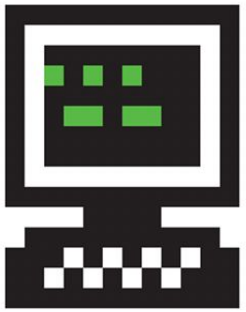
Programming across Paradigms

@AnjanaVakil

GOTO Chicago 2017

hi, I'm @AnjanaVakil!

Über Research



The **Recurse** Center

moz://a

OUTREACHY

The Paradigms of Programming

Robert W. Floyd
Stanford University

Paradigm(pæ·radim, −dɔim) . . . [a. F. *paradigme*, ad. L. *paradigma*, a. Gr. παραδειγμα pattern, example, f. παραδεικνυ·ναι to exhibit beside, show side by side. . .]
1. A pattern, exemplar, example.
1752 J. Gill *Trinity* v. 91
The archetype, paradigm, exemplar, and idea, according to which all things were made.
From the Oxford English Dictionary.

Today I want to talk about the paradigms of programming, how they affect our success as designers of computer programs, how they should be taught, and how they should be embodied in our programming languages.

A familiar example of a paradigm of programming is the technique of *structured programming*, which appears to be the dominant paradigm in most current treatments of programming methodology. Structured programming, as formulated by Dijkstra [6], Wirth [27, 29], and Parnas [21], among others, consists of two phases.

In the first phase, that of top-down design, or stepwise refinement, the problem is decomposed into a very small number of simpler subproblems. In programming the solution of simultaneous linear equations, say, the first level of decomposition would be into a stage of triangularizing the equations and a following stage of back-substitution in the triangularized system. This gradual decomposition is continued until the subproblems that arise are simple enough to cope with directly. In the simultaneous equation example, the back substitution process would be further decomposed as a backwards iteration of a process which finds and stores the value of the *i*th variable from the *i*th equation. Yet further decomposition would yield a fully detailed algorithm.

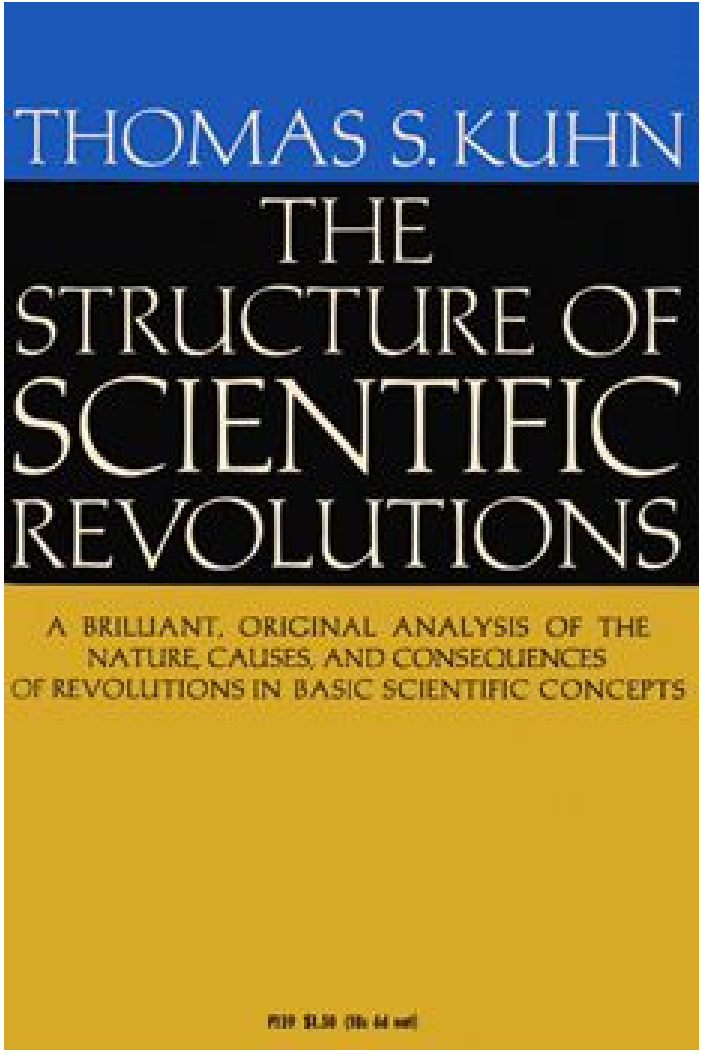
1978 ACM Turing Award Lecture

“I believe the best chance we have to improve the general practice of programming is to attend to our paradigms.”

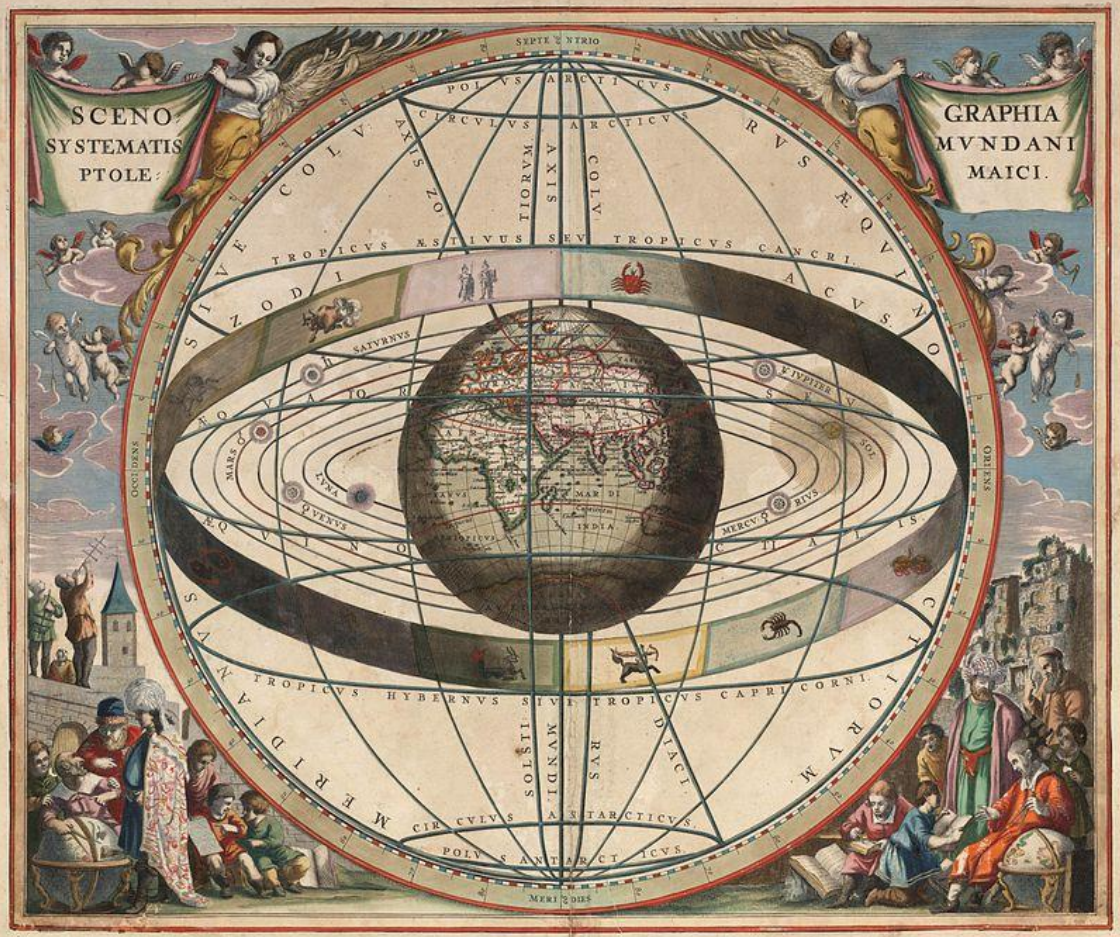
Robert W. Floyd

“The paradigms of programming.”, 1979. p. 456

what is a paradigm?


THOMAS S. KUHN
THE STRUCTURE OF SCIENTIFIC REVOLUTIONS
A BRILLIANT, ORIGINAL ANALYSIS OF THE NATURE, CAUSES, AND CONSEQUENCES OF REVOLUTIONS IN BASIC SCIENTIFIC CONCEPTS

a paradigm is a worldview


SCENO
SYSTEMATIS
PTOLE:
GRAPHIA
MVNDANI
MAICI.
SEPTENTRIO
POLVS ARCTICVS
CIRCVLVS ARCTICVS
AXIS ZODIACI SIVE COLVRVM SOLSTITIORVM
TROPICVS ESTIVVS SEV CANCRI.
ZODIACVS.
SATVRNVS
IVPITER
SOL
LVNA
MARS
MERCVRIVS
VENVS
AEQVINOCTIALIS.
MARE
INDIA
TROPICVS HYBERNVS SIVE CAPRICORNI.
MERIDIANVS
CIRCVLVS ANTARCTICVS
POLVS ANTARCTICVS
MERIDIES
OCCIDENS
ORIENS

a paradigm is a model

a paradigm enables progress

“In learning a paradigm the scientist acquires theory, methods, and standards together, usually in an inextricable mixture.”

