Scientific Programming
A Crash Course

Class 1

Organization

- Meet on Tuesdays and Thursdays
- Please join the Slack and use it if you have questions
- The class will be very practical and interactive, so make sure you have a laptop with you
- Try your best to stay up-to-date with the material

How does the class run?

- I will give an introduction at the start
- You can then download the code and work through the material individually
- Get my attention if you need help; however, if I'm busy with someone else, talk to the people around you
- At the end, we'll have a bit of discussion

Overview of topics

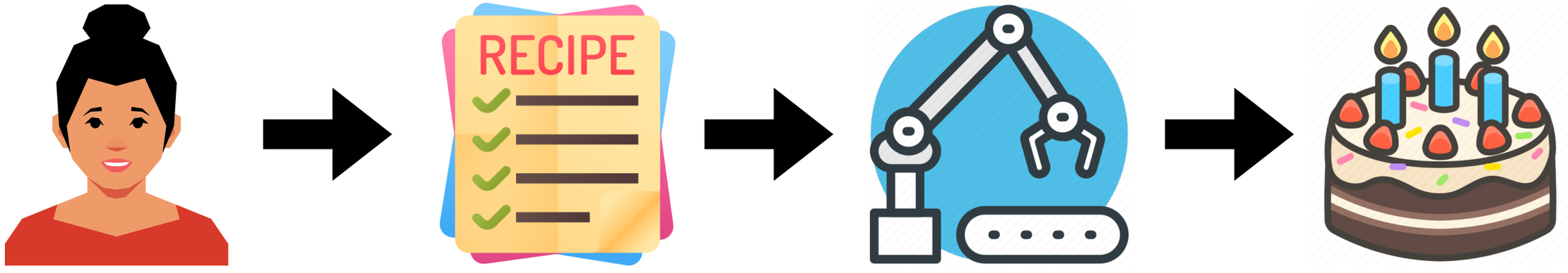
- Core principles of programming
- Data handling
- Data visualization
- Scripting experiments
- Preparation for statistics in R
- Web development (online experiments)
- Theoretical issues in open-science

Why learn programming?

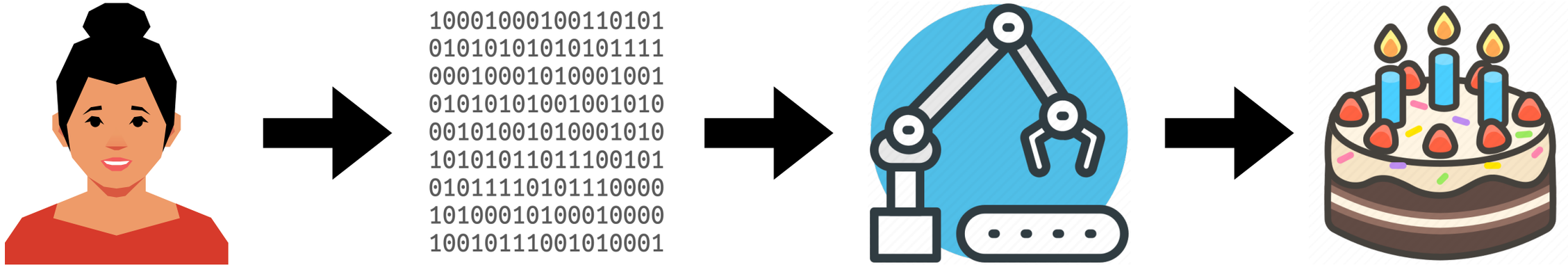
- Automate tasks that are very boring
- Flexibility to do anything, not restricted to what a given app allows you to do
- Minimize mistakes and get consistent reproducible output
- Self-documenting procedure
- Helps you to think about problems in a logical way

What is programming?

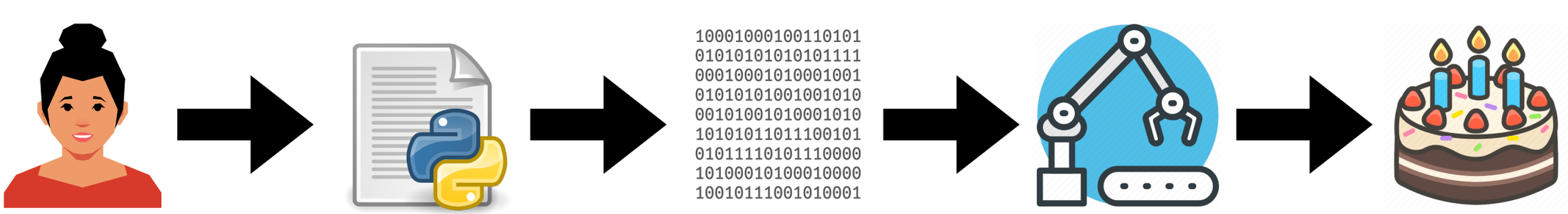
What is programming?

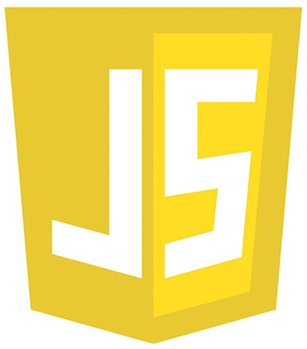


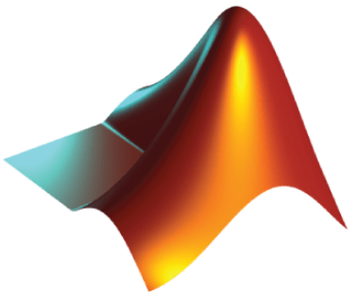
What is programming?



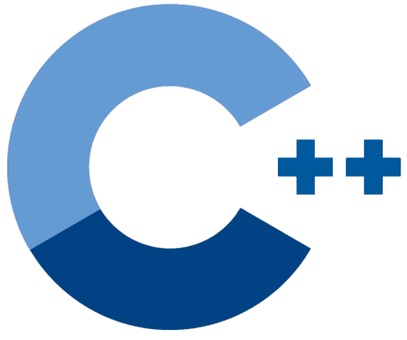
What is programming?







