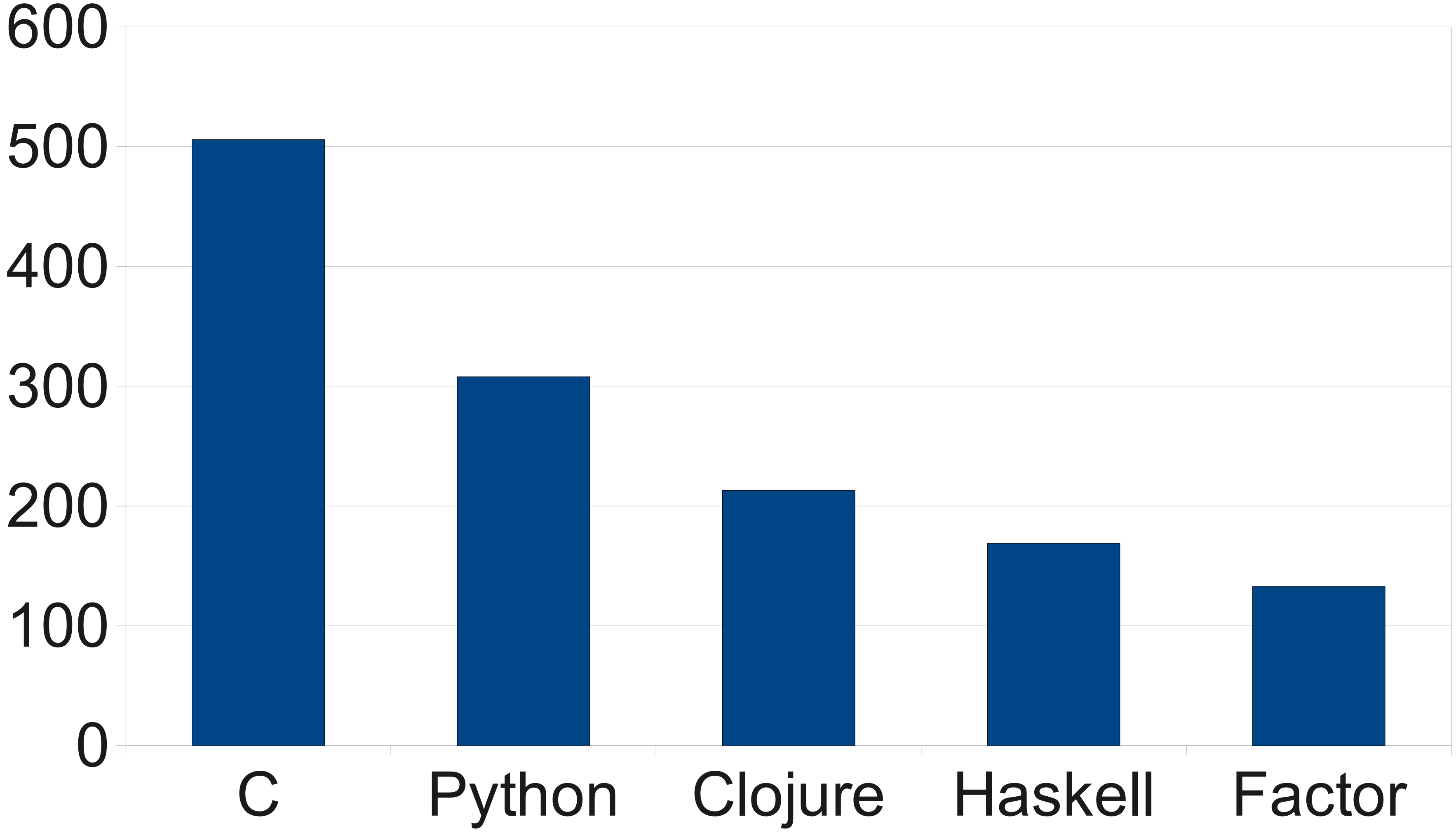
Factor is like

Program Analyses For:

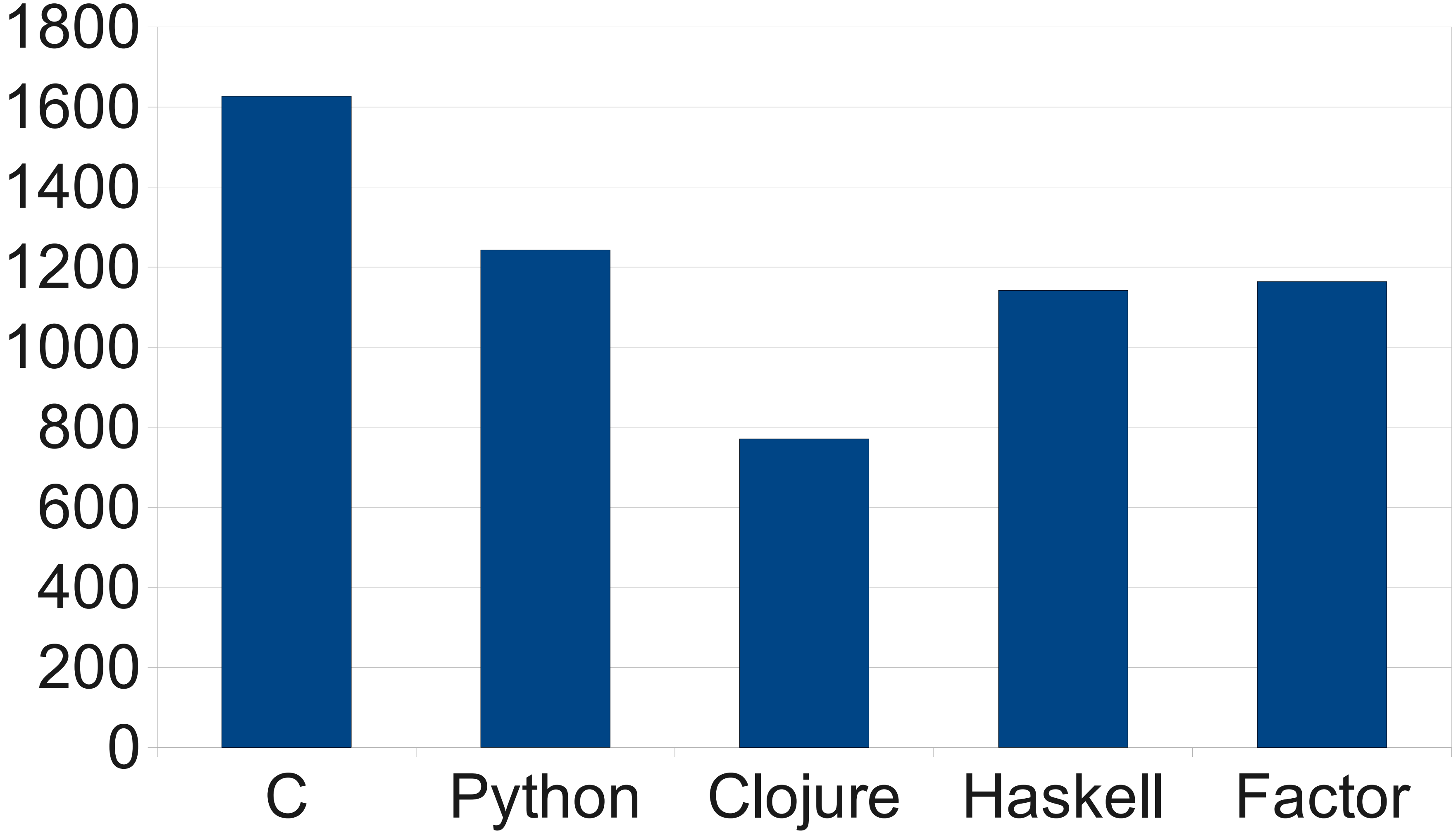
- Total Lines of Code
- Total Tokens
- Average Tokens Per Line
- Execution Time

python™
SECOND EDITION
THE C PROGRAMMING LANGUAGE
ANSI C
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE
PRENTICE HALL SOFTWARE SERIES
FACTOR
A PRACTICAL STACK LANGUAGE
Clojure