Thomas S. Kuhn

The Structure of Scientific Revolutions, (2nd ed.) 1970. p. 109.

theory

what entities make up the universe
how they behave and interact

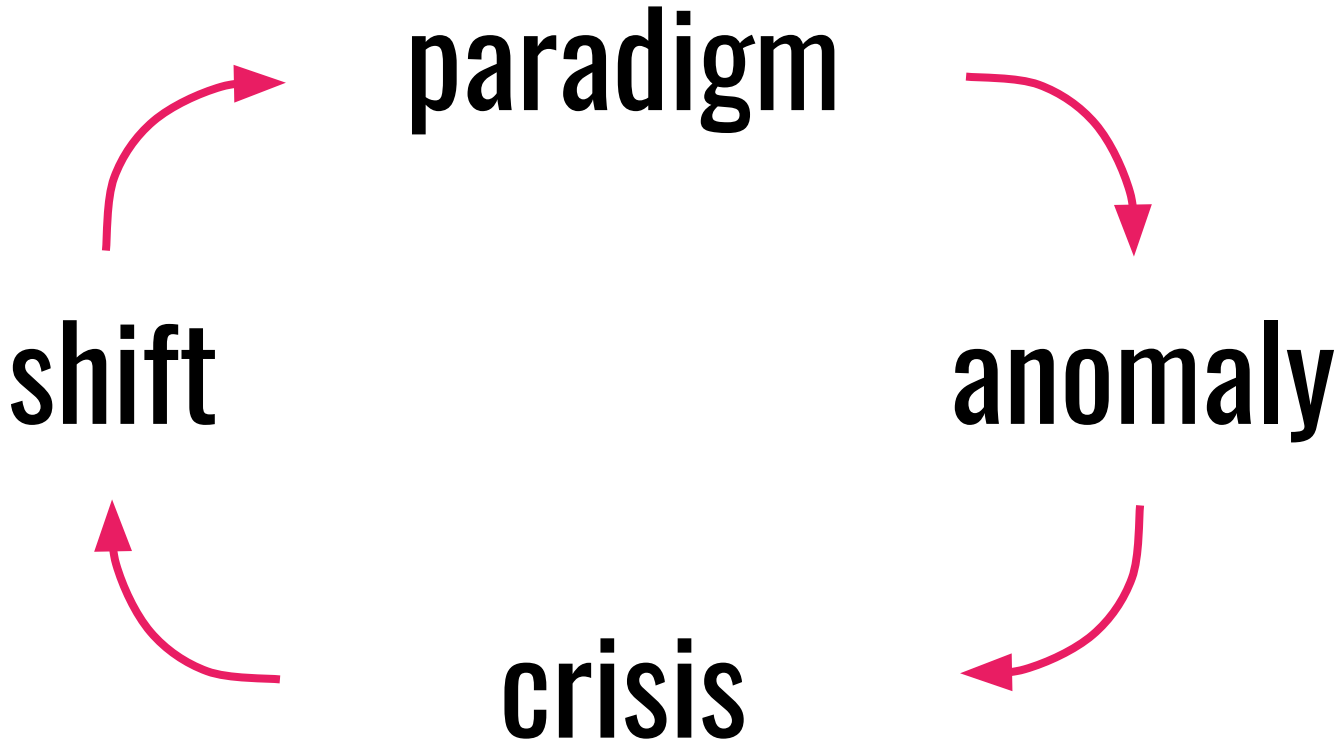
methods & standards

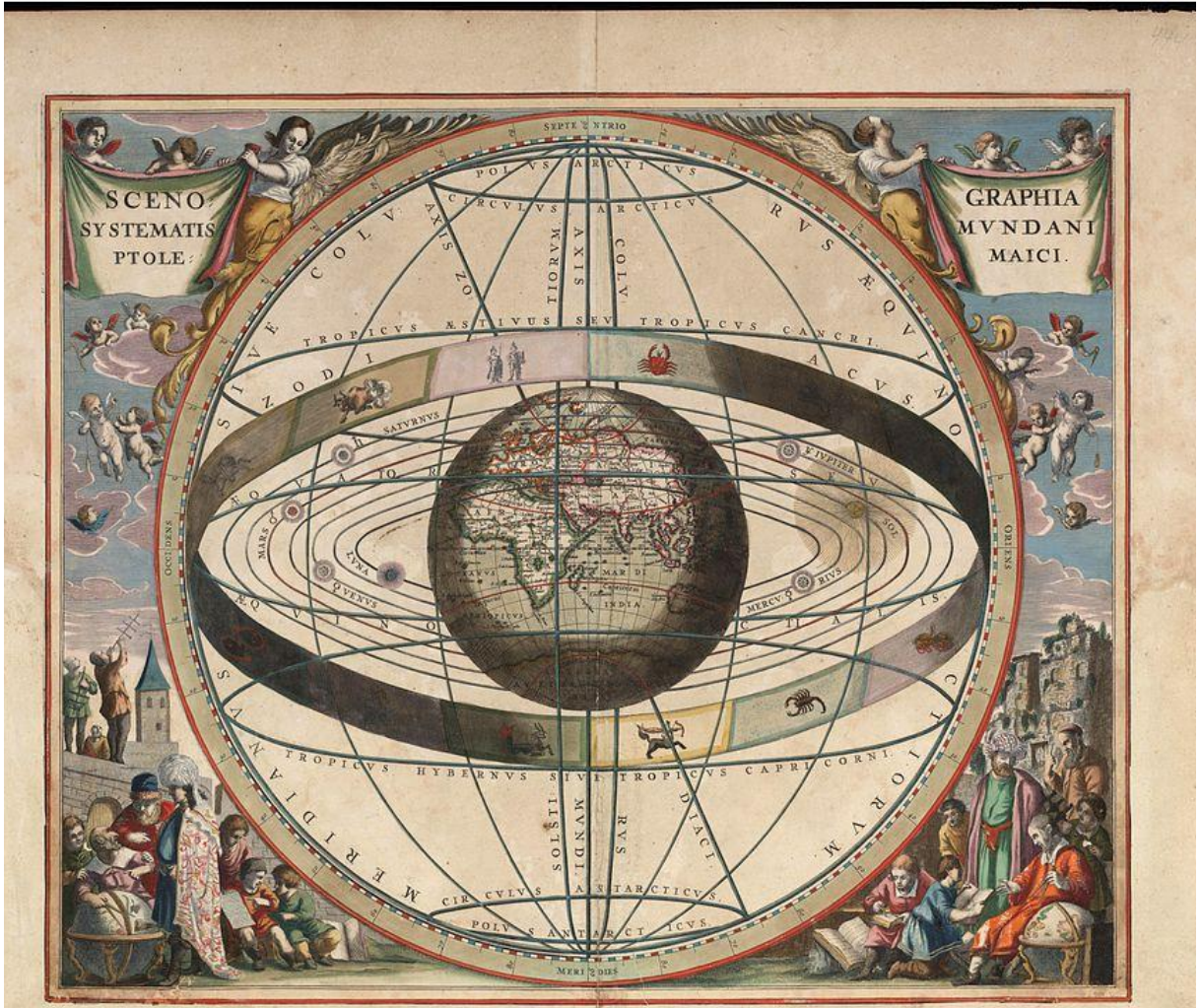
which problems are worth solving
which solutions are legitimate

“All models are wrong”

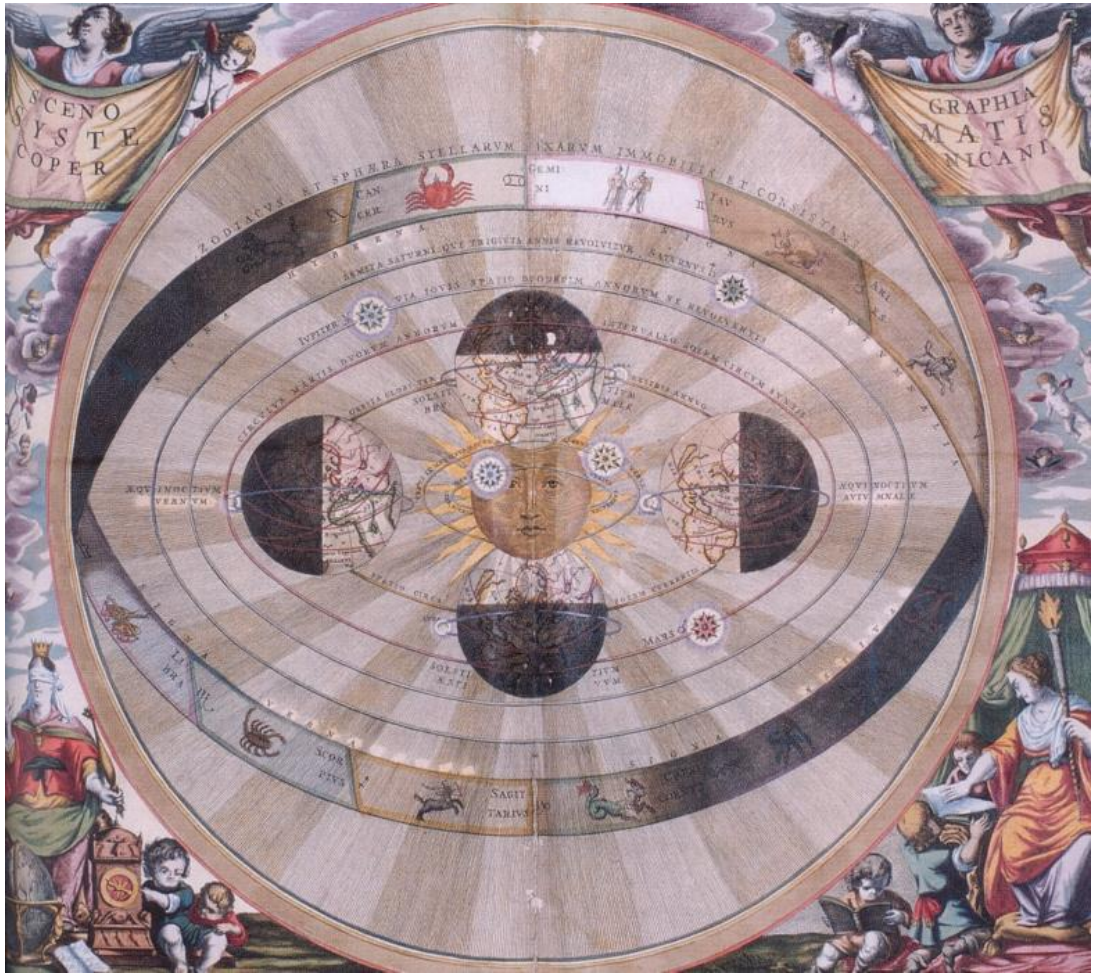
George E. P. Box

"Robustness in the strategy of scientific model building", 1979, p. 202.