MATLAB



FORTRAN

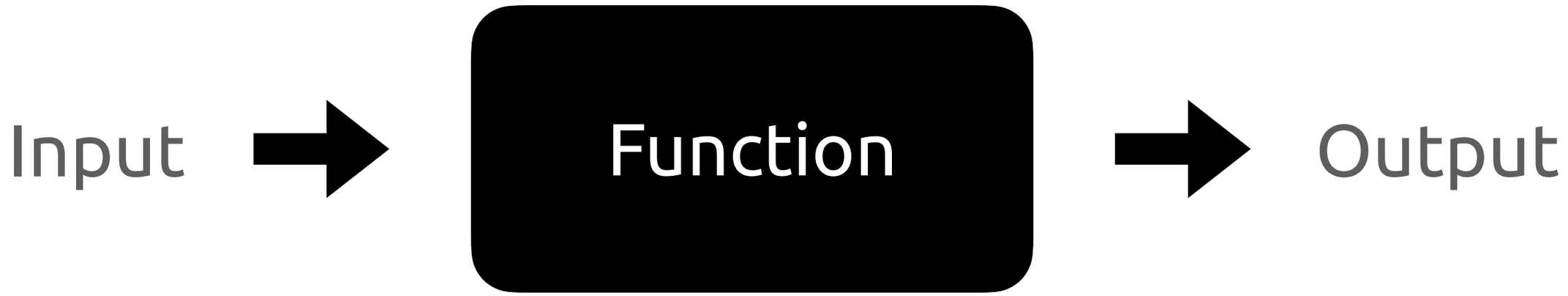
- High-level, clean, readable language
- Easy to learn the basic concepts
- Big community
- General-purpose
- Lots of scientific packages
- Stable, but still under active development

Scientific Programming

A Crash Course

Class 2

Functions



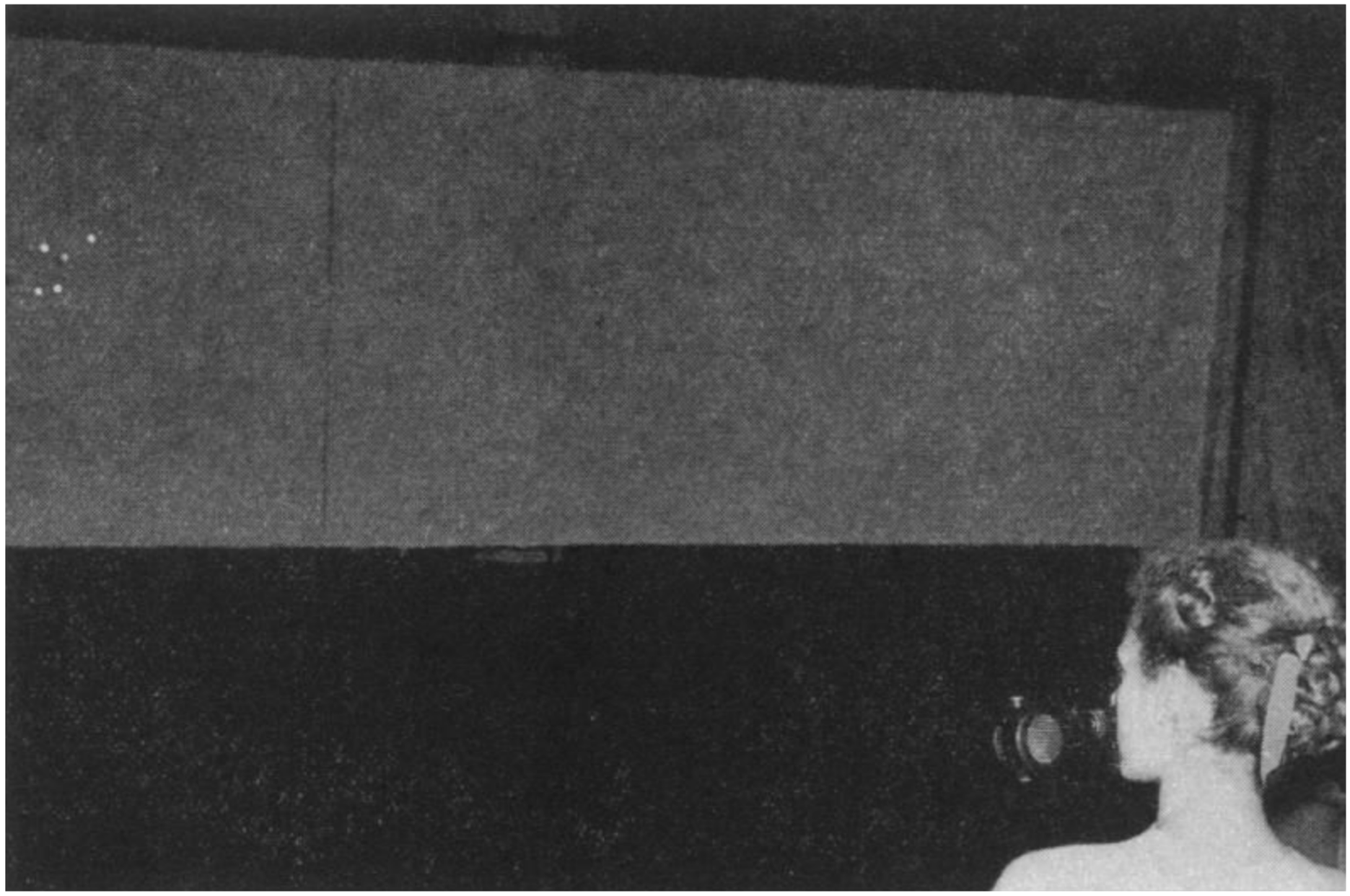
Why functions?

- Write cleaner, more organized code
- Avoid repeating the same lines again and again
- If you need to update the code, just do it once
- Easier to conceptualize the overall purpose
- Forget about details and concentrate on the big picture
- Functions can be tested in isolation