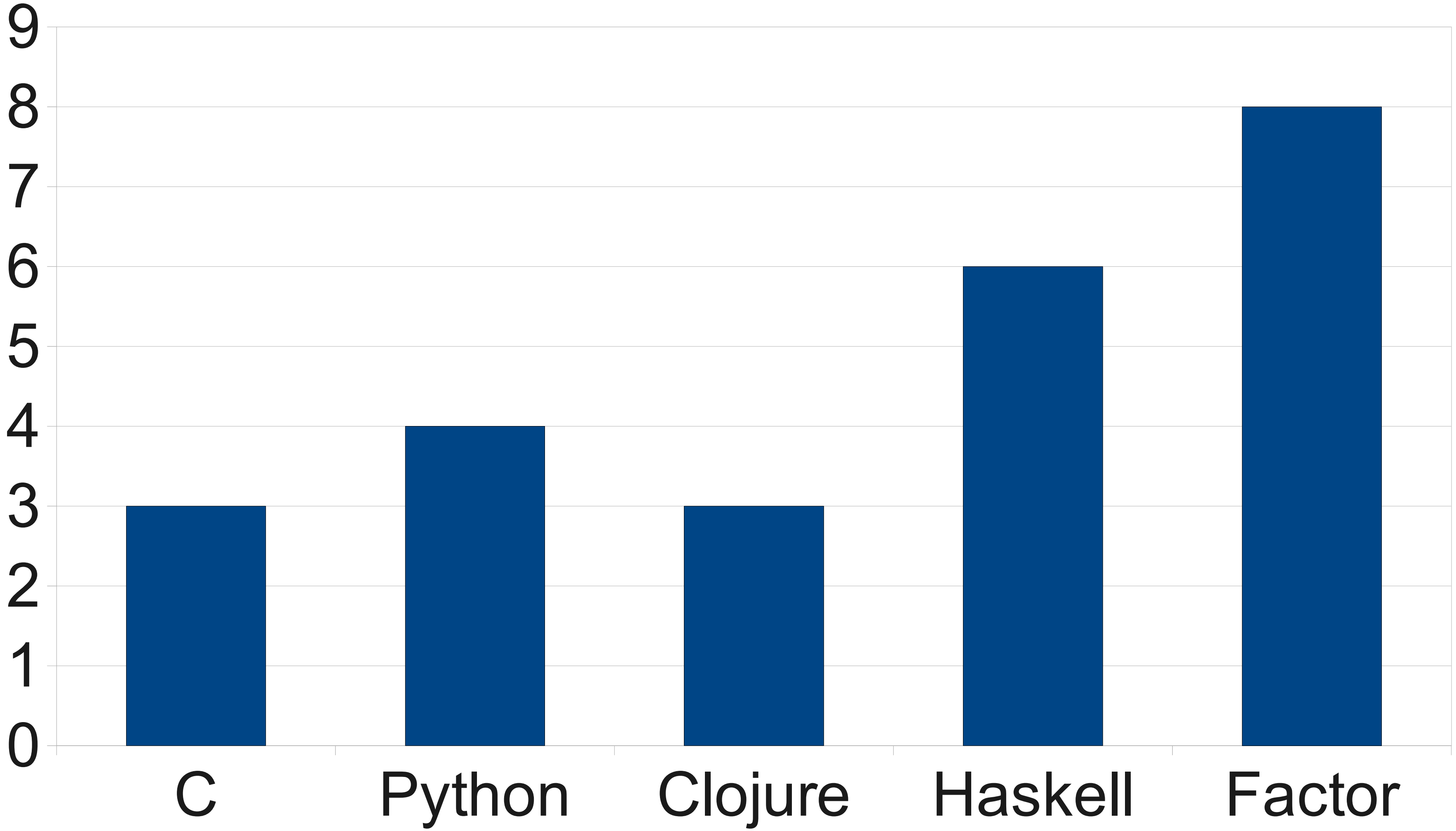
Total Lines of Code



Total Tokens



Average Tokens Per Line



Clojure is like

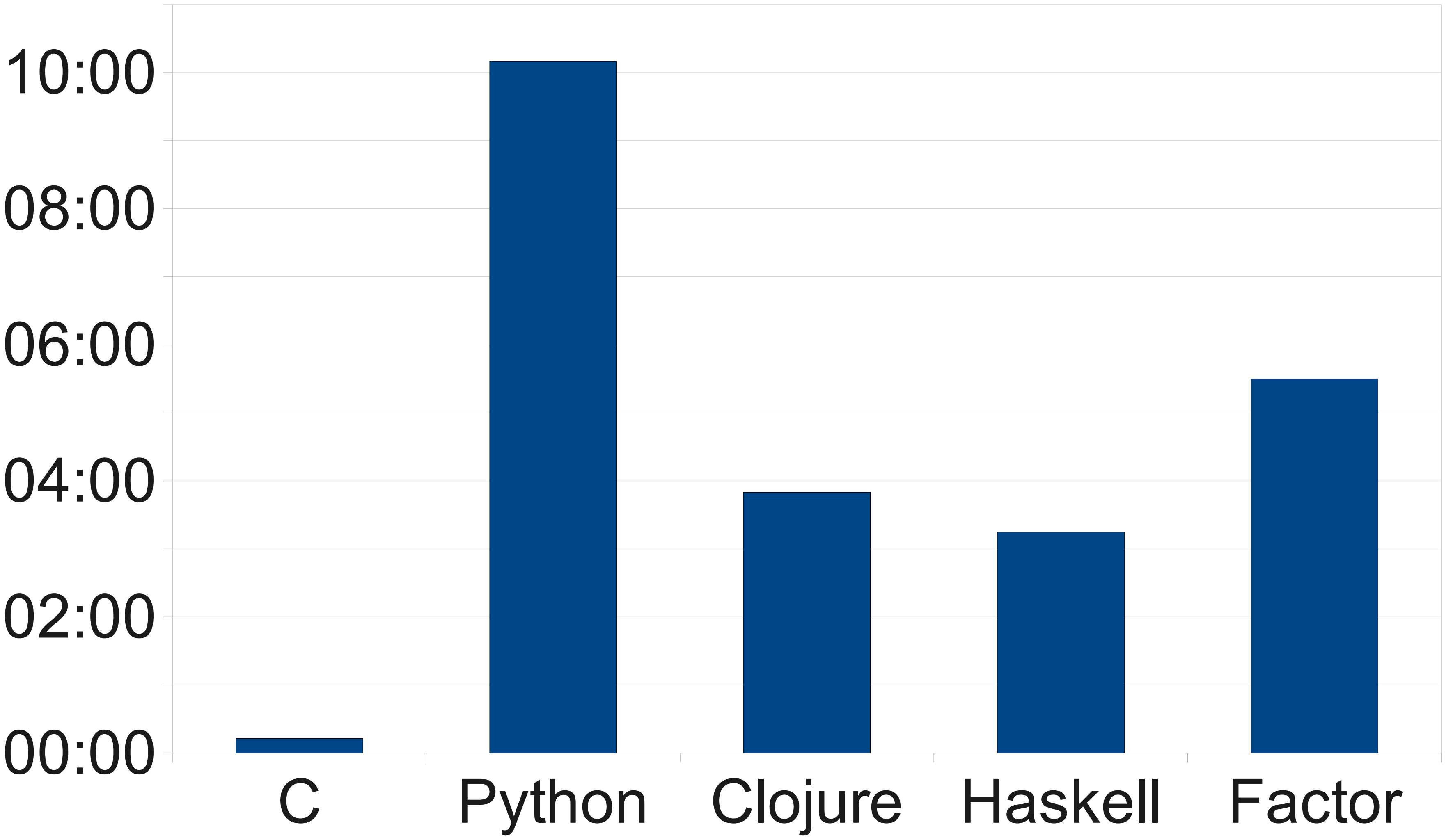
A Conclusion:

Please don't use C

C is like

Unless you have to have high performance.

Speed Tests


10:00
08:00
06:00
04:00
02:00
00:00
C
Python
Clojure
Haskell
Factor

Fast enough is usually fast enough.

Conclusion

A Conclusion:

Learning new paradigms is hard

Programmer Reaction

C **Python** **Clojure** **Haskell** **Factor**

Programmer Reaction

C **Python** **Clojure** **Haskell** **Factor**

Programmer Reaction

C **Python** **Clojure** **Haskell** **Factor**

Programmer Reaction

C

Python

Clojure

Haskell

Factor

Programmer Reaction

C

Python

Clojure

Haskell

Factor

Programmer Reaction

C

Python

Clojure

Haskell

Factor

A Conclusion:

Further investigation is needed

Future Work

Measure speed of programming

Write more idiomatic code

Larger sample size

More languages!

So the next time you start a software project:

Use the best tool for the job!

So the next time you start a software project:

Use the most powerful language in the paradigms you know.

Unless you have lots of time

Then learn

Clojure

Clojure is like

Unless you have even more time

Then learn more about Haskell and Factor and let me know what you find.

I NEED TO KNOW
EVERYTHING

Thanks to:

Cool people for creating these languages

Dr. Nurkkala for being an excellent advisor and proofreader of my papers

My parents for birthing me

YOU, for coming