Programming Languages
A Journey into Abstraction and Composition

Introduction to Programming Languages

Prof. Dr. Guido Salvaneschi

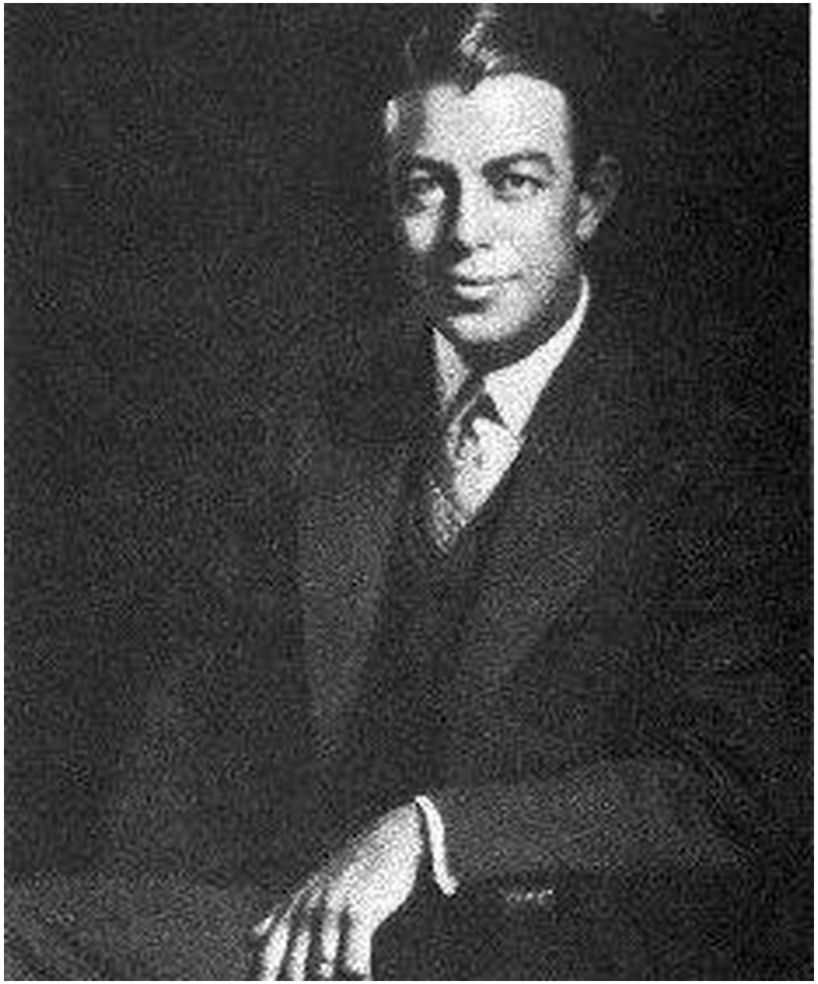
School of Computer Science

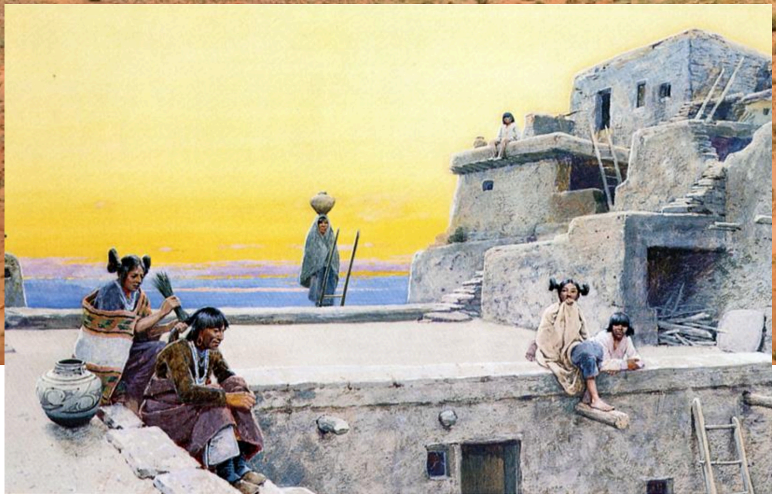
University of St.Gallen

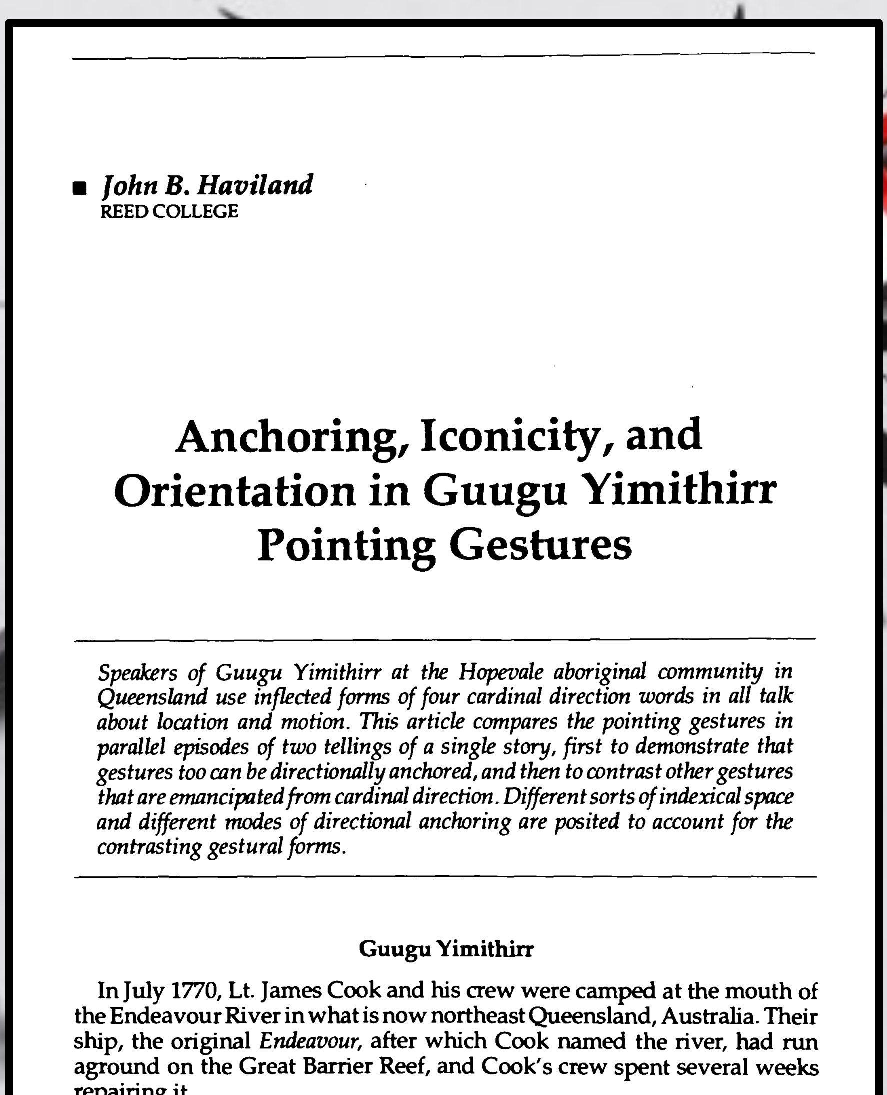
LANGUAGE AND MIND

Language shapes the way we think, and determines what we can think about.

Benjamin Lee Whorf







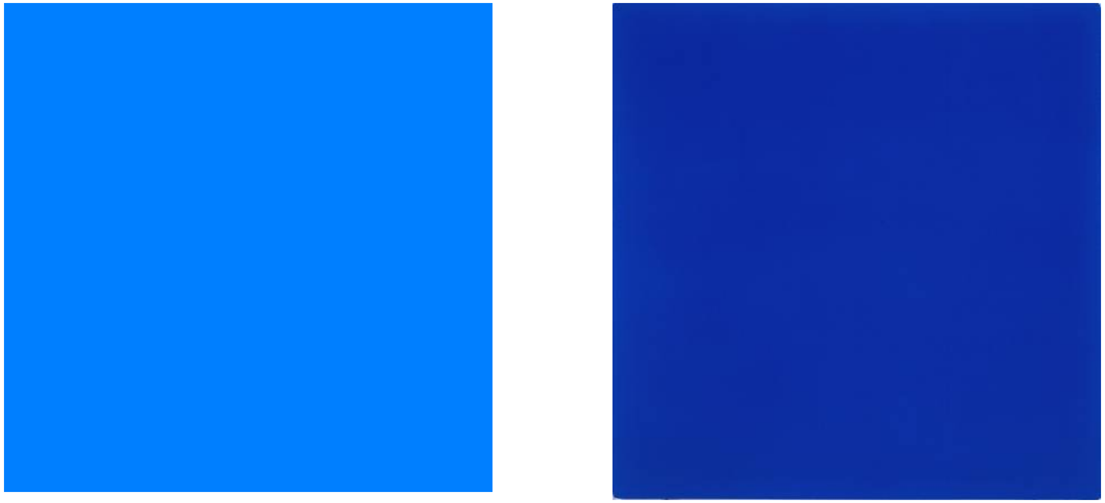
■ *John B. Haviland*
REED COLLEGE

Anchoring, Iconicity, and Orientation in Guugu Yimithirr Pointing Gestures

Speakers of Guugu Yimithirr at the Hopevale aboriginal community in Queensland use inflected forms of four cardinal direction words in all talk about location and motion. This article compares the pointing gestures in parallel episodes of two tellings of a single story, first to demonstrate that gestures too can be directionally anchored, and then to contrast other gestures that are emancipated from cardinal direction. Different sorts of indexical space and different modes of directional anchoring are posited to account for the contrasting gestural forms.

Guugu Yimithirr

In July 1770, Lt. James Cook and his crew were camped at the mouth of the Endeavour River in what is now northeast Queensland, Australia. Their ship, the original *Endeavour*, after which Cook named the river, had run aground on the Great Barrier Reef, and Cook's crew spent several weeks repairing it

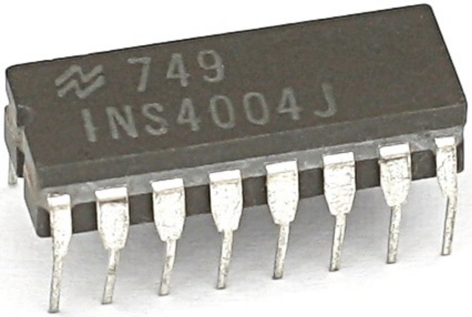


LANGUAGE AND COMPUTERS

How to talk to a computer?

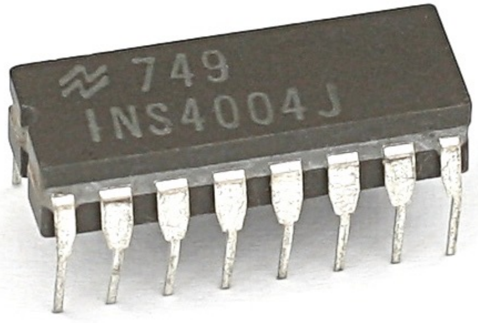
I saw a woman with a telescope wrapped in paper

