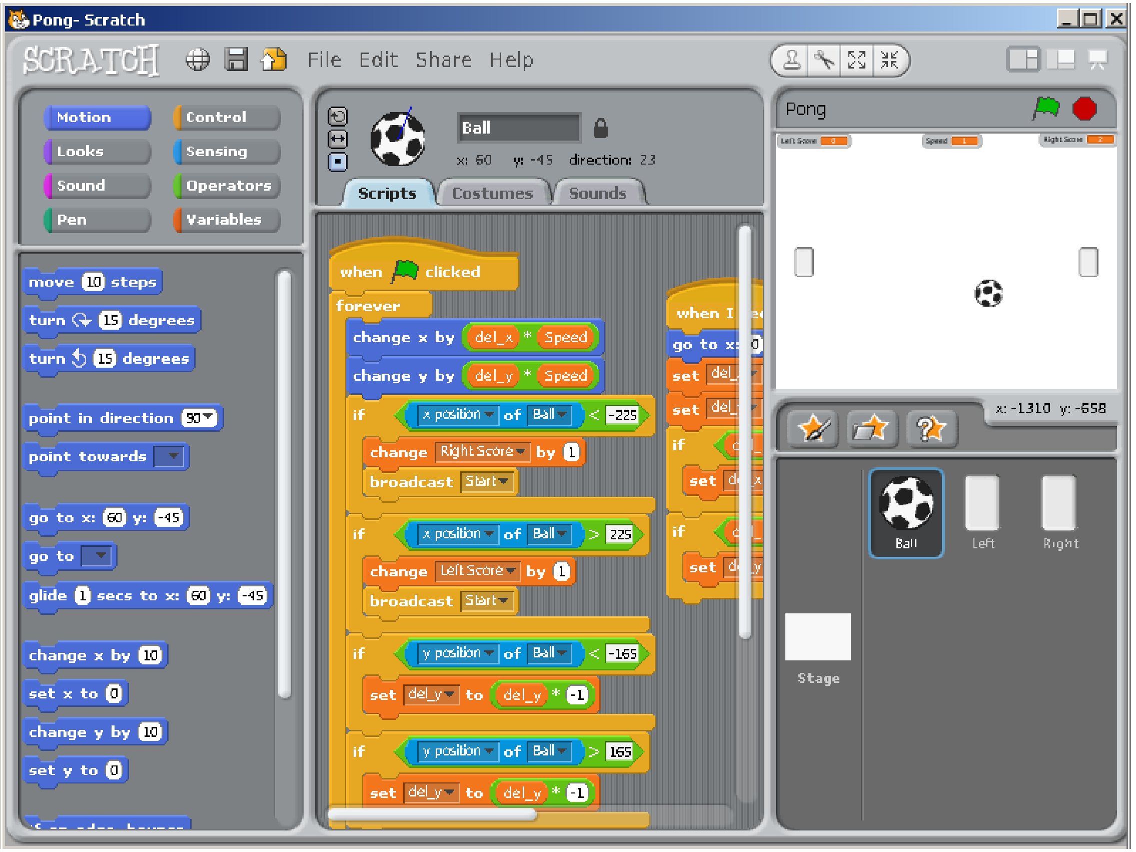
An Expression-Oriented Approach to Programming Education

Exploiting the Synergy between Computing and Math

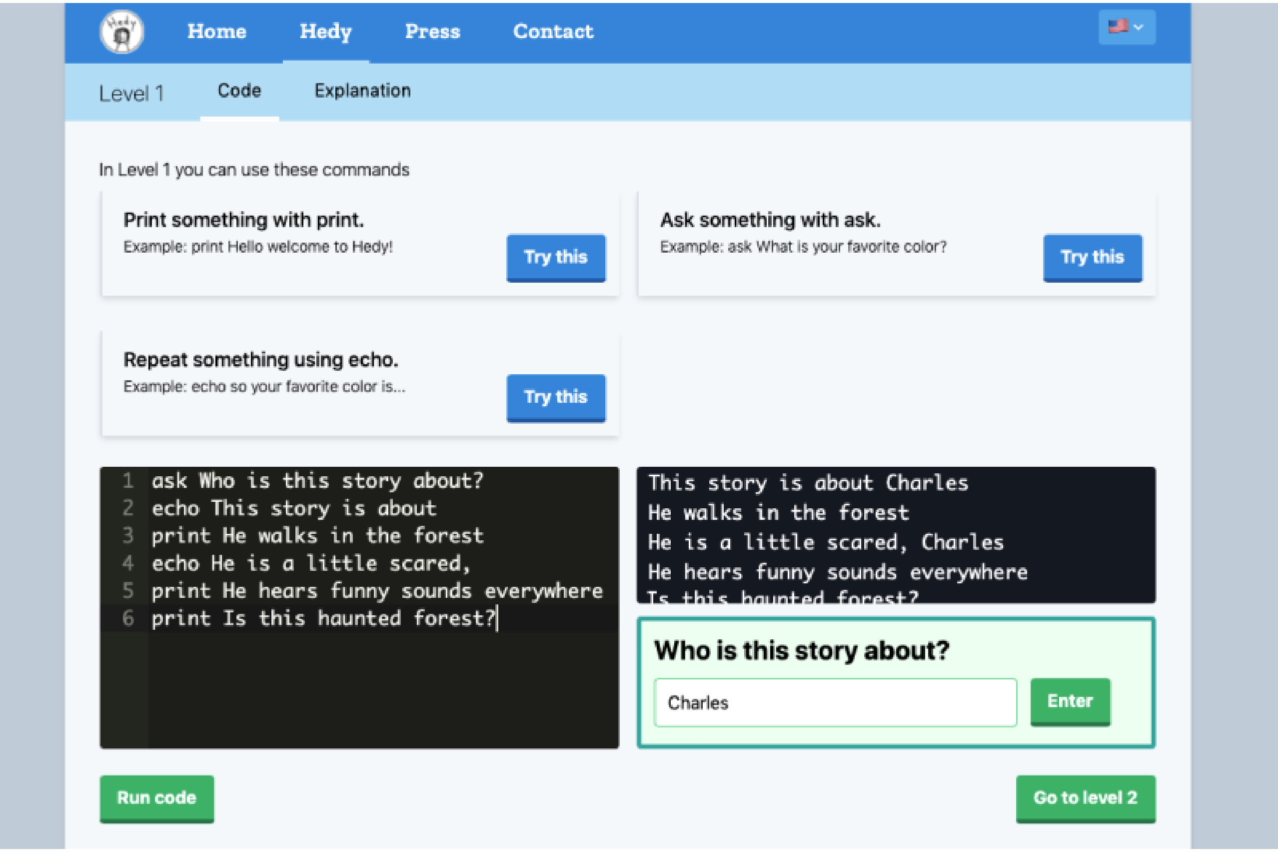
Learning Syntax is Known to be an Obstacle in Programming Education

