Week 2

Mathematics and Computational Methods for Complex Systems, 2023

Dhruva V. Raman

How is notebook 1 going?

Too much

Too little

Too slow

Too fast

Are you using the padlet?

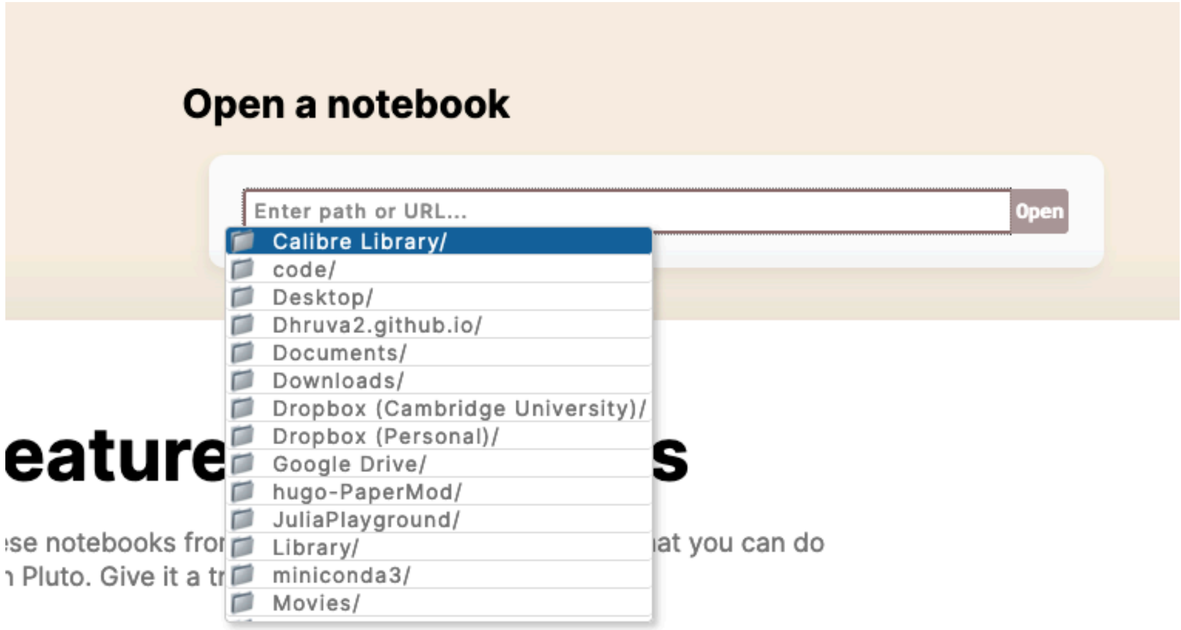
(No)

https://uofsussex.padlet.org/draman2/mcmcs-23-24-nobpvtdw6hzblkkg

Have you read the canvas/course website info?

Some week 1 issues

Pluto notebooks open in strange folder



Some week 1 issues

Pluto notebooks open in strange folder

Solution

```
julia> using Pluto

shell> cd
 /Users/dr360
```

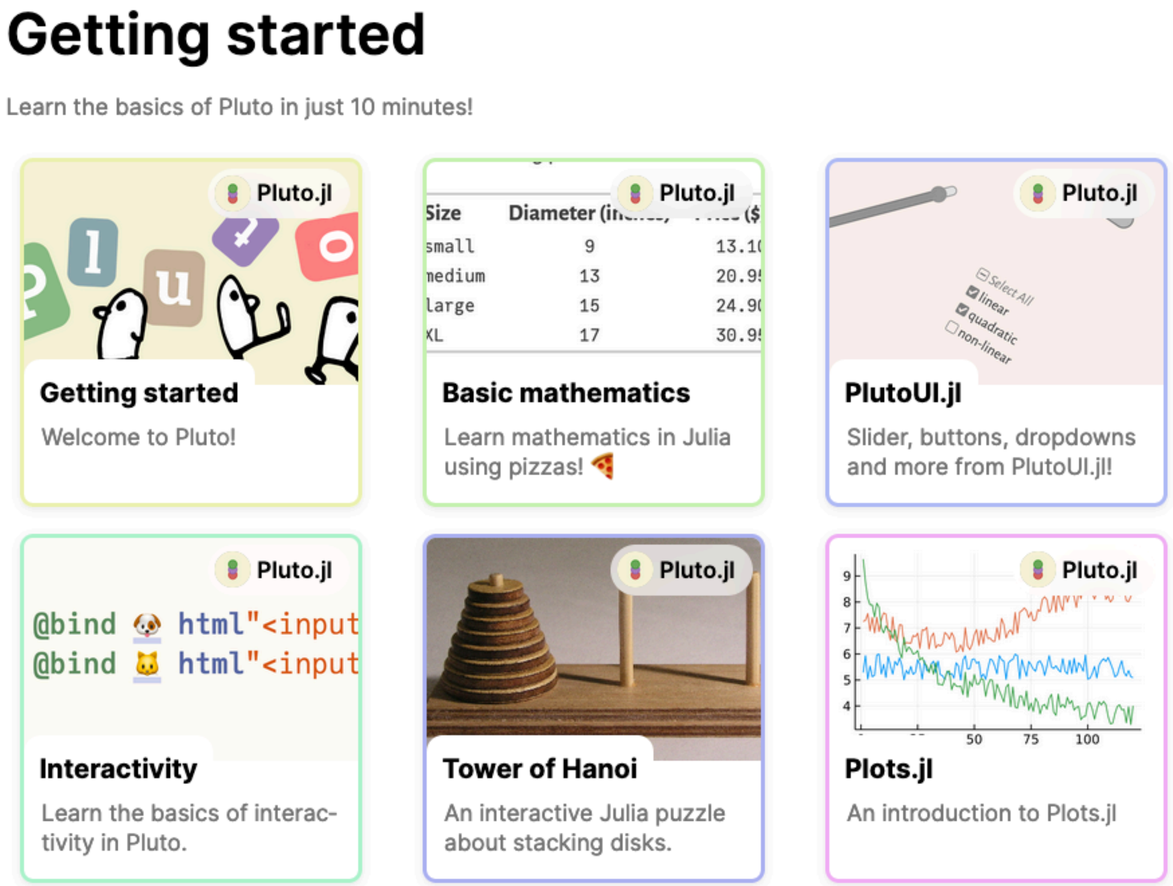
Semicolon ; to switch from Julia to shell

cd = change to home directory

Then open Pluto with Pluto.run()

Some week 1 issues

Sliders not working



Packages

augment languages

```
using Random ✓
```

```
UndefVarError: `eigen` not defined

  1. top-level scope @ Local: 1
```

```
eigen
```

Packages

augment languages

```
using Random
```

```
UndefVarError: `eigen` not defined

  1. top-level scope @ Local: 1
```

```
eigen
```

```
using LinearAlgebra , Random
```

```
eigen (generic function with 17 methods)
```

```
eigen
```

Packages

augment languages

eigen is exported by LinearAlgebra.jl

Principle: only load functions you need!

Avoid confusion from too many function names

```
using LinearAlgebra ✓ , Random ✓

eigen (generic function with 17 methods)

eigen
```

Name conflicts?

LinearAlgebra

eigen

DhruvaAlgebra

eigen

Name conflicts?

LinearAlgebra

eigen

DhruvaAlgebra

eigen

eigen ✗

…use full name: LinearAlgebra.eigen

Week 1 issues

Functions vs methods

```
concatenate (generic function with 2 methods)
function concatenate(a,b)
    println("concatenated two things")
    return [a,b]
end
```

Drag to move cell

```
concatenate (generic function with 2 methods)
function concatenate(a::Number,b,c)
    println("concatenated three things")
    return [a,b,c]
end
```

```
[1, 2, 3]
concatenate(1,2,3)
```

```
concatenated three things
```

Some week 1 issues

Flying planes can be dangerous!