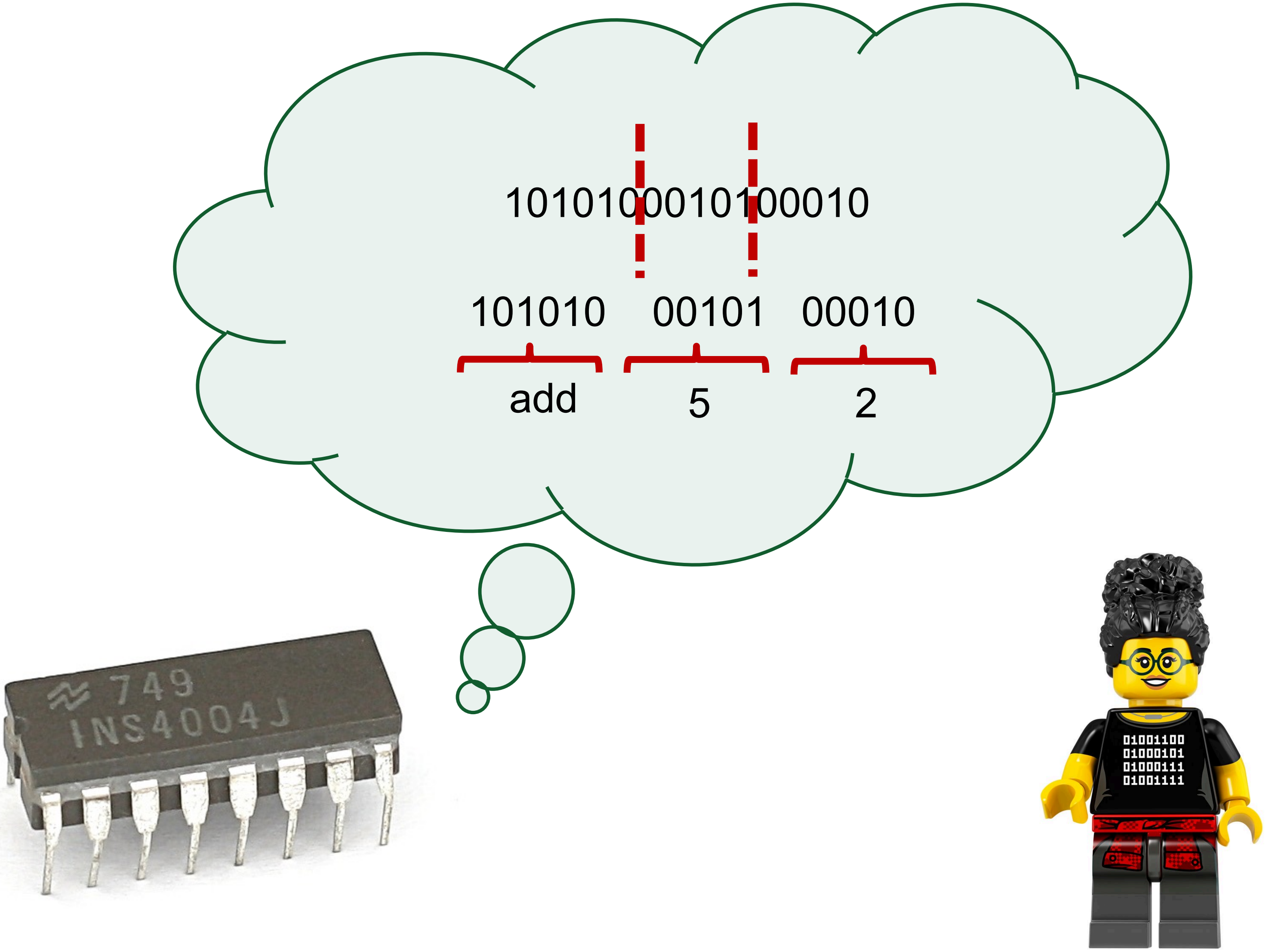
1 0



1010100010100010

1010100010100010



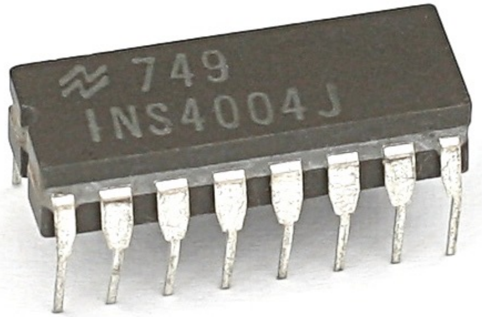


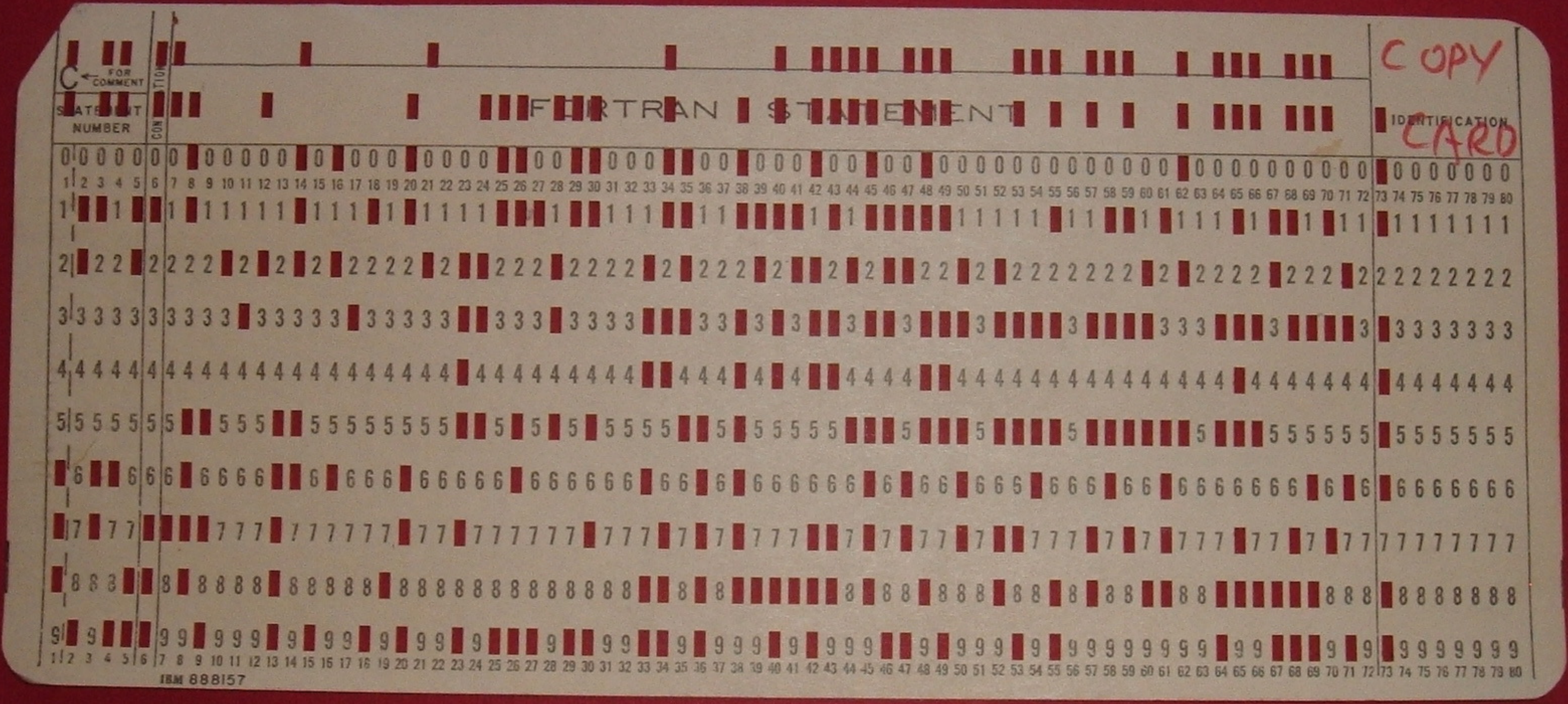
1010100010100010

101010 00101 00010

add 5 2

7




COPY
CARD
C ← FOR COMMENT
STATEMENT NUMBER
CONTINUATION
FORTRAN STATEMENT
IDENTIFICATION
IBM 888157

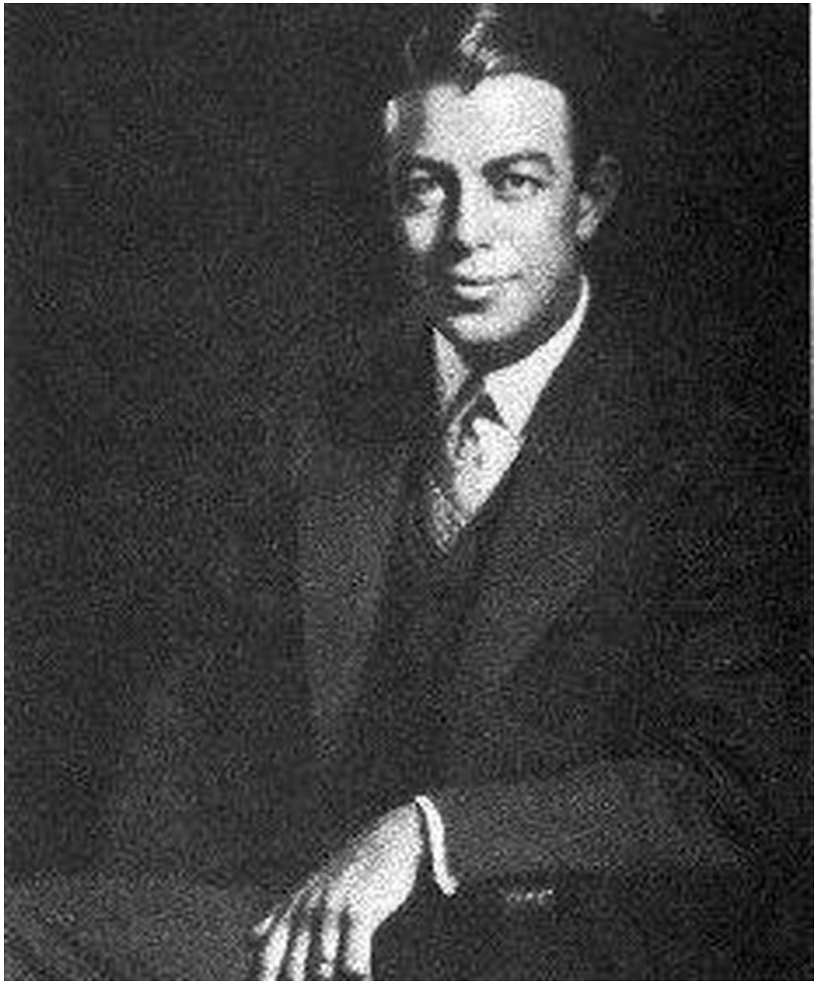
1950's

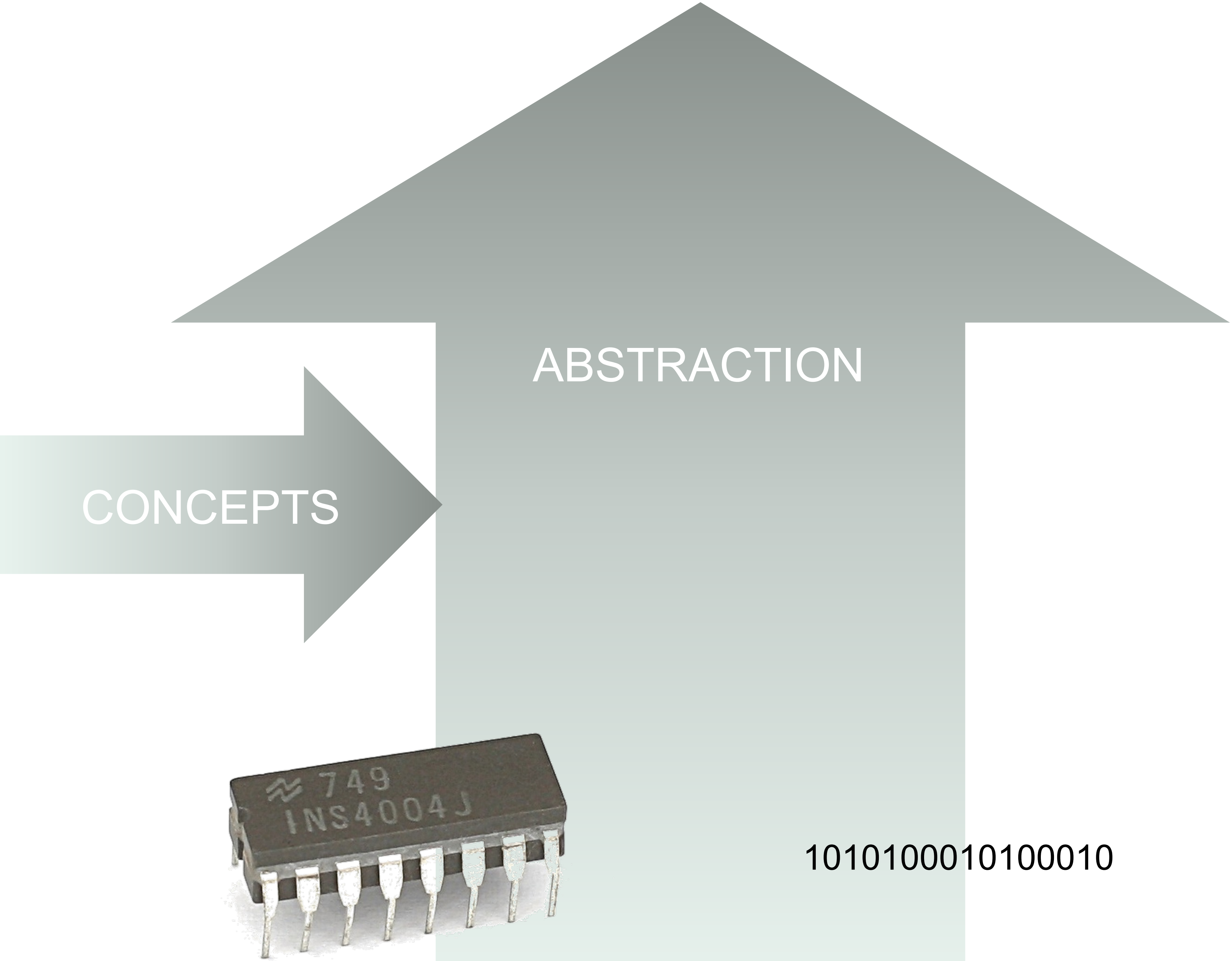
P547 -10

A TALE OF ABSTRACTION

Language shapes the way we think, and determines what we can think about.

Benjamin Lee Whorf





ABSTRACTION

CONCEPTS

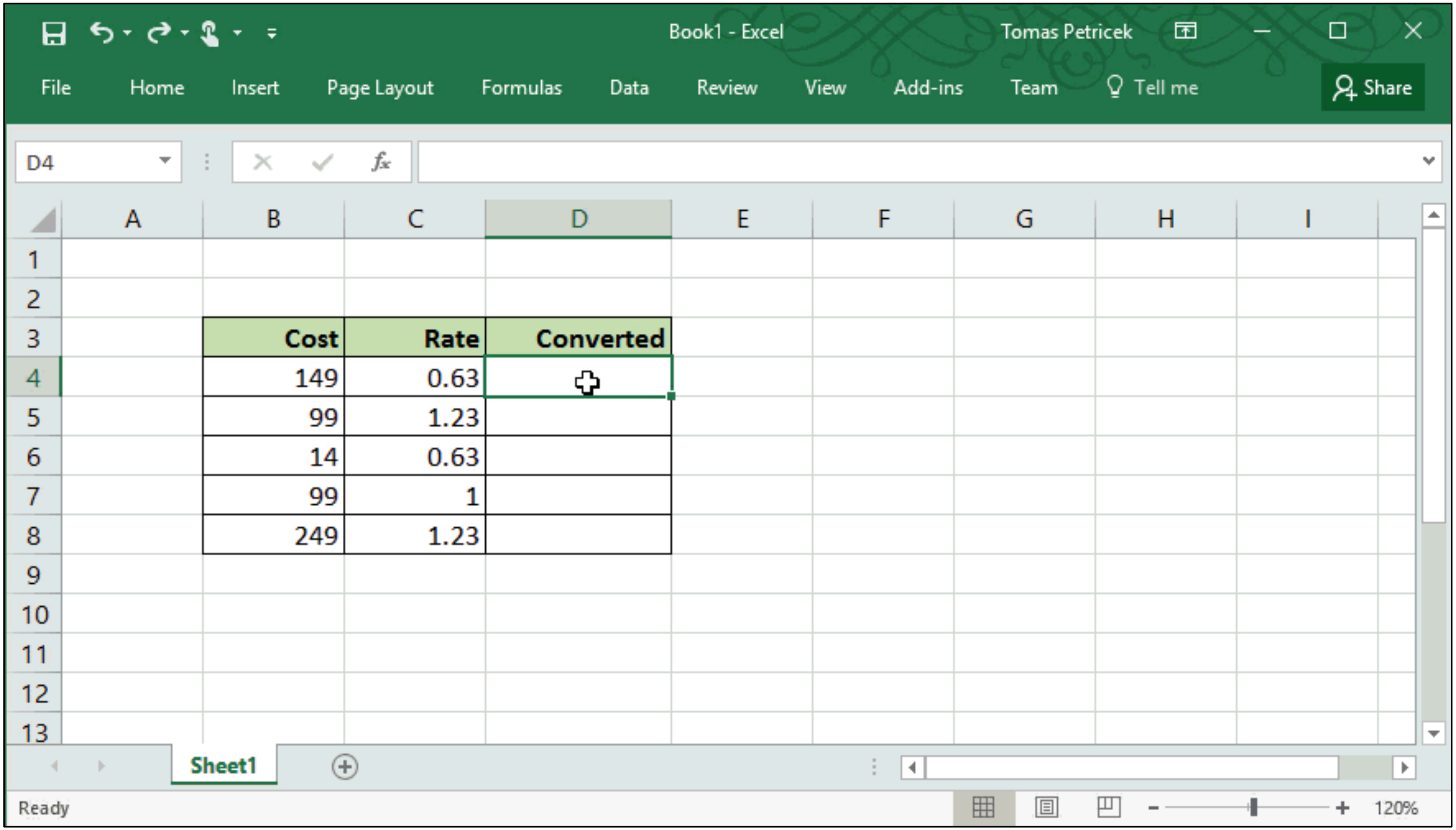
1010100010100010

The Joy of Variables

```
addNumbers() {

    x = 2;
    y = 5;

    z = x + y;
}
```

1010100010100010

Spread Sheets



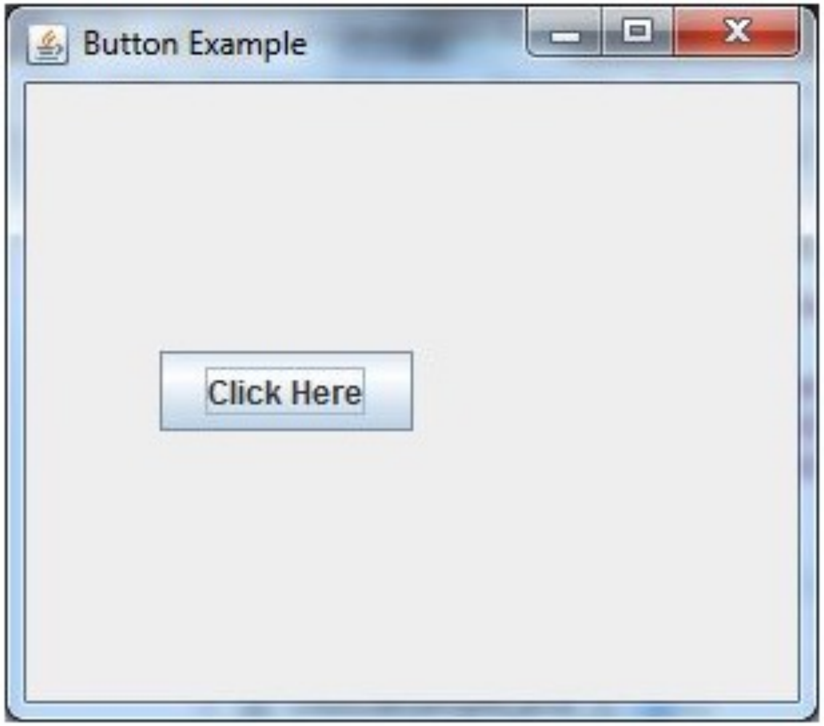
Cost	Rate	Converted
149	0.63	
99	1.23	
14	0.63	
99	1	
249	1.23	

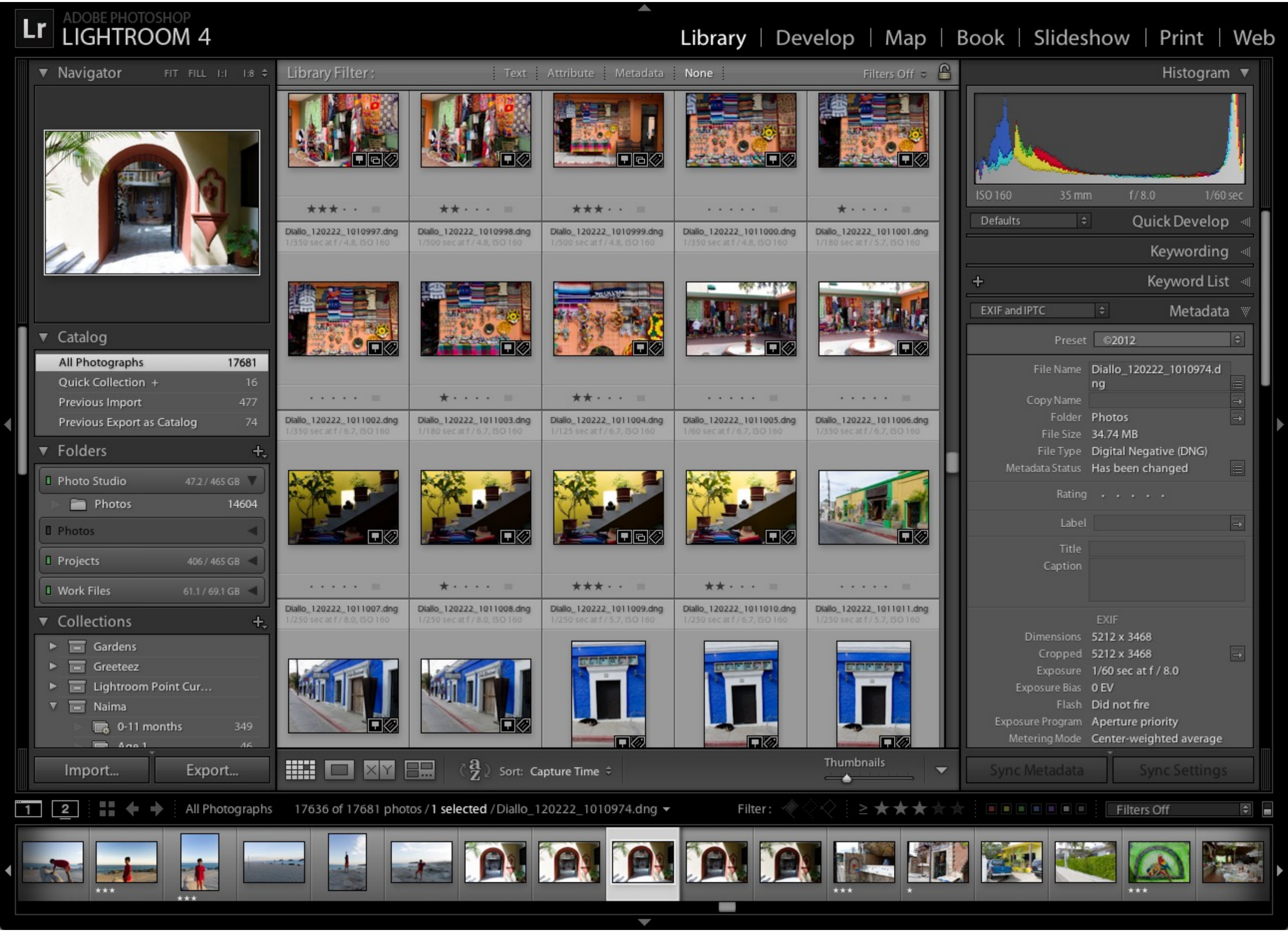
OPEN!

SELF-RISING CRUST
SELF-RISING CRUST

Object-oriented Programming

```
w = new window()

w.setSize(300x300)

w.setTitle(“Button Example”)

w.addButton(“Click Here”)
 …


w.show()
```



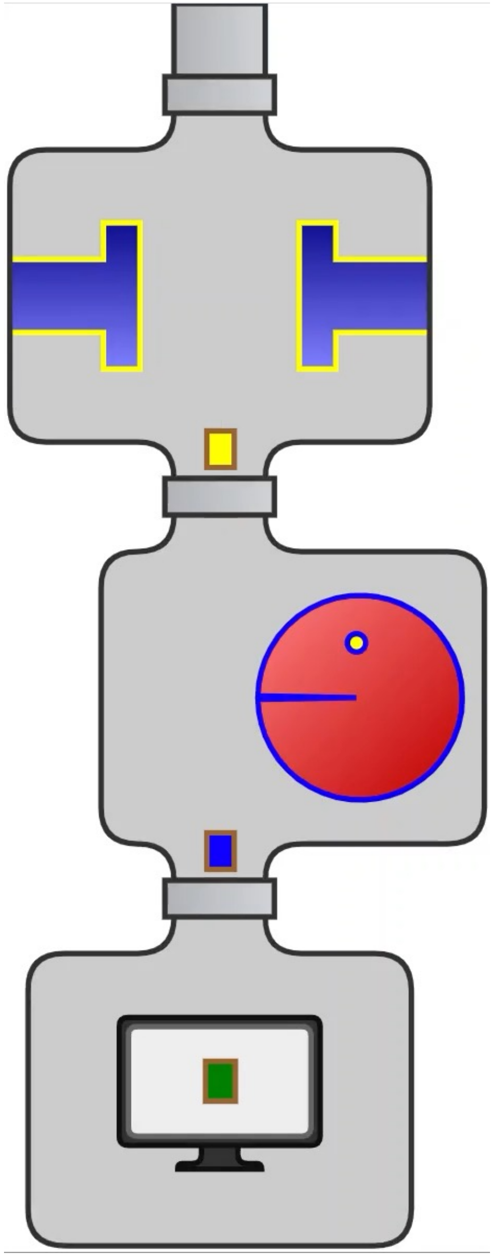


ADOBE PHOTOSHOP
LIGHTROOM 4
Library | Develop | Map | Book | Slideshow | Print | Web
Navigator FIT FILL 1:1 1:8
Library Filter : Text Attribute Metadata None Filters Off
Histogram
Catalog
All Photographs 17681
Quick Collection + 16
Previous Import 477
Previous Export as Catalog 74
Folders
Photo Studio 47.2 / 465 GB
Photos 14604
Photos
Projects 406 / 465 GB
Work Files 61.1 / 69.1 GB
Collections
Gardens
Greeteez
Lightroom Point Cur…
Naima
0-11 months 349
Age 1 46
Import...
Export...
Diallo_120222_1010997.dng
1/350 sec at f / 4.8, ISO 160
Diallo_120222_1010998.dng
1/500 sec at f / 4.8, ISO 160
Diallo_120222_1010999.dng
1/500 sec at f / 4.8, ISO 160
Diallo_120222_1011000.dng
1/350 sec at f / 4.8, ISO 160
Diallo_120222_1011001.dng
1/180 sec at f / 5.7, ISO 160
Diallo_120222_1011002.dng
1/350 sec at f / 6.7, ISO 160
Diallo_120222_1011003.dng
1/180 sec at f / 6.7, ISO 160
Diallo_120222_1011004.dng
1/125 sec at f / 6.7, ISO 160
Diallo_120222_1011005.dng
1/60 sec at f / 6.7, ISO 160
Diallo_120222_1011006.dng
1/350 sec at f / 6.7, ISO 160
Diallo_120222_1011007.dng
1/250 sec at f / 8.0, ISO 160
Diallo_120222_1011008.dng
1/250 sec at f / 8.0, ISO 160
Diallo_120222_1011009.dng
1/250 sec at f / 5.7, ISO 160
Diallo_120222_1011010.dng
1/250 sec at f / 6.7, ISO 160
Diallo_120222_1011011.dng
1/250 sec at f / 5.7, ISO 160
Sort: Capture Time
Thumbnails
ISO 160 35 mm f / 8.0 1/60 sec
Defaults Quick Develop
Keywording
Keyword List
EXIF and IPTC Metadata
Sync Metadata
Sync Settings
1 2
Filter :
Filters Off

Preset ©2012
File Name Diallo_120222_1010974.dng
Copy Name
Folder Photos
File Size 34.74 MB
File Type Digital Negative (DNG)
Metadata Status Has been changed
Rating · · · · ·
Label
Title
Caption
EXIF
Dimensions 5212 x 3468
Cropped 5212 x 3468
Exposure 1/60 sec at f / 8.0
Exposure Bias 0 EV
Flash Did not fire
Exposure Program Aperture priority
Metering Mode Center-weighted average

All Photographs 17636 of 17681 photos / 1 selected / Diallo_120222_1010974.dng

```
shapesEventStream
        .transform(toSquare)
        .filter(notYellow)
        .show(screen)
```



How Different are Programming Languages, Really?

Calculate the square of the even numbers between 1 and 10

How Different are Programming Languages, Really?