Responses: Block Coding (replace syntax with shapes) & Gradual Languages (relaxed syntax rules)

Scratch [MIT Media Lab]



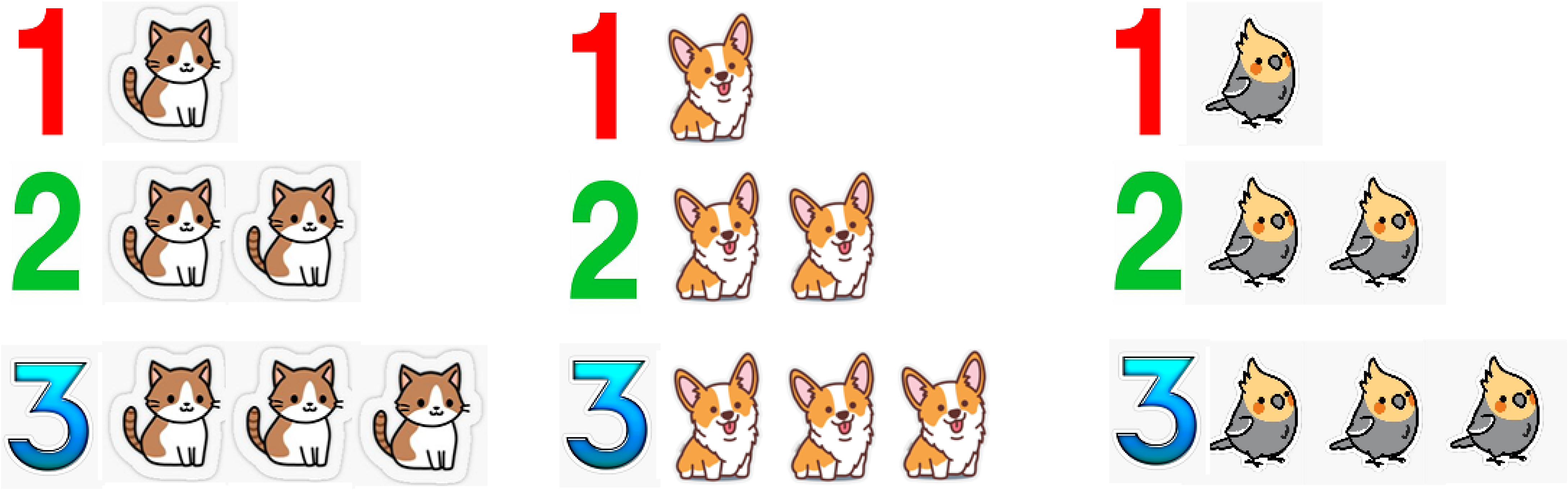
Hedy [Felienne Hermans, Leiden University]



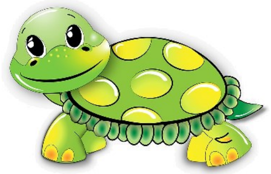
Alternative Approach

Exploit Synergy with Math ⇔ Embrace Syntax

Math Abstractions … Baby Steps



and I am **one** baby syntax turtle ➔



positional numeral system, operations, operator precedence, fractions, (oh my!) …

Math Abstractions …

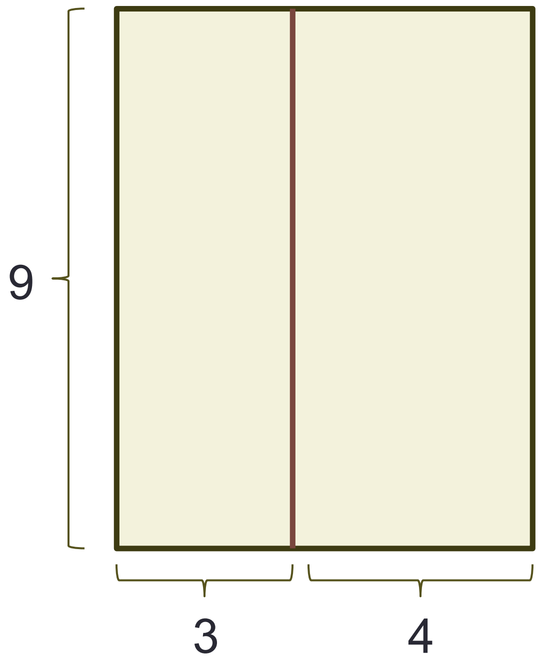
$8\ 1\ 9\ 2 = 8 \times 1000 + 1 \times 100 + 9 \times 10 + 2 \times 1$

$1{,}000{,}000 - 997 = 999{,}003$

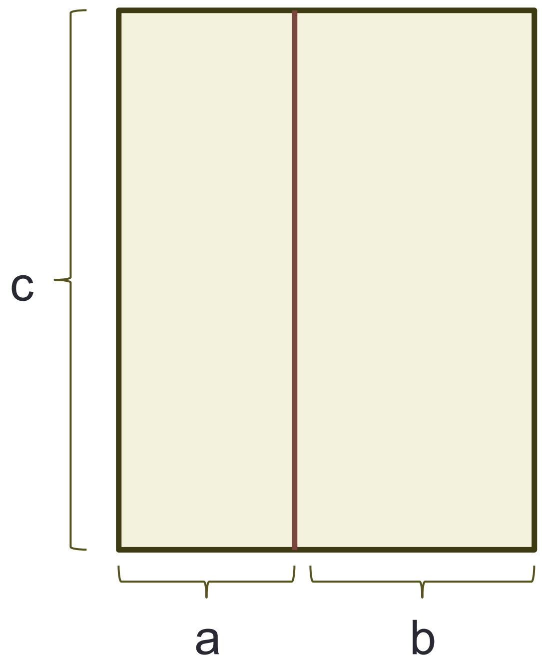
$$\begin{array}{r} 11\ + \\ 31 \\ \hline 42 \end{array}$$