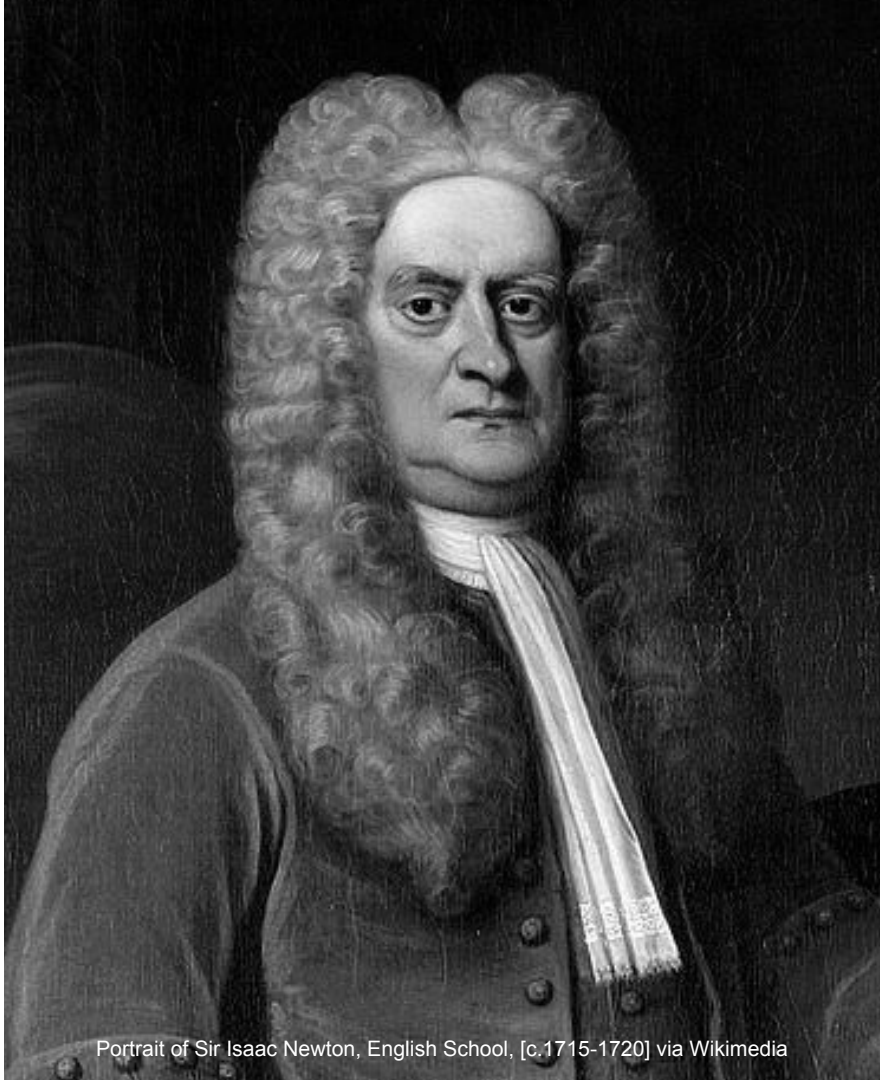

paradigm
anomaly
crisis
shift



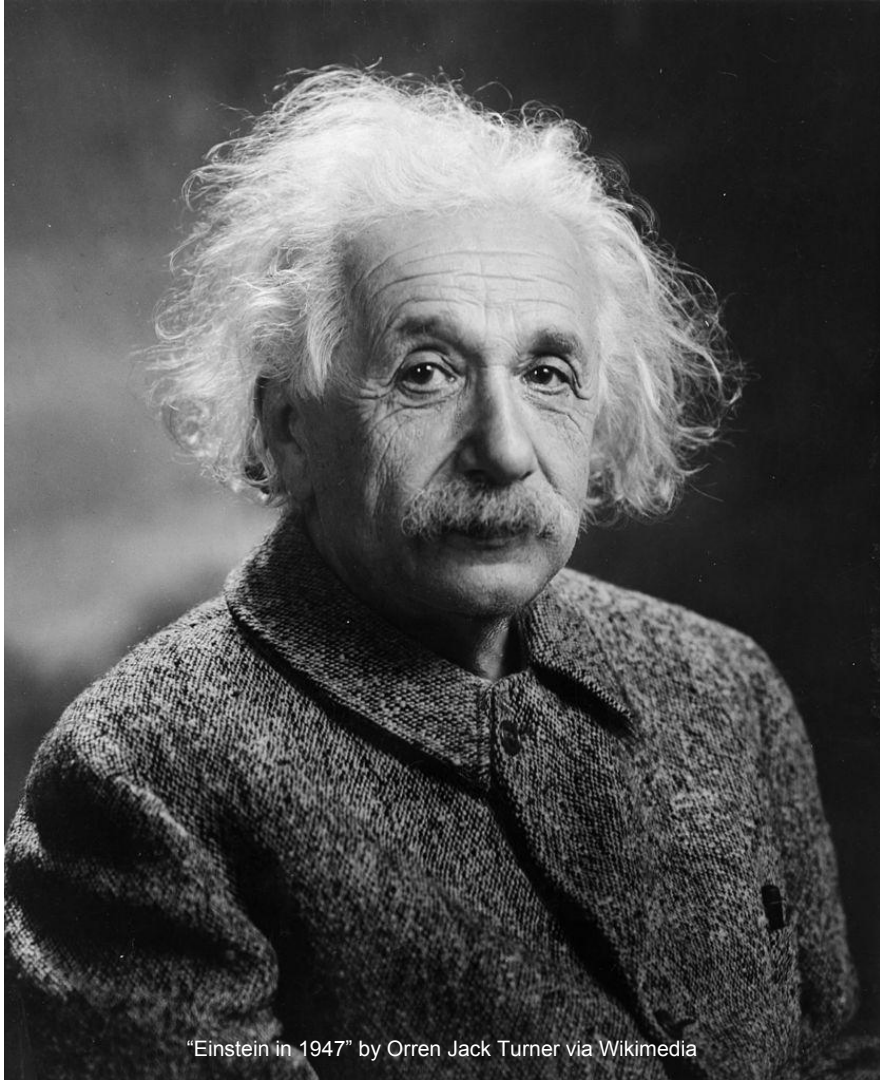
Scenography of the Ptolemaic cosmography by Loon, J. van (Johannes), ca. 1611–1686. via Wikimedia



Scenographia Systematis Copernicani (The Copernican System), 1686. via Wikimedia



Portrait of Sir Isaac Newton, English School, [c.1715-1720] via Wikimedia



“Einstein in 1947” by Orren Jack Turner via Wikimedia

what are some

major paradigms?

imperative programming

follow my commands

in the order I give them

remember state

imperative programming

"Clocks Gears Wallpaper" via Wallpoper

C

GitHub, Inc. (US) https://github.com/python/cpython/blob/master/Parser/node.c

```
21  /* See comments at XXXROUNDUP below.  Returns -1 on overflow. */
22  static int
23  fancy_roundup(int n)
24  {
25      /* Round up to the closest power of 2 >= n. */
26      int result = 256;
27      assert(n > 128);
28      while (result < n) {
29          result <<= 1;
30          if (result <= 0)
31              return -1;
32      }
33      return result;
34  }
```