Calculate the square of the even numbers between 1 and 10

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Even: 2, 4, 6, 8, 10

Squares: 4, 16, 36, 64, 100

```
import java.util.*;

public class HelloWorld{

    public static void main(String []args){

      ArrayList<Integer> list1 =
          new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
      ArrayList<Integer> list2 = new ArrayList();

      for (Integer e : list1){

          if (isEven(e)){
              list2.add((int) Math.pow(e, 2));
          }
      }

      System.out.println(list2);
    }
}
```

Java

```
[ x^2 | x <- [1..10], even x ]
```

Haskell

```
import java.util.*;
```

```
  lic c     ello      {

public static void main(String []args){

        ArrayList<Integer> list1 =
            new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
        ArrayList<Integer> list2 = new ArrayList();

        for (Integer e : list1){

            if (isEven(e)){
                list2.add((int) Math.pow(e, 2));
            }
        }

        System.out.println(list2);
    }
}
```

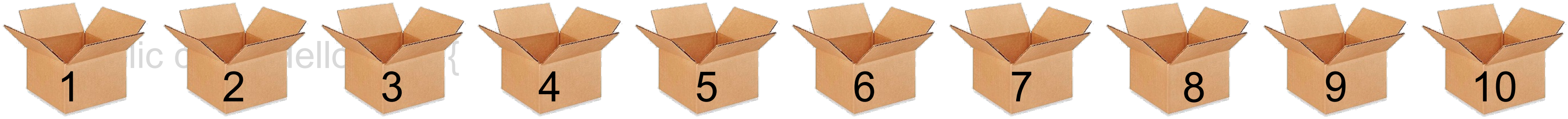
Java

```
import java.util.*;

   lic c      ello         {

public static void main(String []args){

     ArrayList<Integer> list1 =
        new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
     ArrayList<Integer> list2 = new ArrayList();

     for (Integer e : list1){

        if (isEven(e)){
           list2.add((int) Math.pow(e, 2));
        }
     }

     System.out.println(list2);
   }
}
```

1
2
3
4
5
6
7
8
9
10

Java

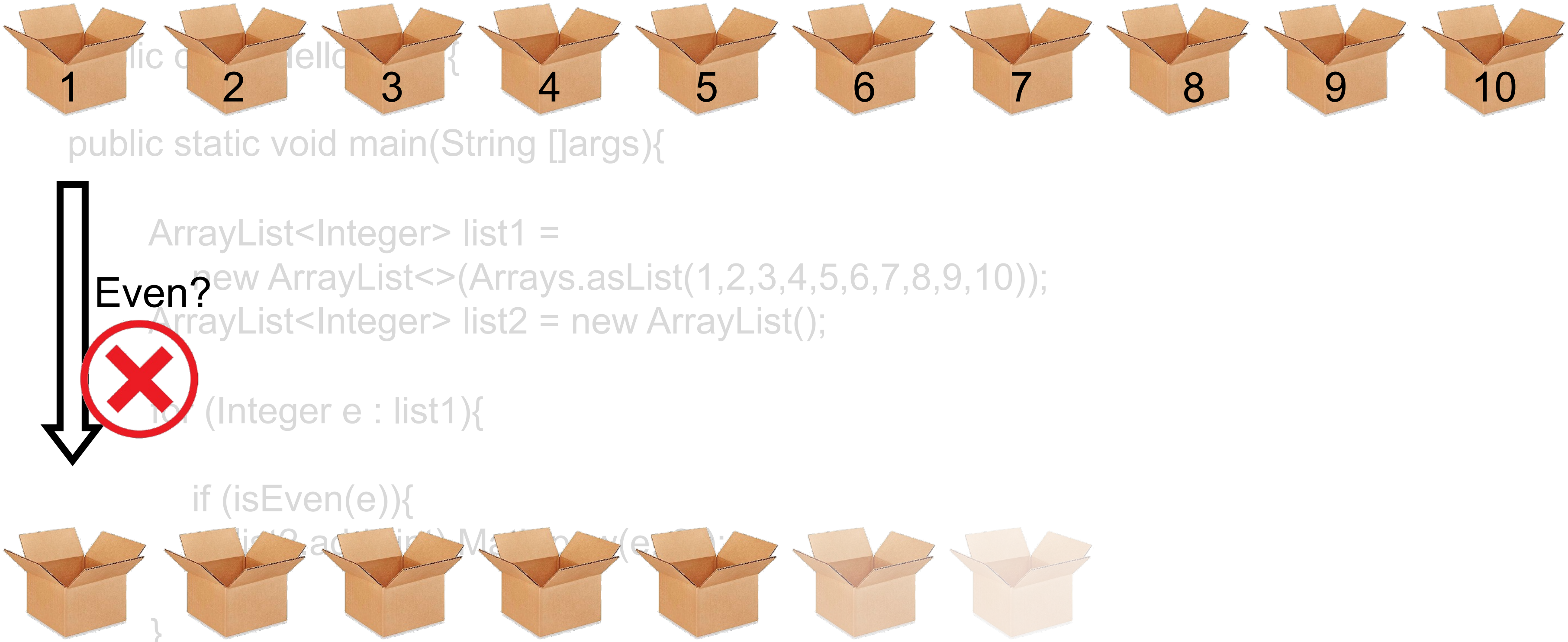
```
import java.util.*;
```



```
[illegible]lic c[illegible]ello[illegible]{

public static void main(String []args){

    ArrayList<Integer> list1 =
        new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
    ArrayList<Integer> list2 = new ArrayList();

    for (Integer e : list1){

        if (isEven(e)){
            li[illegible]t2.ad[illegible]int) Ma[illegible]w(e[illegible]);
```

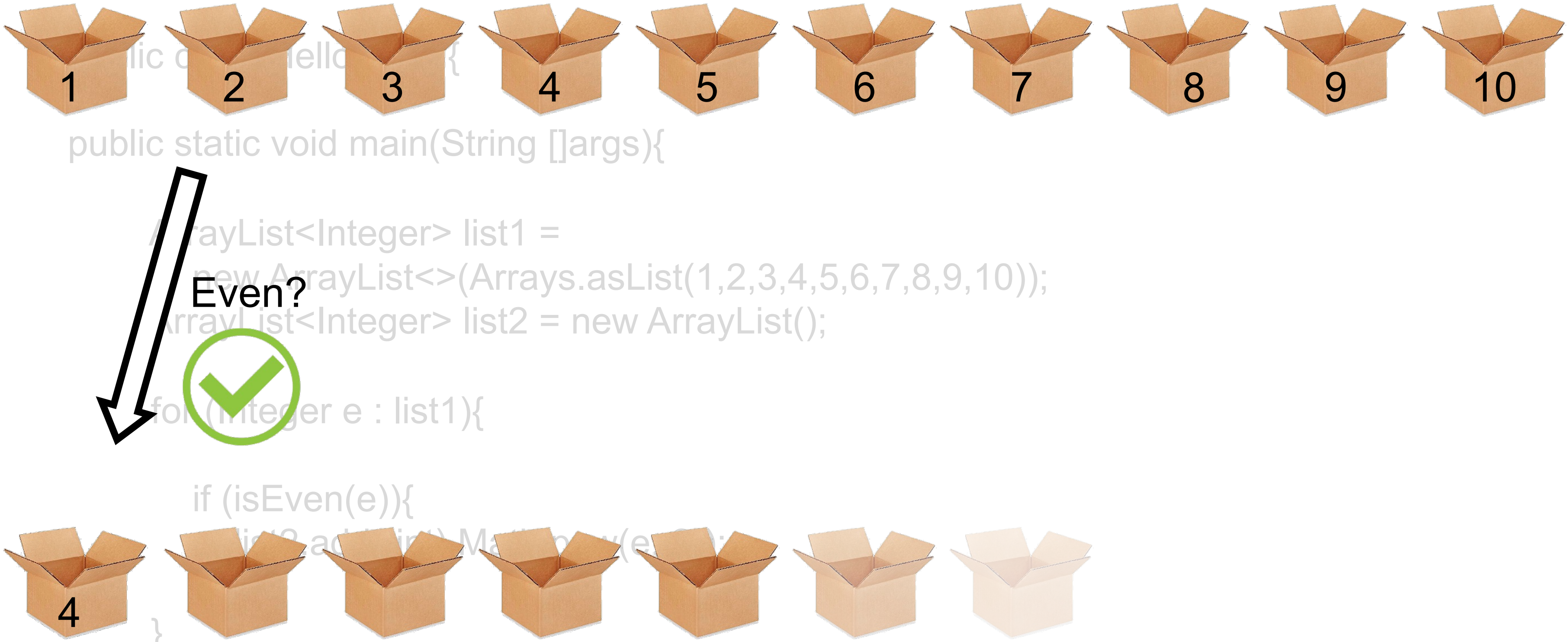
```
    }

    System.out.println(list2);
  }
}
```

Java



```
import java.util.*;

public class Hello{

public static void main(String []args){

    ArrayList<Integer> list1 =
        new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
    ArrayList<Integer> list2 = new ArrayList();

    for (Integer e : list1){

        if (isEven(e)){
            list2.add((int) Math.pow(e, 2));

    }

    System.out.println(list2);
  }
}
```

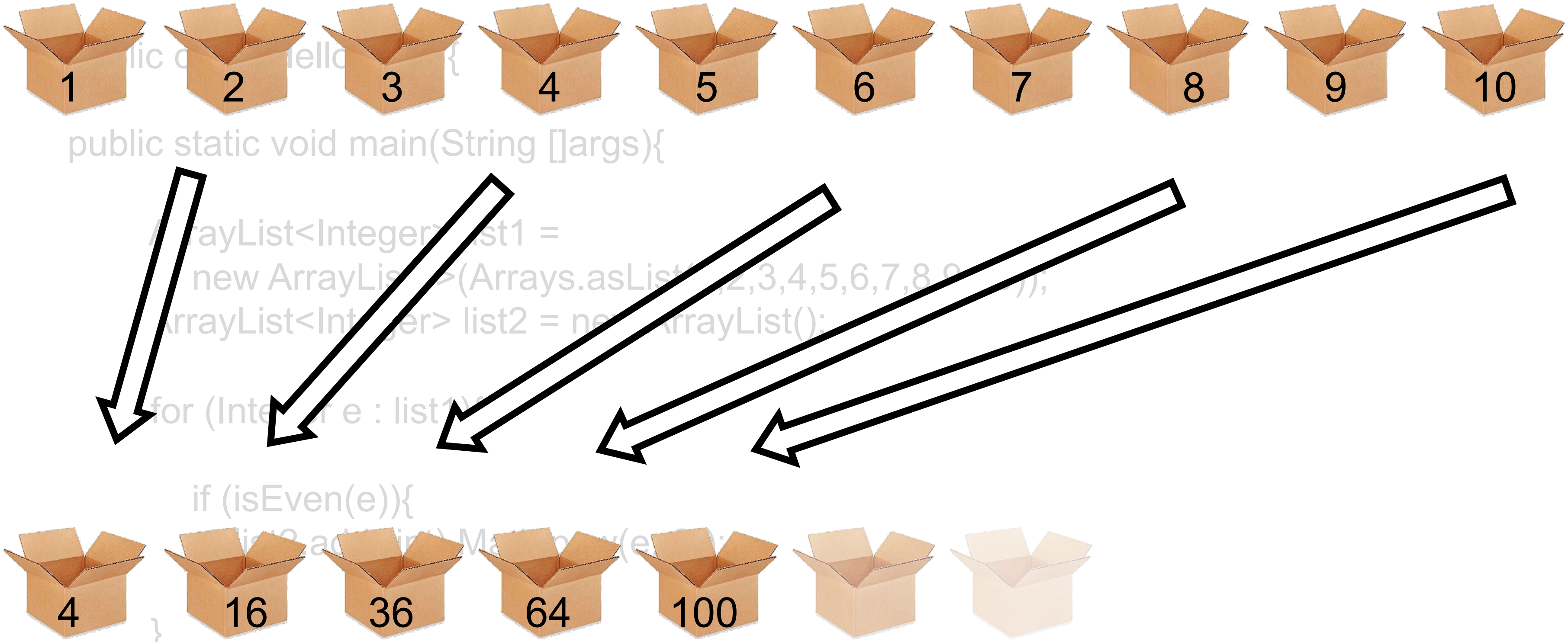
Java



```
import java.util.*;

public class Hello{

public static void main(String []args){

         ArrayList<Integer> list1 =
             new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
         ArrayList<Integer> list2 = new ArrayList();

         for (Integer e : list1){

             if (isEven(e)){
                 list2.add((int) Math.pow(e, 2));

         }

         System.out.println(list2);
      }
}
```

Java

```
import java.util.*;

public class Hello {

public static void main(String []args){

    ArrayList<Integer> list1 =
        new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
    ArrayList<Integer> list2 = new ArrayList();

    for (Integer e : list1){

        if (isEven(e)){
            list2.add((int) Math.pow(e, 2));
    }

    System.out.println(list2);
  }
}
```



Java

```
import java.util.*;

public class HelloWorld{

public static void main(String []args){

    ArrayList<Integer> list1 =
      new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
    ArrayList<Integer> list2 = new ArrayList();

    for (Integer e : list1){

      if (isEven(e)){
        list2.add((int) Math.pow(e, 2));
      }
    }

    System.out.println(list2);
   }
}
```

Java

```
[ x^2 | x <- [1..10], even x ]
```

Haskell

```
import java.util.*;

public class HelloWorld{

public static void main(String []args){

    ArrayList<Integer> list1 =
      new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));
    ArrayList<Integer> list2 = new ArrayList();

    for (Integer e : list1){

      if (isEven(e)){
        list2.add((int) Math.pow(e, 2));
      }
    }

    System.out.println(list2);
  }
}
```

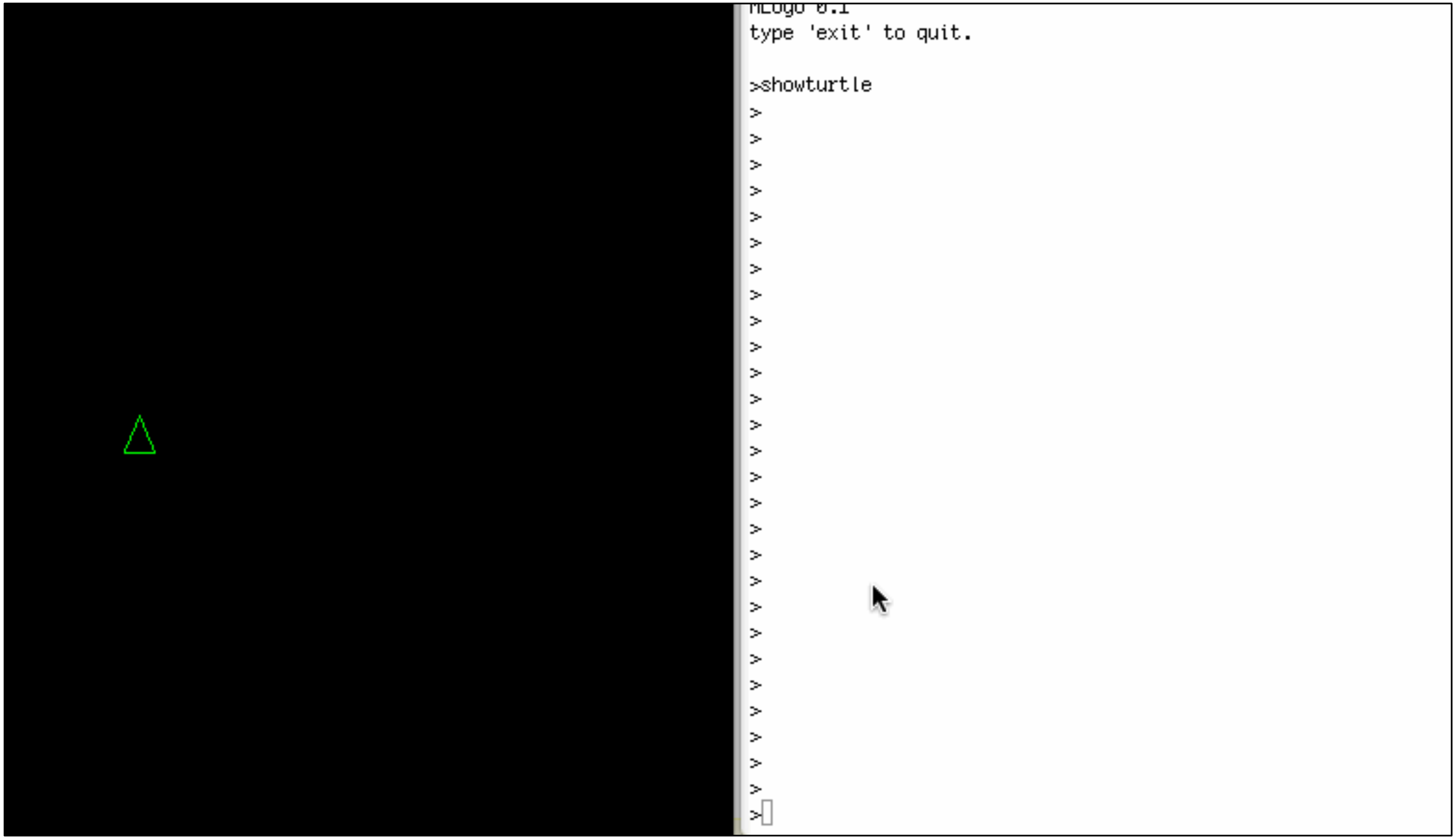
$$\{\, x^2 \mid 1 \leq x \leq 10, x\%2 = 0, x \in \mathbb{N}\}$$

```
[ x^2 | x <- [1..10], even x ]
```