$6 \times 7 = 42$

$$\frac{42}{18} = \frac{21}{9}$$



$9 \times (3 + 4) = 63$



$c \times (a + b) = ca + cb$

$2 / 7 + 3 / 7 = 5 / 7$

The turtles … they are multiplying (!) – here we are, in middle school and high school:

Math Abstractions …

$$E = mc^2$$

$$y = m \cdot x + q$$

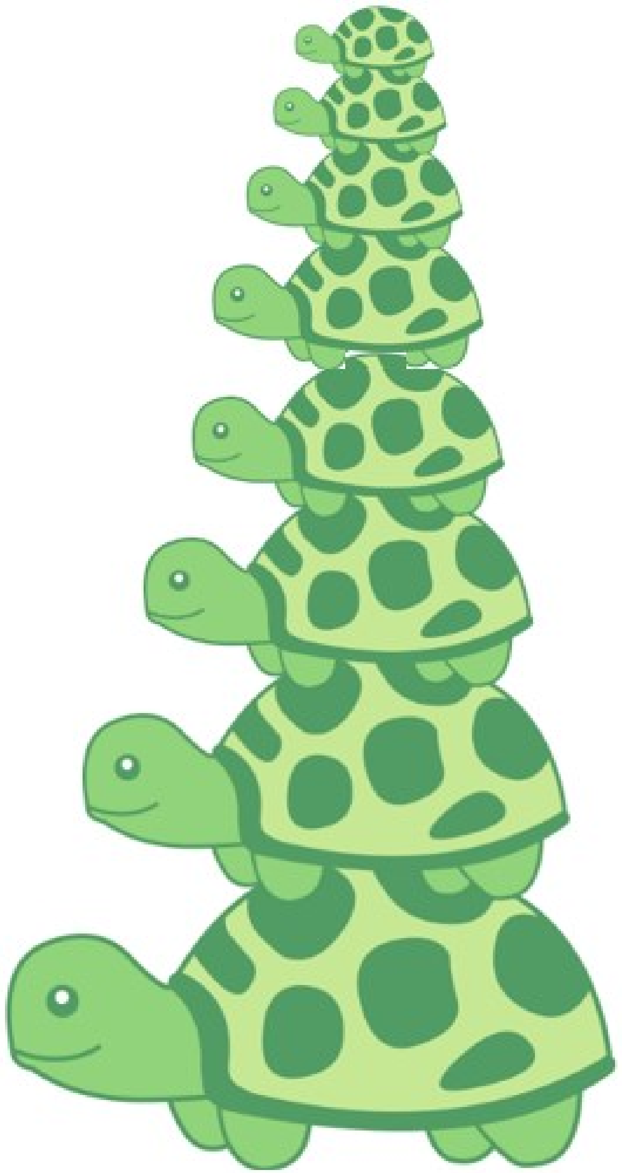
$$sin^2 x + cos^2 x = 1$$

$$\frac{sin\ \ x}{cos\ \ x} = tan\ \ x$$

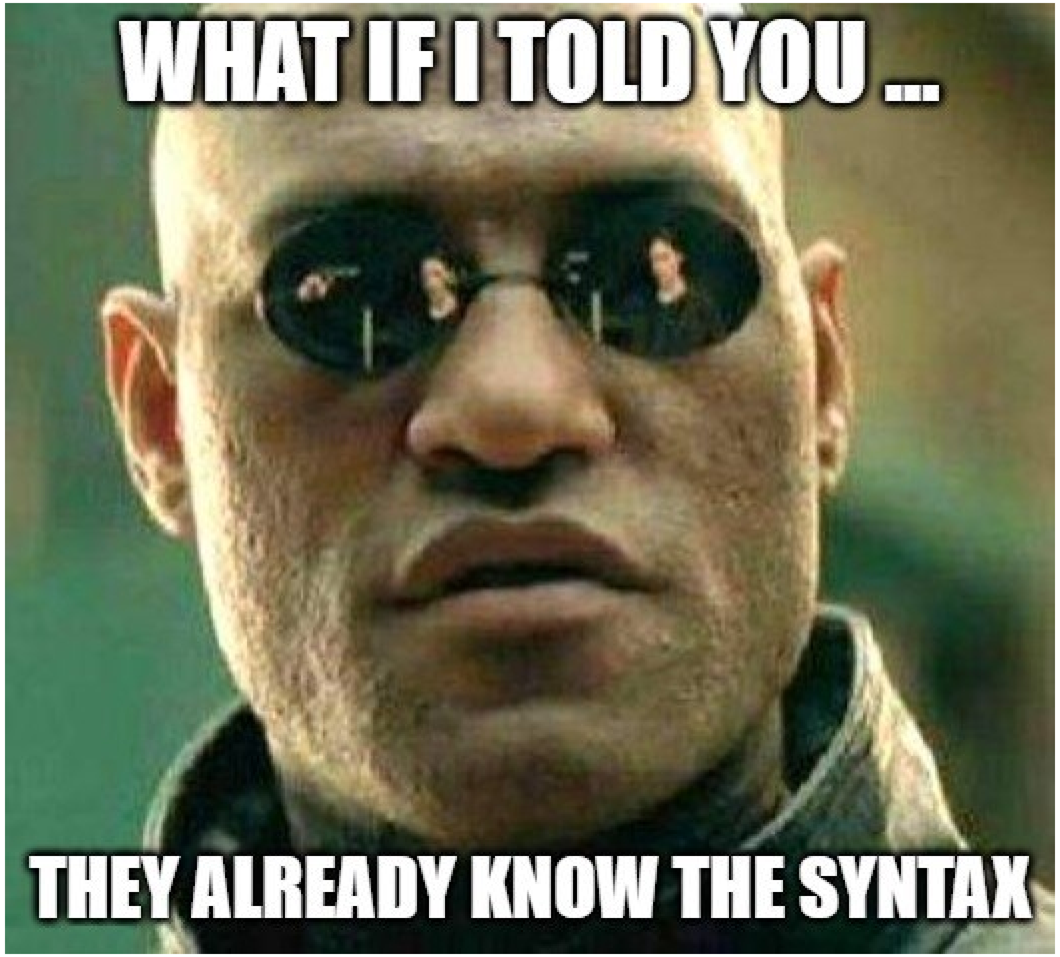
$$(a - b)(a + b) = a^2 - b^2$$

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$F = G\ \frac{m_1\ \ m_2}{d^2}$$



How much syntax should we feed students to ***start learning programming***?