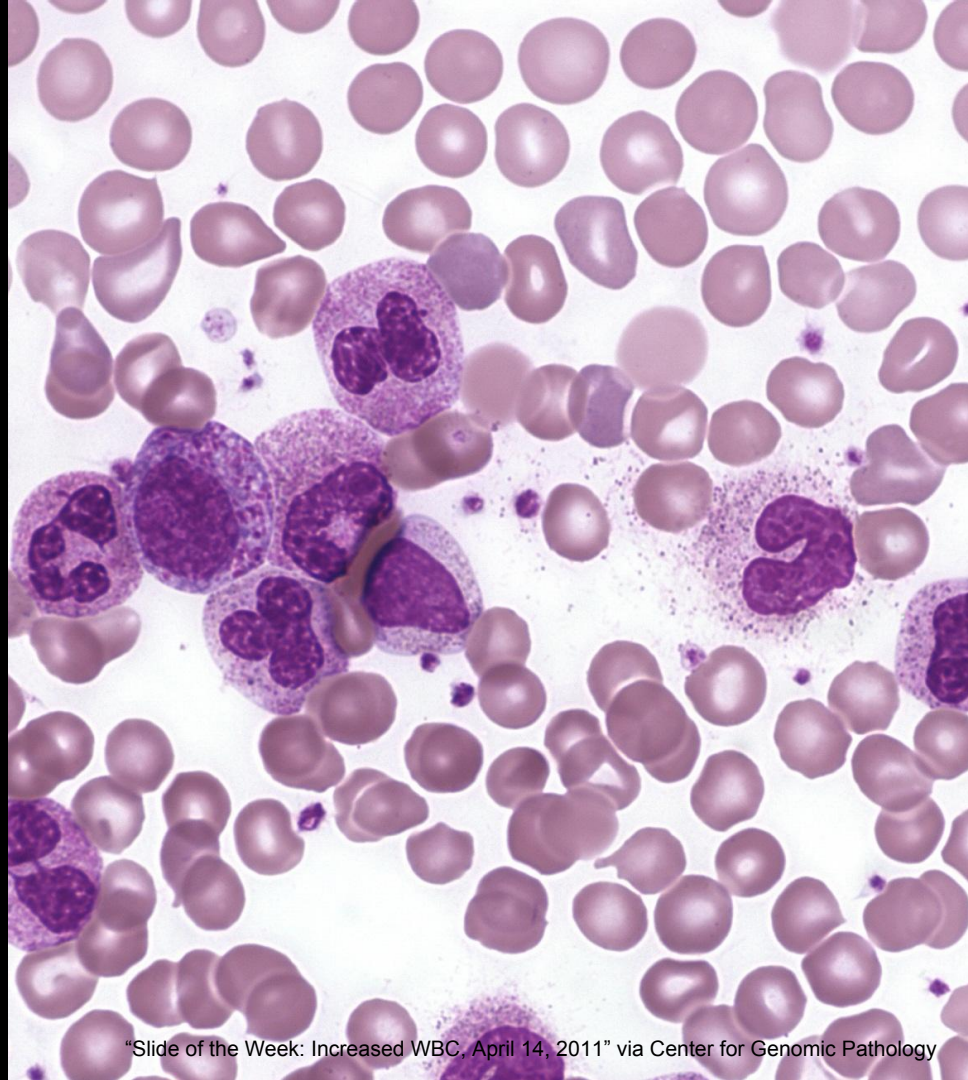
object-oriented programming

keep your state to yourself

receive my messages

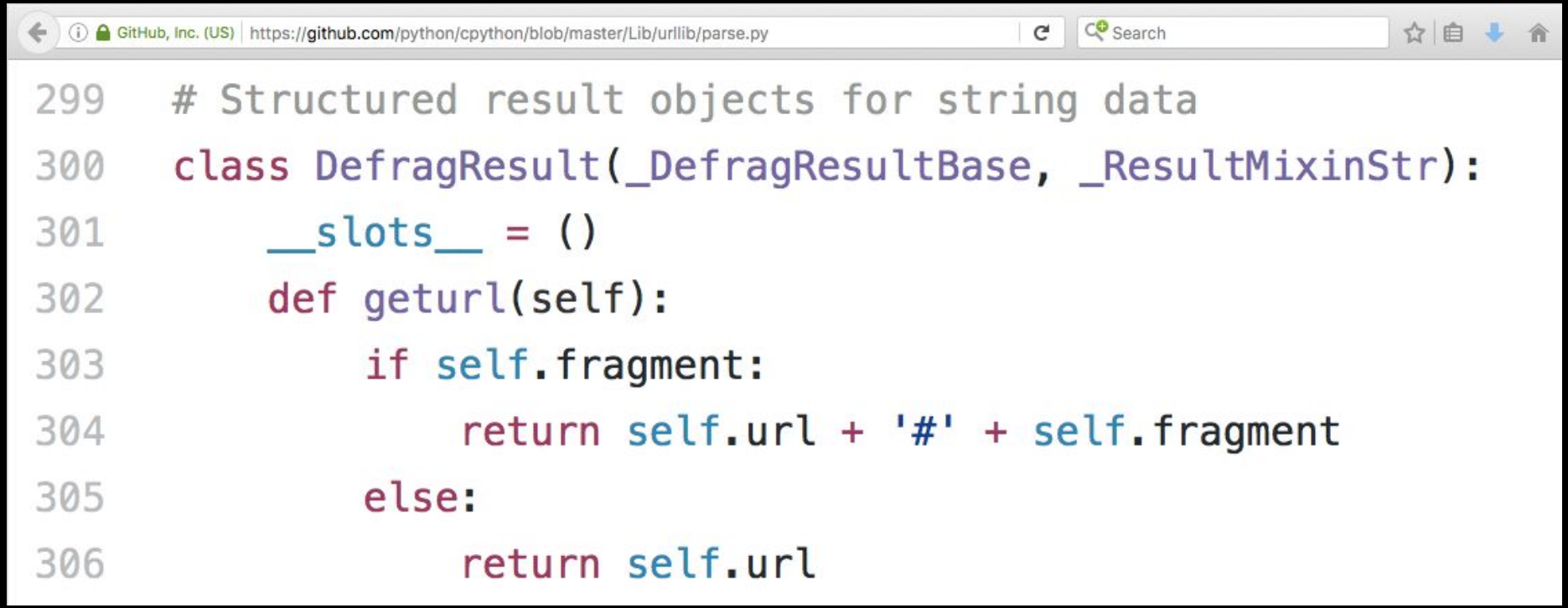
respond as you see fit

object-oriented programming



"Slide of the Week: Increased WBC, April 14, 2011" via Center for Genomic Pathology

Python


GitHub, Inc. (US) https://github.com/python/cpython/blob/master/Lib/urllib/parse.py
Search
299 # Structured result objects for string data
300 class DefragResult(_DefragResultBase, _ResultMixinStr):
301 __slots__ = ()
302 def geturl(self):
303 if self.fragment:
304 return self.url + '#' + self.fragment
305 else:
306 return self.url

functional programming

mutable state is dangerous

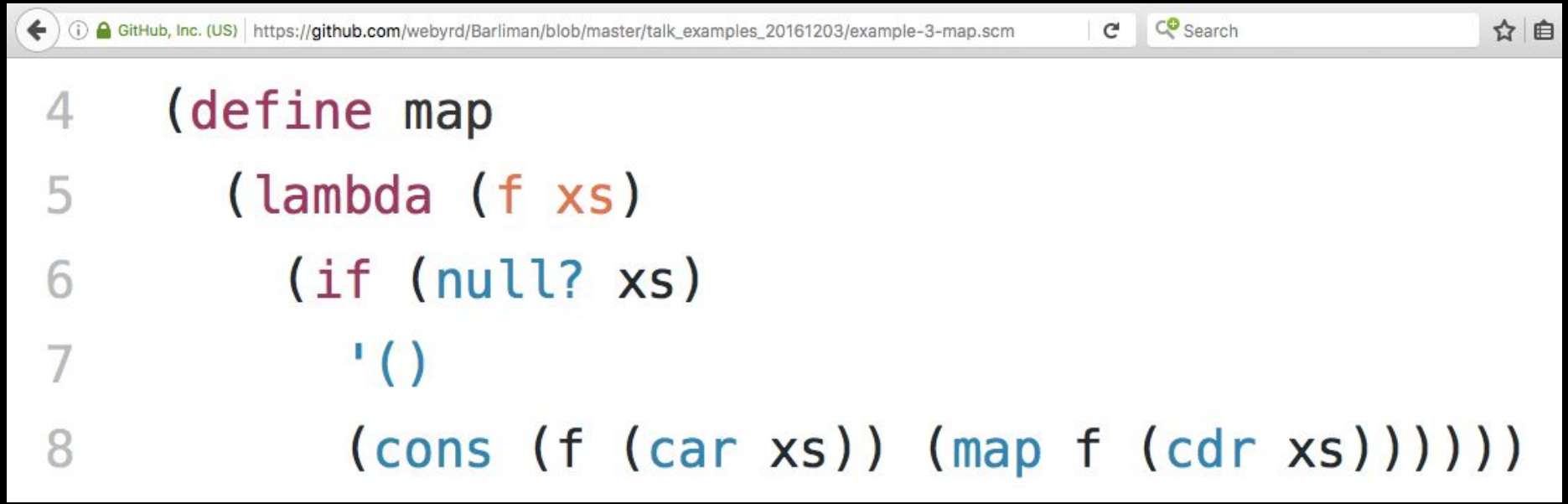
pure functions are safe

data goes in
data comes out

functional programming

"2012 Chevrolet Cruze on the production line at Lordstown Assembly in Lordstown, Ohio" via GM

Scheme (Lisp)




```
4   (define map
5     (lambda (f xs)
6       (if (null? xs)
7           '()
8           (cons (f (car xs)) (map f (cdr xs))))))
```

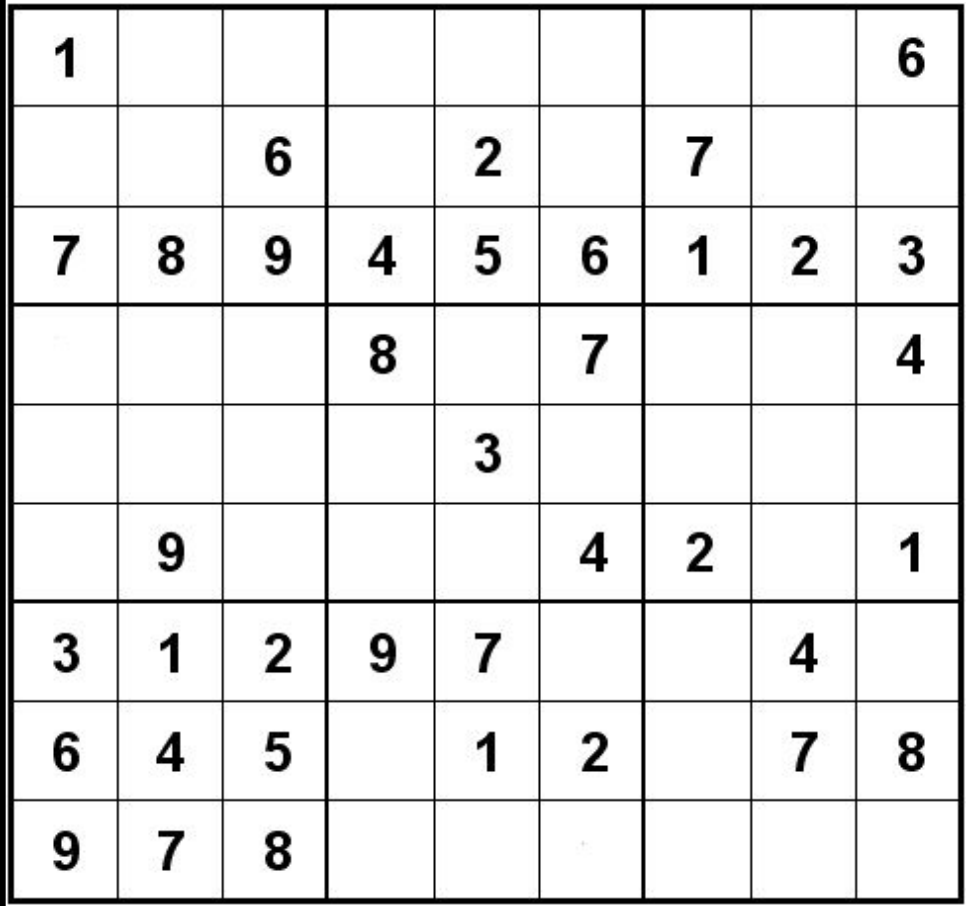
declarative programming

these are the facts

this is what I want

I don’t care how you do it

declarative programming



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

SQL

```
SELECT isbn,
       title,
       price,
       price * 0.06 AS sales_tax
FROM   Book
WHERE  price > 100.00
ORDER BY title;
```