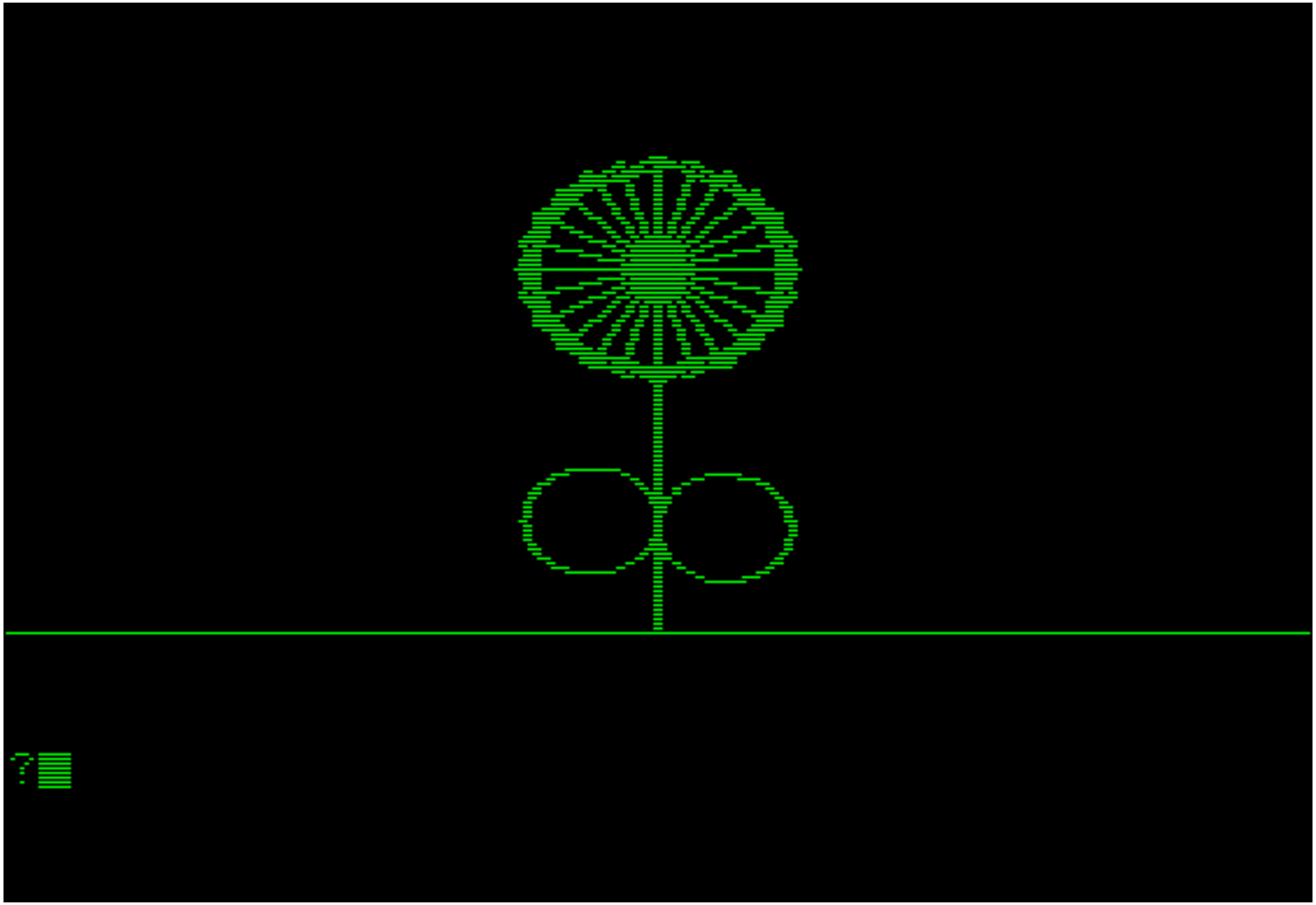
Java

Haskell

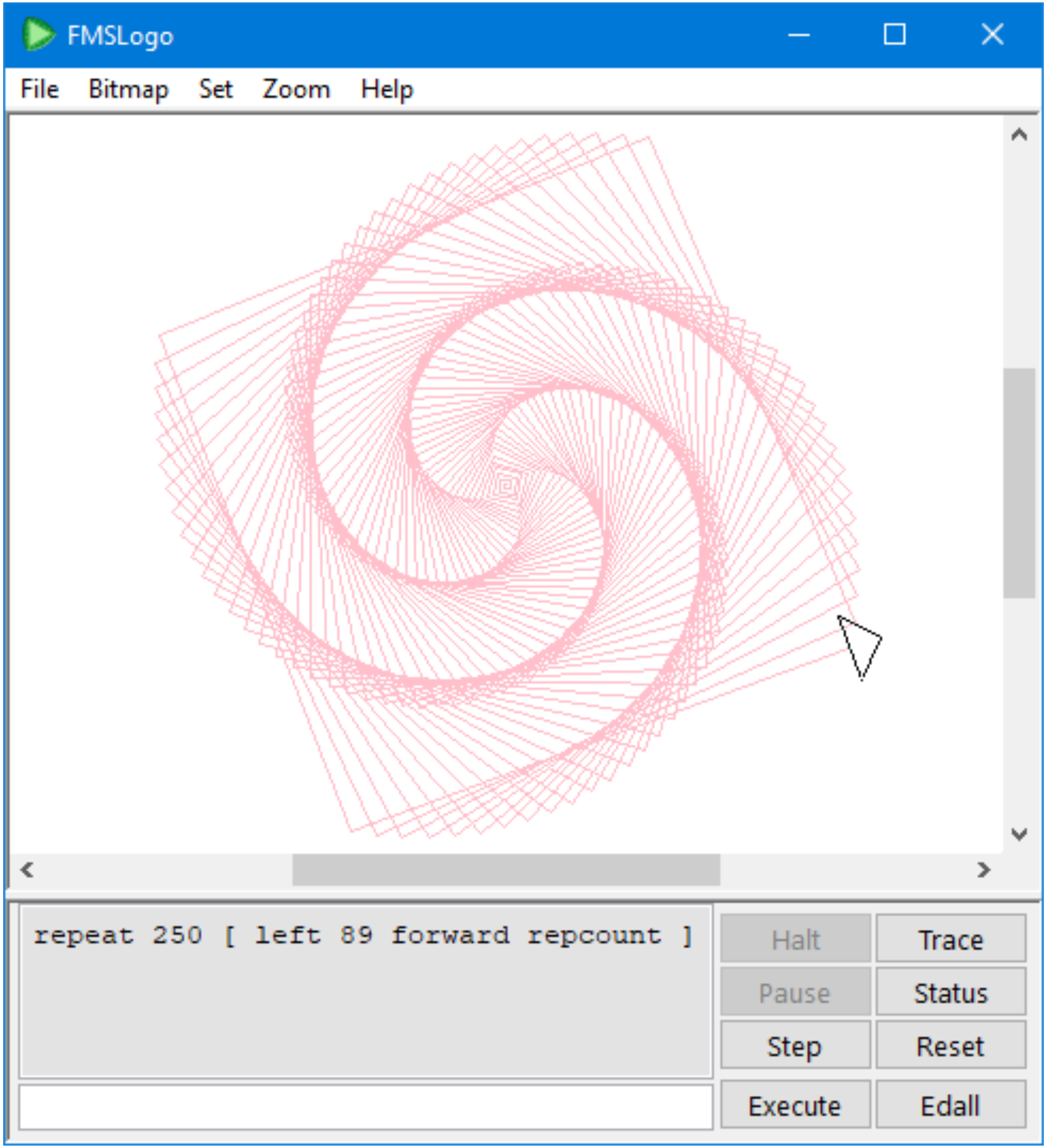
The Logo Programming language



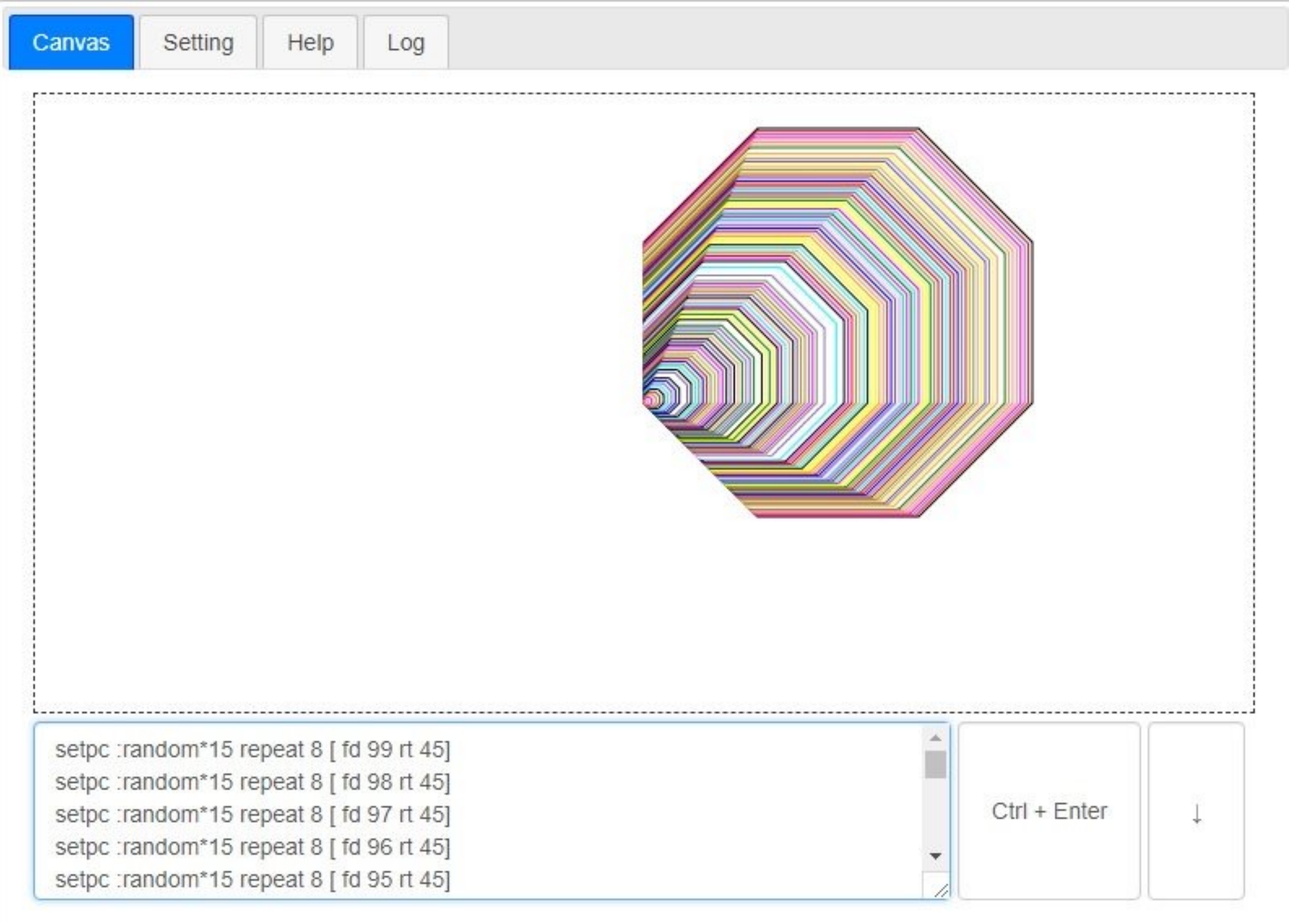
```
MLogo 0.1
type 'exit' to quit.

>showturtle
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
```



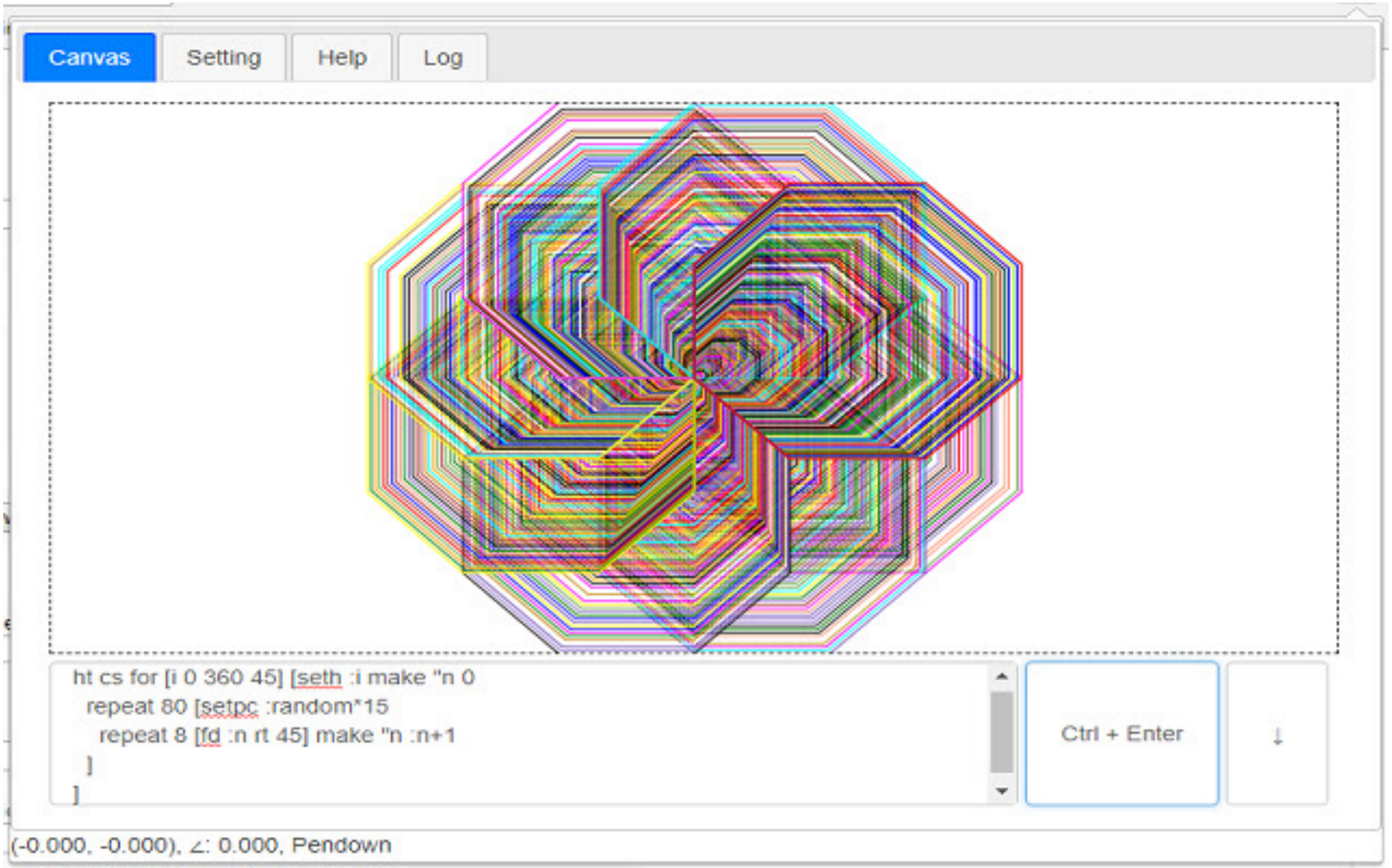
```
?
```



```
repeat 250 [ left 89 forward repcount ]
```



```
setpc :random*15 repeat 8 [ fd 99 rt 45]
setpc :random*15 repeat 8 [ fd 98 rt 45]
setpc :random*15 repeat 8 [ fd 97 rt 45]
setpc :random*15 repeat 8 [ fd 96 rt 45]
setpc :random*15 repeat 8 [ fd 95 rt 45]
```



```
ht cs for [i 0 360 45] [seth :i make "n 0
  repeat 80 [setpc :random*15
    repeat 8 [fd :n rt 45] make "n :n+1
  ]
]
```

(-0.000, -0.000), ∠: 0.000, Pendown

Domain-specific Languages

A Domain-Specific Language for Payroll Calculations: a Case Study at DATEV

Markus Voelter, Sergej Koščejev, Marcel Riedel, Anna Deitsch and Andreas Hinkelmann

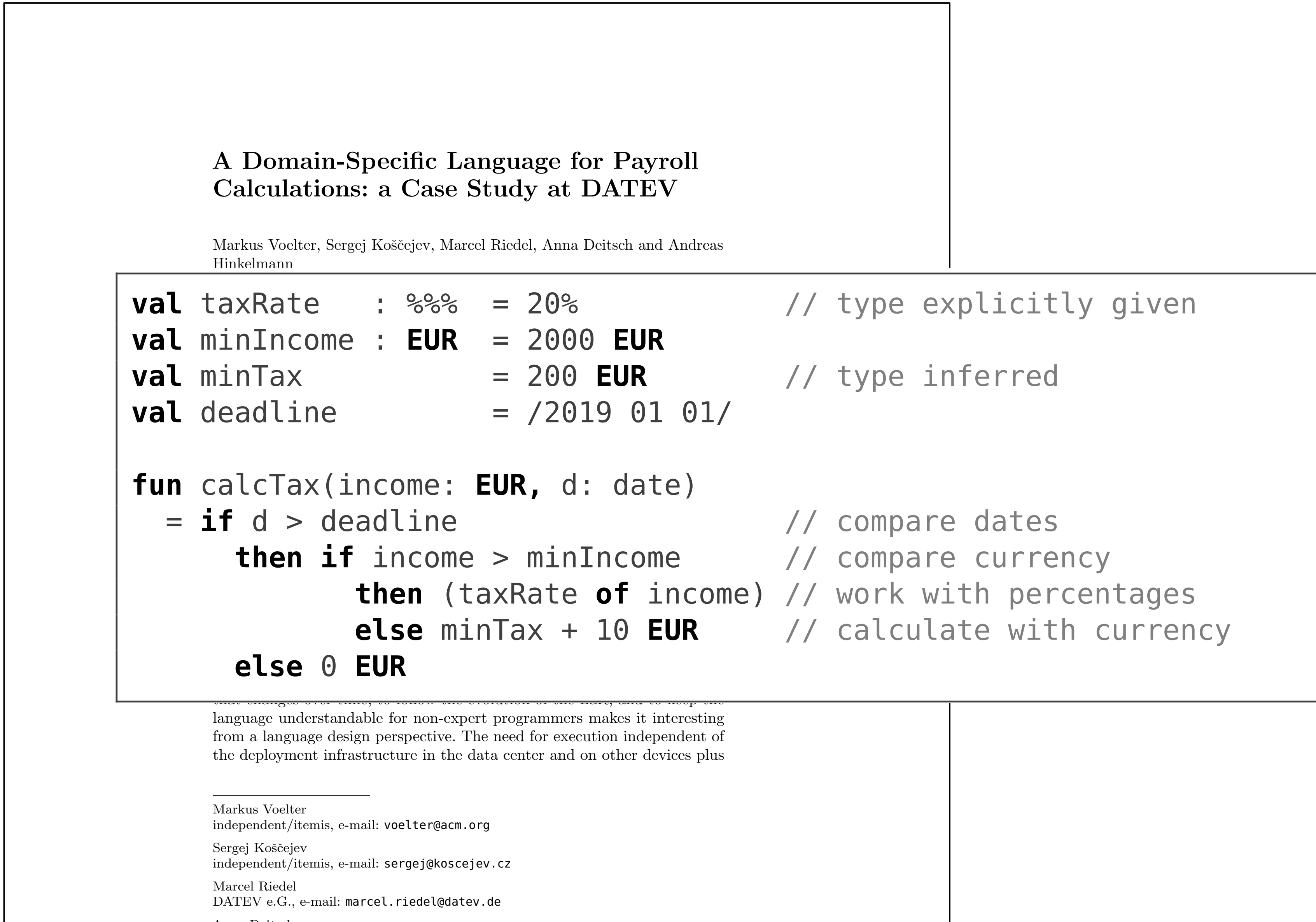
1 Introduction

Over the last three years, DATEV, a leading German payroll services provider, has been developing a domain-specific language (DSL) for expressing the calculation logic at the core of their payroll systems. The goal is to allow the business programmers to express and test the calculations and their evolution over time in a way that is completely independent of the technical infrastructure that is used to execute them in the data center. Business programmers are people who are experts in the intricacies of the payroll domain and its governing laws and regulations (LaR) – but not in software development – which leads to interesting tradeoffs in the design of the DSL. The specific set of challenges that motivated the development of the DSL are given in Sec. 3.2. Payroll might seem dull and not too complicated ("just a bunch of decisions and some math"). However, the need to work on data that changes over time, to follow the evolution of the LaR, and to keep the language understandable for non-expert programmers makes it interesting from a language design perspective. The need for execution independent of the deployment infrastructure in the data center and on other devices plus

Markus Voelter
independent/itemis, e-mail: voelter@acm.org

Sergej Koščejev
independent/itemis, e-mail: sergej@koscejev.cz

Domain-specific languages



A Domain-Specific Language for Payroll Calculations: a Case Study at DATEV

Markus Voelter, Sergej Koščejev, Marcel Riedel, Anna Deitsch and Andreas Hinkelmann

```
val taxRate   : %%%  = 20%                  // type explicitly given
val minIncome : EUR  = 2000 EUR
val minTax           = 200 EUR              // type inferred
val deadline         = /2019 01 01/

fun calcTax(income: EUR, d: date)
  = if d > deadline                         // compare dates
      then if income > minIncome            // compare currency
             then (taxRate of income)       // work with percentages
             else minTax + 10 EUR           // calculate with currency
      else 0 EUR
```

that changes over time, to follow the evolution of the Law, and to keep the language understandable for non-expert programmers makes it interesting from a language design perspective. The need for execution independent of the deployment infrastructure in the data center and on other devices plus

Markus Voelter
independent/itemis, e-mail: voelter@acm.org

Sergej Koščejev
independent/itemis, e-mail: sergej@koscejev.cz

Marcel Riedel
DATEV e.G., e-mail: marcel.riedel@datev.de

HMusic: A domain specific language for music programming and live coding

André Rauber Du Bois
Programa de Pós-Graduação em Computação
Universidade Federal de Pelotas
Pelotas - RS - Brazil
dubois@inf.ufpel.edu.br

Rodrigo Geraldo Ribeiro
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Ouro Preto
Ouro Preto - MG - Brazil
rodrigo@decsi.ufop.br

ABSTRACT

This paper presents HMusic, a domain specific language based on music patterns that can be used to write music and live coding. The main abstractions provided by the language are patterns and tracks. Code written in HMusic looks like patterns and multi-tracks available in music sequencers, drum machines and DAWs. HMusic provides primitives to design and compose patterns generating new patterns. The basic abstractions provided by the language have an inductive definition and HMusic is embedded in the Haskell functional programming language, hence programmers can design functions to manipulate music on the fly. The current implementation of the language is compiled into Sonic Pi [10] and can be downloaded from [9].

Author Keywords

Live coding, Functional Programming, Haskell

CCS Concepts

•**Applied computing** → **Sound and music computing;** *Performing arts;* •**Software and its engineering** → **Functional languages;**

1. INTRODUCTION

Computers are generic abstract machines that can be programmed with different goals in a variety of domains, including arts in general, and music. Computer music is usually associated with the use of software applications to create music, but on the other hand, there is a growing interest in programming languages that let artists write software as an expression of art. There are a number of programming languages that allow artists to write music, e.g., CSound [2], Max [13, 28], Pure Data [23], Supercollider [19], Chuck [27], FAUST [22], to name a few. Besides writing songs, all these languages also allow the live coding of music. Live coding is the idea of writing programs that represent music

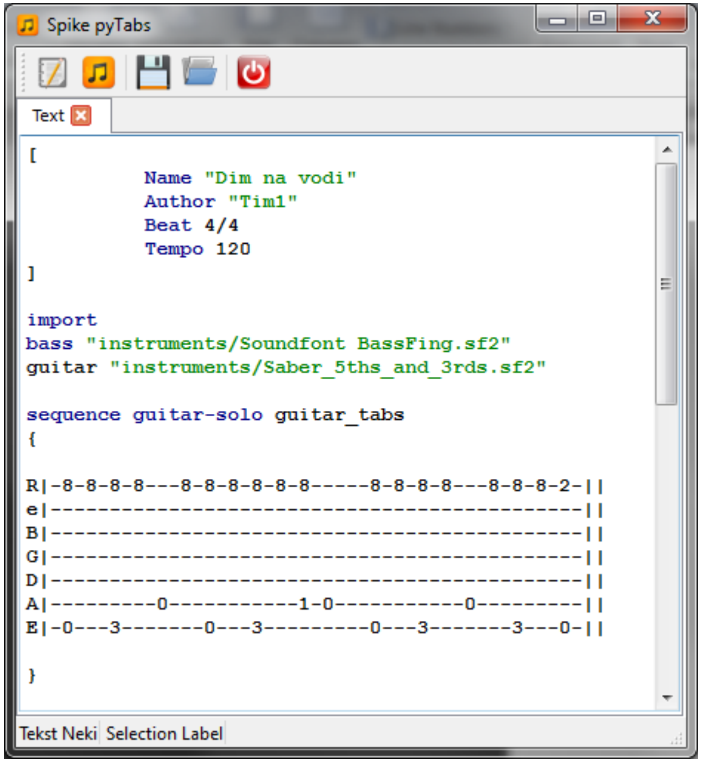
very similar to the grids available in sequencers, drum machines and DAWs. The difference is that these abstractions have an inductive definition, hence programmers can write functions that manipulate these tracks in real time. As the DSL is embedded in Haskell, it is possible to use all the power of functional programming in our benefit to define new abstractions over patterns of songs. To understand the paper the reader needs no previous knowledge of Haskell, although some knowledge of functional programming and recursive definitions would help. We try to introduce the concepts and syntax of Haskell needed to understand the paper as we go along.

The contributions of this paper are as follows:

- We describe the design and implementation of HMusic, a DSL for music programming that provides the abstractions of patterns and tracks, together with a set of functions to manipulate and combine these abstractions. The interesting aspect of the language is that basic programs look like the grids available in drum machines and sequencers, which is a concept familiar to music composers.
- We describe a simple interface for live coding based on looping tracks and function application to modify tracks in real time.

In the current implementation of HMusic, tracks can load pre-recorded samples. As it is currently compiled into Sonic Pi [10], any sample accessible by the Sonic Pi environment can be loaded and manipulated in tracks. The current implementation of the HMusic language can be downloaded from [9].