Note: MATH stands for Generally accepted Math notation

They Already Know The Syntax!

	Raise to Power	Multiplication	Division	Addition	Subtraction	Less than	String Concatenation
MATH	x^y	$xy \mid x*y$	$x \div y \mid x/y$	$x+y$	$x-y$	$x<y$	$xy \mid x \cdot y$
FORTRAN	x ** y	x * y	x / y	x + y	x – y	x .LT. y	x // y
LISP	(pow x y)	(* x y)	(/ x y)	(+ x y)	(– x y)	(< x y)	(concatenate x y)
C / C++	pow(x, y)	x * y	x / y	x + y	x – y	x < y	x + y
Haskell	x ^ y | x ** y	x * y	x / y	x + y	x – y	x < y	x ++ y
Python	x ** y	x * y	x / y	x + y	x – y	x < y	x + y
Java	Math.pow(x, y)	x * y	x / y	x + y	x – y	x < y	x + y
JavaScript	x ** y	x * y	x / y	x + y	x – y	x < y	x + y
OCaml	x ** y	x * y | x *. y	x / y | x /. y	x + y | x +. y	x – y | x –. y	x < y	x ^ y
MS-Excel	x ^ y	x * y	x / y	x + y	x – y	x < y	x & y

.:. OK, sure, but those are ***just expressions***. Computing them is not programming, right? Expressions aren't enough, right? ➔

A User-Centred Approach to Functions in Excel
30th June 2003

Simon Peyton Jones
Microsoft Research

Alan Blackwell
Cambridge University

Margaret Burnett
Oregon State University

“It may seem odd to describe a spreadsheet as a programming language. Indeed, one of the great merits of spreadsheets is that users need not think of themselves as doing “programming”, let alone functional programming — rather, they simply “write formulae” or “build a model”. However, one can imagine printing the cells of a spreadsheet in textual form, like this:

```
A1 = 3
A2 = A1-32
A3 = A2 * 5/9
```

and then it plainly is a (functional) program.”

to program with spreadsheets all the syntax you need is that of expressions!

But traditional spreadsheets have issues that make them unacceptable for education and other purposes …

Critique of the Traditional Spreadsheet Core

- **Lack of functional abstraction**
  - Considerable research work has been done on this
  - December 3rd, 2020: Microsoft Research announced LAMBDA

- **Overly simplistic type system**
  - All top-level variables must be a worksheet
  - Worksheets are non-composable cell containers
  - All cells are unitype and must be referenced via coordinates
  - **A1 notation should be considered harmful**

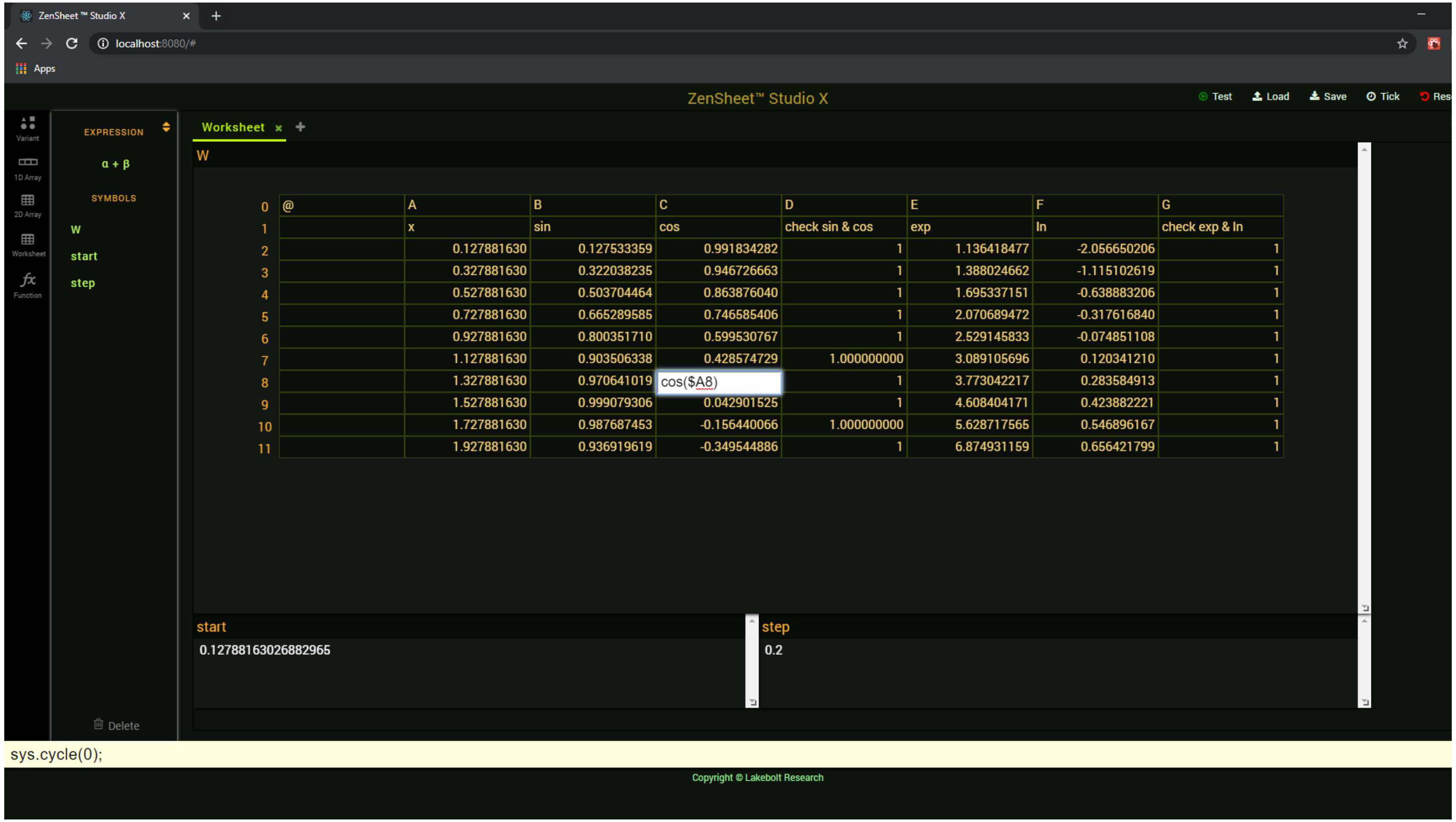
- **Entanglement of model and visualization**
  - Worksheets are the only true variables of the core
    - They are containers that hold state, which includes unreduced expressions
  - Worksheets are also the primary element of the presentation
    - They play an important role as UI layout managers