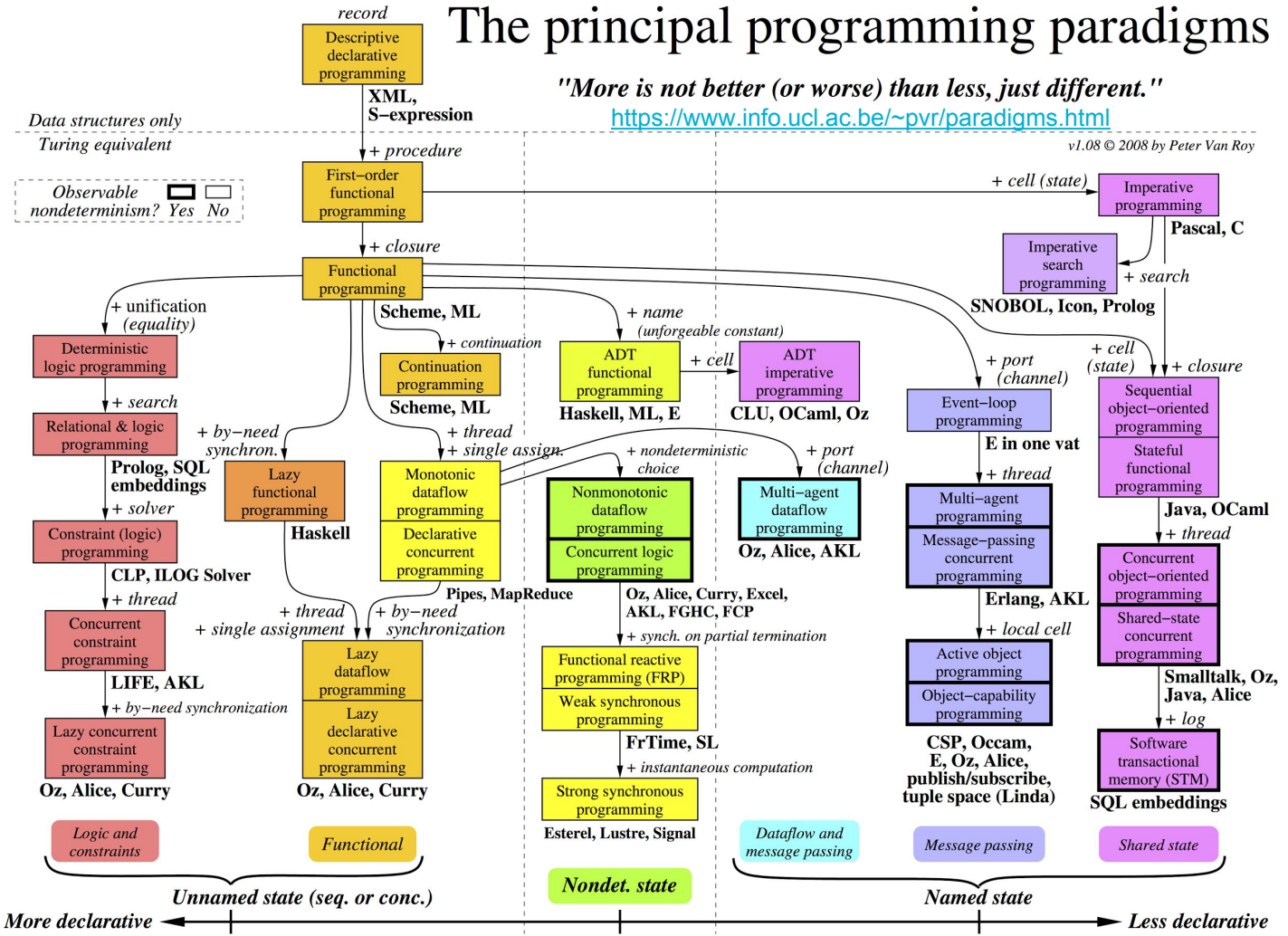
Prolog

```
parent_child(juan, ana).
parent_child(kim, ana).
parent_child(kim, mai).

sibling(X,Y) :- parent_child(Z,X), parent_child(Z,Y).


?- sibling(ana, mai).
Yes
```

The principal programming paradigms

"More is not better (or worse) than less, just different."

https://www.info.ucl.ac.be/~pvr/paradigms.html

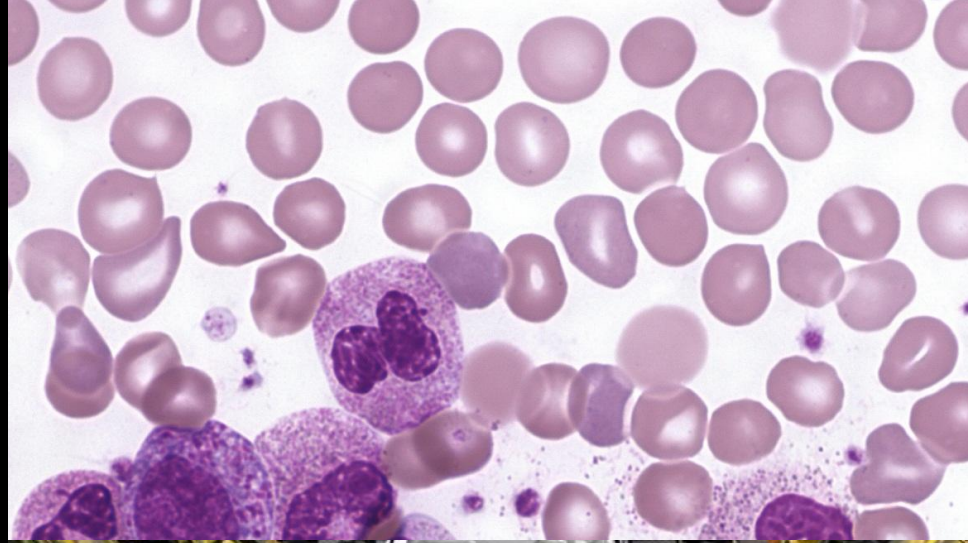


what do they have
in common?