The paper is organized as follows. First we describe the main constructors for pattern (Section 2.1) and track (Section 2.2) design and their basic operations. Next, we examine the important abstraction of track composition, i.e., combining different multi-tracks to form a new track (Section 2.3). The abstraction provided by HMusic for live cod-



```
[
            Name "Dim na vodi"
            Author "Tim1"
            Beat 4/4
            Tempo 120
]

import
bass "instruments/Soundfont BassFing.sf2"
guitar "instruments/Saber_5ths_and_3rds.sf2"

sequence guitar-solo guitar_tabs
{

R|-8-8-8-8---8-8-8-8-8-8-----8-8-8-8---8-8-8-2-||
e|---------------------------------------------||
B|---------------------------------------------||
G|---------------------------------------------||
D|---------------------------------------------||
A|---------0-----------1-0-----------0---------||
E|-0---3-------0---3---------0---3-------3---0-||

}
```

```
[
    Name "Dim na vodi"
    Author "Tim1"
    Beat 4/4
    Tempo 120
]
```

song meta data

```
import
bass "instruments/Soundfont BassFing.sf2"
guitar "instruments/Saber_5ths_and_3rds.sf2"
```

import section

```
sequence guitar-solo bass_tabs
{

R|-8-8-8-8---8-8-8-8-8-8-----8-8-8-8---8-8-8-2-|
G|---------------------------------------------|
D|---------------------------------------------|
A|---------------------------------------------|
E|-0-0-0-0-0-0-0-0-0-0-3-2-1-0-0-3-3-5-5-3-3-0-|

}

sequence guitar-rhythm guitar_chords
{
    A(4) B(4) C(4) D(4) E(4) F(4) G(4)
}
```

sequences section

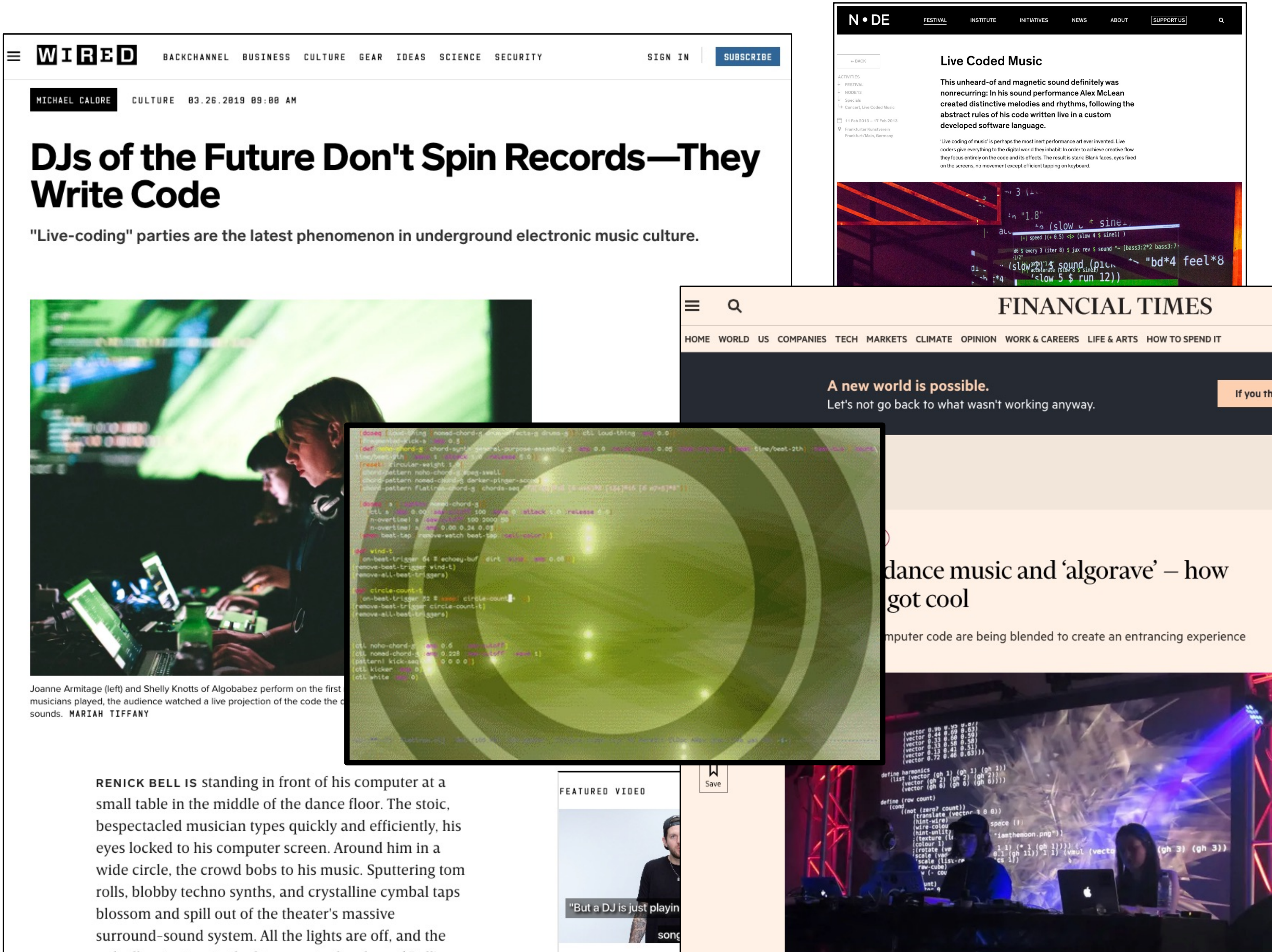
```
segment Chorus
{
    bass_tabs : bass
    guitar_chords : guitar
}
```

song segments section

```
timeline
{
    Chorus
}
```

song timeline

https://github.com/E2Music/pyTabs



MICHAEL CALORE CULTURE 03.26.2019 09:00 AM

DJs of the Future Don't Spin Records—They Write Code

"Live-coding" parties are the latest phenomenon in underground electronic music culture.

Joanne Armitage (left) and Shelly Knotts of Algobabez perform on the first musicians played, the audience watched a live projection of the code the sounds. MARIAH TIFFANY

RENICK BELL IS standing in front of his computer at a small table in the middle of the dance floor. The stoic, bespectacled musician types quickly and efficiently, his eyes locked to his computer screen. Around him in a wide circle, the crowd bobs to his music. Sputtering tom rolls, blobby techno synths, and crystalline cymbal taps blossom and spill out of the theater's massive surround-sound system. All the lights are off, and the

Live Coded Music

ACTIVITIES
FESTIVAL
NODE13
Specials
Concert, Live Coded Music

11 Feb 2013 – 17 Feb 2013

Frankfurter Kunstverein
Frankfurt/Main, Germany

This unheard-of and magnetic sound definitely was nonrecurring: In his sound performance Alex McLean created distinctive melodies and rhythms, following the abstract rules of his code written live in a custom developed software language.

'Live coding of music' is perhaps the most inert performance art ever invented. Live coders give everything to the digital world they inhabit: In order to achieve creative flow they focus entirely on the code and its effects. The result is stark: Blank faces, eyes fixed on the screens, no movement except efficient tapping on keyboard.

FINANCIAL TIMES
HOME WORLD US COMPANIES TECH MARKETS CLIMATE OPINION WORK & CAREERS LIFE & ARTS HOW TO SPEND IT

dance music and 'algorave' – how got cool

mputer code are being blended to create an entrancing experience

EVERYTHING YOU ALWAYS WANTED TO KNOW ABOUT PROGRAMMING LANGUAGES

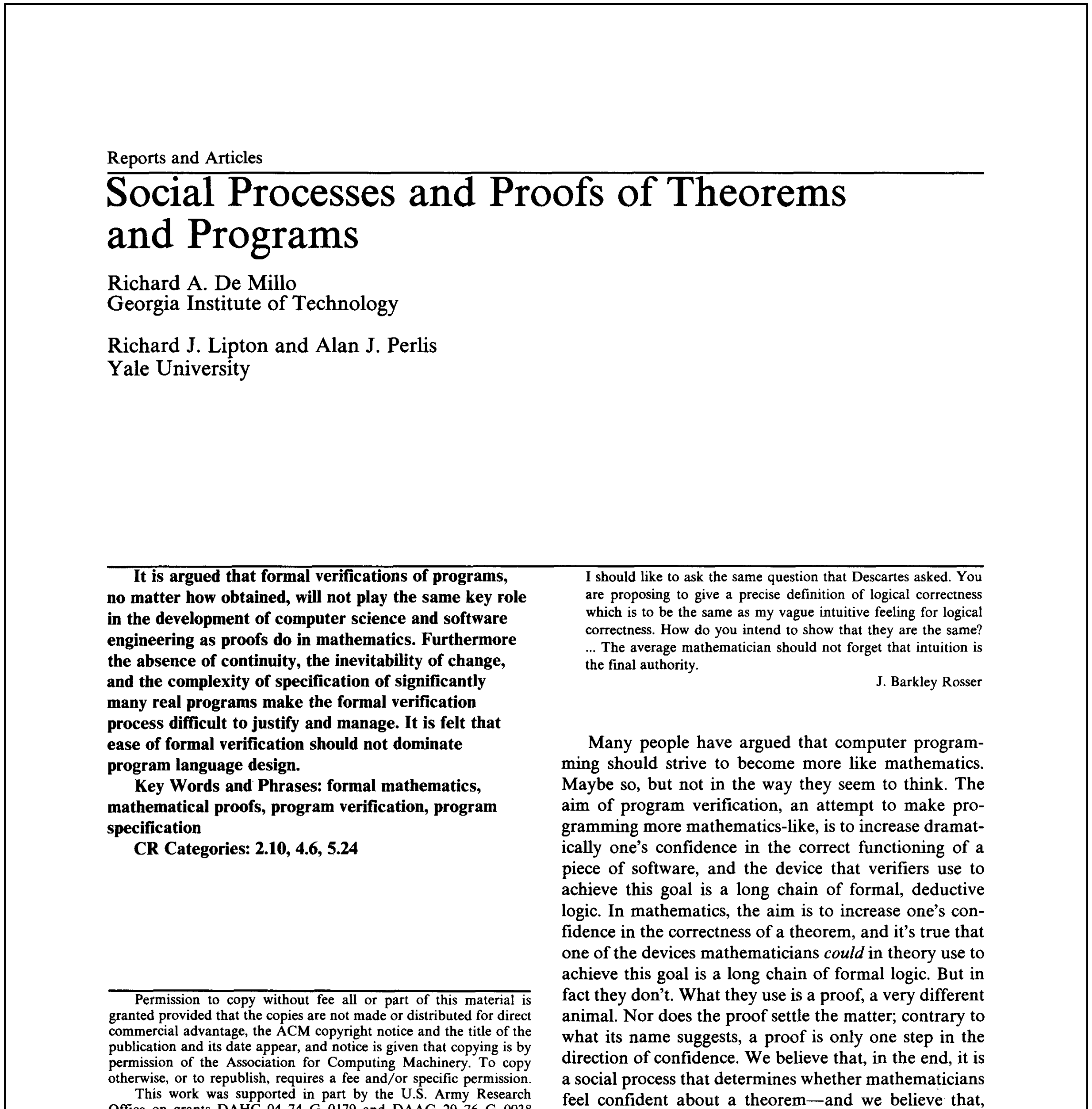
Why don’t we use **the same** PL?

What is **the best** PL?

How many PLs are there?

Why don’t we **add all** possible concepts to a single PL?

Programming Languages as a Social Process



Reports and Articles

Social Processes and Proofs of Theorems and Programs

Richard A. De Millo
Georgia Institute of Technology

Richard J. Lipton and Alan J. Perlis
Yale University

It is argued that formal verifications of programs, no matter how obtained, will not play the same key role in the development of computer science and software engineering as proofs do in mathematics. Furthermore the absence of continuity, the inevitability of change, and the complexity of specification of significantly many real programs make the formal verification process difficult to justify and manage. It is felt that ease of formal verification should not dominate program language design.

Key Words and Phrases: formal mathematics, mathematical proofs, program verification, program specification

CR Categories: 2.10, 4.6, 5.24

> I should like to ask the same question that Descartes asked. You are proposing to give a precise definition of logical correctness which is to be the same as my vague intuitive feeling for logical correctness. How do you intend to show that they are the same? ... The average mathematician should not forget that intuition is the final authority.
>
> J. Barkley Rosser

Many people have argued that computer programming should strive to become more like mathematics. Maybe so, but not in the way they seem to think. The aim of program verification, an attempt to make programming more mathematics-like, is to increase dramatically one's confidence in the correct functioning of a piece of software, and the device that verifiers use to achieve this goal is a long chain of formal, deductive logic. In mathematics, the aim is to increase one's confidence in the correctness of a theorem, and it's true that one of the devices mathematicians *could* in theory use to achieve this goal is a long chain of formal logic. But in fact they don't. What they use is a proof, a very different animal. Nor does the proof settle the matter; contrary to what its name suggests, a proof is only one step in the direction of confidence. We believe that, in the end, it is a social process that determines whether mathematicians feel confident about a theorem—and we believe that,

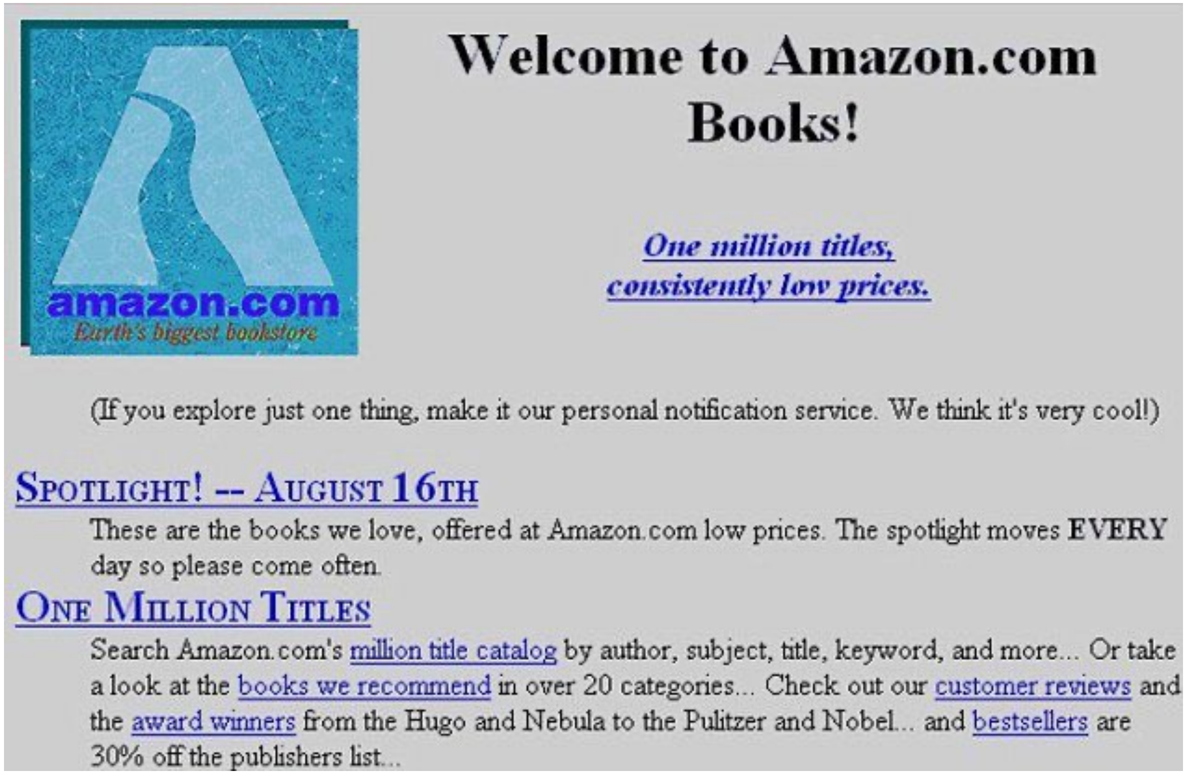
This work was supported in part by the U.S. Army Research Office on grants DAHC 04-74-G-0179 and DAAG 29-76-G-0038

A good analogy: artificial languages are similar to **human languages**

- Cultural heritage, social groups

Effect of companies and market

- Microsoft: C#
- Apple: Objective-C, Swift
- Google: Go, Dart



Welcome to Amazon.com Books!

One million titles, consistently low prices.

(If you explore just one thing, make it our personal notification service. We think it's very cool!)

SPOTLIGHT! -- AUGUST 16TH

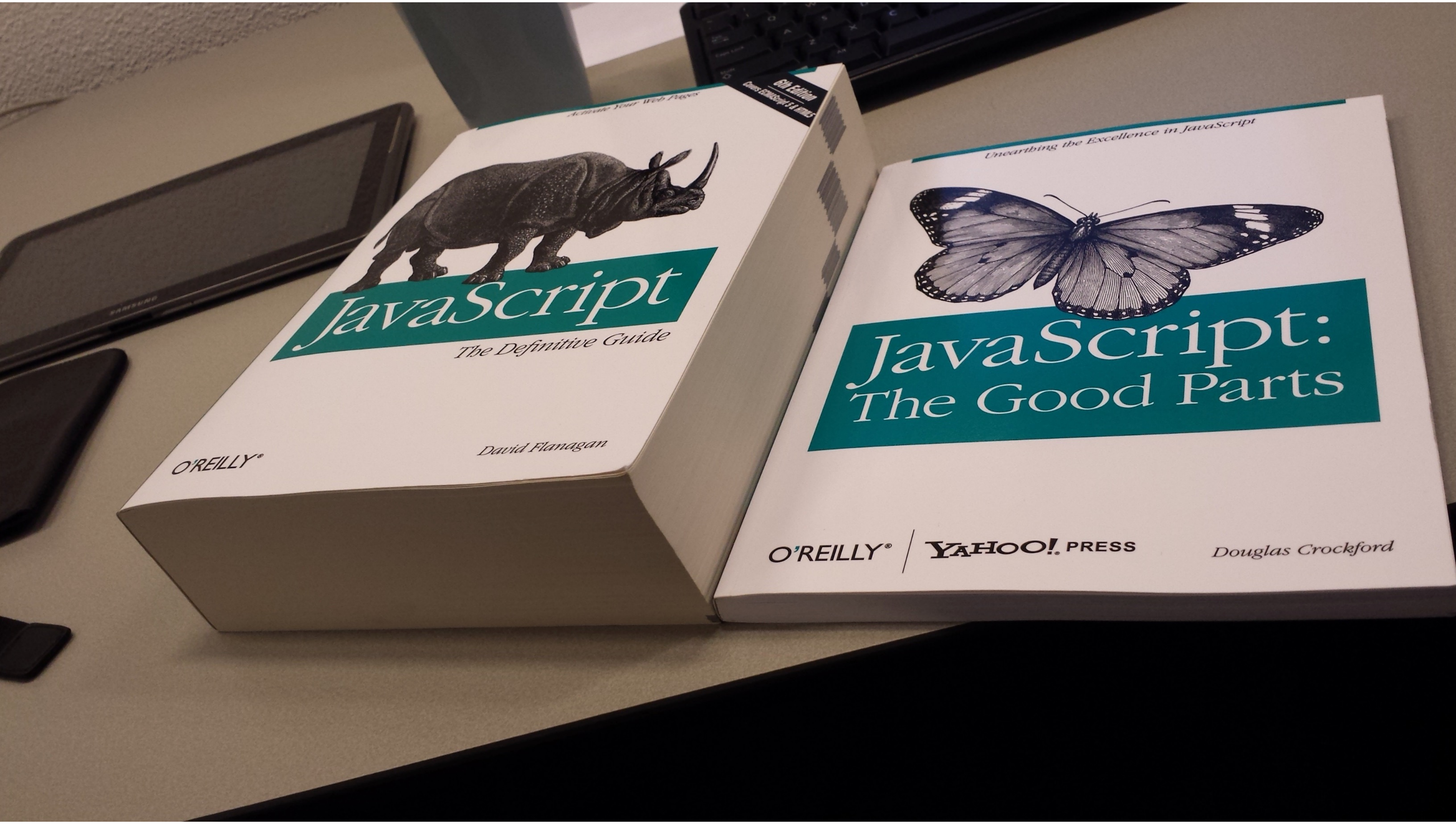
These are the books we love, offered at Amazon.com low prices. The spotlight moves EVERY day so please come often.

ONE MILLION TITLES

Search Amazon.com's million title catalog by author, subject, title, keyword, and more... Or take a look at the books we recommend in over 20 categories... Check out our customer reviews and the award winners from the Hugo and Nebula to the Pulitzer and Nobel... and bestsellers are 30% off the publishers list...

Interplay between technologies

- Early ‘90s: e-commerce becomes popular
- April 1995: at Netscape, Brendan Eich designs **JavaScript**



Activate Your Web Pages
6th Edition
Covers ECMAScript 5 & HTML5
JavaScript
The Definitive Guide
O'REILLY®
David Flanagan
Unearthing the Excellence in JavaScript
JavaScript:
The Good Parts
O'REILLY®
YAHOO! PRESS
Douglas Crockford

WAT

JavaScript: let’s Figure out the Semantics

Sergio Maffeis, John C. Mitchell, and Ankur Taly. 2008. **An Operational Semantics for JavaScript.** In Proceedings of the 6th Asian Symposium on Programming Languages and Systems (APLAS '08), G. Ramalingam (Ed.). Springer-Verlag, Berlin, Heidelberg, 307-325.

Arjun Guha, Claudiu Saftoiu, and Shriram Krishnamurthi. 2010. **The essence of javascript**. In Proceedings of the 24th European conference on Object-oriented programming (ECOOP'10), Theo D'Hondt (Ed.). Springer-Verlag, Berlin, Heidelberg, 126-150.