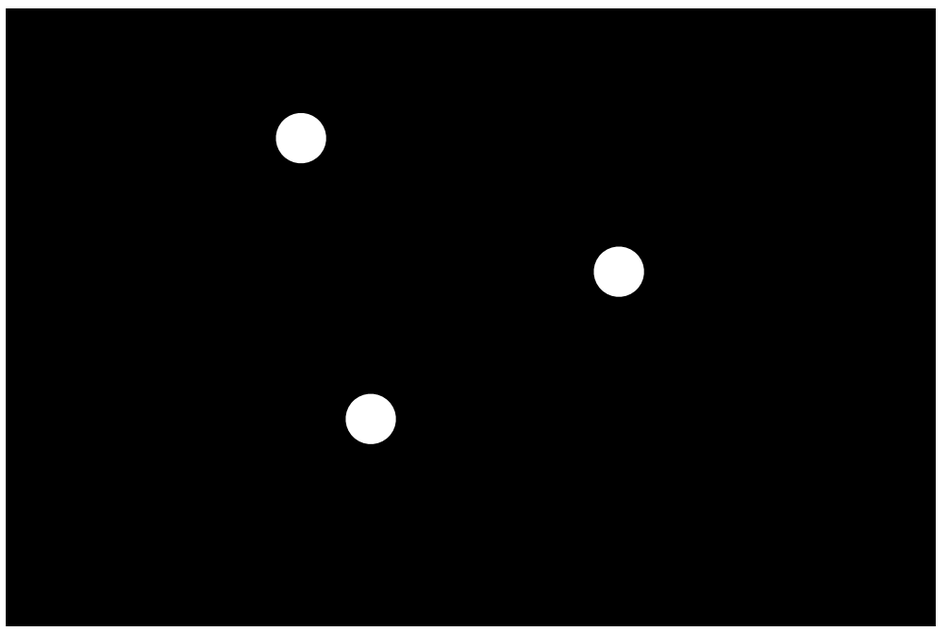
Scientific Programming

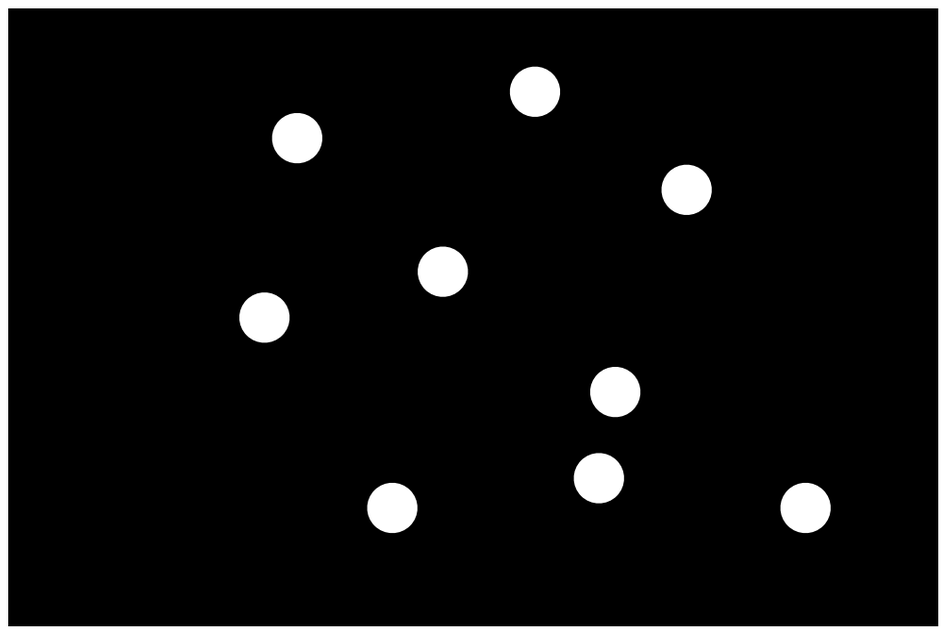
A Crash Course

Class 5

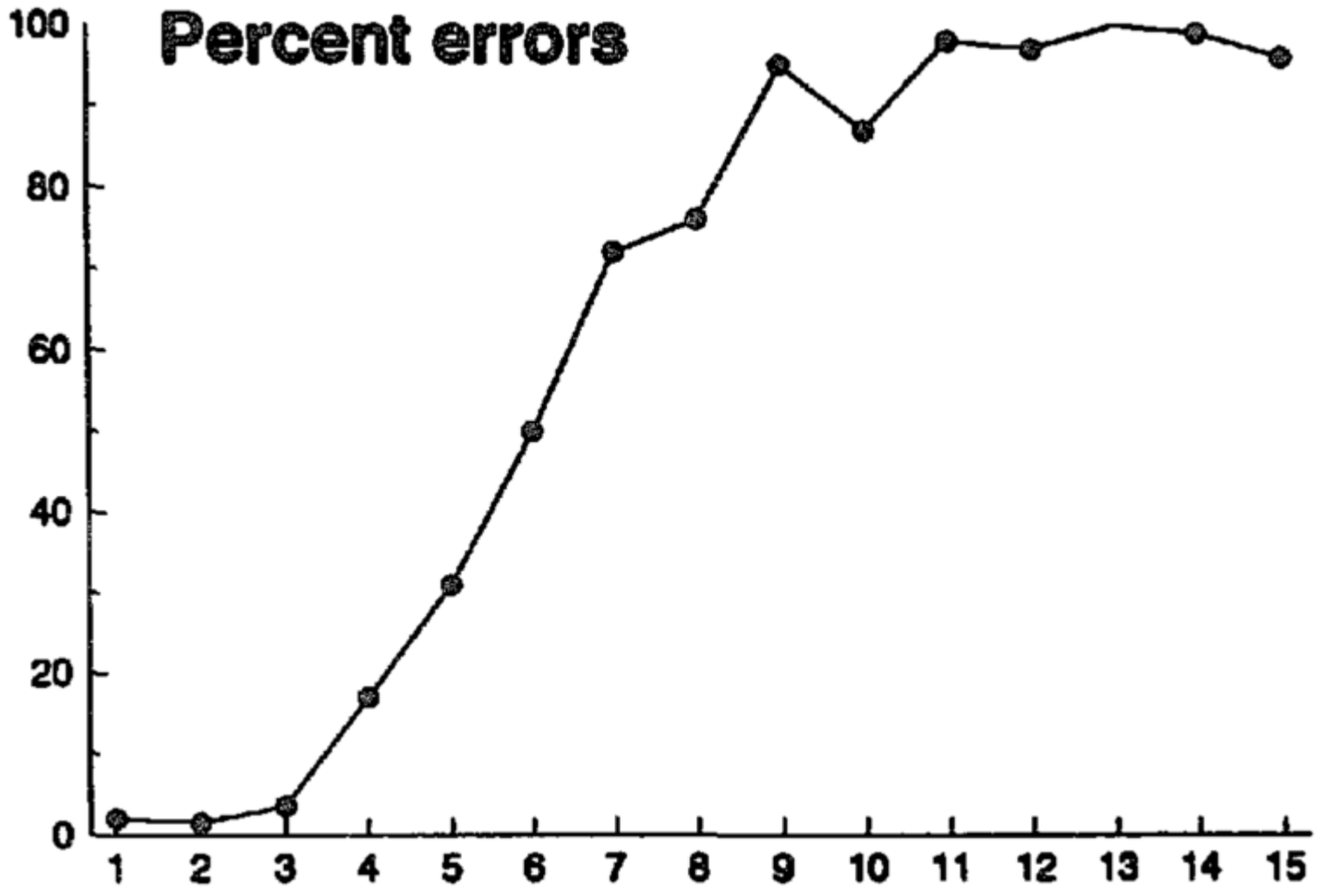


Numerosity estimation

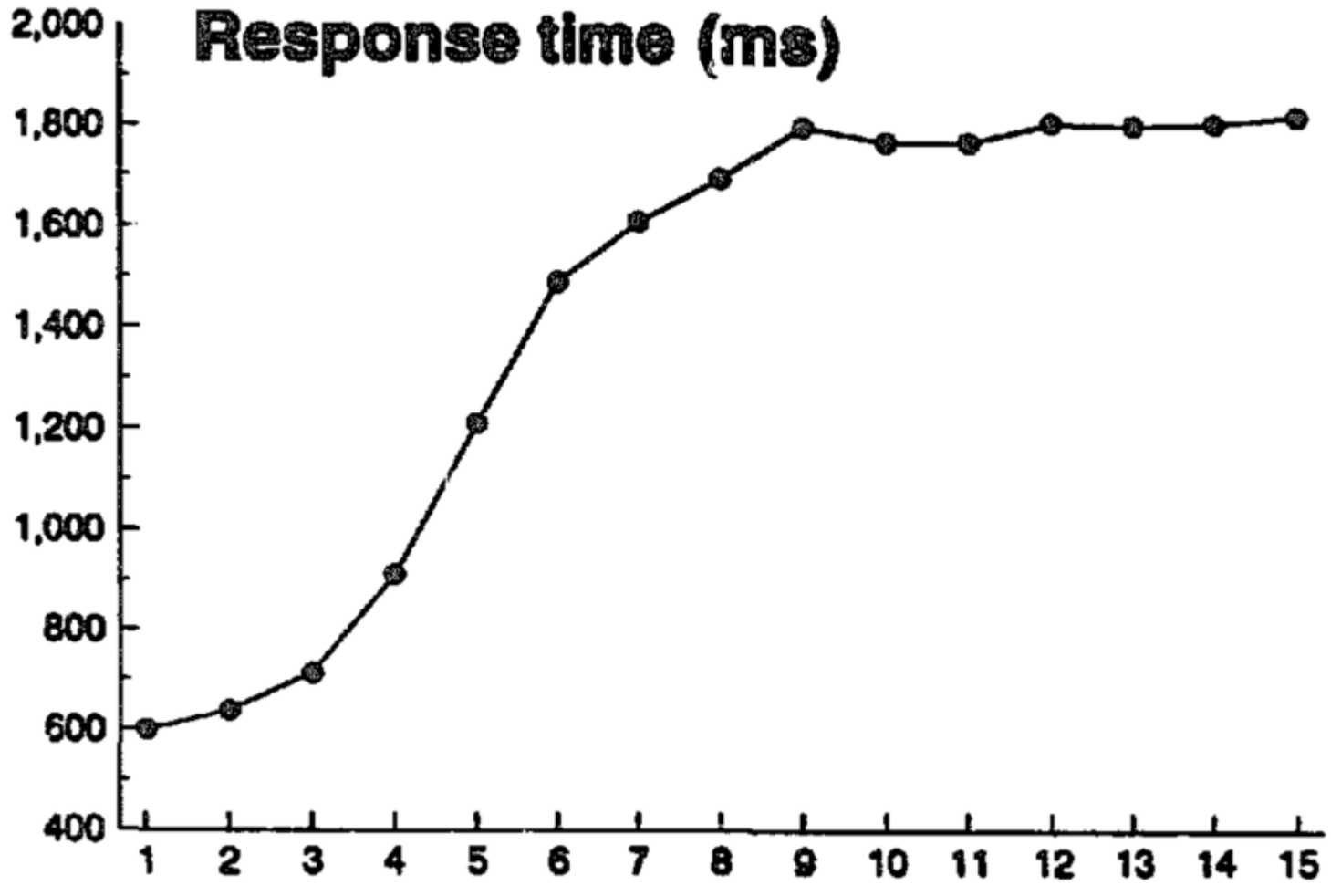




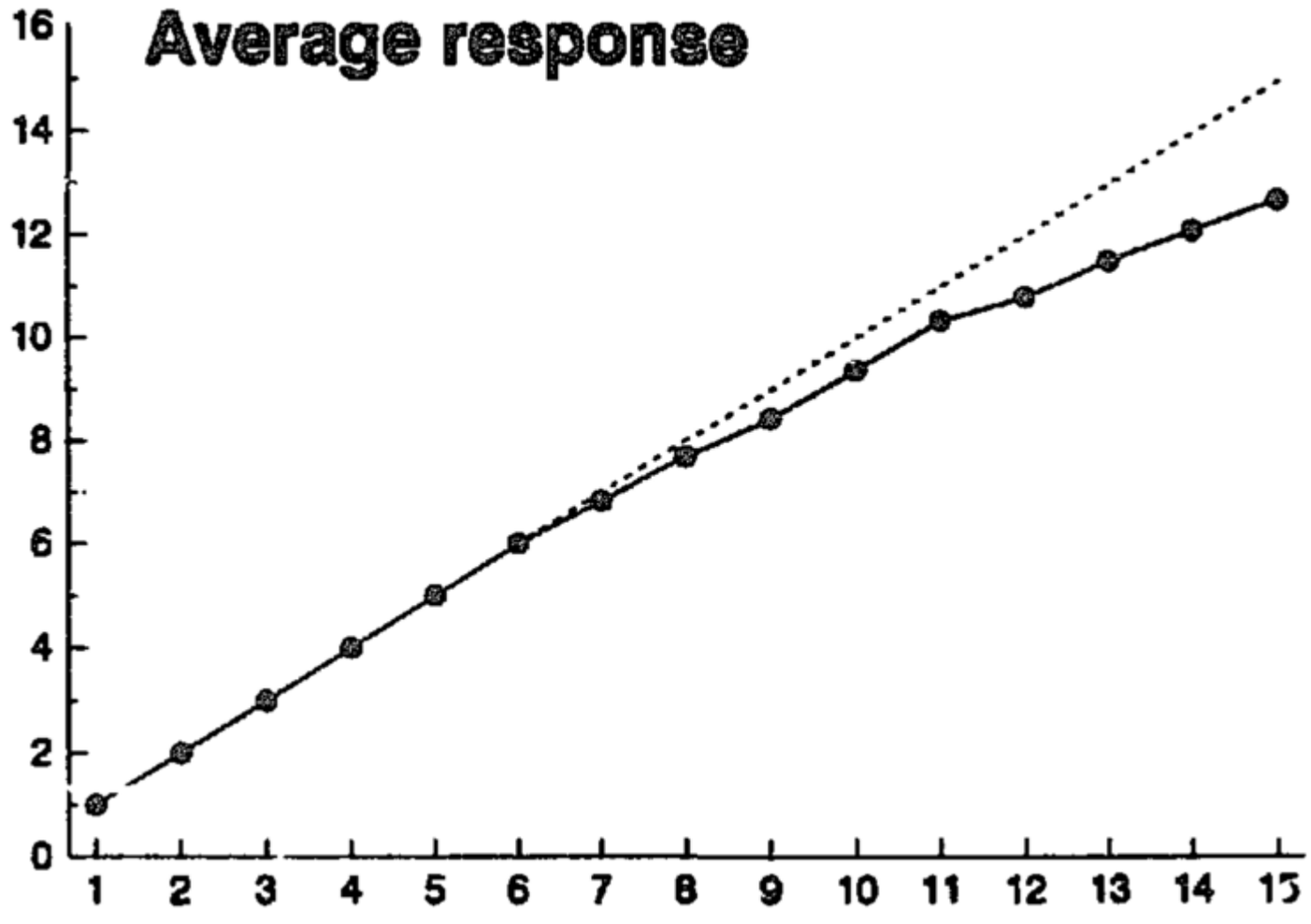
Numerosity estimation



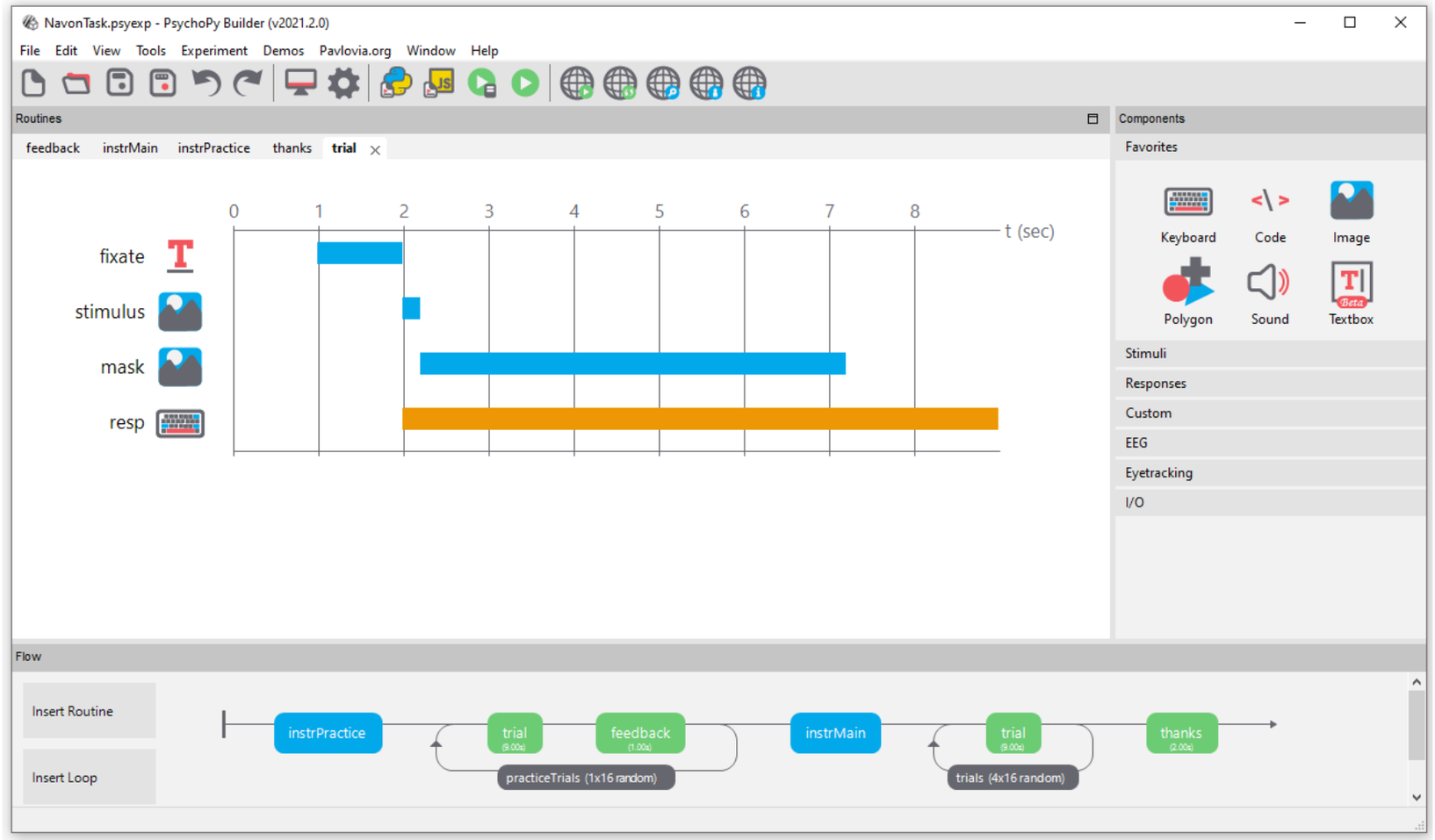
Numerosity estimation



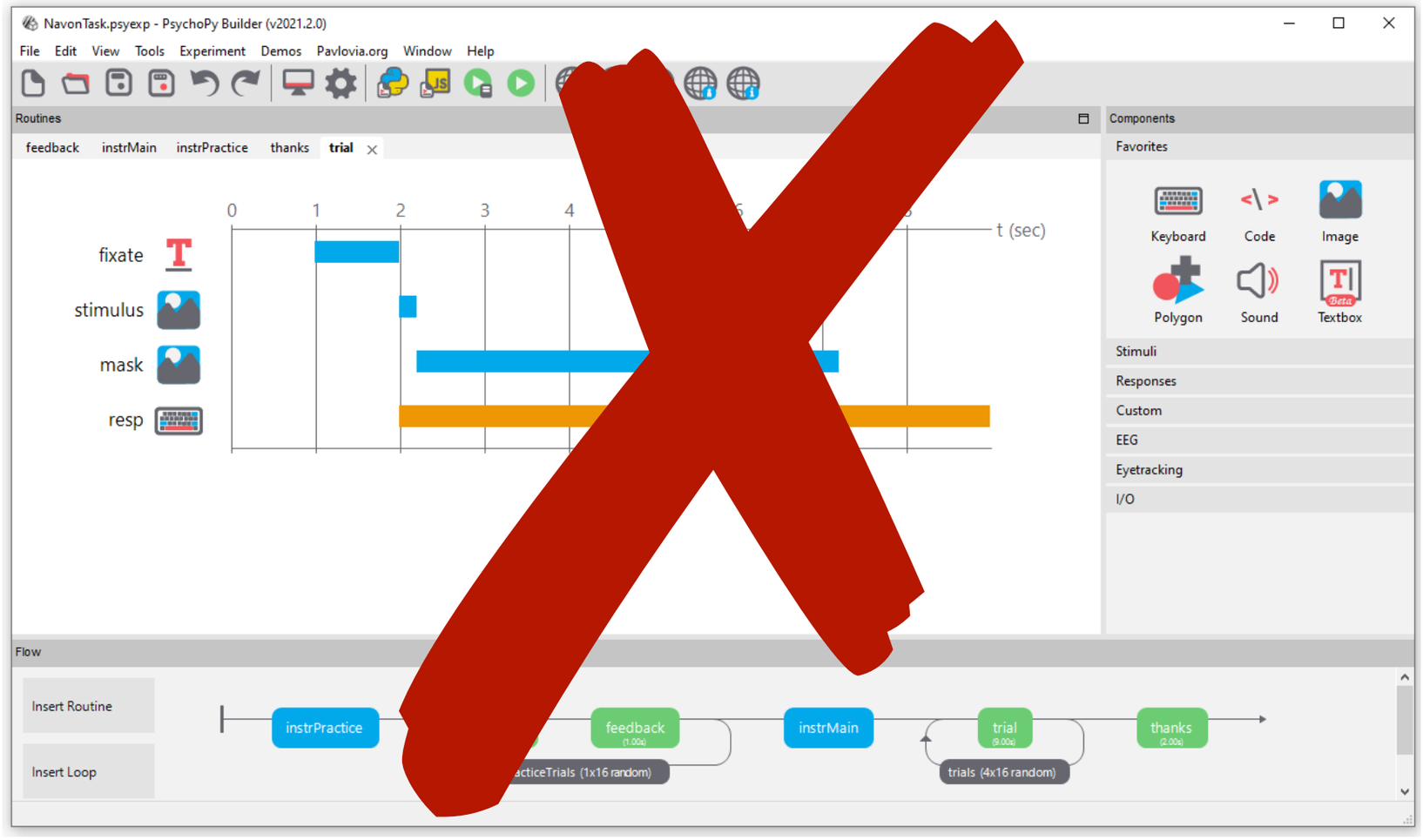
Numerosity estimation



PsychoPy


NavonTask.psyexp - PsychoPy Builder (v2021.2.0)