Noam Chomsky (1965)
Aspects of the Theory of Syntax

Some week 1 issues

Try to state things twice, in complementary ways, especially when giving a definition

Donald Knuth (1987)
Mathematical writing

Language types

	NATURAL	PROGRAMMING
Ambiguous	X	
Redundant	X	
Literal		X

Some week 1 issues

```
julia> using pluto
ERROR: ArgumentError: Package pluto not found in current path.
- Run `import Pkg; Pkg.add("pluto")` to install the pluto package.
Stacktrace:
```

```
julia> using Pluto

julia>
```

Debugging

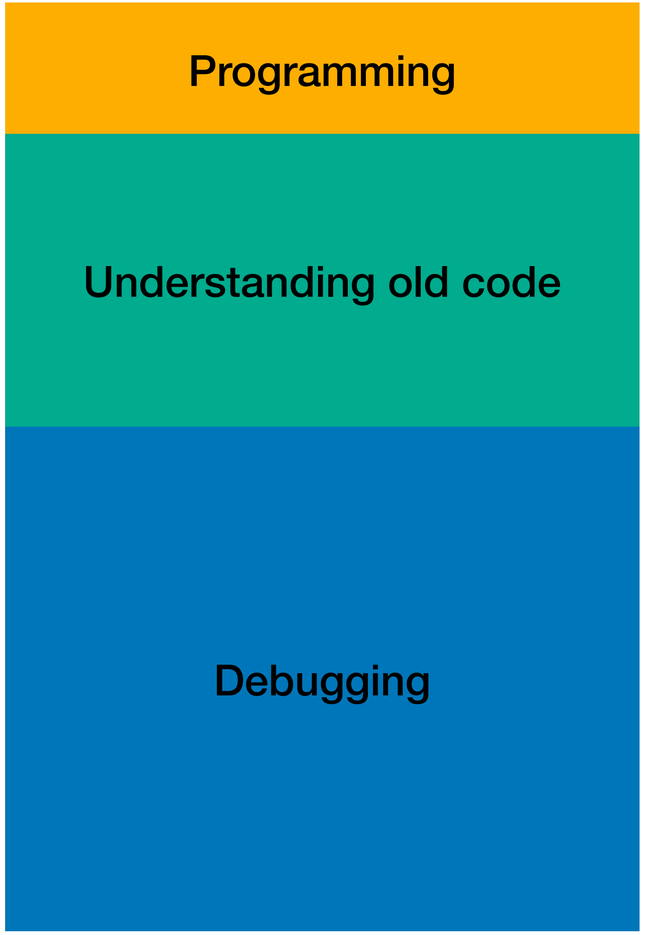

WHAT ARE YOU WORKING ON?
TRYING TO FIX THE PROBLEMS I CREATED WHEN I TRIED TO FIX THE PROBLEMS I CREATED WHEN I TRIED TO FIX THE PROBLEMS I CREATED WHEN...

Debugging

Good debugger

>

Good programmer



Debugging

WIKIPEDIA
The Free Encyclopedia

Search Wikipedia | Search

Rubber duck debugging

Article Talk

From Wikipedia, the free encyclopedia

Contents [hide]

(Top)

The Duck House Brighton
https://www.theduckhousebrighton.online

Home | The Duck House Brighton | Rubber Duck Shop

Welcome to The Duck House! Here you will find over 400 different types of imaginative **rubber ducks**. There's one for everyone! Browse our website or come and ...

All Ducks
There's a Rubber Duck for Everyone! Black Facebook Icon ...

Most Popular
There's a Rubber Duck for Everyone! · DC- Catwoman ...

About Us
At The Duck House you will find over four hundred different ...

Special Occasions
There's a Rubber Duck for Everyone! · Love Eco Rubber ...

More results from theduckhousebrighton.online »

Goals

Mathematics and Computational Methods for Complex Systems, 2023

How do computers do maths?

How do we make sure computer maths is efficient?

Getting the answer is not enough

Mathematics and Computational Methods for Complex Systems, 2023

We often know how to analyse a complex system

Writing an efficient algorithm is the bottleneck

Structure

What are programs? How are they run?

Interpreted vs compiled languages

Why Julia?

Memory models for programming

Allocations

Types

….next week maths!

What is a computer program?

Inputs

Keyboard
File
Network
Other device

Program

Outputs

Display on screen
Save in file
Send over network

CPU
(Central processing unit)

What is a computer program?

Program

Basic maths operations
(+,-,…)

Conditional execution
(run code if … is true)

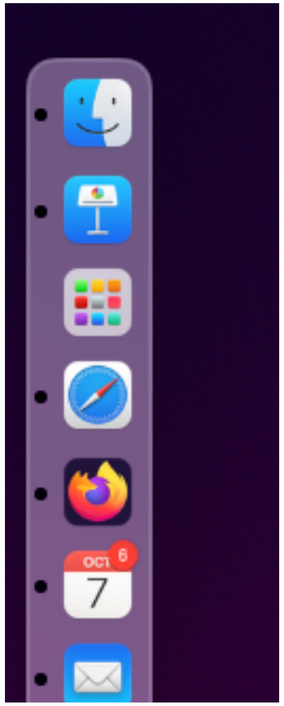
Repetition
(run things repeatedly,
with varationse)

….that’s it!!!

How do we run a program?

Graphical user interface (GUI)

…you click!

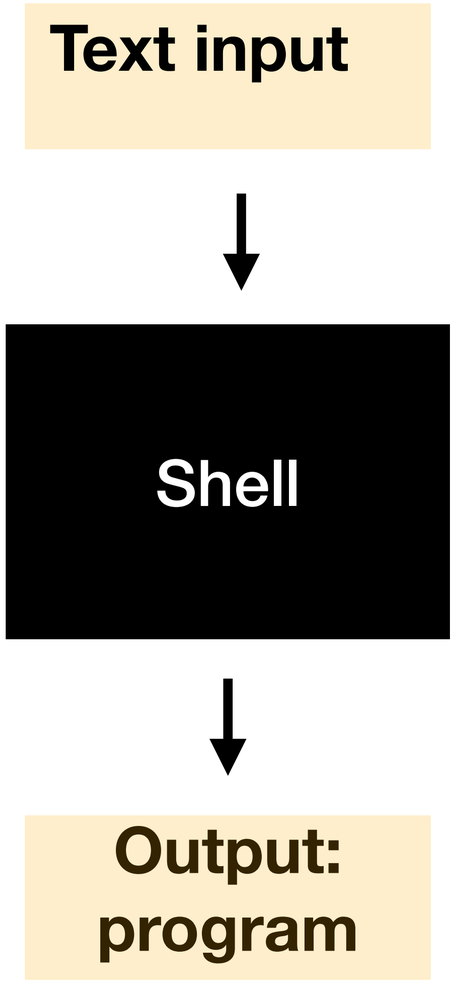


How do we run a program?

Command line interface (CLI)

Terminal / powershell

… a program that runs programs!



How do we run a program?

Shell

```
(base) ➜  ~ julia some_program.jl
```

What is a script?