[8] E. Alda et al. 2021. "Towards Wide-Spectrum Spreadsheet Computing."
*4th International Conference on Information and Computer Technologies (**ICICT 2021**).*
HI, USA, 2021, pp. 233-242. doi: 10.1109/ICICT52872.2021.00046.
https://ieeexplore.ieee.org/document/9476942

A language-centric redesign of spreadsheets has been shown to work

ZenSheet / Lilly

ZenSheet supports functional abstraction and composable data structures.
2D arrays can be used as worksheets: it truly generalizes spreadsheets!



0	@	A	B	C	D	E	F	G
1		x	sin	cos	check sin & cos	exp	ln	check exp & ln
2		0.127881630	0.127533359	0.991834282	1	1.136418477	-2.056650206	1
3		0.327881630	0.322038235	0.946726663	1	1.388024662	-1.115102619	1
4		0.527881630	0.503704464	0.863876040	1	1.695337151	-0.638883206	1
5		0.727881630	0.665289585	0.746585406	1	2.070689472	-0.317616840	1
6		0.927881630	0.800351710	0.599530767	1	2.529145833	-0.074851108	1
7		1.127881630	0.903506338	0.428574729	1.000000000	3.089105696	0.120341210	1
8		1.327881630	0.970641019	cos($A8)	1	3.773042217	0.283584913	1
9		1.527881630	0.999079306	0.042901525	1	4.608404171	0.423882221	1
10		1.727881630	0.987687453	-0.156440066	1.000000000	5.628717565	0.546896167	1
11		1.927881630	0.936919619	-0.349544886	1	6.874931159	0.656421799	1

Types

ZT.1) T → **null** | **error** | **bool** | **number** | **string**

ZT.2) T → **fun**(T, …, T) => T

ZT.3) T → **array**[, …,] => T

ZT.4) T → **struct**(T, …, T)

ZT.5) T → **lazy** T

ZT.6) T → **var**

ZT.7) T → <symbol>

Expressions

XLS.1) E → ? | <error> | **true** | **false** | <number> | <string>

XLS.3) E → < A1 > | <symbol> ! < A1 >

ZSE.1) E → <symbol>

ZSE.2) E → λ(T <symbol>, …, T <symbol>) -> E

ZSE.3) E → E(E, …, E)

ZSE.4) E → (E, …, E)

ZSE.5) E → [E, …, E]

ZSE.6) E → E[E, …, E]

ZSE.7) E → E:E

ZSE.8) E → E..E

ZSE.9) E → ‘E’

Actions

ZSA.1) A → **type** <symbol> = T;

ZSA.2) A → T <symbol> := E;

ZSA.3) A → E := E;

Attending Program Tracks Organization Search Series Sign in Sign up

SPLASH 2017 (series) / LIVE 2017 (series) / LIVE 2017 /

ZenSheet: a live programming environment for reactive computing

Who	*Enzo Alda*, *Monica Figuera*
Track	LIVE 2017
When	**Tue 24 Oct 2017 15:30 - 15:50 at Regency D** - Winter
Abstract	We introduce ZenSheet: an experimental live programming environment for reactive computing. We emphasize the most relevant design and implementation choices, and provide a glimpse about future work. The ZenSheet project aims to generalize spreadsheets in an intuitive way. It implements a superset of the core functionality of traditional spreadsheets, adding concepts from modern programming languages. The end purpose is to make spreadsheet computing valuable to an even wider audience spectrum.
File attachments	PDF Preprint (ZenSheet.pdf) — 475KiB

Enzo Alda
Lakebolt Research
United States

Monica Figuera
Universidad Simón Bolívar

lambda DAYS

13–14 FEBRUARY 2020
KRAKÓW | POLAND

ZenSheet™ Studio X

Test · Load · Save · Cycle · Tick · Reset

EXPRESSION

α + β

SYMBOLS

W
start
step

Worksheet

W

	@	A	B	C	D
1		x	sin	cos	check sin & cos
2		0.189434686	0.188303724	0.982110843	1
3		0.389434686	0.379665490	0.925123838	1
4		0.589434686	0.555891191	0.831255065	1
5		0.789434686	B1($A5)	0.704246776	1
6		0.989434686	0.835715663	0.549162390	1
7		1.189434686	0.928158714	0.372184633	1
8		1.389434686	0.983599007	0.180369049	1
9		1.589434686	0.999826311	-0.018637280	1
10		1.789434686	0.976193695	-0.216900599	1
11		1.989434686	0.913643317	-0.406516776	1