Daejun Park, Andrei Stefănescu, and Grigore Roşu. 2015. **KJS: a complete formal semantics of JavaScript.** In Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '15). ACM, New York, NY, USA, 346-356.

…

Why don’t we all use the same language?

Why don’t we all use the same language?

Which one?

Uh… What about the most common?

Why don’t we all speak Chinese?

What is the best language?

No clear answer (what’s the “best” human language!?)

Some languages are very effective for certain tasks

- E.g., Python for data science

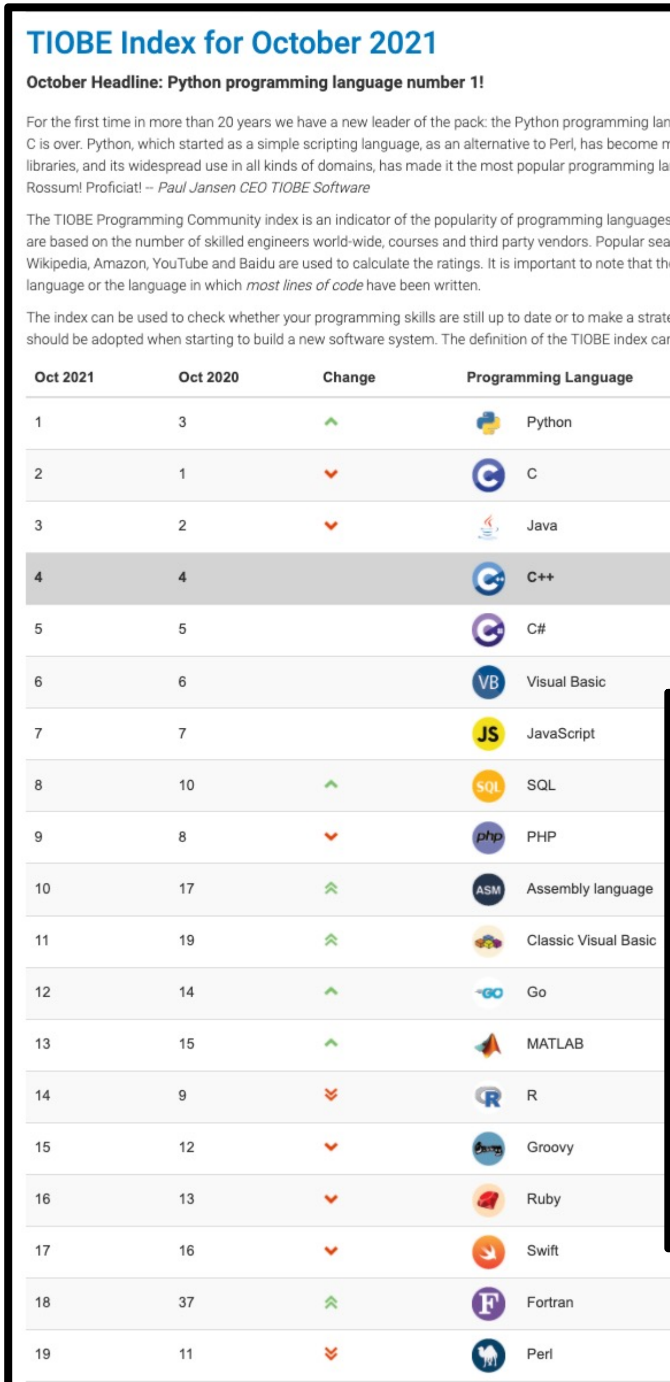
How many languages exist?

Wikipedia lists 700 “recognized” languages

In practice, many more. Could be ~10.000

What is a ‘new’ language anyway?

For comparison, ~6000 human languages



TIOBE Index for October 2021

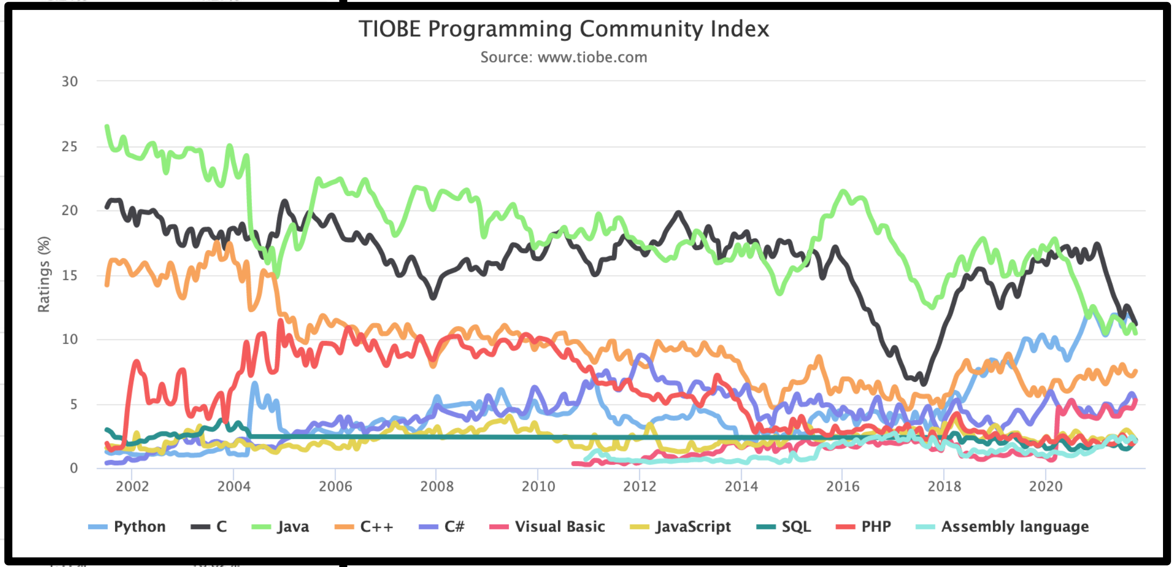
October Headline: Python programming language number 1!

For the first time in more than 20 years we have a new leader of the pack: the Python programming language. The long-standing hegemony of Java and C is over. Python, which started as a simple scripting language, as an alternative to Perl, has become mature. Its ease of learning, its huge amount of libraries, and its widespread use in all kinds of domains, has made it the most popular programming language of today. Congratulations Guido van Rossum! Proficiat! -- *Paul Jansen CEO TIOBE Software*

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

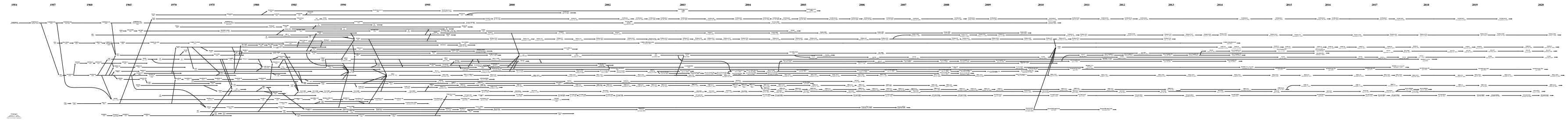
The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

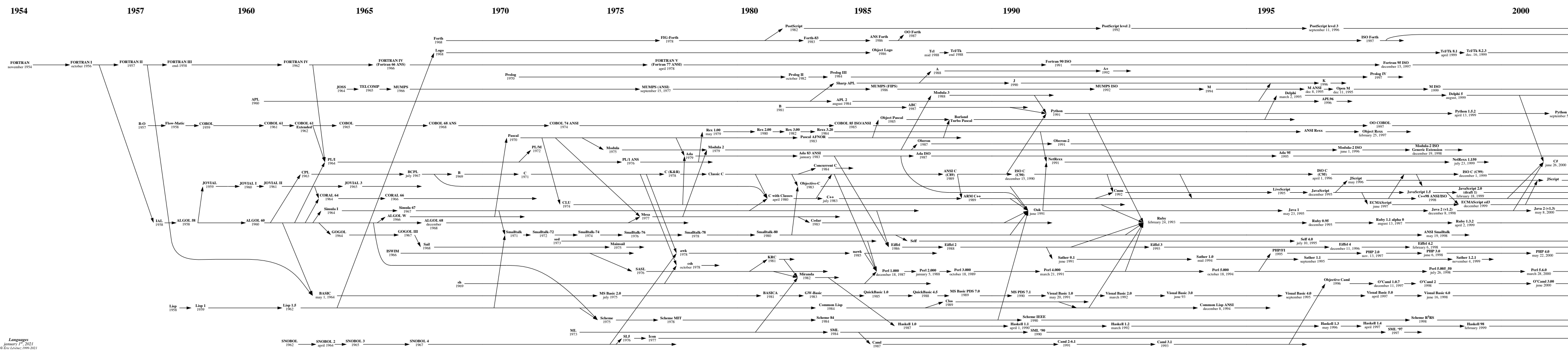
Oct 2021	Oct 2020	Change	Programming Language	Ratings	Change
1	3	⌃	Python	11.27%	-0.00%
2	1	⌄	C	11.16%	-5.79%
3	2	⌄	Java	10.46%	-2.11%
4	**4**		**C++**	**7.50%**	**+0.57%**
5	5		C#	5.26%	+1.10%
6	6		Visual Basic	5.24%	+1.27%
7	7		JavaScript		
8	10	⌃	SQL		
9	8	⌄	PHP		
10	17	⌃⌃	Assembly language		
11	19	⌃⌃	Classic Visual Basic		
12	14	⌃	Go		
13	15	⌃	MATLAB		
14	9	⌄⌄	R		
15	12	⌄	Groovy		
16	13	⌄	Ruby		
17	16	⌄	Swift	1.11%	+0.02%
18	37	⌃⌃	Fortran	1.08%	+0.70%
19	11	⌄⌄	Perl	0.93%	-0.49%



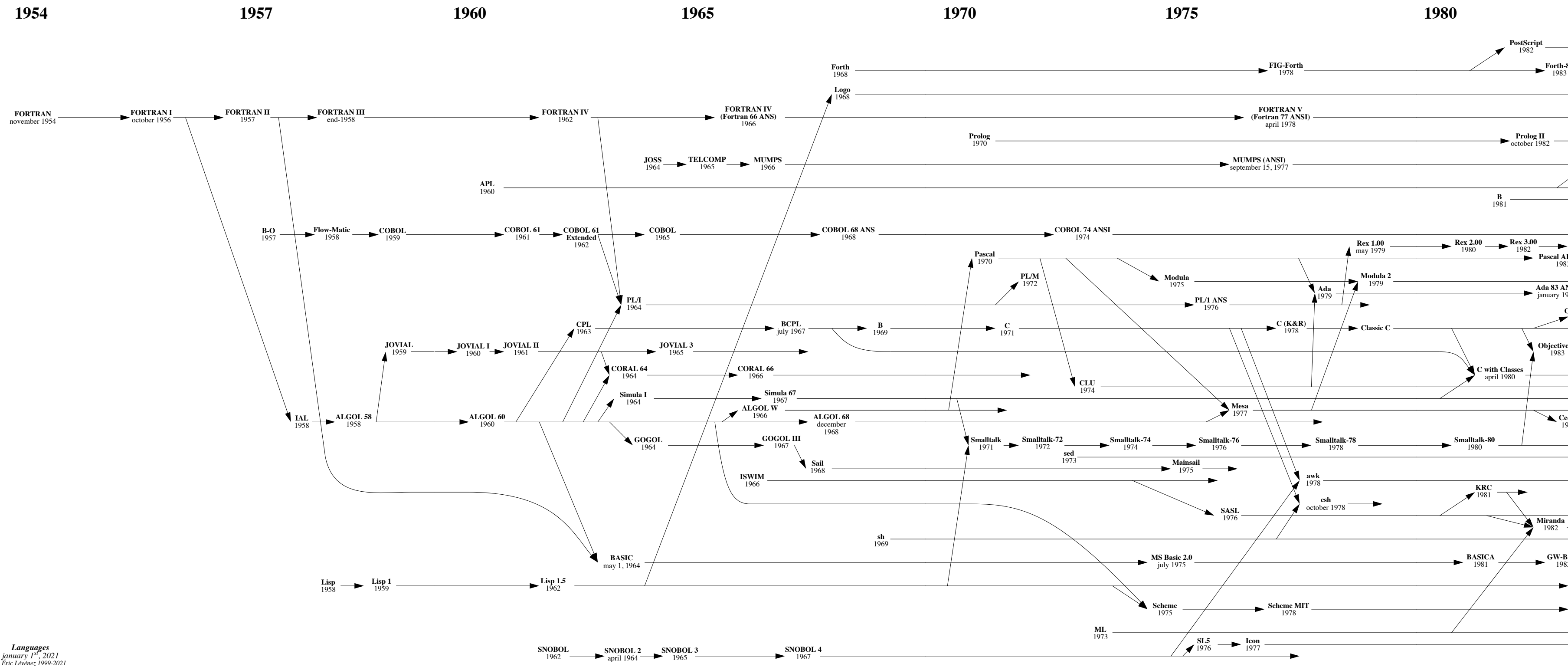
TIOBE Programming Community Index

Source: www.tiobe.com

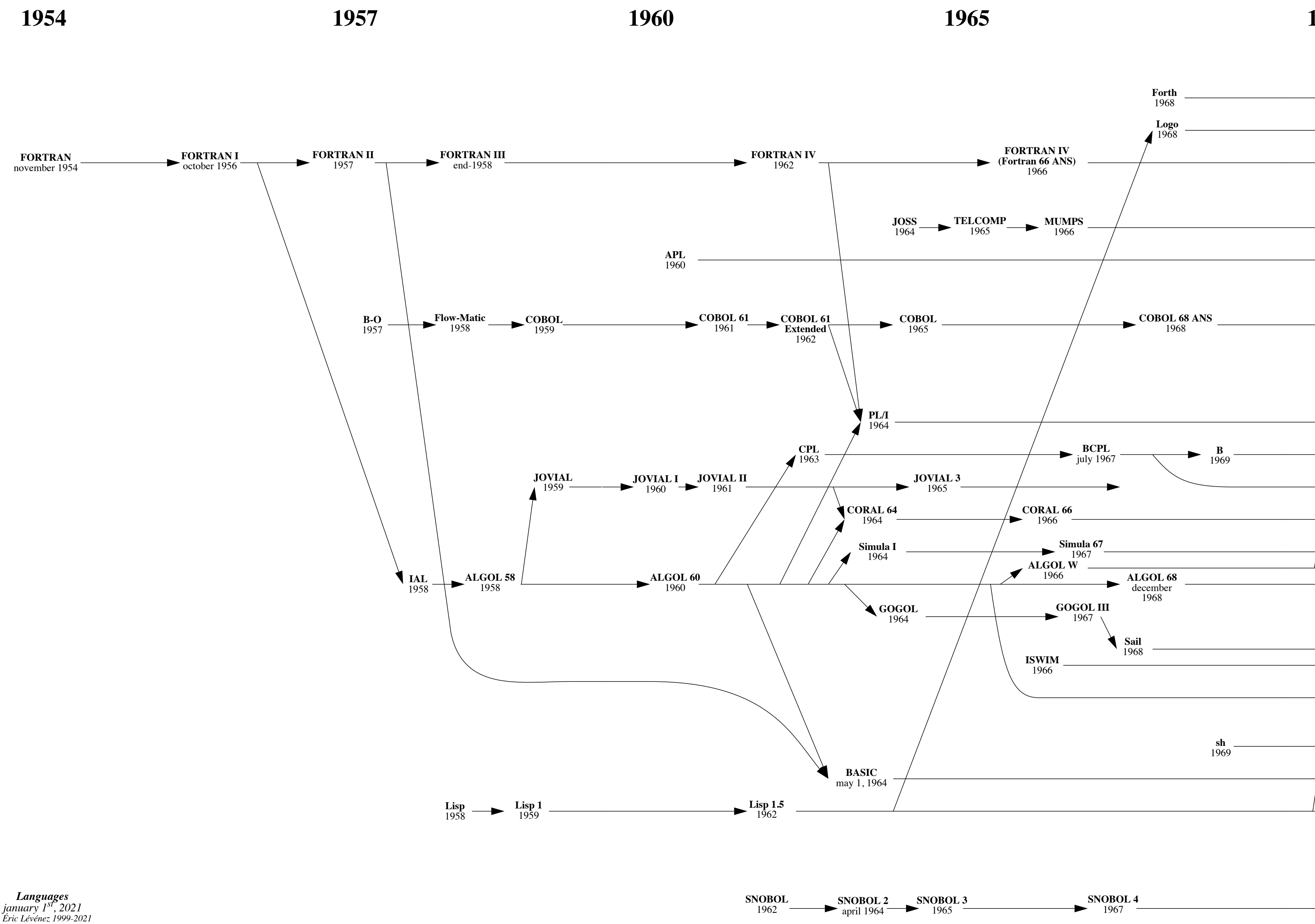

1954
1957
1960
1965
1970
1975
1980
1985
1990
1995
2000
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020



Languages
january 1st, 2021
© Éric Lévénez 1999-2021
<http://www.levenez.com/lang/>



Languages
january 1st, 2021
© Éric Lévénez 1999-2021
<http://www.levenez.com/lang/>



1954 1957 1960 1965

Forth 1968

Logo 1968

FORTRAN november 1954 → FORTRAN I october 1956 → FORTRAN II 1957 → FORTRAN III end-1958 → FORTRAN IV 1962 → FORTRAN IV (Fortran 66 ANS) 1966

JOSS 1964 → TELCOMP 1965 → MUMPS 1966

APL 1960

B-O 1957 → Flow-Matic 1958 → COBOL 1959 → COBOL 61 1961 → COBOL 61 Extended 1962 → COBOL 1965 → COBOL 68 ANS 1968

PL/I 1964

CPL 1963 → BCPL july 1967 → B 1969

JOVIAL 1959 → JOVIAL I 1960 → JOVIAL II 1961 → JOVIAL 3 1965

CORAL 64 1964 → CORAL 66 1966

Simula I 1964 → Simula 67 1967

ALGOL W 1966

IAL 1958 → ALGOL 58 1958 → ALGOL 60 1960 → ALGOL 68 december 1968

GOGOL 1964 → GOGOL III 1967 → Sail 1968

ISWIM 1966

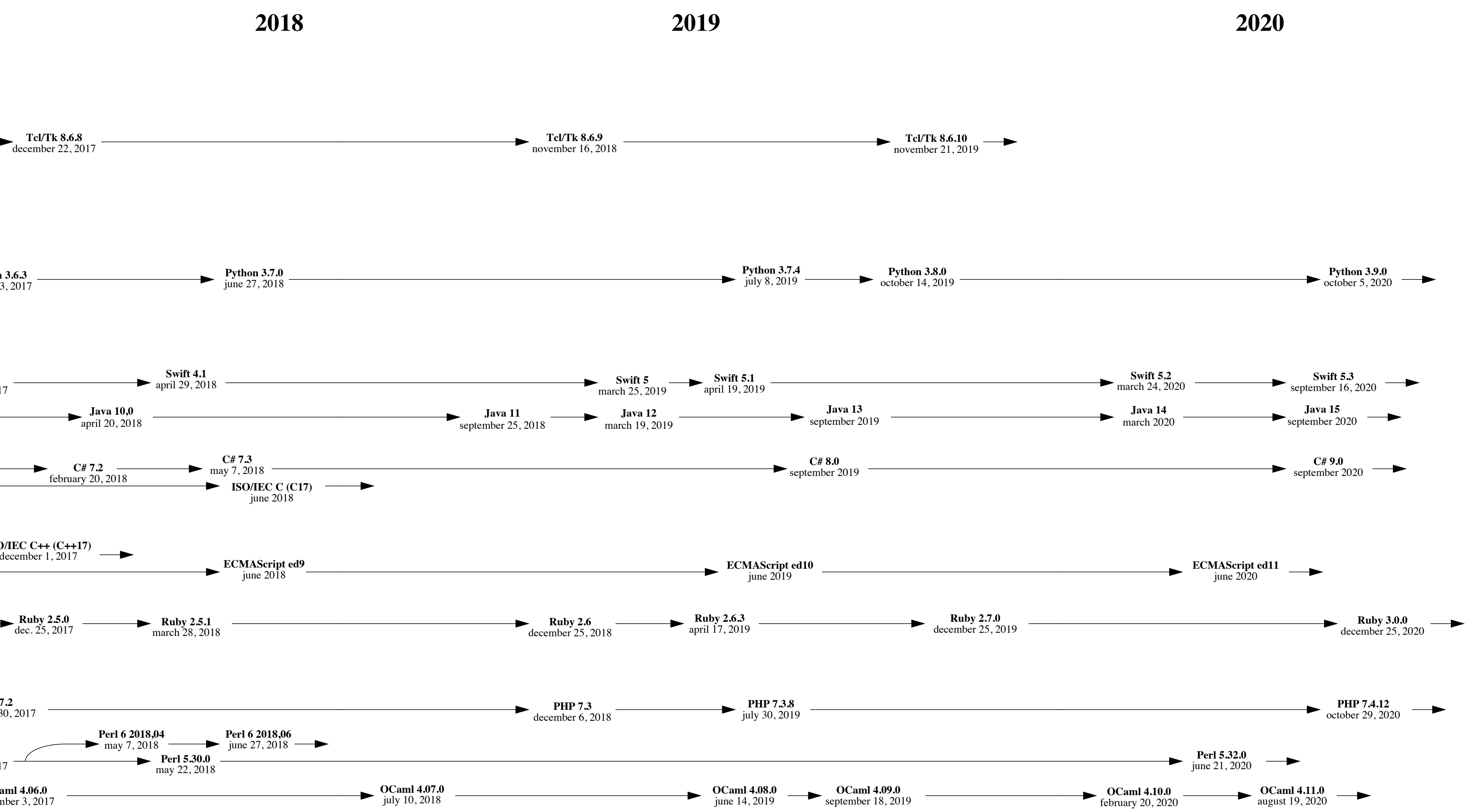
sh 1969

BASIC may 1, 1964

Lisp 1958 → Lisp 1 1959 → Lisp 1.5 1962

SNOBOL 1962 → SNOBOL 2 april 1964 → SNOBOL 3 1965 → SNOBOL 4 1967

Languages
january 1st, 2021
© Éric Lévénez 1999-2021
<http://www.levenez.com/lang/>



2018

2019

2020

Tcl/Tk 8.6.8
december 22, 2017

Tcl/Tk 8.6.9
november 16, 2018

Tcl/Tk 8.6.10
november 21, 2019

n 3.6.3
3, 2017

Python 3.7.0
june 27, 2018

Python 3.7.4
july 8, 2019

Python 3.8.0
october 14, 2019

Python 3.9.0
october 5, 2020

17

Swift 4.1
april 29, 2018

Swift 5
march 25, 2019

Swift 5.1
april 19, 2019

Swift 5.2
march 24, 2020

Swift 5.3
september 16, 2020

Java 10,0
april 20, 2018

Java 11
september 25, 2018

Java 12
march 19, 2019

Java 13
september 2019

Java 14
march 2020

Java 15
september 2020

C# 7.2
february 20, 2018

C# 7.3
may 7, 2018

C# 8.0
september 2019

C# 9.0
september 2020

ISO/IEC C (C17)
june 2018

D/IEC C++ (C++17)
december 1, 2017

ECMAScript ed9
june 2018

ECMAScript ed10
june 2019

ECMAScript ed11
june 2020

Ruby 2.5.0
dec. 25, 2017

Ruby 2.5.1
march 28, 2018

Ruby 2.6
december 25, 2018

Ruby 2.6.3
april 17, 2019

Ruby 2.7.0
december 25, 2019

Ruby 3.0.0
december 25, 2020

7.2
30, 2017

PHP 7.3
december 6, 2018

PHP 7.3.8
july 30, 2019

PHP 7.4.12
october 29, 2020

Perl 6 2018,04
may 7, 2018

Perl 6 2018,06
june 27, 2018

17

Perl 5.30.0
may 22, 2018

Perl 5.32.0
june 21, 2020

aml 4.06.0
mber 3, 2017

OCaml 4.07.0
july 10, 2018

OCaml 4.08.0
june 14, 2019

OCaml 4.09.0
september 18, 2019

OCaml 4.10.0
february 20, 2020

OCaml 4.11.0
august 19, 2020

Why don’t we add all possible concepts to a single language?