A list of instructions, with no defined input or output

```
[9, 10]
begin
    x = 4
    y = [5,6]
    z = x .+y
end
```

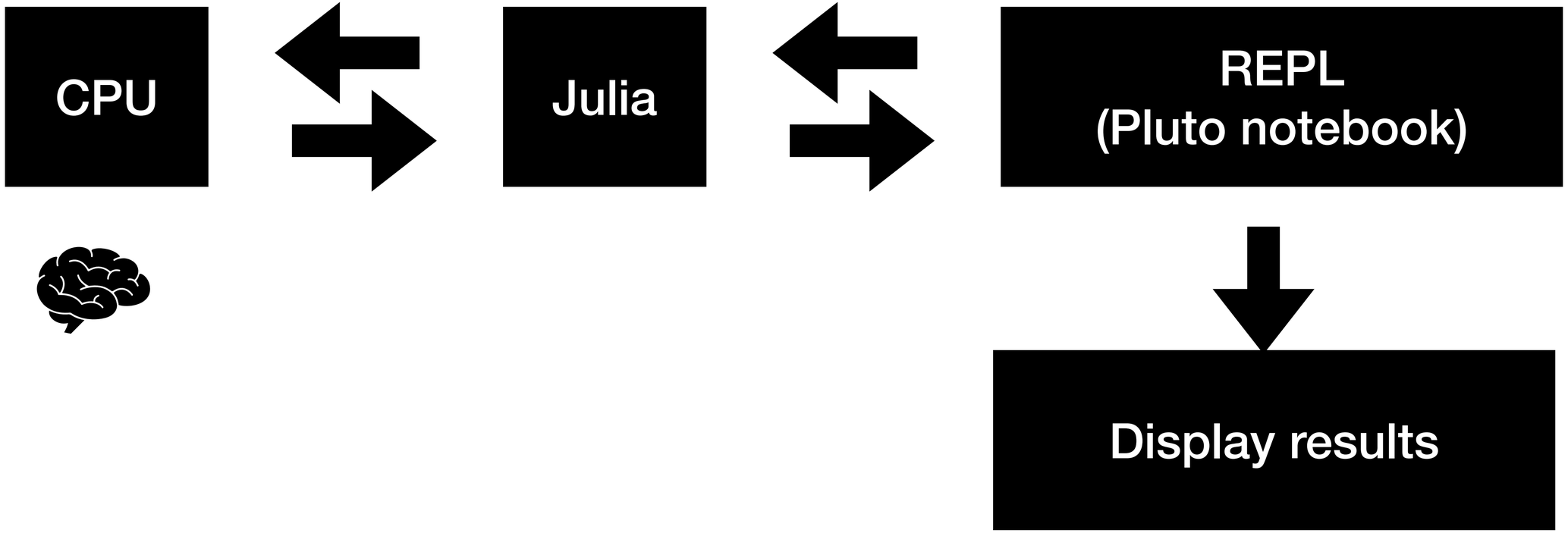
(Output of last line is shown, unless suppressed with semicolon **;**)

```
begin
    x = 4
    y = [5,6]
    z = x.+y
end;
```

Scripts are interpreted by a REPL program

(Read-eval-print loop)

```julia
begin
    x = 4
    y = [5,6]
    z = x.+y
end;
```



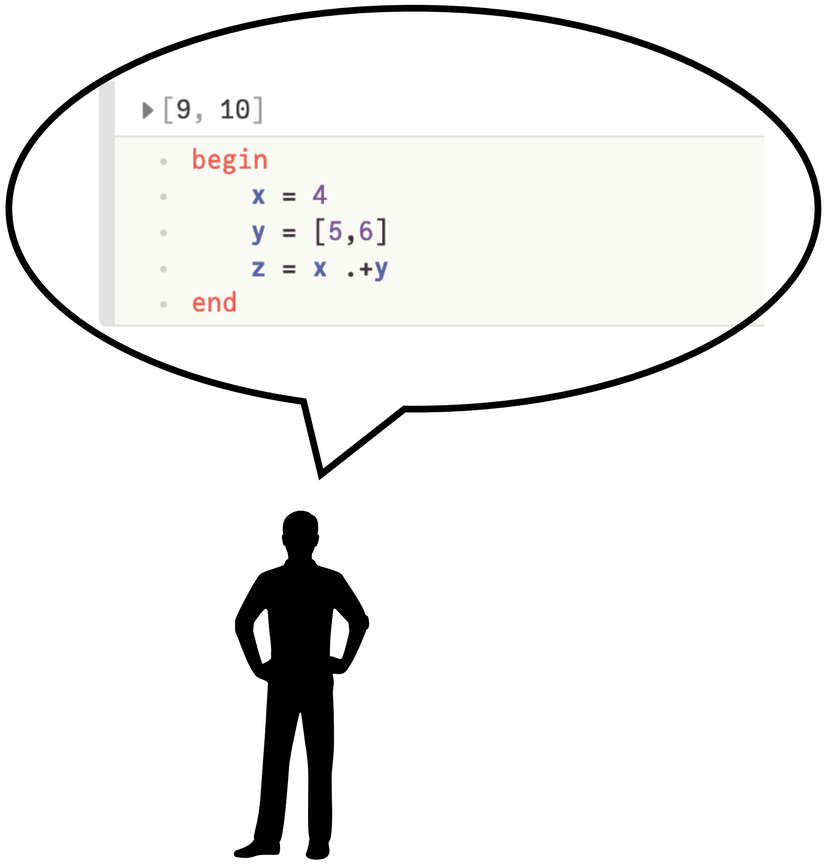
REPL scripts are for exploratory programming

Python	IPython3
Julia	The Julia REPL / Pluto
Eg C++	**Nothing**

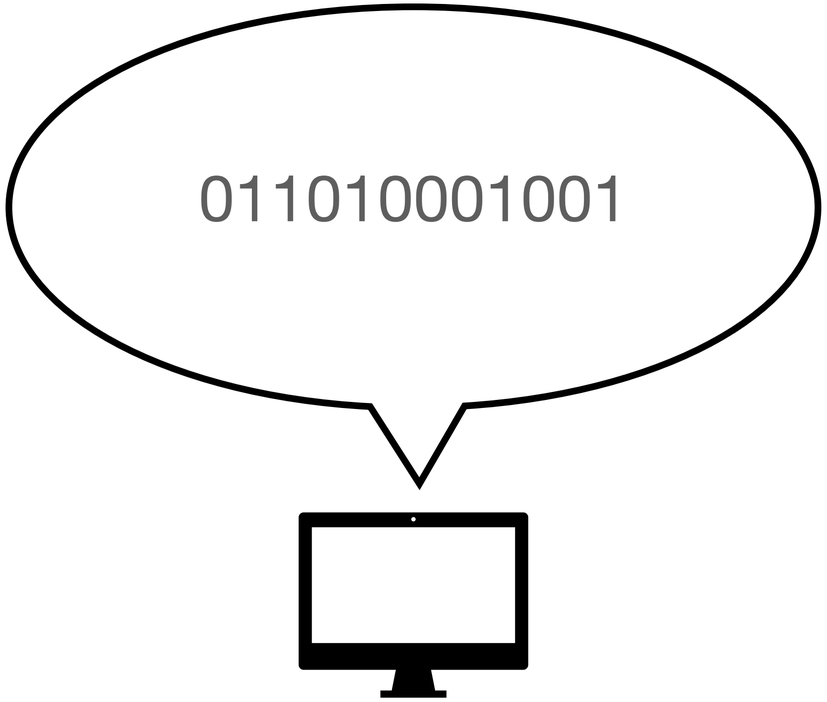
Not all programming is in the REPL!