shared
mutable state

~~shared~~

~~mutable state~~

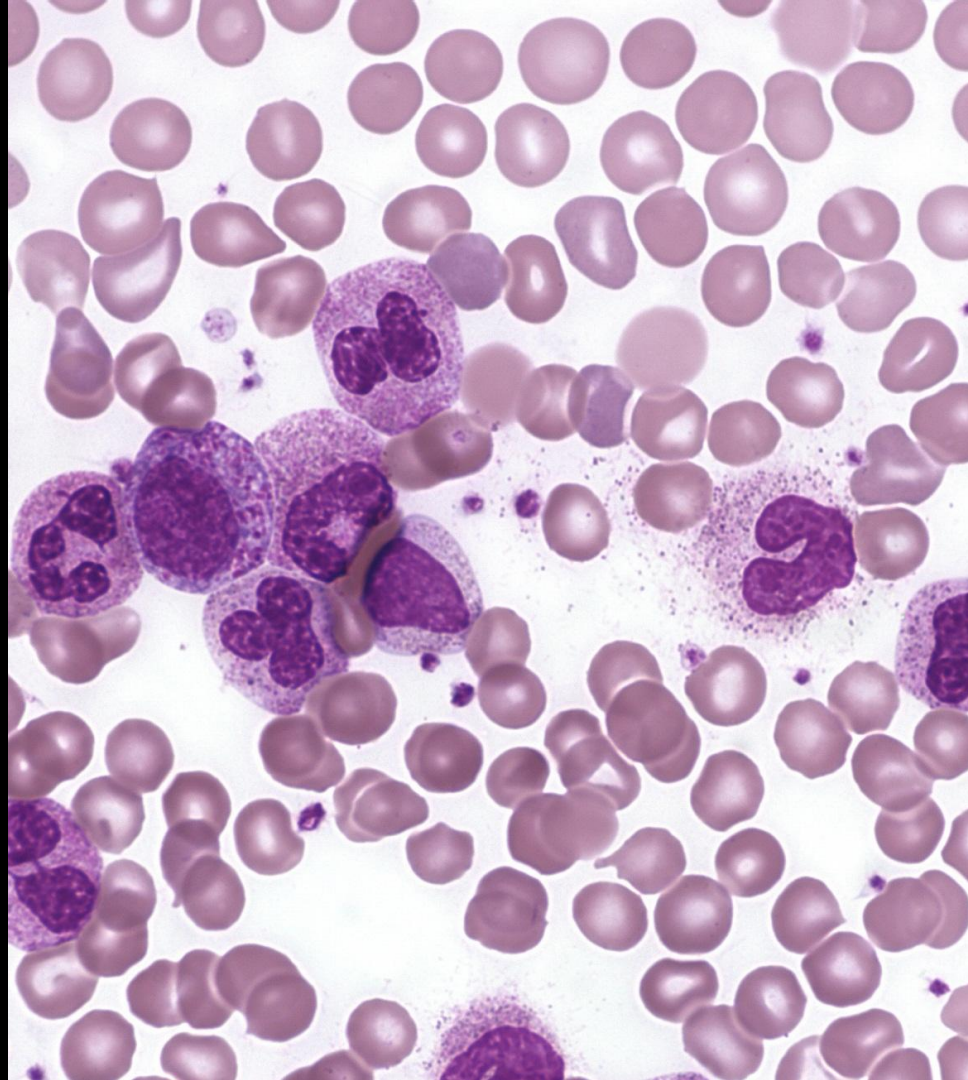


shared

~~mutable state~~

~~shared~~

mutable state

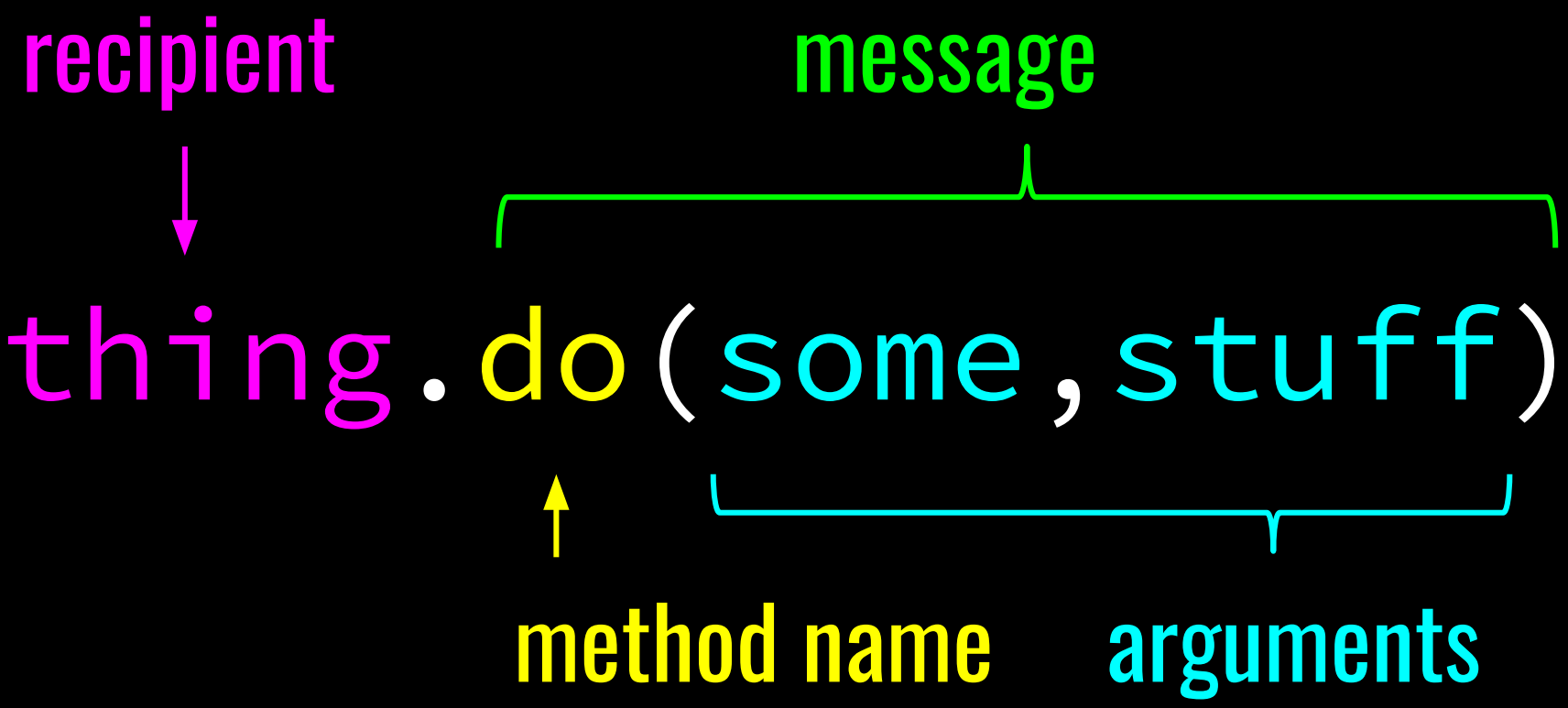


“I' m sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea. The big idea is "messaging".”

Alan Kay

Message to Smalltalk/Squeak mailing list, 1998

```
thing.do(some,stuff)
```


recipient
message
thing.do(some,stuff)
method name
arguments

Ruby

```ruby
thing.do(some,stuff)

thing.send(:do,some,stuff)
```

```
class Friend:
    def __init__(self, friends):
        self.friends = friends
    def is_friend_of(self, name):
        return name in self.friends


buddy = Friend(['alan', 'alonzo'])
buddy.is_friend_of('guy')    # False
```

```
buddy.is_friend_of('guy')
```

```
buddy.is_friend_of('guy')

buddy.send(‘is_friend_of’, 'guy')
```

```
buddy.is_friend_of('guy')

buddy.send(‘is_friend_of’, 'guy')

buddy('is_friend_of', 'guy')
```

```
def Friend(friend_names):
  def is_my_friend(name):
    return name in friend_names
  def instance(method, *args):
    if method == 'is_friend_of':
      return is_my_friend(*args)
  return instance

buddy = Friend(['alan', 'alonzo'])
buddy('is_friend_of','guy') # False
```

```
class Friend:
    def __init__(self, friends):
        self.friends = friends
    def is_friend_of(self, name):
        return name in self.friends


buddy = Friend(['alan', 'alonzo'])
buddy.is_friend_of('guy')    # False
```

which paradigm is the best?

“All models are wrong...