Chinese characters + German grammar +
UK humor + Latin declinations + …

@bogartcreek

DOES IT MATTER?

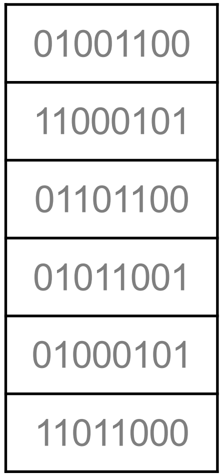
When Abstraction Goes Wrong

```
addNumbers() {

    x = 2;
    y = 5;

    z = x + y;
}
```

1010100010100010

When Abstraction Goes Wrong

```
addNumbers() {

    x = 2147483647;
    y = 5;

    z = x + y;
}
```



01001100

11000101

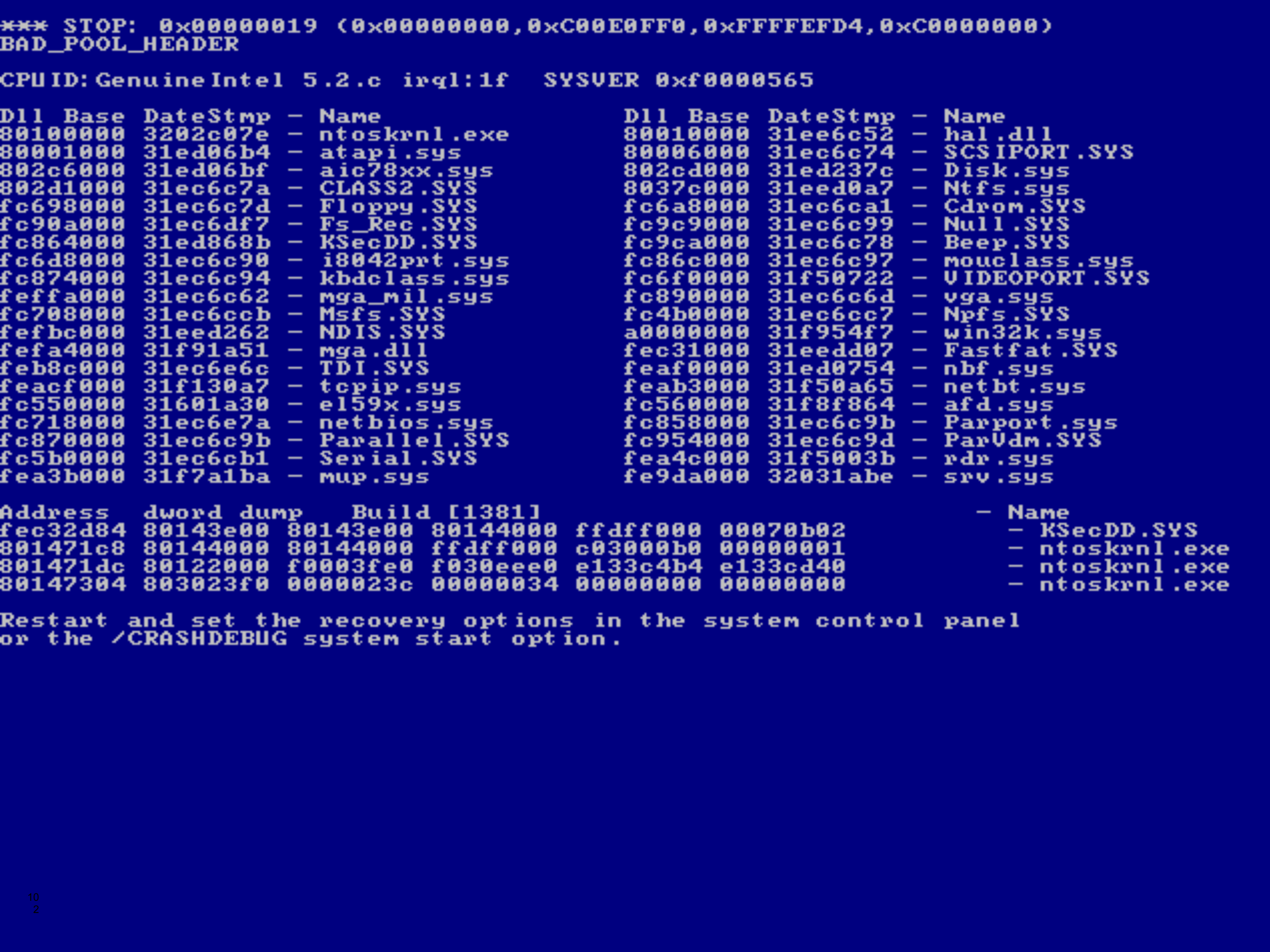
01101100

01011001

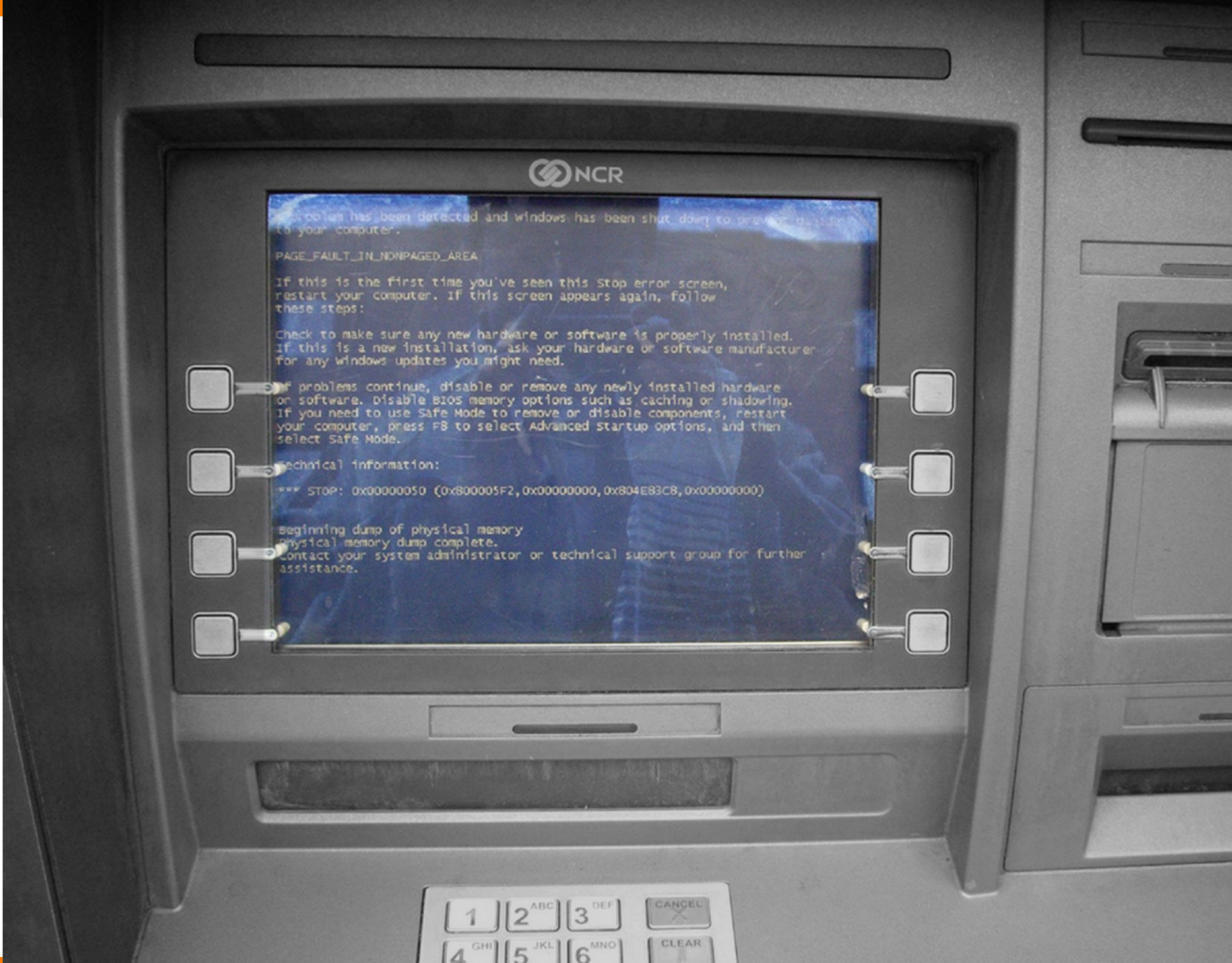
01000101

11011000

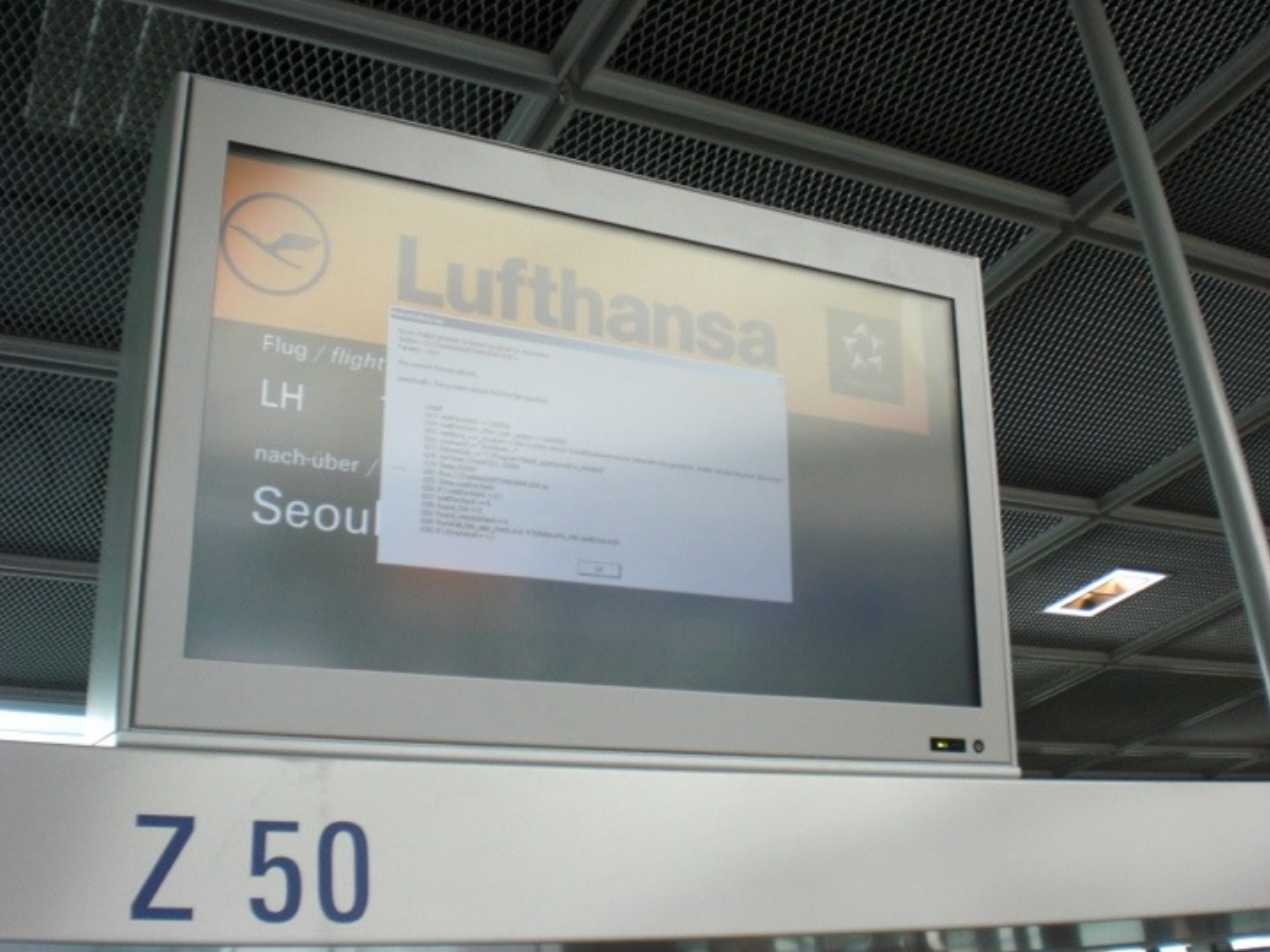
Abstraction Gone Wrong: The Ariane 5



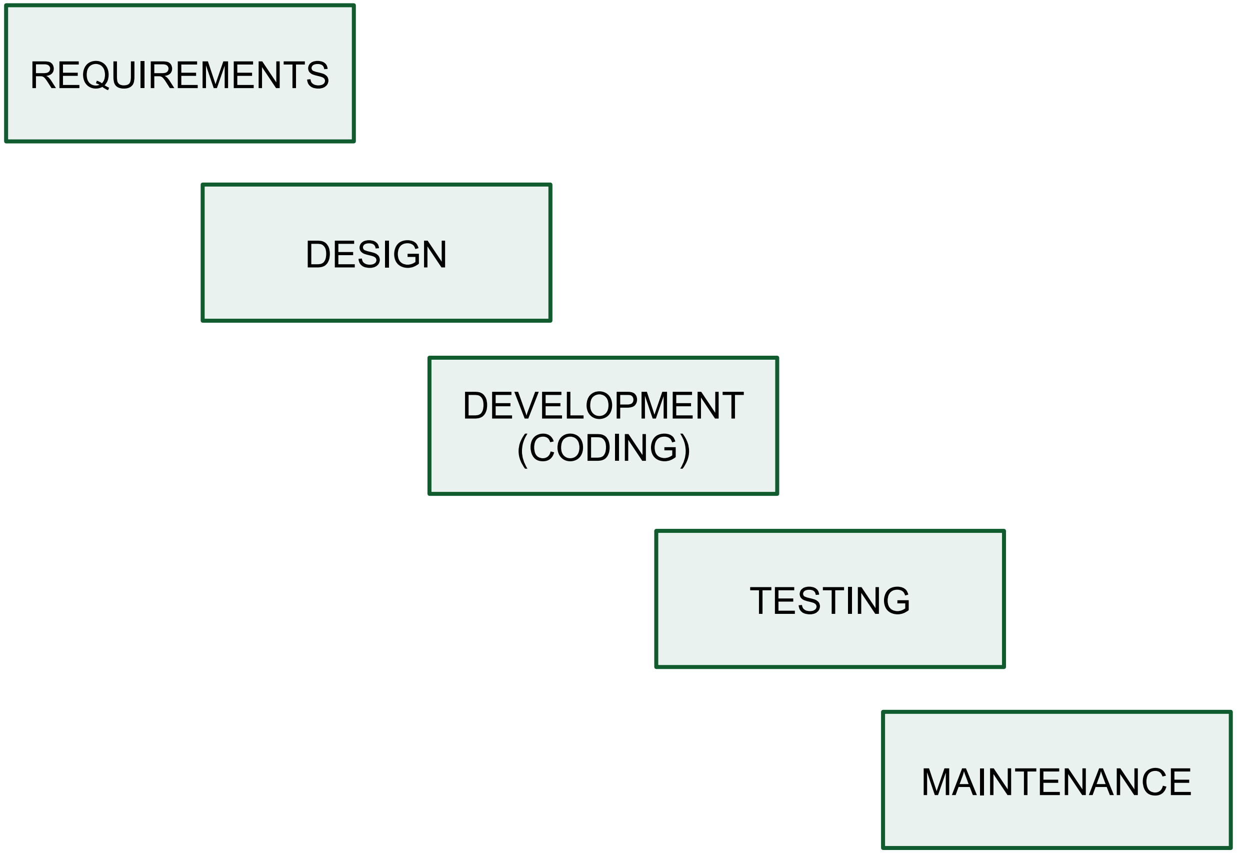
```
*** STOP: 0x00000019 (0x00000000,0xC00E0FF0,0xFFFFEFD4,0xC0000000)
BAD_POOL_HEADER

CPUID:GenuineIntel 5.2.c irql:1f  SYSVER 0xf0000565

Dll Base DateStmp - Name                Dll Base DateStmp - Name
80100000 3202c07e - ntoskrnl.exe        80010000 31ee6c52 - hal.dll
80001000 31ed06b4 - atapi.sys           80006000 31ec6c74 - SCSIPORT.SYS
802c6000 31ed06bf - aic78xx.sys         802cd000 31ed237c - Disk.sys
802d1000 31ec6c7a - CLASS2.SYS          8037c000 31eed0a7 - Ntfs.sys
fc698000 31ec6c7d - Floppy.SYS          fc6a8000 31ec6ca1 - Cdrom.SYS
fc90a000 31ec6df7 - Fs_Rec.SYS          fc9c9000 31ec6c99 - Null.SYS
fc864000 31ed868b - KSecDD.SYS          fc9ca000 31ec6c78 - Beep.SYS
fc6d8000 31ec6c90 - i8042prt.sys        fc86c000 31ec6c97 - mouclass.sys
fc874000 31ec6c94 - kbdclass.sys        fc6f0000 31f50722 - VIDEOPORT.SYS
feffa000 31ec6c62 - mga_mil.sys         fc890000 31ec6c6d - vga.sys
fc708000 31ec6ccb - Msfs.SYS            fc4b0000 31ec6cc7 - Npfs.SYS
fefbc000 31eed262 - NDIS.SYS            a0000000 31f954f7 - win32k.sys
fefa4000 31f91a51 - mga.dll             fec31000 31eedd07 - Fastfat.SYS
feb8c000 31ec6e6c - TDI.SYS             feaf0000 31ed0754 - nbf.sys
feacf000 31f130a7 - tcpip.sys           feab3000 31f50a65 - netbt.sys
fc550000 31601a30 - el59x.sys           fc560000 31f8f864 - afd.sys
fc718000 31ec6e7a - netbios.sys         fc858000 31ec6c9b - Parport.sys
fc870000 31ec6c9b - Parallel.SYS        fc954000 31ec6c9d - ParVdm.SYS
fc5b0000 31ec6cb1 - Serial.SYS          fea4c000 31f5003b - rdr.sys
fea3b000 31f7a1ba - mup.sys             fe9da000 32031abe - srv.sys

Address  dword dump   Build [1381]                            - Name
fec32d84 80143e00 80143e00 80144000 ffdff000 00070b02         - KSecDD.SYS
801471c8 80144000 80144000 ffdff000 c03000b0 00000001         - ntoskrnl.exe
801471dc 80122000 f0003fe0 f030eee0 e133c4b4 e133cd40         - ntoskrnl.exe
80147304 803023f0 0000023c 00000034 00000000 00000000         - ntoskrnl.exe

Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option.
```



```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0x800005F2,0x00000000,0x804E83C8,0x00000000)


Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```


Lufthansa
Flug / flight
LH
nach-über /
Seoul
OK
Z 50

The Software Engineering Process



REQUIREMENTS

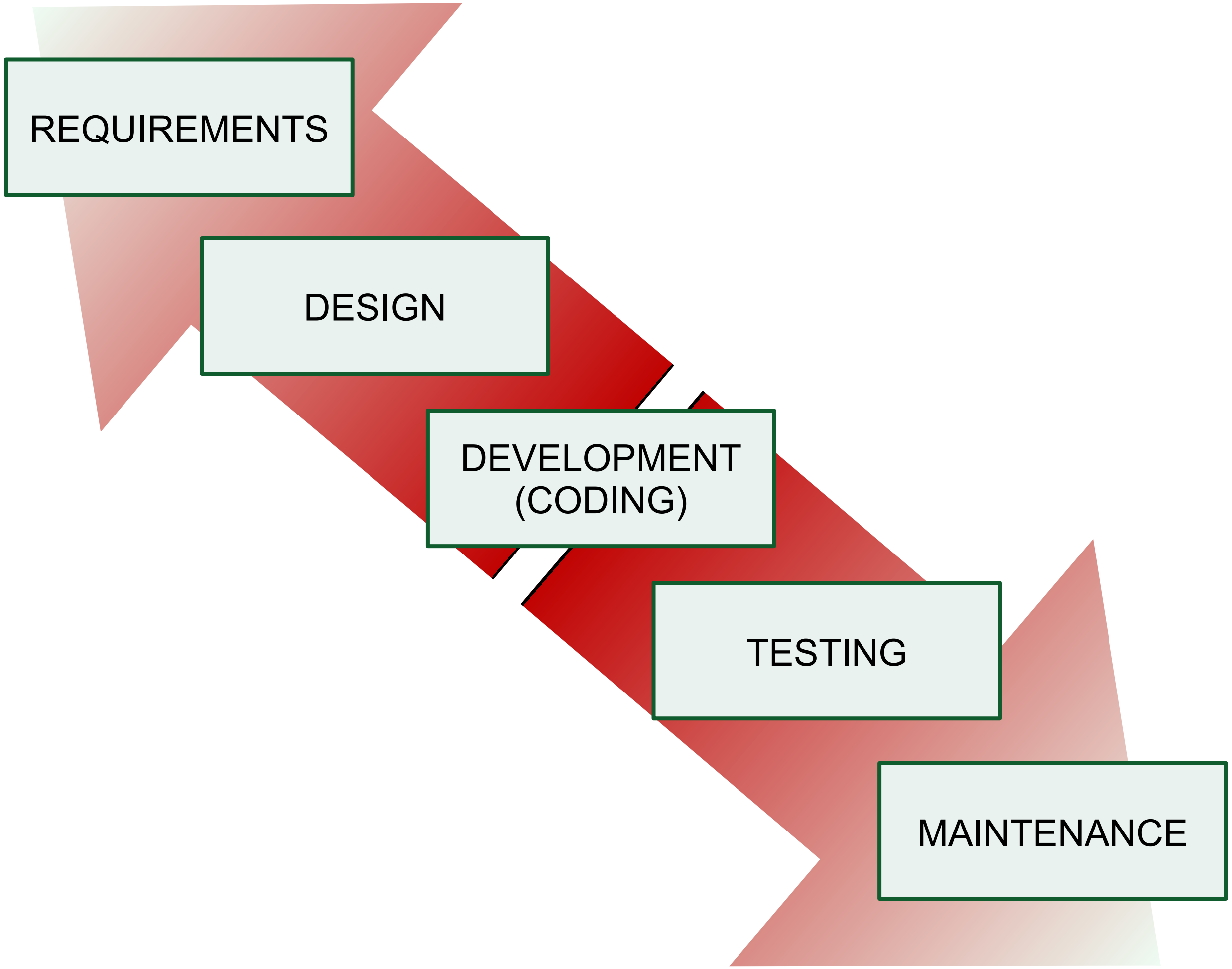
DESIGN

DEVELOPMENT (CODING)

TESTING

MAINTENANCE

The Software Engineering Process



REQUIREMENTS

DESIGN

DEVELOPMENT
(CODING)

TESTING

MAINTENANCE

Ferguson’s Covid 19 Epidemic Model

“From tomorrow, if you have coronavirus symptoms, however mild […] you should stay at home for at least 7 days to protect others […]”

Boris Johnson, March 12th, 2020