But that's all we'll use in MCMCS…

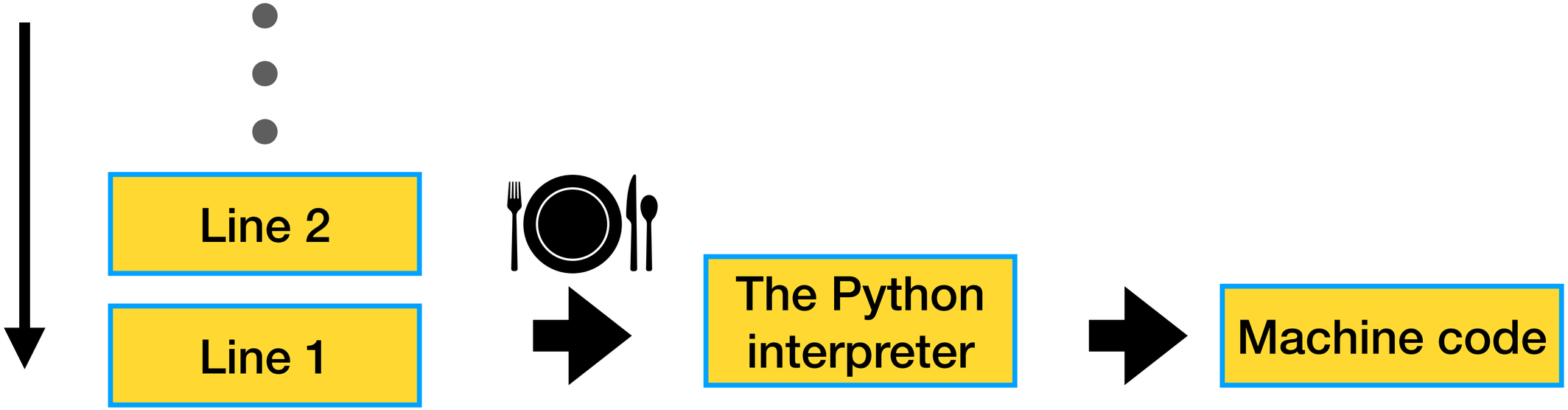
Running programs



Machine code



Interpreted languages

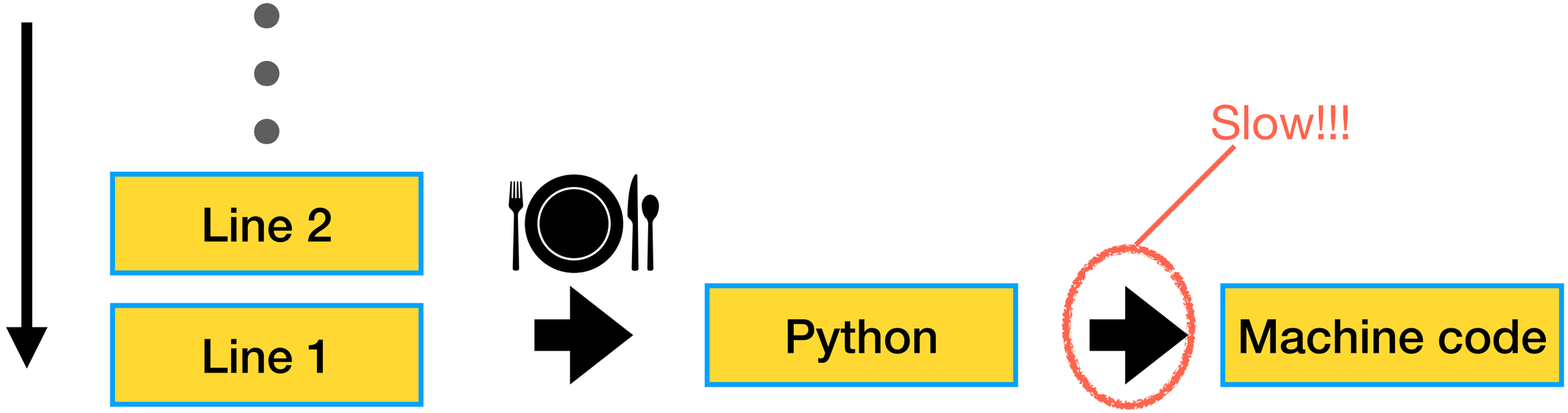


Interpreted languages

```
▶ [9, 10]
```

```
begin
    x = 4
    y = [5,6]
    z = x .+y
end
```

Interpreted languages



Interpreted languages

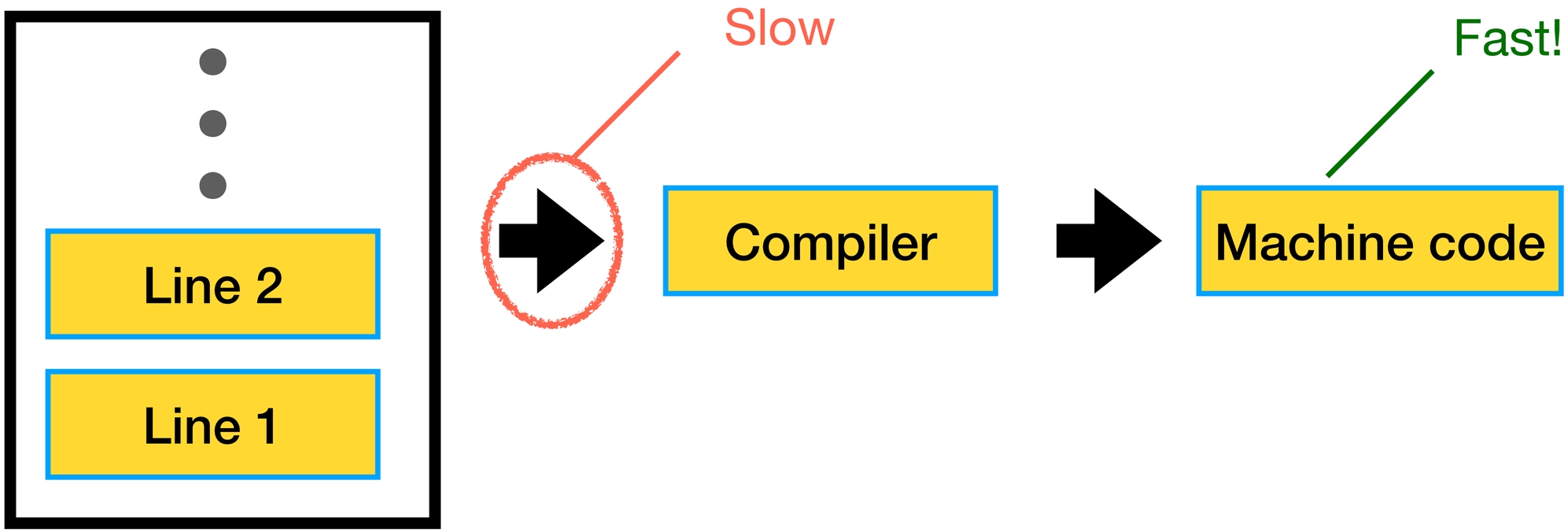
...should never do this!

```
lots_of_lines (generic function with 1 method)
```

```julia
function lots_of_lines(a::Int64)
    for i = 1:100000
        a +=1
    end
end
```

Executing each line takes time. x100000

Compiled languages
...are much faster



Compiled languages

...will happily munch this

```
lots_of_lines (generic function with 1 method)
```

```julia
function lots_of_lines(a::Int64)
    for i = 1:100000
        a +=1
    end
end
```

Compiled language example

C/C++

Your code

gcc is a compiler program that produces an executable

```
gcc sourcefile_name.c -o outputfile.exe
```

Unreadable to humans, but computer will run it

```
outputfile.exe
```

Machine code

Production software is usually written in compiled languages

…but this isn’t a software development course

Always use compiled languages?

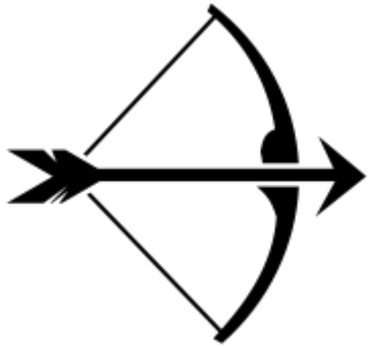
```
gcc sourcefile_name.c -o outputfile.exe
```

Every change needs recompilation!

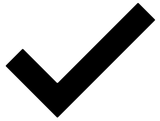
Compiling induces delayed feedback

Compiled languages are usually harder to learn

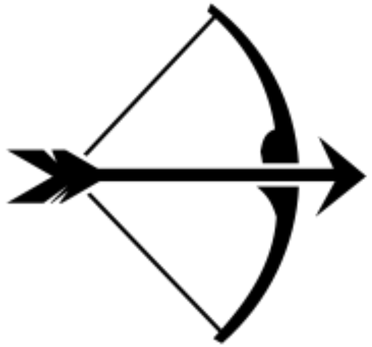
English laws



All English males over the age of 14 are to carry out two hours of longbow practice every week, supervised by the local clergy.

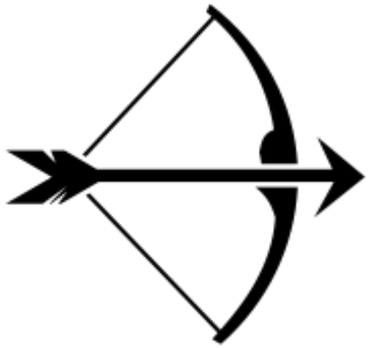


English laws



All English males over the age of 14 are to carry out two hours of longbow practice every week, supervised by the local clergy.