File Edit View Tools Experiment Demos Pavlovia.org Window Help
Routines
feedback instrMain instrPractice thanks trial
0 1 2 3 4 5 6 7 8 t (sec)
fixate
stimulus
mask
resp
Components
Favorites
Keyboard Code Image
Polygon Sound Textbox
Stimuli
Responses
Custom
EEG
Eyetracking
I/O
Flow
Insert Routine
Insert Loop
instrPractice
trial (9.00s)
feedback (1.00s)
practiceTrials (1x16 random)
instrMain
trial (9.00s)
trials (4x16 random)
thanks (2.00s)

PsychoPy


NavonTask.psyexp - PsychoPy Builder (v2021.2.0)
File Edit View Tools Experiment Demos Pavlovia.org Window Help
Routines
feedback instrMain instrPractice thanks trial
0 1 2 3 4 t (sec)
fixate
stimulus
mask
resp
Components
Favorites
Keyboard Code Image
Polygon Sound Textbox
Stimuli
Responses
Custom
EEG
Eyetracking
I/O
Flow
Insert Routine
Insert Loop
instrPractice
feedback (1.00s)
acticeTrials (1x16 random)
instrMain
trial (9.00s)
trials (4x16 random)
thanks (2.00s)

Scientific Programming

A Crash Course

Class 6

Why run online?

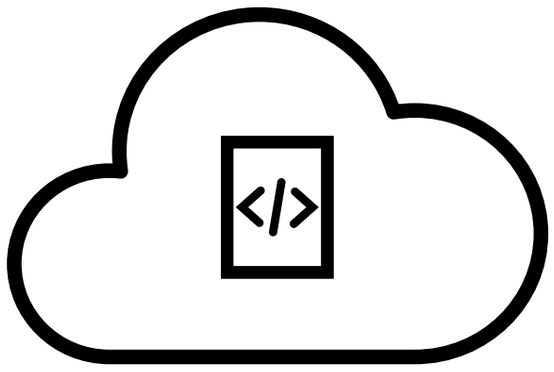
- Get results fast
- Save time and money
- More diverse sample (less WEIRD?)
- Fewer bureaucratic hurdles
- As good as lab-based studies (maybe?)

What can you do?

- Questionnaires and surveys
- Behavioral experiments (learning, categorization, reaction times...)
- Recording from the microphone/camera
- Real-time communication experiments
- Rating tasks (similarity judgments etc.)

What can't you do?