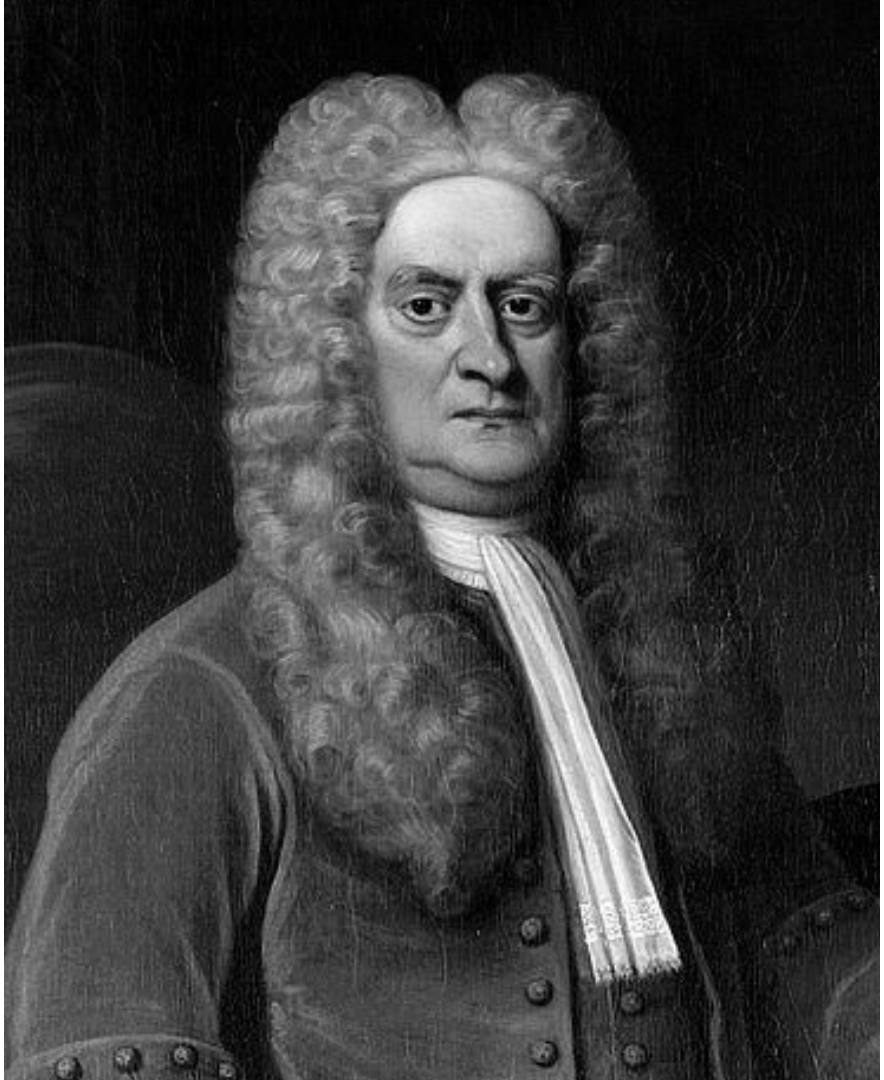
George E. P. Box

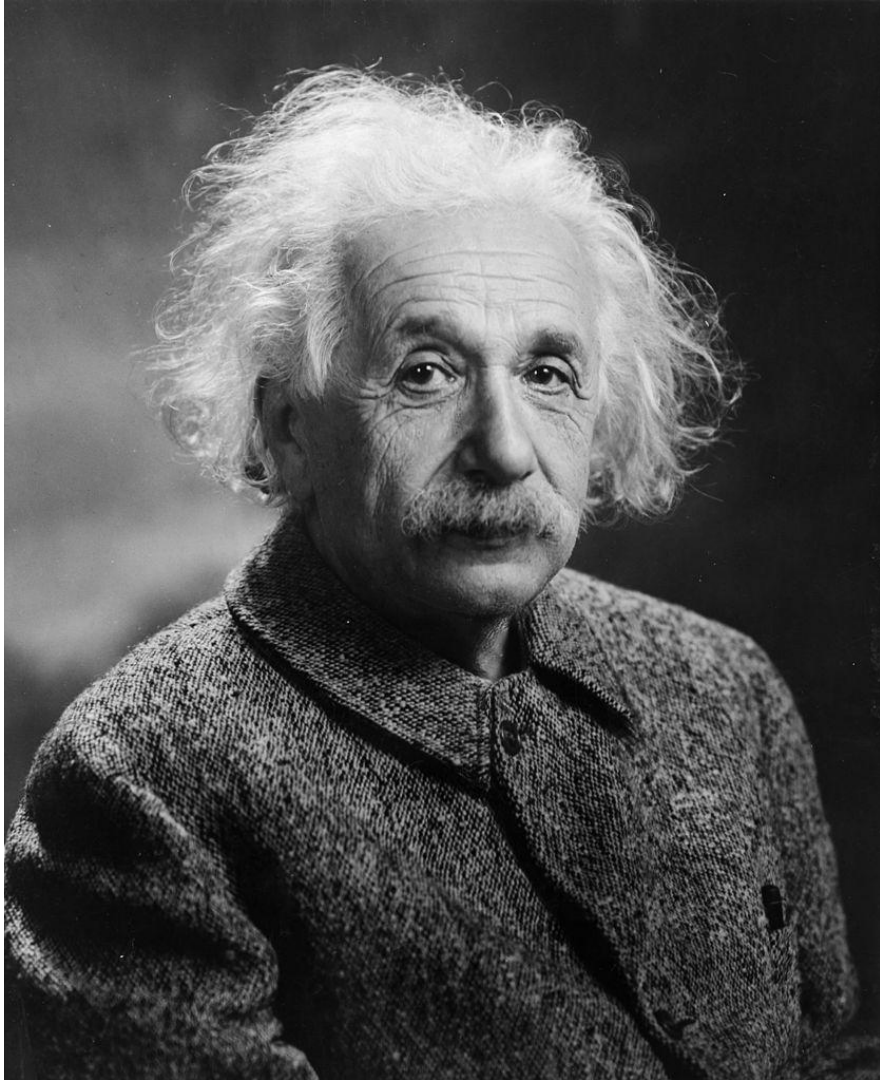
"Robustness in the strategy of scientific model building", 1979, p. 202.

“All models are wrong
but some are useful”

George E. P. Box

"Robustness in the strategy of scientific model building", 1979, p. 202.





Each paradigm supports a set of concepts that makes it the best for a certain kind of problem.

Peter Van Roy

"Programming paradigms for dummies: What every programmer should know", 2009, p. 10.

~~“Is the model true?”~~

“Is the model illuminating and useful?”

George E. P. Box

"Robustness in the strategy of scientific model building", 1979, p. 202.

what can a paradigm teach me?

be explicit
understand
implementation

```
1 # global.py
2 for i in range(10**8):
3     i
```

time: 9.185s

```
1 # in_fn.py
2 def run_loop():
3     for i in range(10**8):
4         i
5 run_loop()
```

time: 5.738s

Adapted from A. Vakil, “Exploring Python Bytecode”, 2015. https://youtu.be/GNPKBICTF2w

```
1 # global.py
2 for i in range(10**8):
3     i
```

time: 9.185s

```
2   0 SETUP_LOOP    24 (to 27)
    ...
    16 STORE_NAME   1 (i)
3   19 LOAD_NAME    1 (i)
    ...
```

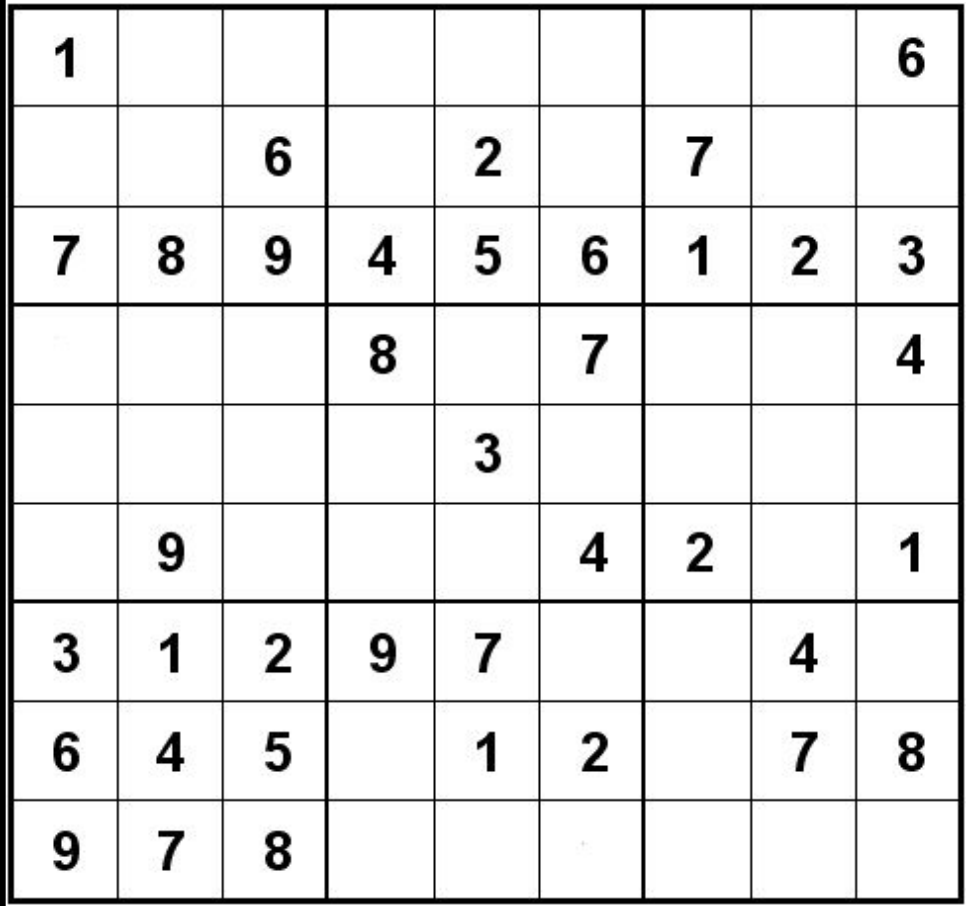
```
1 # in_fn.py
2 def run_loop():
3     for i in range(10**8):
4         i
5 run_loop()
```

time: 5.738s

```
3   0 SETUP_LOOP    24 (to 27)
    ...
    16 STORE_FAST   0 (i)
4   19 LOAD_FAST    0 (i)
    ...
```

Adapted from A. Vakil, “Exploring Python Bytecode”, 2015. https://youtu.be/GNPKBICTF2w

be abstract
understand
domain

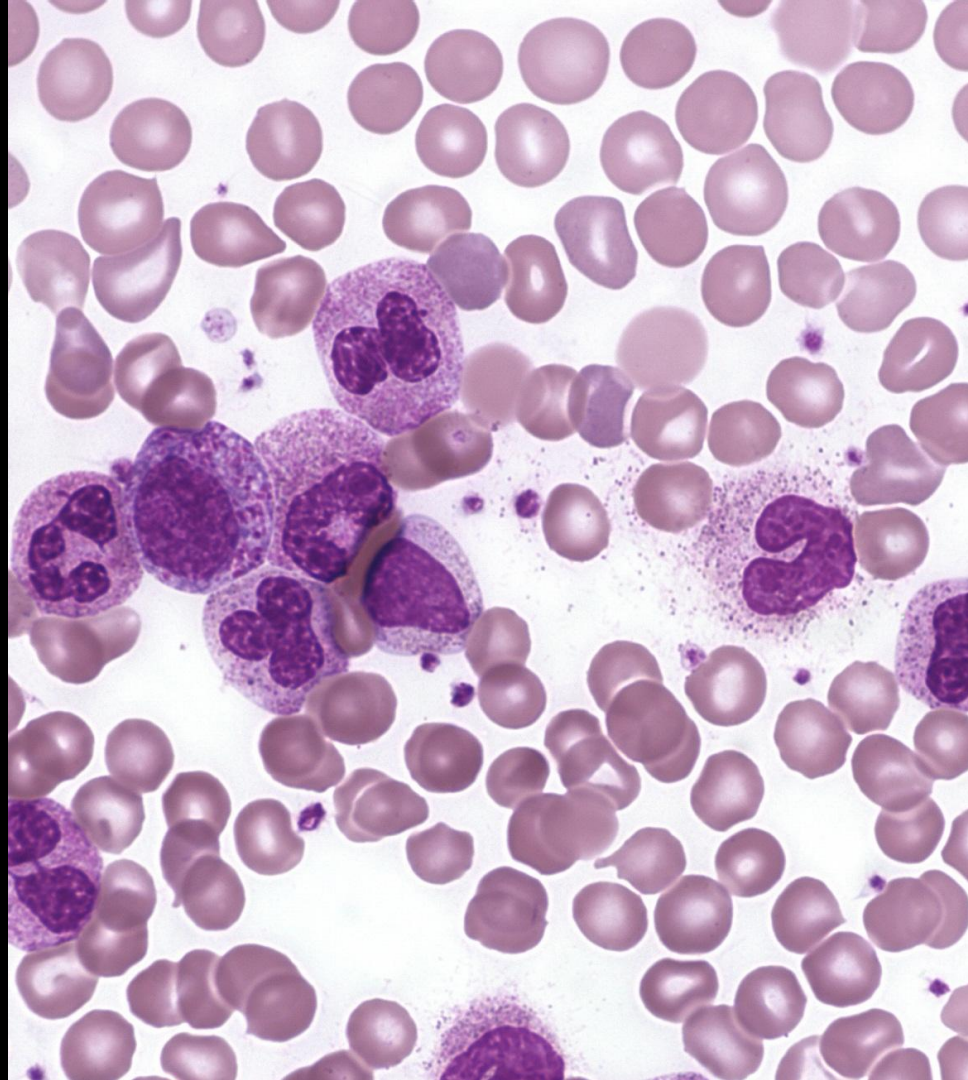


Embedded DSL in Java

```
cal = new Calendar();
cal.event("GOTO Chicago")
    .on(2017, 5, 2)
    .from("09:00")
    .to("17:00")
    .at("Swissôtel");
```