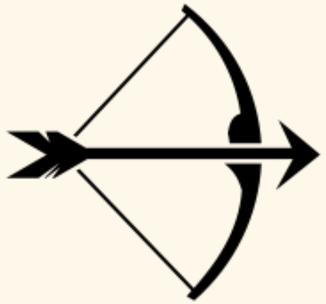
It is legal to shoot a Welshman with a longbow in Hereford, if he is within the city walls after midnight



$\geq$

?

Programming vs shooting



Compiled languages

C, C++, Fortran, Rust…

Interpreted languages

Python, Matlab, PHP, Ruby, …

Buying computing power is cheaper than training a programmer

New laws?

All English residents over the age of 14 are to carry out two hours of coding practice every week, supervised by the local TA's?

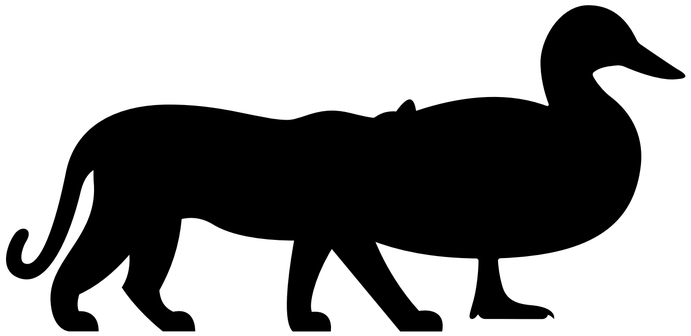
New laws?

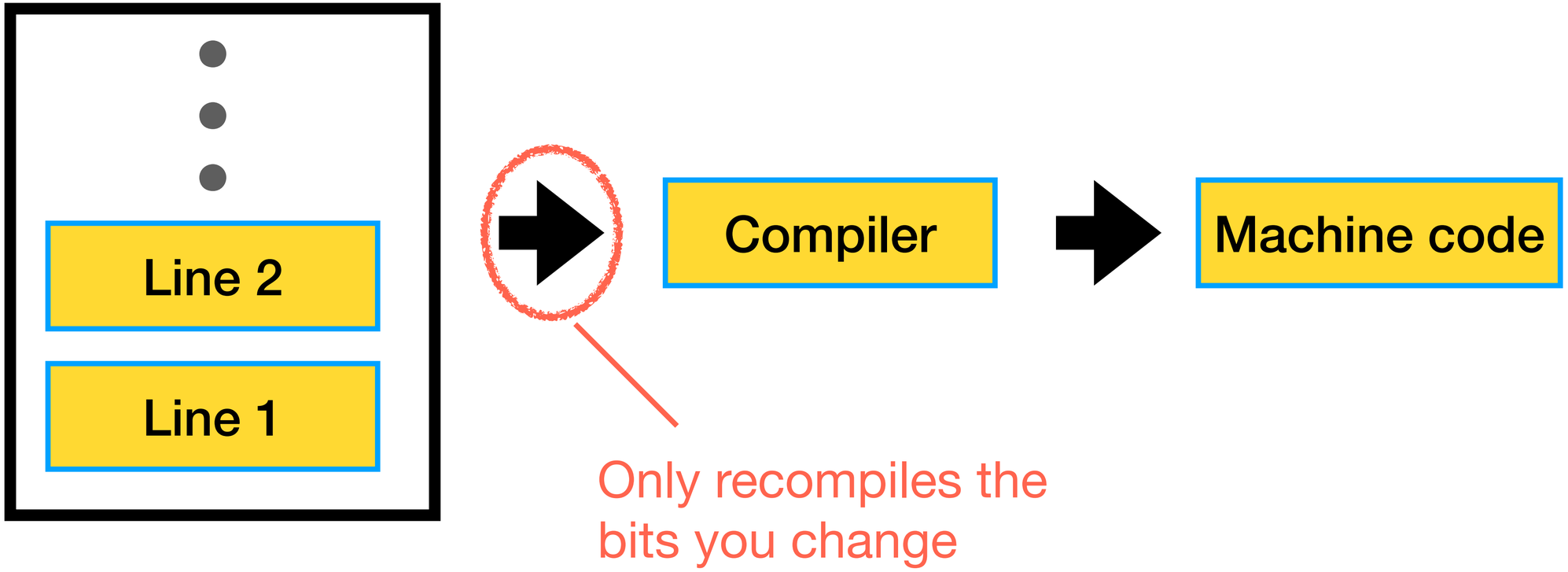
All English residents over the age of 14 are to carry out two hours of coding practice every week, supervised by the local TA's?

⬇

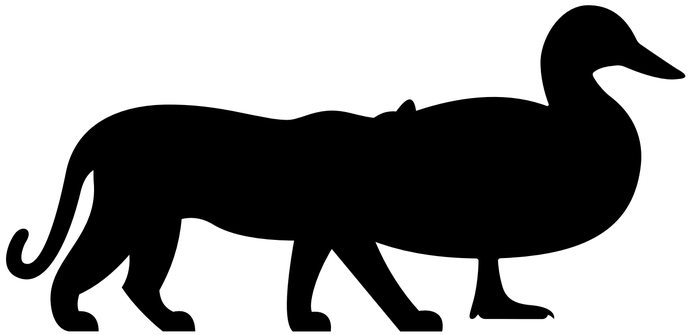
Compiled languages would be more popular!

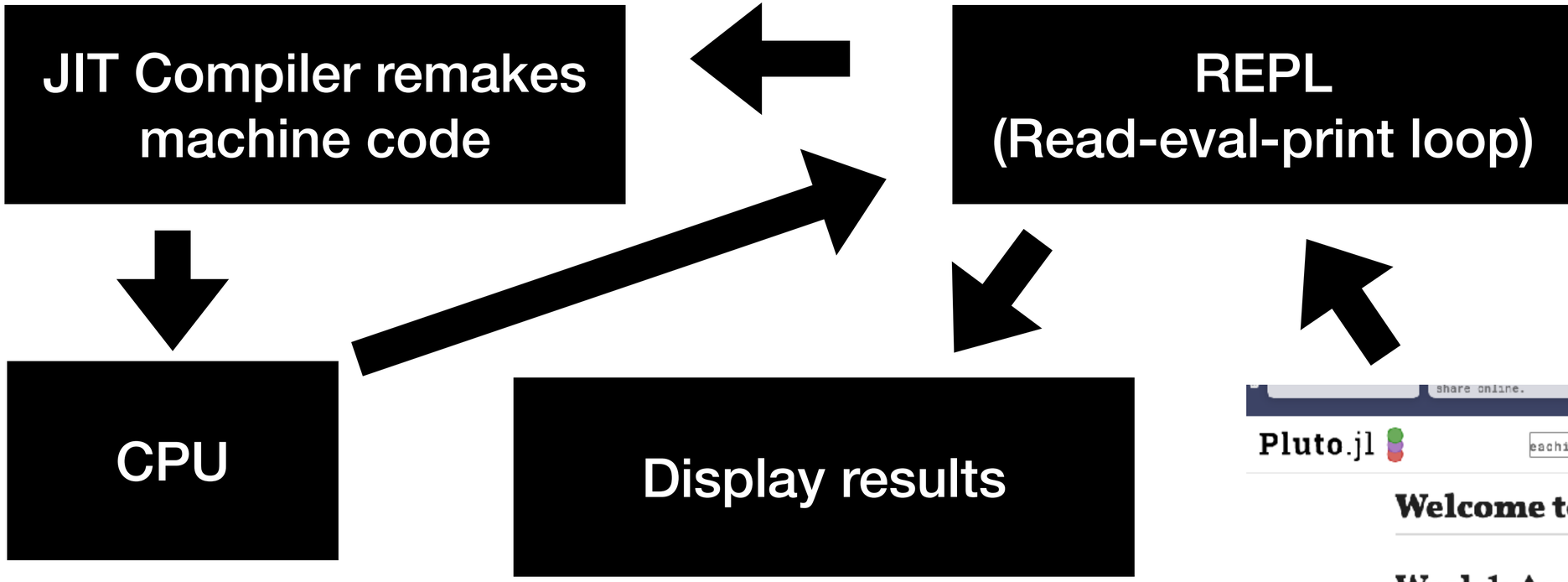
Julia is a just-in-time (JIT) compiled language