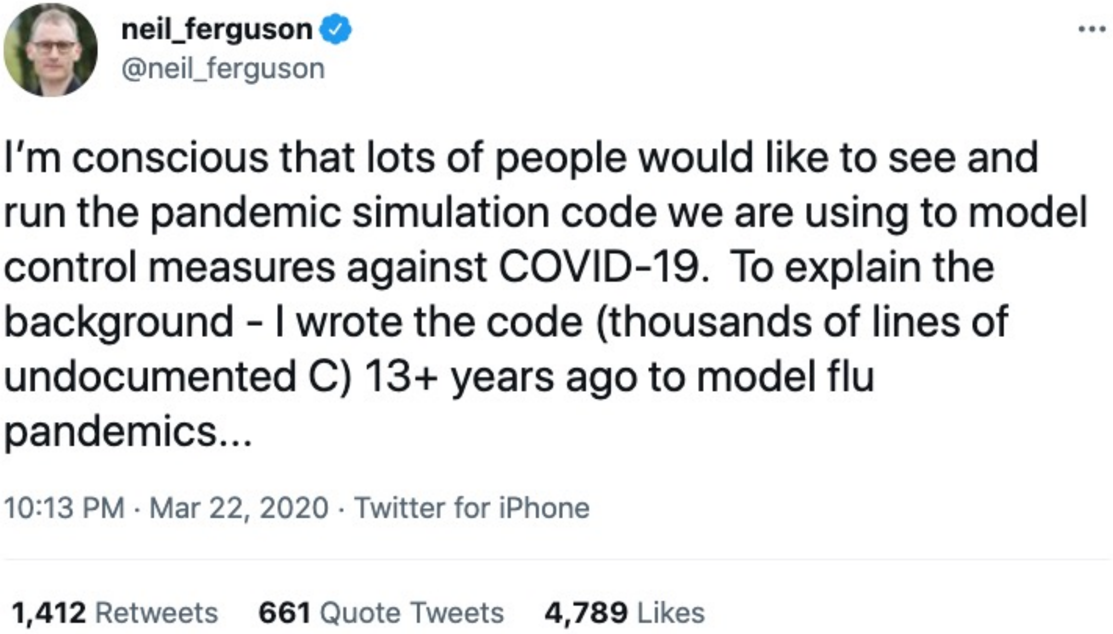
Ferguson’s Covid 19 Epidemic Model

“From tomorrow, if you have coronavirus symptoms, however mild […] you should stay at home for at least 7 days to protect others […]”

Boris Johnson, March 12th, 2020

March 16th: Imperial team’s model, released

- With no countermeasures, UK’s health service overwhelmed.
- UK: As many as 500,000 deaths
- United States might face 2.2 million deaths.

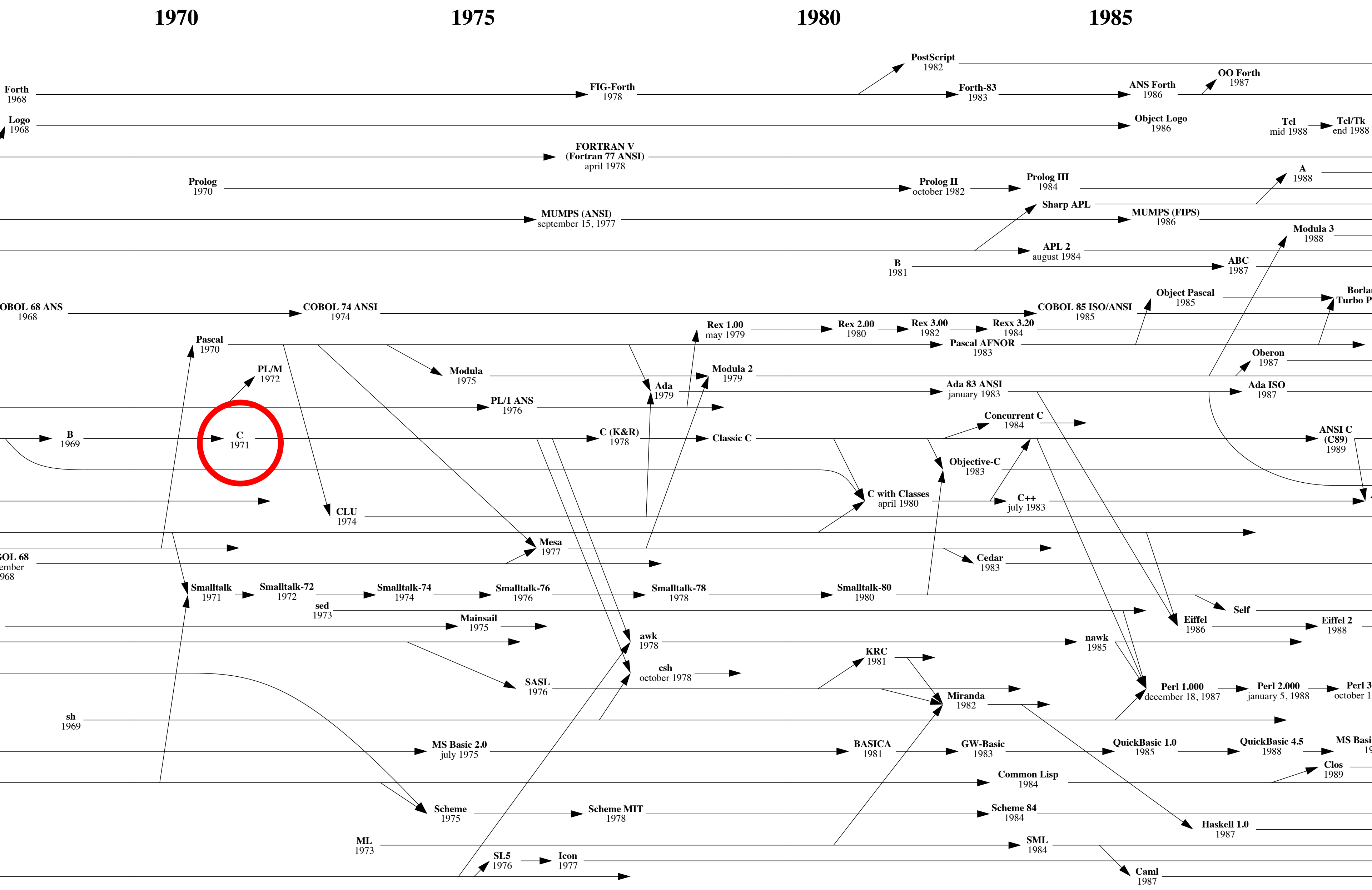


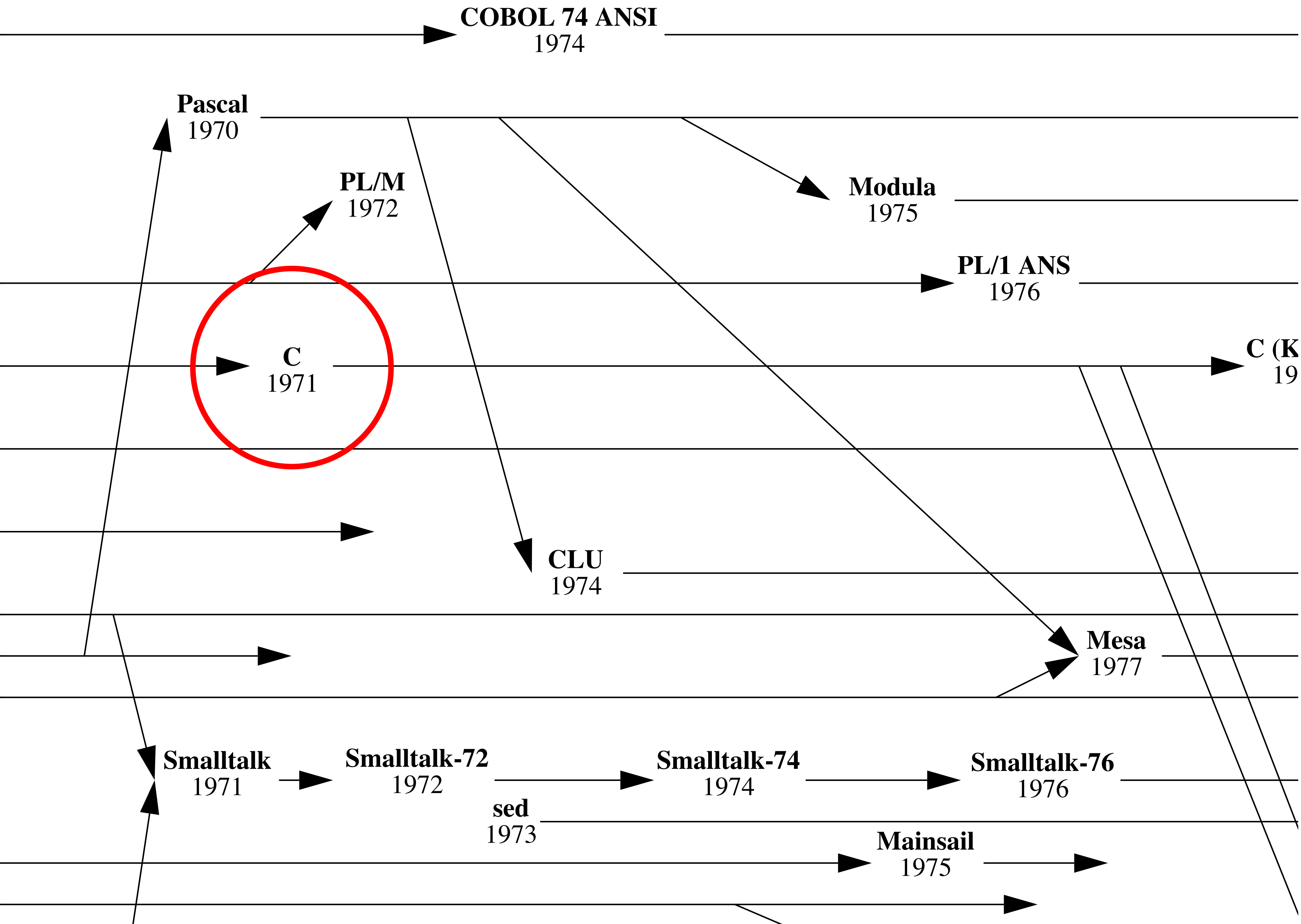
neil_ferguson
@neil_ferguson

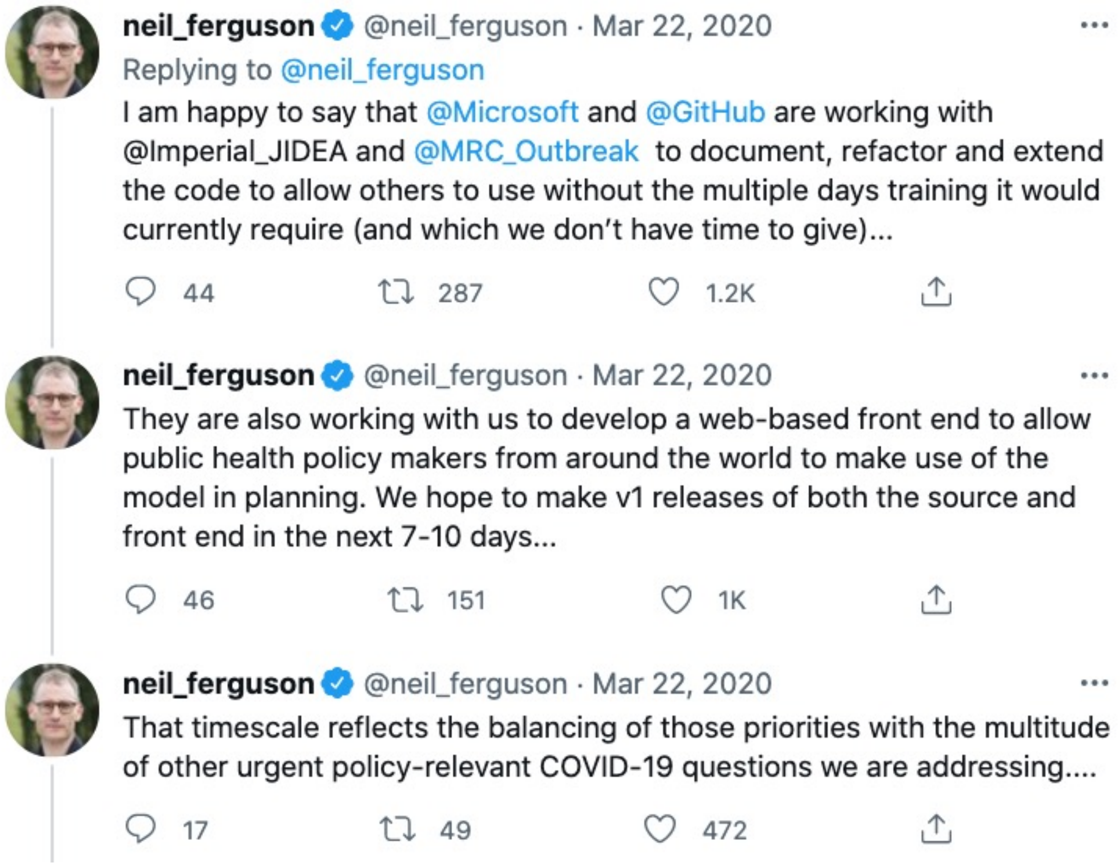
I’m conscious that lots of people would like to see and run the pandemic simulation code we are using to model control measures against COVID-19. To explain the background - I wrote the code (thousands of lines of undocumented C) 13+ years ago to model flu pandemics...

10:13 PM · Mar 22, 2020 · Twitter for iPhone

1,412 Retweets **661** Quote Tweets **4,789** Likes


1970
1975
1980
1985
PostScript 1982
Forth 1968
FIG-Forth 1978
Forth-83 1983
ANS Forth 1986
OO Forth 1987
Logo 1968
Object Logo 1986
Tcl mid 1988
Tcl/Tk end 1988
FORTRAN V (Fortran 77 ANSI) april 1978
Prolog 1970
Prolog II october 1982
Prolog III 1984
A 1988
Sharp APL
MUMPS (ANSI) september 15, 1977
MUMPS (FIPS) 1986
Modula 3 1988
APL 2 august 1984
B 1981
ABC 1987
Object Pascal 1985
Borlan
Turbo P
OBOL 68 ANS 1968
COBOL 74 ANSI 1974
COBOL 85 ISO/ANSI 1985
Rex 1.00 may 1979
Rex 2.00 1980
Rex 3.00 1982
Rexx 3.20 1984
Pascal 1970
Pascal AFNOR 1983
Oberon 1987
PL/M 1972
Modula 1975
Modula 2 1979
Ada 1979
Ada 83 ANSI january 1983
Ada ISO 1987
PL/1 ANS 1976
Concurrent C 1984
B 1969
C 1971
C (K&R) 1978
Classic C
ANSI C (C89) 1989
Objective-C 1983
C with Classes april 1980
C++ july 1983
CLU 1974
Mesa 1977
OL 68 ember 968
Cedar 1983
Smalltalk 1971
Smalltalk-72 1972
Smalltalk-74 1974
Smalltalk-76 1976
Smalltalk-78 1978
Smalltalk-80 1980
sed 1973
Self
Mainsail 1975
Eiffel 1986
Eiffel 2 1988
awk 1978
nawk 1985
KRC 1981
csh october 1978
SASL 1976
Miranda 1982
Perl 1.000 december 18, 1987
Perl 2.000 january 5, 1988
Perl 3 october 1
sh 1969
MS Basic 2.0 july 1975
BASICA 1981
GW-Basic 1983
QuickBasic 1.0 1985
QuickBasic 4.5 1988
MS Basi 19
Clos 1989
Common Lisp 1984
Scheme 1975
Scheme MIT 1978
Scheme 84 1984
Haskell 1.0 1987
ML 1973
SML 1984
SL5 1976
Icon 1977
Caml 1987


COBOL 74 ANSI
1974
Pascal
1970
PL/M
1972
Modula
1975
PL/1 ANS
1976
C
1971
C (K
19
CLU
1974
Mesa
1977
Smalltalk
1971
Smalltalk-72
1972
Smalltalk-74
1974
Smalltalk-76
1976
sed
1973
Mainsail
1975



neil_ferguson @neil_ferguson · Mar 22, 2020

Replying to @neil_ferguson

I am happy to say that @Microsoft and @GitHub are working with @Imperial_JIDEA and @MRC_Outbreak to document, refactor and extend the code to allow others to use without the multiple days training it would currently require (and which we don't have time to give)...

neil_ferguson @neil_ferguson · Mar 22, 2020

They are also working with us to develop a web-based front end to allow public health policy makers from around the world to make use of the model in planning. We hope to make v1 releases of both the source and front end in the next 7-10 days...

neil_ferguson @neil_ferguson · Mar 22, 2020

That timescale reflects the balancing of those priorities with the multitude of other urgent policy-relevant COVID-19 questions we are addressing....