start	step
0.18943468560200596	0.2

Delete

```
W!B3:C11 := formula(W!B2:C2);
```

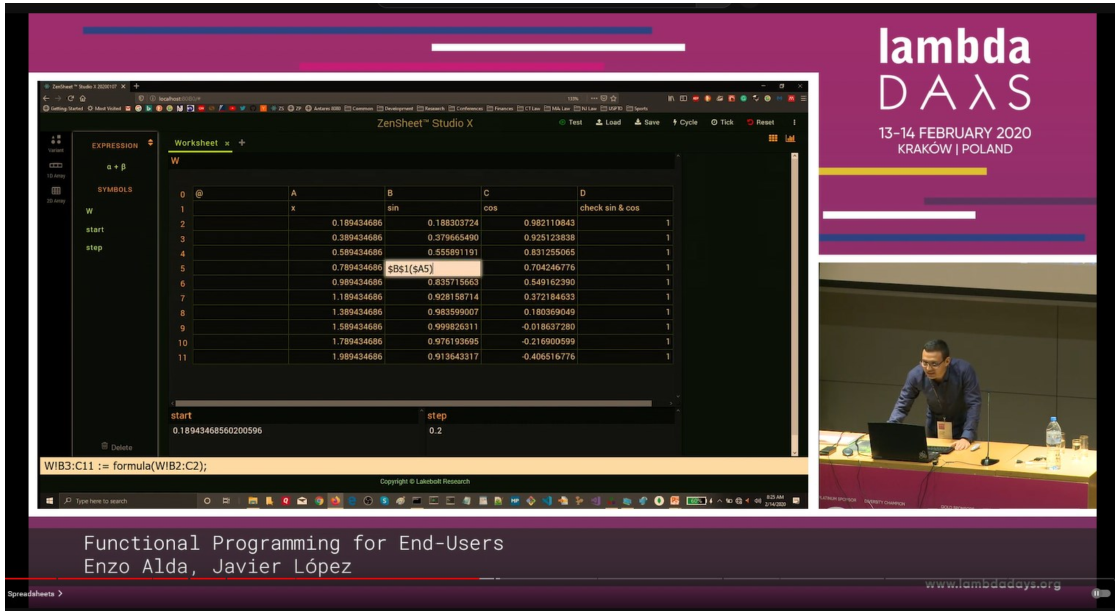

Functional Programming for End-Users
Enzo Alda, Javier López

lambda
DAΛS

SPEAKERS PROGRAMME WORKSHOPS SPONSORS VENUE CONTACT OTHER EVENTS

ENZO ALDA

functional programming
for everyone!