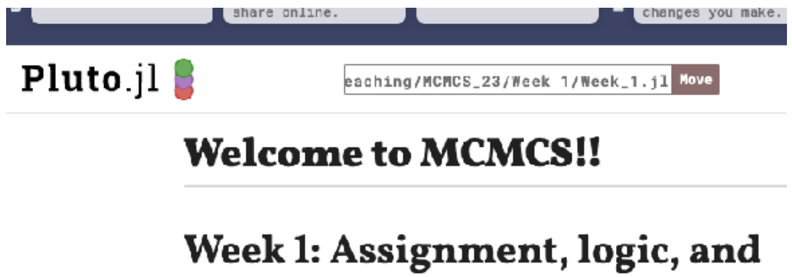




Julia is a just-in-time (JIT) compiled language







See the machine code

..which is processor dependent

```
@show @code_native lots_of_lines(0)
```

```
	.section	__TEXT,__text,regular,pure_instructions
	.build_version macos, 13, 0
	.globl	_julia_lots_of_lines_4521	## -- Begin function julia_lots_of_
lines_4521
	.p2align	4, 0x90
_julia_lots_of_lines_4521:		## @julia_lots_of_lines_4521
;  @ /Users/dr360/.julia/pluto_notebooks/Wild report.jl#==#43425e91-760d-4f5a-
8386-149a7c692e3b:1 within `lots_of_lines`
	.cfi_startproc
## %bb.0:		## %top
;  @ /Users/dr360/.julia/pluto_notebooks/Wild report.jl#==#43425e91-760d-4f5a-
8386-149a7c692e3b:2 within `lots_of_lines`
	leaq	100000(%rdi), %rax
;  @ /Users/dr360/.julia/pluto_notebooks/Wild report.jl#==#43425e91-760d-4f5a-
8386-149a7c692e3b:3 within `lots_of_lines`
	retq
	.cfi_endproc
;
		## -- End function
.subsections_via_symbols
#= /Users/dr360/.julia/pluto_notebooks/Wild report.jl#==#617452bc-ee7b-4f91-9e4
0-2ba28813ec83:1 =# @code_native(lots_of_lines(0)) = nothing
```

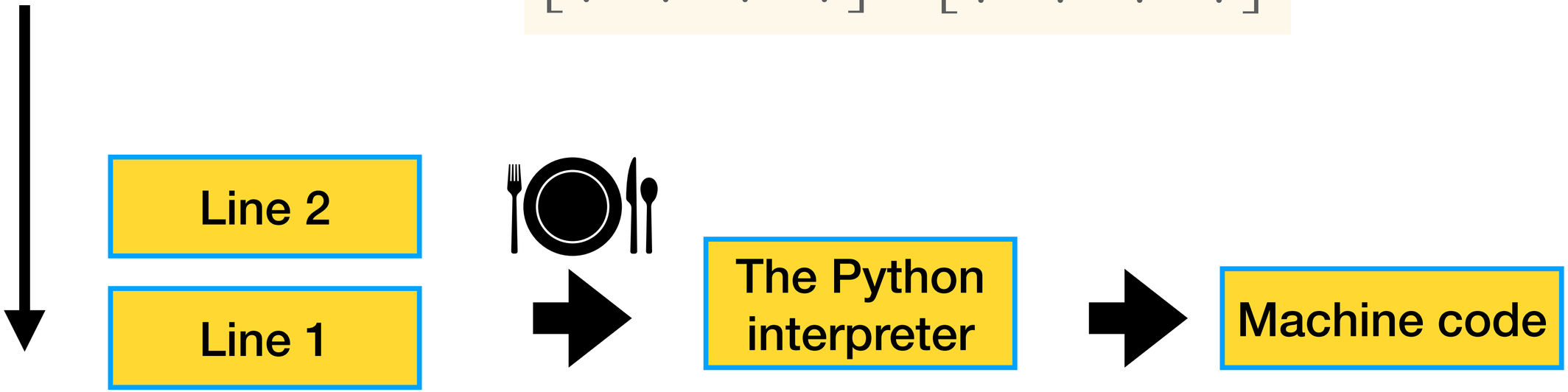
13.2 m

…and thank god we're not in the

But why do … use Python?

Lots of lines of code needed

$$\begin{bmatrix} 1 & 3 & 4 & \dots \\ 2 & 7 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \times \begin{bmatrix} 5 & 7 & 2 & \dots \\ 2 & 5 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$



But why do … use Python?

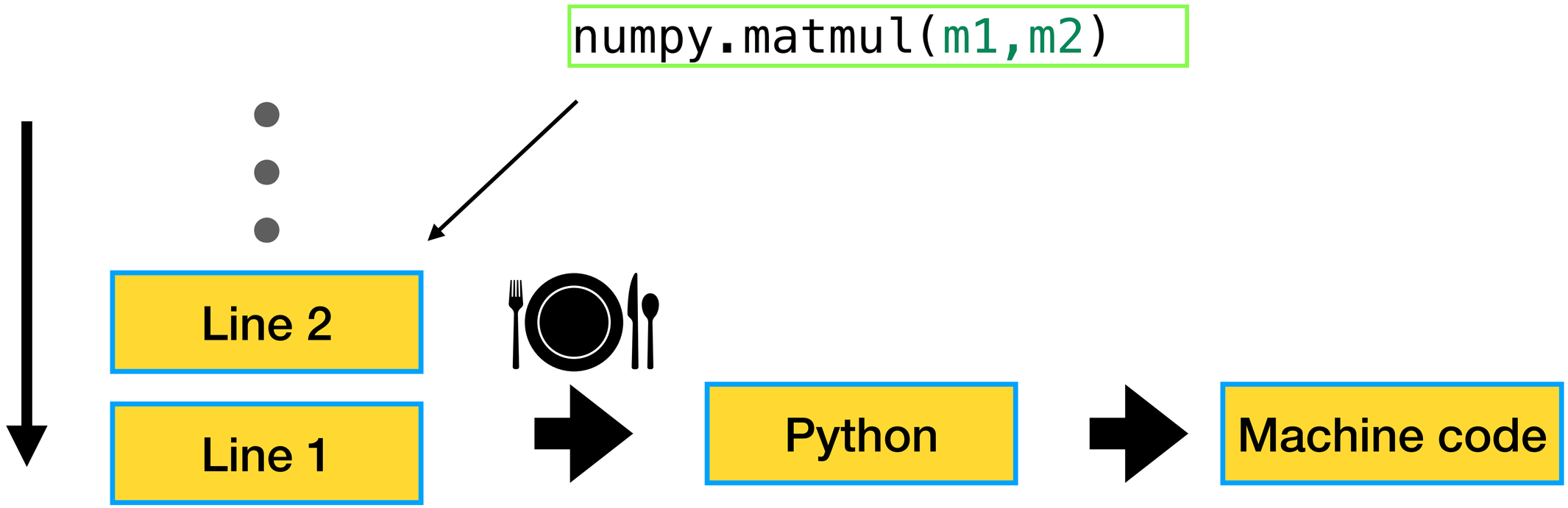
Lots of lines of code needed

$$\begin{bmatrix} 1 & 3 & 4 & \dots \\ 2 & 7 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \times \begin{bmatrix} 5 & 7 & 2 & \dots \\ 2 & 5 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

```
numpy.matmul(m1,m2)
```

This is a program
(written in fast language C++)

Python is a glue language



Ideally, each line is a specialised program
written in a fast language and doing lots of work

Writing fast code

JIT/Compiled languages

Write the algorithm yourself

Help the compiler: give it lots of info

Writing fast code

JIT/Compiled languages

Write the algorithm yourself

Help the compiler: give it lots of info

Interpreted languages

Build algorithm by gluing specialised programs

Minimise lines of code

Writing fast code

JIT/Compiled languages

Write the algorithm yourself

Help the compiler: give it lots of info

Interpreted languages

Build algorithm by gluing specialised programs

Minimise lines of code

Future? Python is turning into a JIT compiled language!!! (Mojo, Jax etc)

Other reasons we are using Julia

Transferable

Most concepts we learn have a direct analogue in Python

Closer to the metal

Easier for maths

Multilinguality is important!!