Adapted from M. Fowler & R. Parsons, *Domain Specific Languages*, 2011, p. 345.

encapsulate
communicate

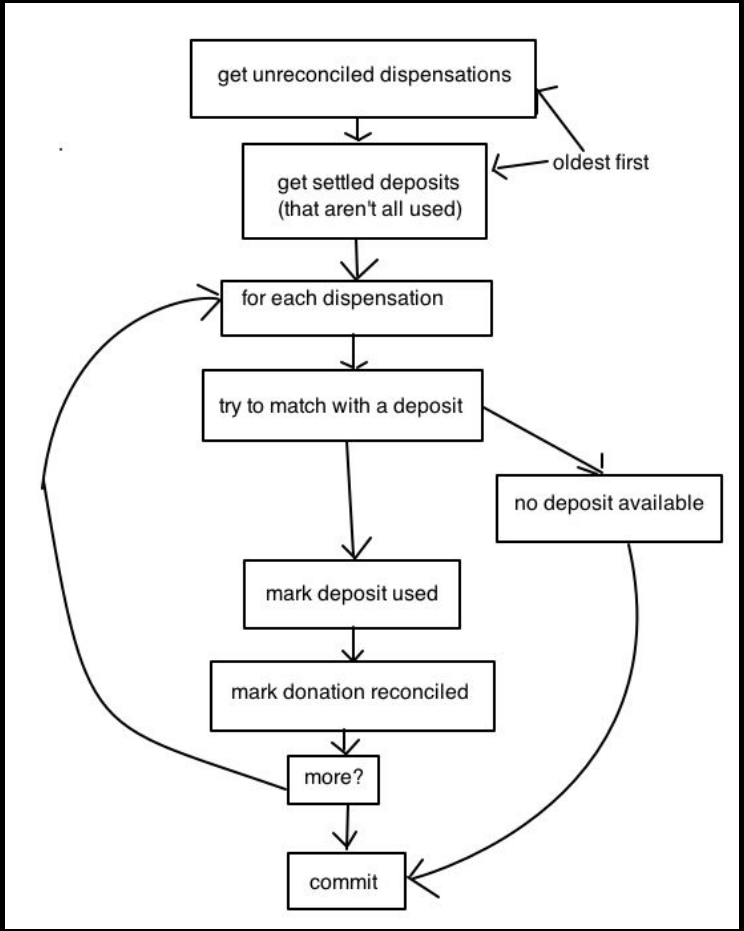


Context-aware API in F#

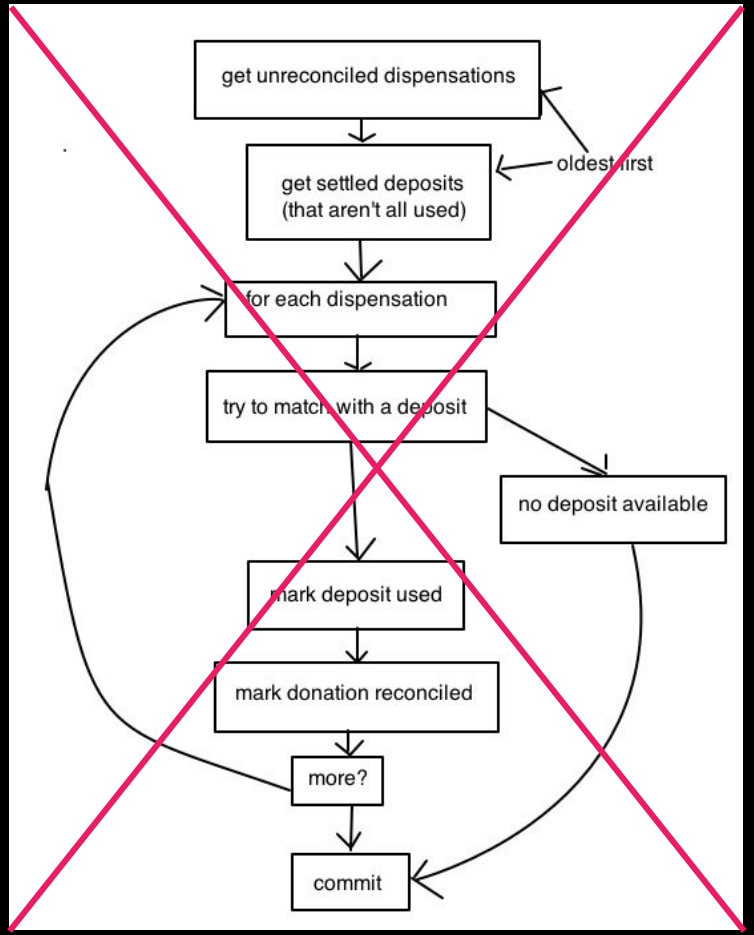
```
module MyApi =
  let fnA dep1 dep2 dep3 arg1 = doAWith dep1 dep2 dep3 arg1
  let fnB dep1 dep2 dep3 arg2 = doBWith dep1 dep2 dep3 arg2


type MyParametricApi(dep1, dep2, dep3) =
  member __.FnA arg1 = doAWith dep1 dep2 dep3 arg1
  member __.FnB arg2 = doBWith dep1 dep2 dep3 arg2
```

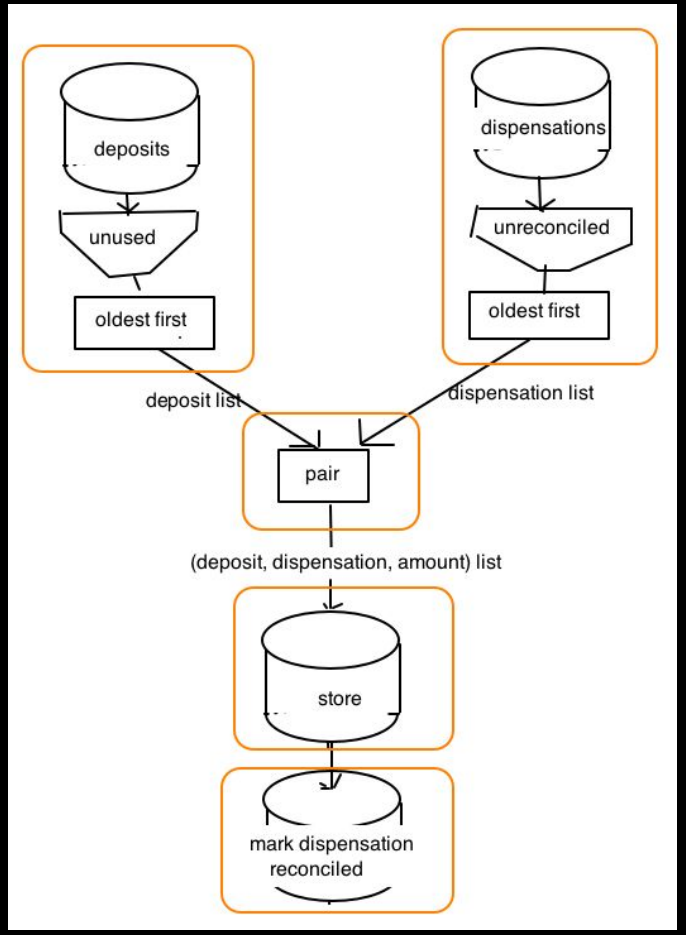
Adapted from E. Tsarpalis, “Why OOP Matters (in F#)”, 2017.

specialize
transform data



Adapted from J. Kerr, “Why Functional Matters: Your white board will never be the same”, 2012.





Adapted from J. Kerr, “Why Functional Matters: Your white board will never be the same”, 2012.

no paradigm is best absolutely
each is best for a certain case

“If the advancement of the general art of programming requires the continuing invention and elaboration of paradigms, ...

Robert W. Floyd

“The paradigms of programming.”, 1979. p. 456

“If the advancement of the general art of programming requires the continuing invention and elaboration of paradigms, advancement of the art of the individual programmer requires that [t]he[y] expand [their] repertory of paradigms.”

Robert W. Floyd

“The paradigms of programming.”, 1979. p. 456

learn new paradigms
try multi-paradigm languages

what's the point?

paradigms enable programming

paradigms define programming

don’t fight your paradigm,
embrace it

be open to shift

attend to your paradigms

David Albert, Darius Bacon, Julia Evans
& the Recurse Center

GOTO Chicago organizers

thank you!

@AnjanaVakil

References & further reading/watching

Box, G. E. P. (1979), "Robustness in the strategy of scientific model building", in Launer, R. L.; Wilkinson, G. N., *Robustness in Statistics*, Academic Press, pp. 201–236.

Floyd, R. W. (1979). “The paradigms of programming.” *Commun. ACM* 22, 8, 455-460.

Fowler, Martin, with Parsons, Rebecca. (2011). *Domain Specific Languages*. Addison-Wesley.

Kay, Alan (1998). Message to Smalltalk/Squeak mailing list. wiki.c2.com/?AlanKayOnMessaging

Kerr, Jessica (2012). “Why Functional Matters: Your white board will never be the same”. blog.jessitron.com/2012/06/why-functional-matters-your-white-board.html

Kerr, Jessica (2014). “Functional Principles for Object-Oriented Development”, GOTO Chicago. https://youtu.be/GpXsQ-NIKXY

Kuhn, Thomas S. (1970). *The Structure of Scientific Revolutions*. (2nd ed.). University of Chicago Press.

Tsarpalis, Eirik. (2017). “Why OO matters (in F#)”. https://eiriktsarpalis.wordpress.com/2017/03/20/why-oo-matters-in-f/

Van Roy, Peter. (2009). “Programming paradigms for dummies: What every programmer should know.” In *New computational paradigms for computer music*, p. 104.

Williams, Ashley. (2015). “If you wish to learn ES6/2015 from scratch, you must first invent the universe”, JSConf. https://youtu.be/DN4yLZB1vUQ