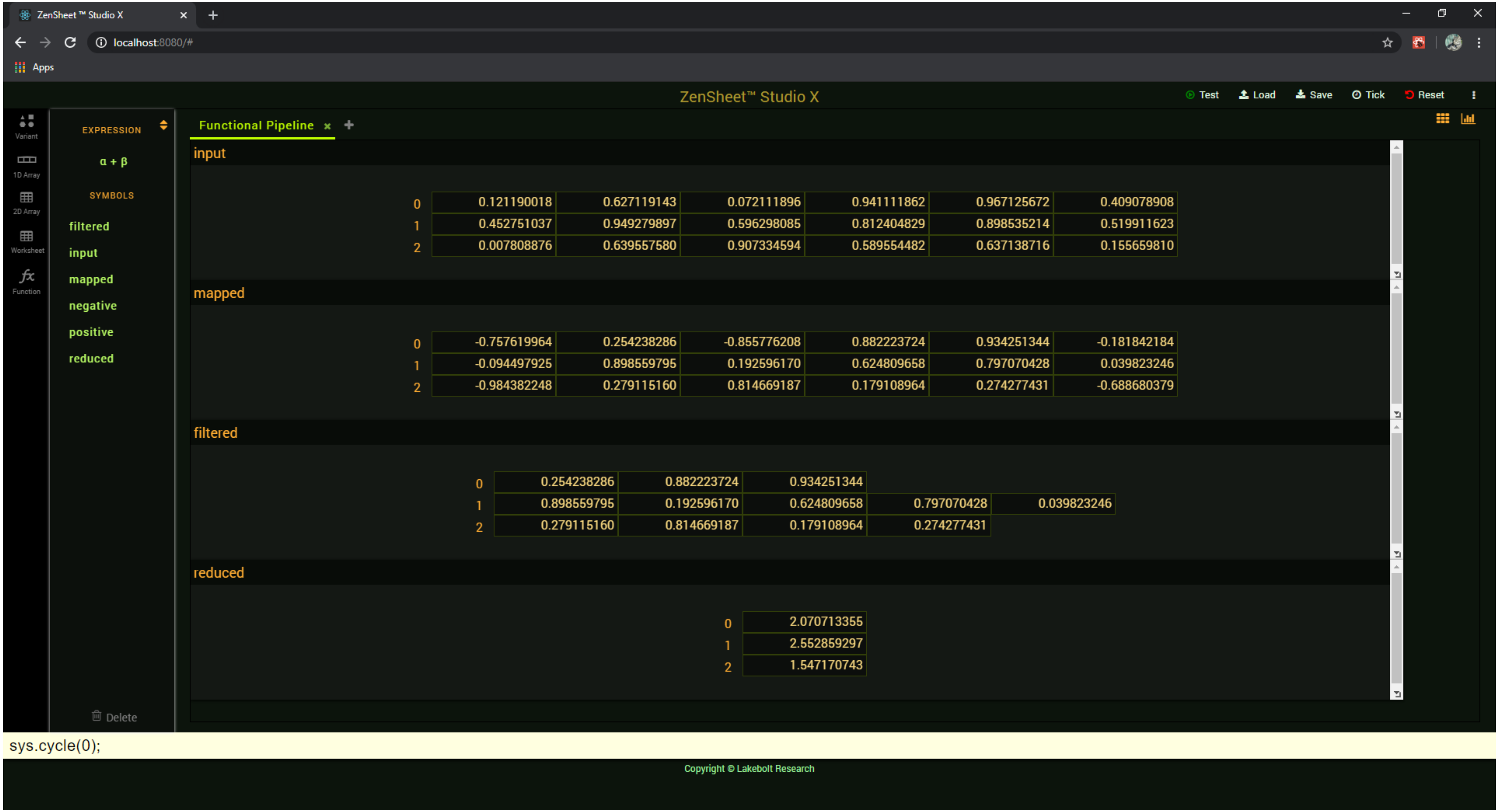


https://www.youtube.com/watch?v=mJa0_gKE6xo

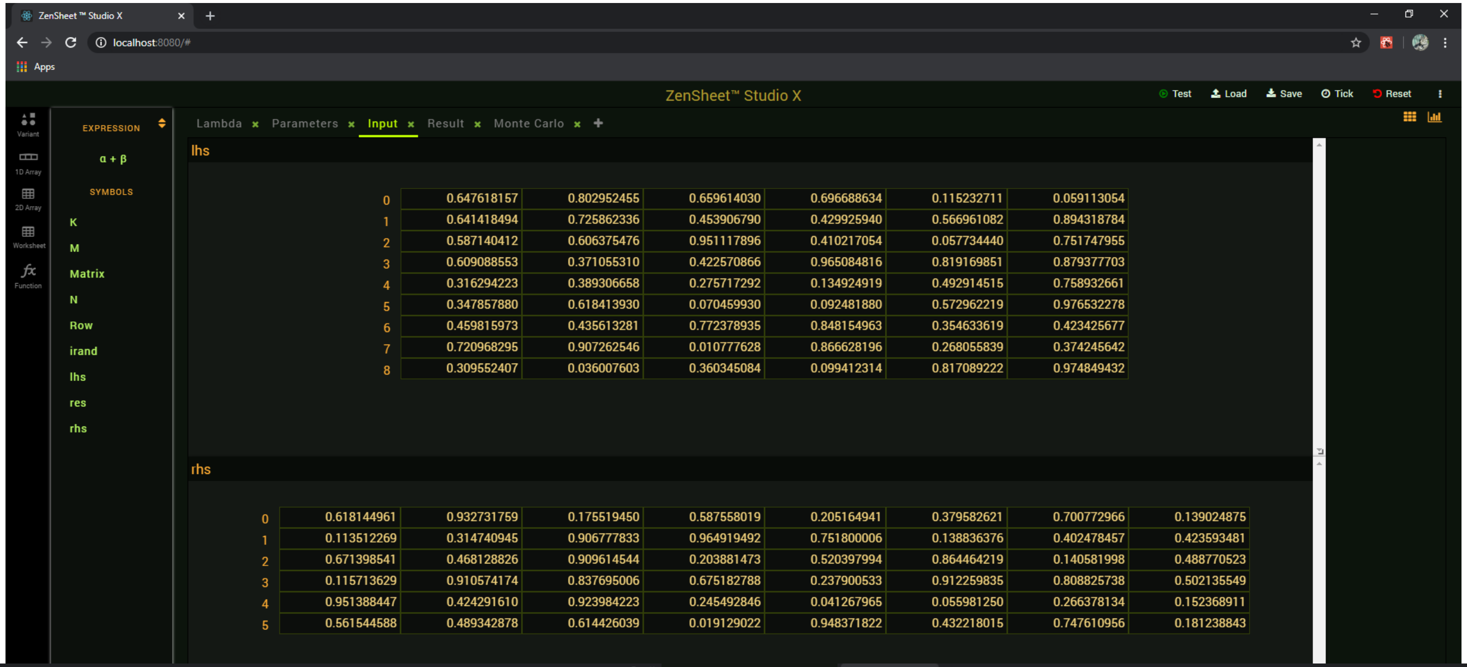
JAVIER LÓPEZ

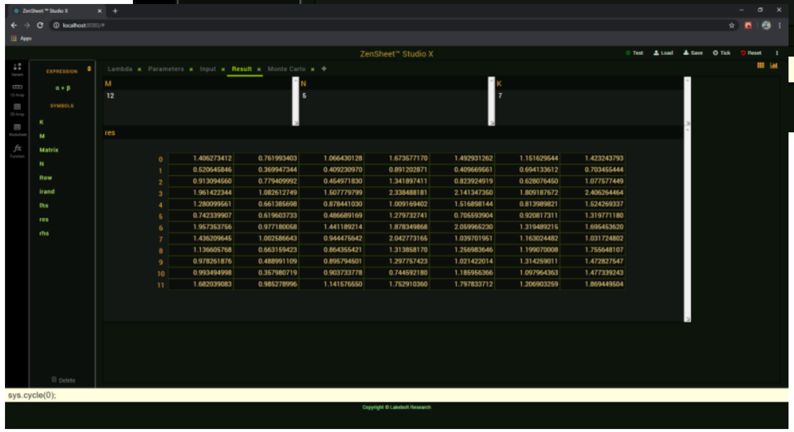
For the love of compilers,
interpreters, and dragons
therein

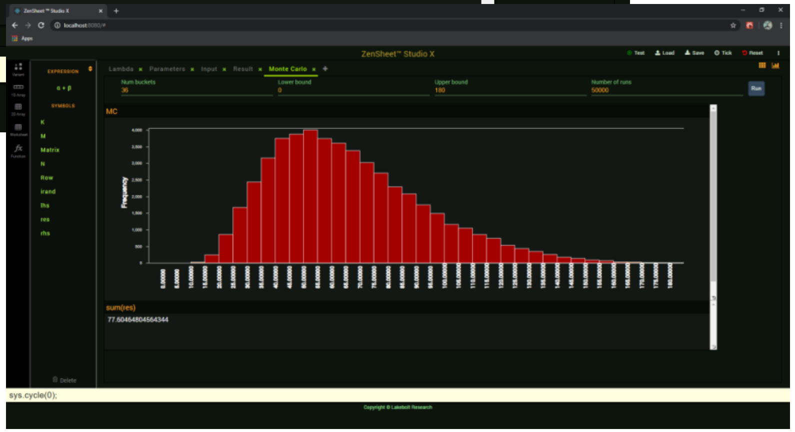
Higher-Order Functions



Rich Type System: Dynamic Arrays , Tuples, Structs







Wide-Spectrum Computing

Experience Report:
Expressions-First Programming 101

Experience Report:

Modernizing Programming 101 with an Expressions First Approach

- **Setting**
  - **Location**: Simón Bolívar University – Venezuela
    - Ranked top 5 among Latin American universities in the 80s
  - **Course**: Programming 101 for engineers, excluding software engineering students
    - Nearly all students have no programming experience
  - **Delivery**: interactive online (programming theory & lab) plus recorded video
    - Only 10 usable weeks (20 lectures) for theory and 8 usable weeks (16 hours) for lab instruction
    - Students set up their own lab, with material and some technical assistance from the lab instructor
    - Key tools: Git & GitHub (https://github.com/), MSYS2 (https://www.msys2.org/), code editors

Experience Report:
Modernizing Programming 101 with an Expressions First Approach

- Objectives
  - **Old** course objectives:
    - students must learn basic C programming
    - emphasis is placed on array processing and I/O
  - **New** course objectives: form students who exhibit
    1. A clear understanding of **basic programming concepts**
    2. Confidence in their **ability to learn other languages**
    3. **Basic proficiency** in one or more languages