An annoying difference…

Zero-indexed languages

Python

PHP

Java

C

Ruby

One-indexed languages

Julia

Fortran

MATLAB

An annoying difference…

One-indexed languages

```
a = [2, 4, 7, 9]
```

```
a = [2,4,7,9]
```

```
4
```

```
a[2]
```

Zero-indexed languages

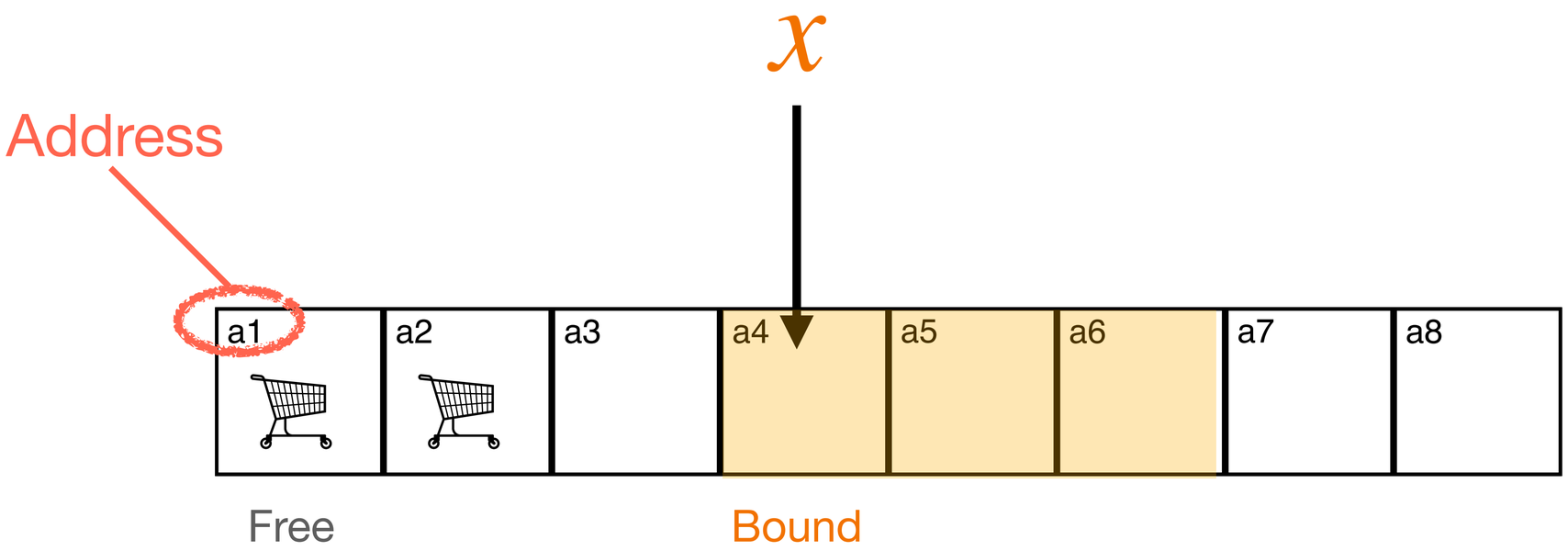
```
a[2] = 7
```

Memory models in programming

```
x = 4
```

What’s going on under the hood?

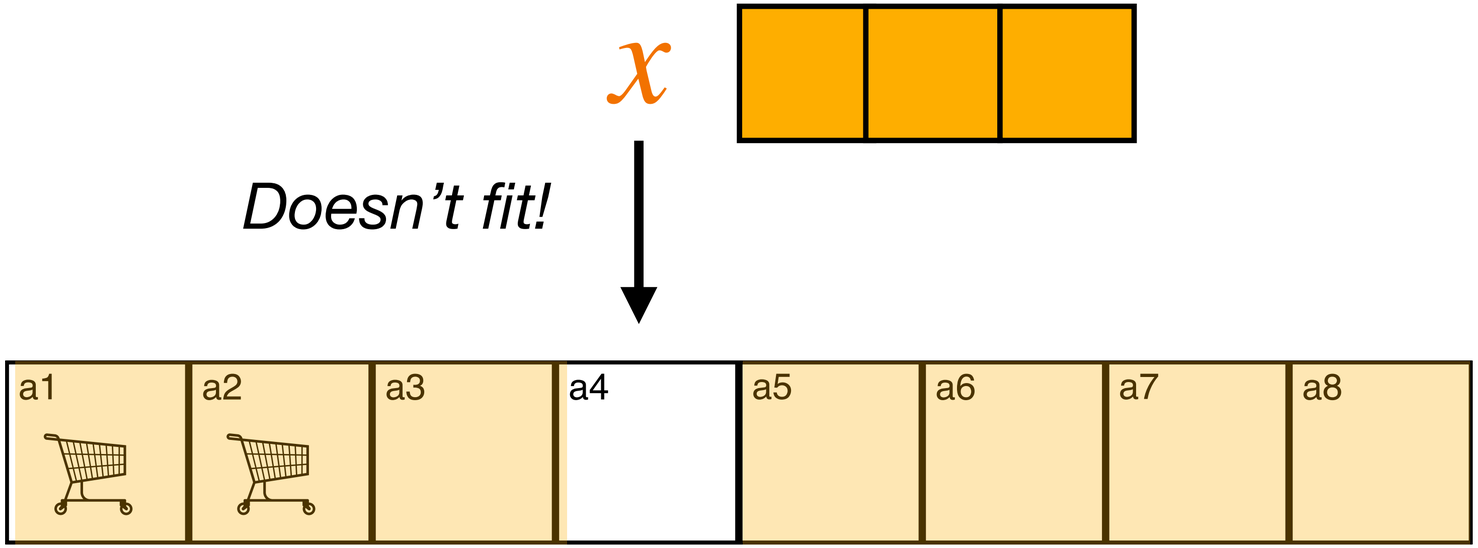
Computer storage



Computer stores each variable

Like a website address

Lots of variables?



Where do I put them!

Takes time for computer to figure out

Compiler does some organisation during compilation

(As much as possible)

Redefining x 5

```julia
@time begin
    bb = [4,5,2,6]
    for i = 1:5
        bb = 2bb
    end
end
```

```
0.000004 seconds (6 allocations: 576 bytes)
```

Important to run twice
(first run includes compilation)

```
[128, 160, 64, 192]
```

```julia
bb
```

Redefining x 5

```julia
@time begin
    bb = [4,5,2,6]
    for i = 1:5
        bb = 2bb
    end
end
```

```
0.000004 seconds (6 allocations: 576 bytes)
```

No redefining

Why would this be slow in Python?

```julia
@time begin
    aa = [4,5,2,6]
    for i = 1:5
        for j in 1:length(aa)
            aa[j] = 2*aa[j]
        end
    end
end
```

```
0.000004 seconds (1 allocation: 96 bytes)
```

Preallocation is good

Goal

aa = [1,2,3,4,5,6,7,…, 1000]

Preallocation is good

3.6 ms

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0
```

```
zeros(1000)
```

16.0 μs

```
@time begin
    aa = zeros(1000)
    for i = 1:1000
        aa[i]=i
    end
end
```

```
0.000005 seconds (1 allocation: 7.938 KiB)
```

Changing variable size is slow

Previous (preallocation)

```
@time begin
    aa = zeros(1000)
    for i = 1:1000
        aa[i]=i
    end
end
```

```
0.000005 seconds (1 allocation: 7.938 KiB)
```

New (dynamic allocation)

```
@time begin
    bb = []
    for i = 1:1000
        push!(bb,i)
    end
end
```