Expressions

Math	C++	JavaScript	Lilly
$a(c+b) = ac + ab$	a * (b + c) == a*c + a*b	a * (b + c) == a*c + a*b	a * (b + c) = a*c + a*b
$\frac{1}{\left(1+\frac{1}{n}\right)^n}$	1 / pow(1 + 1.0/n, n)	1 / (1 + 1/n)**n	1 / (1 + 1/n)^n
πr^2	M_PI * pow(r, 2)	Math.PI * r**2	pi() * r^2
$sin^2 x$	pow(sin(x), 2)	Math.sin(x)**2	sin(x)^2
$sin\, x^2$	sin(pow(x, 2))	Math.sin(x**2)	sin(x^2)
$sin\, sin\, x$	sin(sin(x))	Math.sin(Math.sin(x))	sin(sin(x))

Functional Abstraction
dynamically typed languages

Math	JavaScript	Lilly
$A = \pi r^2$	let area = r => Math.PI * r**2;	:: area := fn(r) -> pi() * r^2;
		type Coord = struct { x; y; };
	let point1 = { x: 3, y: 2 }; let point2 = { x: 6, y: 6 };	:: point1 = Coord(3, 2); :: point2 = Coord(6, 6);
$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$	**function distance (p1, p2) { return Math.sqrt((p1.x – p2.x)**2 + (p1.y – p2.y)**2) }**	**:: distance := fn (p1, p2) -> sqrt((p1.x – p2.x)^2 + (p1.y – p2.y)^2);**
	distance(point1, point2)	distance(point1, point2)

Functional Abstraction

statically typed language

Math

$$A = \pi r^2$$

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

C++

```
double area(double r) {
    return M_PI * pow(r, 2);
}

struct Coord {
    double x;
    double y;
};

Coord point1 = { 3, 2 };
Coord point2 = { 6, 6 };

double distance (Coord p1, Coord p2) {
    return sqrt(pow(p1.x – p2.x, 2) + pow(p1.y – p2.y, 2));
}
```

Lilly

```
:: area := fn(double r) => double -> pi() * r^2;

type Coord = struct {
    double x;
    double y;
};

:: point1 = Coord(3, 2);
:: point2 = Coord(6, 6);

:: distance := fn (Coord p1, Coord p2) => double ->
                sqrt((p1.x – p2.x)^2 + (p1.y – p2.y)^2);
```

Inductive Math definitions => Recursion

```
/// fibonacci
/// fibo(0) ==> 0
/// fibo(1) ==> 1
/// fibo(n) ==> fibo(n-1) + fibo(n-2) // ... for n > 1

// C++
int fibo(int n) {
  return n < 2 ? n : fibo(n - 1) + fibo(n - 2);
}

// JavaScript
let fibo = n => (n < 2 ? n : fibo(n - 1) + fibo(n - 2));

// Lilly (dynamically typed)
:: fibo := fn(n) -> if(n < 2, n, fibo(n - 1) + fibo(n - 2));

// Lilly (statically typed)
:: fibo := fn(int n) => int -> if(n < 2, n, fibo(n - 1) + fibo(n - 2));
```

Expressions-First Course Plan

Level 0

- Values and basic types
- Computing expressions using literal constants
- Variable definitions: using variables in expressions
- Computing with simple values and tuples
- Defining functions by abstracting expressions

Level 1

- Making algorithmic decisions: conditional expressions
- Defining recursive functions from inductive definitions
- Computing functional reductions over sequences
- Understanding tail recursion and iteration

Level 2

- Higher order functions
- Curry transformations
- Nested functions and scope rules

Level 3

- Pointers and references
- Associative maps and mutable arrays
- Persistence

Level 4

- Recursive types: trees
- Sum types: dealing with polymorphism
- Polymorphism in dynamically typed languages
- Polymorphism in statically typed languages
- Polymorphism in functional languages

Experience Report:
Modernizing Programming 101 with an Expressions First Approach

Findings

So far, students have shown:

1. a practical understanding of the **difference between dynamically typed and statically typed languages**, well beyond a merely abstract notion, having been exposed to JavaScript, C++, and Lilly for several weeks;

2. a good grasp of the **difference between the functional and imperative programming styles**, due to spending the first 4 weeks without performing any state mutation besides defining and initializing immutable variables, essentially using an SSA-form (single static assignment) style of programming in JavaScript, C++, and Lilly;

3. ability to **code in JavaScript and C++**, completing programming assignments for the levels covered so far.

References

- [1] Lamb, Annette and Johnson, Larry April 2011. “Scratch: Computer Programming for 21st Century Learners.” *Teacher Librarian Magazine* 38, 4; ProQuest Central
- [2] Hermans, Felienne August 2020. “Hedy: A Gradual Language for Programming Education.” *ICER ’20* Virtual Event, New Zealand
- [3] Wolfram, Conrad 2011. “Teaching kids real math with computers.” *TED Talk.* https://www.youtube.com/watch?v=60OVlfAUPJg
- [4] Wolfram, Conrad 2014. “Stop Teaching Calculating, Start Learning Maths!” *WISE Talk: World Innovation Summit for Education.* https://www.youtube.com/watch?v=xYONRn3EbYY
- [5] Peyton Jones, Simon; Blackwell, Alan; Burnett, Margaret 2003. “A User-Centred Approach to Functions in Excel.” *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP ’03)* ACM, New York, NY, USA, 165–176. https://doi.org/10.1145/944705.944721
- [6] Alda, Enzo and Figuera, Monica October 2017. “ZenSheet: a live programming environment for reactive computing.” *SPLASH’17, LIVE 2017 Workshop*. Vancouver, British Columbia, Canada. https://2017.splashcon.org/details/live-2017/5/ZenSheet-a-live-programming-environment-for-reactive-computing
- [7] Figuera, Monica October 2017. “ZenSheet Studio: A Spreadsheet-Inspired Environment for Reactive Computing.” *SPLASH’17 SRC - SPLASH Companion’17.* Vancouver, British Columbia, Canada. ACM ISBN 978-1-4503-5514-8/17/10. https://doi.org/10.1145/3135932.3135949
- [8] E. Alda et al. 2021. "Towards Wide-Spectrum Spreadsheet Computing." *4th International Conference on Information and Computer Technologies (ICICT).* HI, USA, 2021, pp. 233-242. doi: 10.1109/ICICT52872.2021.00046.

Thank You!