```
0.000023 seconds (495 allocations: 29.484 KiB)
```

Changing variable size is slow

Previous (preallocation)

```
@time begin
    aa = zeros(1000)
    for i = 1:1000
        aa[i]=i
    end
end
```

```
0.000005 seconds (1 allocation: 7.938 KiB)
```

Best = preallocation

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
```

```
@time collect(1:1000)
```

```
0.000005 seconds (1 allocation: 7.938 KiB)
```

Pass by reference

Both variables access the same block of memory

Think of a,b as shortcuts to the same website

Changing website changes output of both links

```
7
```

```
begin
    a = [3,4]
    b = a
    a[1] = 7;
end
```

```
[7, 4]
```

```
a
```

```
[7, 4]
```

```
b
```

Pass by value

b is a physical copy of the contents of a

Think of b as printing a new website

…with the same information as a

```
7
begin
    a = [3,4]
    b = deepcopy(a)
    a[1] = 7;
end
```

```
[7, 4]
a
```

```
[3, 4]
b
```

Pass by what?

```
5
begin
    c = 4
    d = c
    c = c+1
end
```

```
4
d
```

Pass by value

```
5
begin
    c = 4
    d = c
    c = c+1
end
```

```
4
d
```

Like Python

Immutable types are pass-by-value

```
false
ismutable(d)
```

Pass by reference

```
7
```

```
begin
    a = [3,4]
    b = a
    a[1] = 7;
end
```

```
[7, 4]
```

```
a
```

```
[7, 4]
```

```
b
```

Like Python

Mutable types are pass-by-reference

```
true
```

```
ismutable(a)
```

Type annotation

Every variable, in every language, has a type

```
Int64
typeof(1)
```

```
Float64
typeof(1.)
```

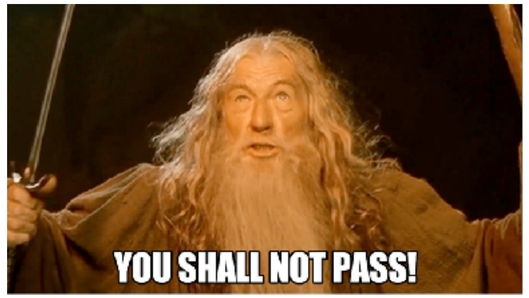
```
Float64
typeof(1.0)
```

Type annotation

x::Type is a type assertion

```
1
  1::Int64

TypeError: in typeassert, expected Float64, got a value of type Int64
  1. top-level scope @ Local: 1 [inlined]
  1::Float64
```



Operations on different types

Empathising with the compiler

Often necessary in ‘untyped’ languages like Python

```
1::Int64    +    2.0::Float64
```

1. Shit, different types!
2. Promote to a common type
3. Do the addition

Operations on different types

Empathising with the compiler

Often necessary in ‘untyped’ languages like Python

```
1::Int64    +    2.0::Float64
```

1. Shit, different types!
2. Promote to a common type
3. Do the addition

```
1.0::Float64    +    2.0::Float64
```

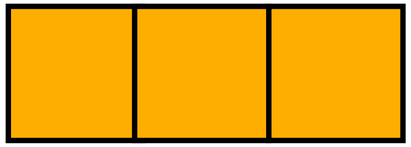
1. Guaranteed same type :)
2. Do the addition

Type annotation

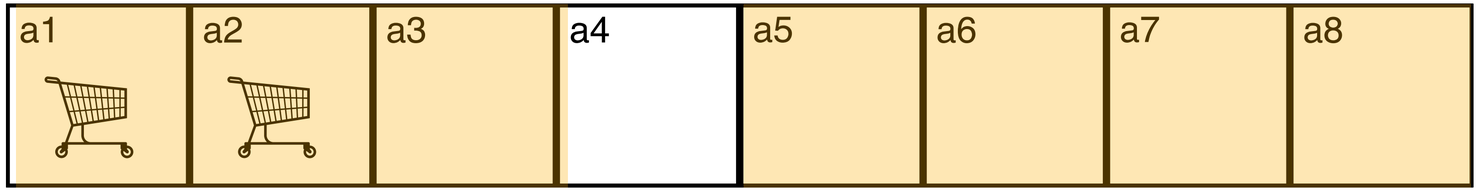
```
struct Fungus <: Organism
    weight
end
```

How big? Don't know!

Fungus



Doesn't fit!



Type annotation

Compiler knows how much storage to allocate each Fungus

```
struct Fungus <: Organism
    weight
end
```

Vs

```
struct Fungus <: Organism
    weight::Float64
end
```

Type annotation

Compiler knows how much storage to allocate each Fungus

Can do better compile-time memory management

Run-time is faster, leaner

```
struct Fungus <: Organism
    weight
end
```

Vs

```
struct Fungus <: Organism
    weight::Float64
end
```

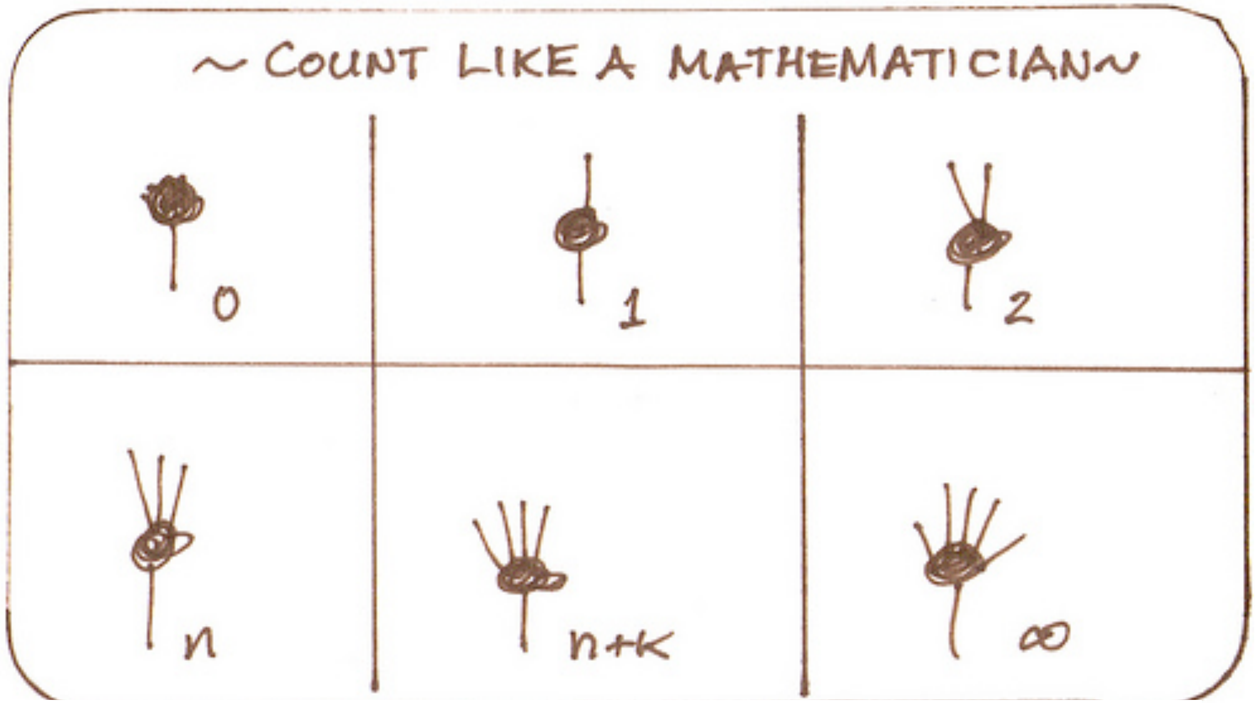
Type annotation

…can be done anywhere!

Type assertion of function output

```
add_weights (generic function with 2 methods)
add_weights(o1::Fungus, o2::Fungus)::Float64 = o1.weight + o2.weight
```

NB compilers are clever and can often infer types by detective work




```
no_inputs_or_outputs()
```

```
hi
```