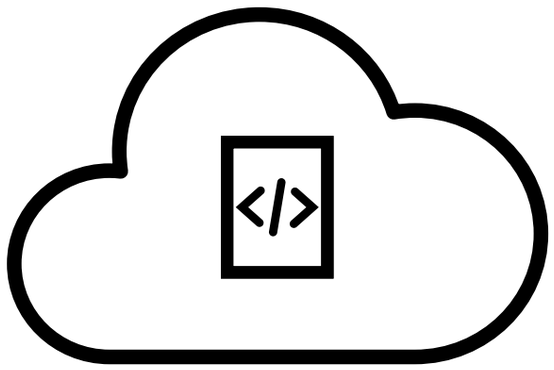
- Imaging studies, obviously
- Eye tracking (although, see WebGazer.js)
- Longitudinal studies can be tricky
- Experiments with children/animals
- Experiments involving extreme levels of precision



server

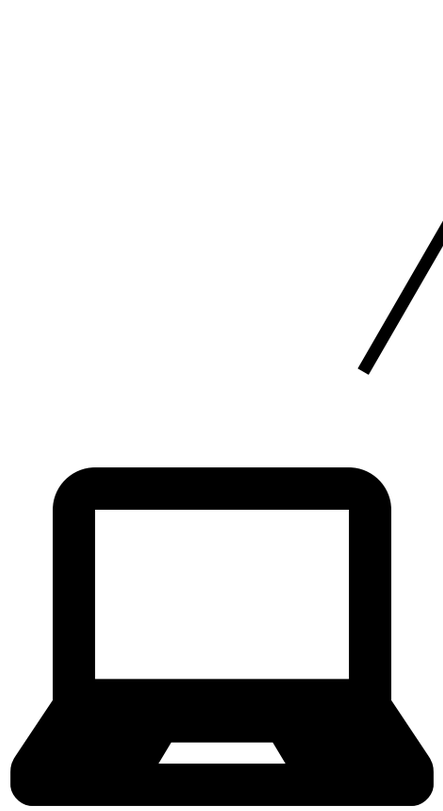
platform

client

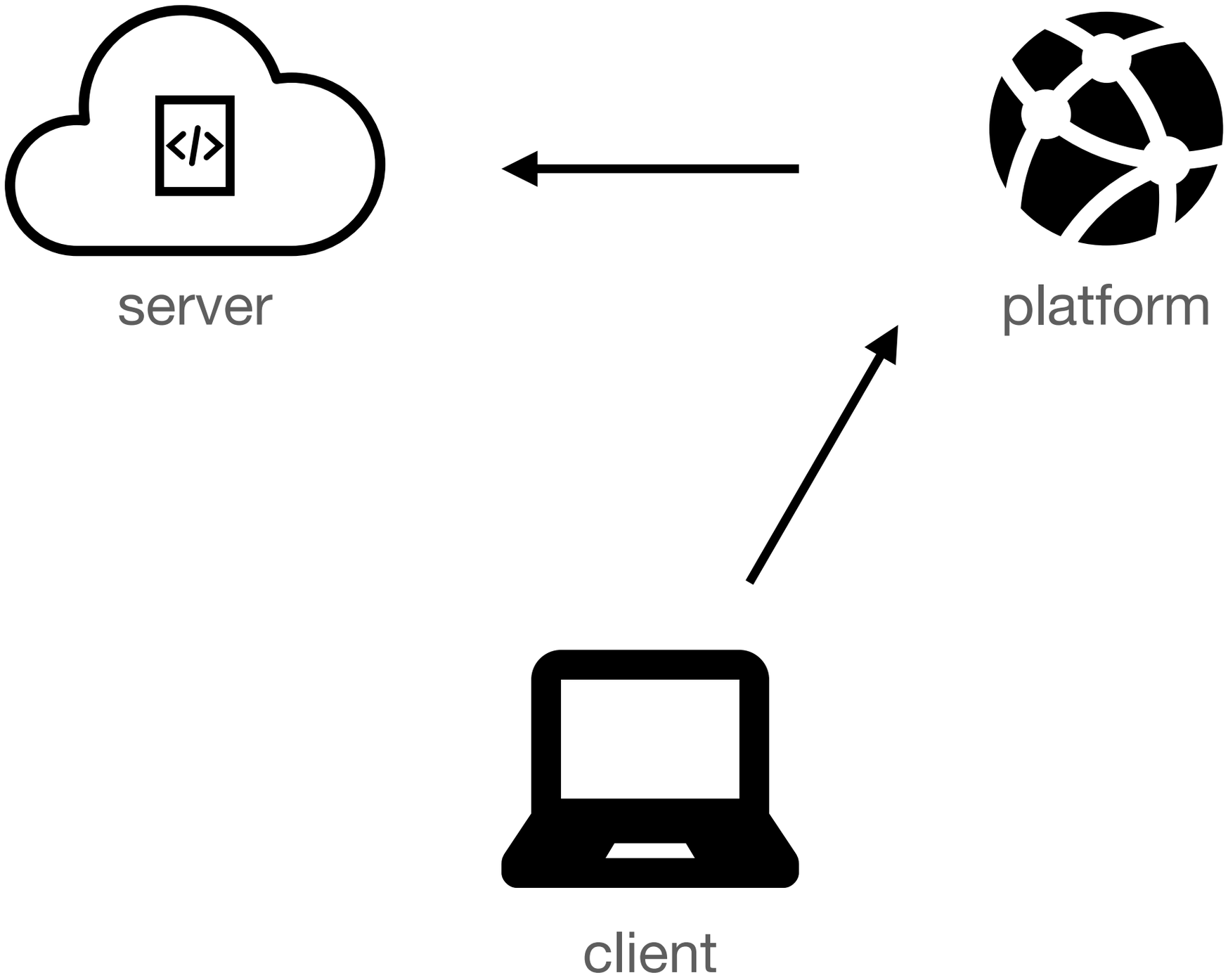


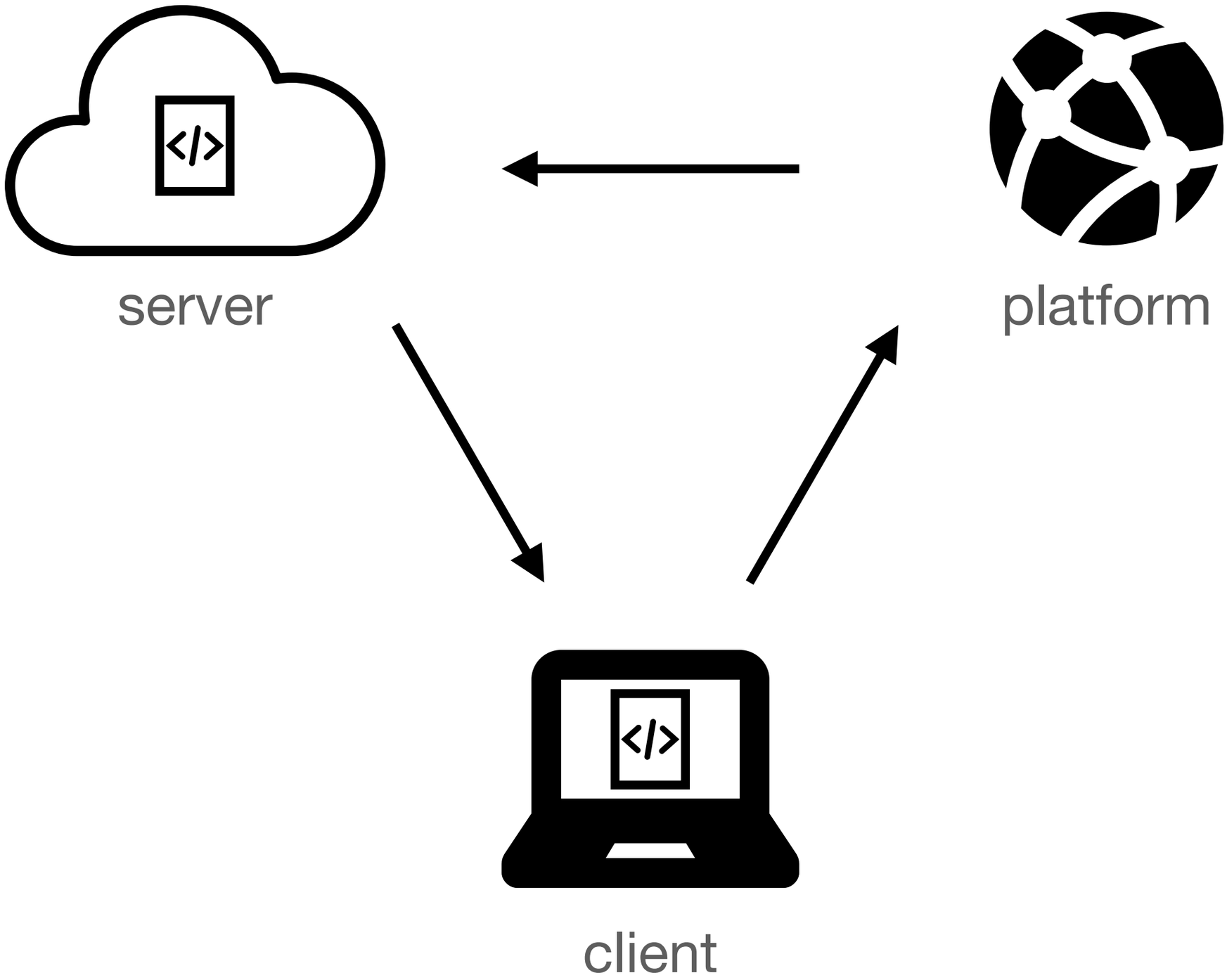
server

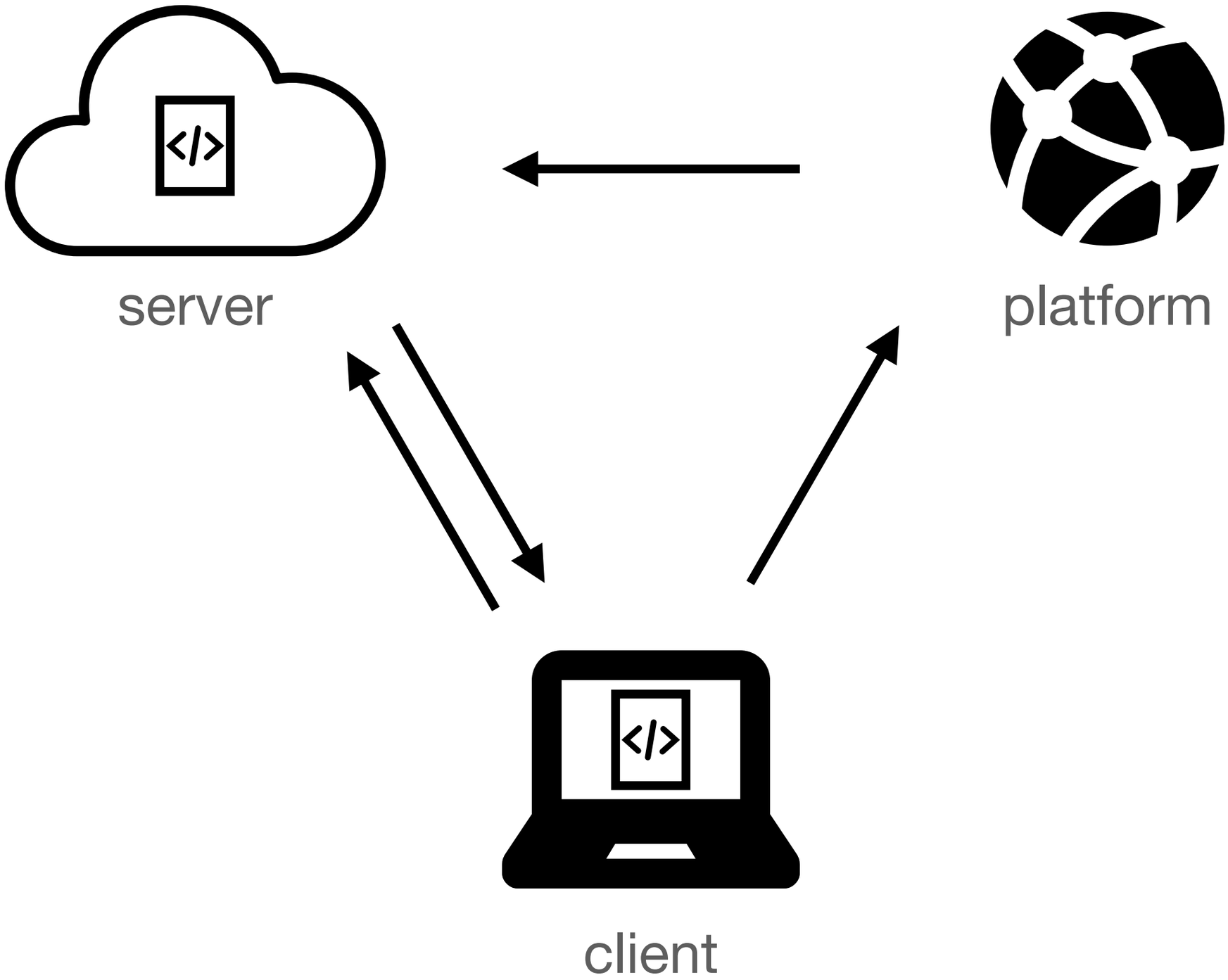
platform

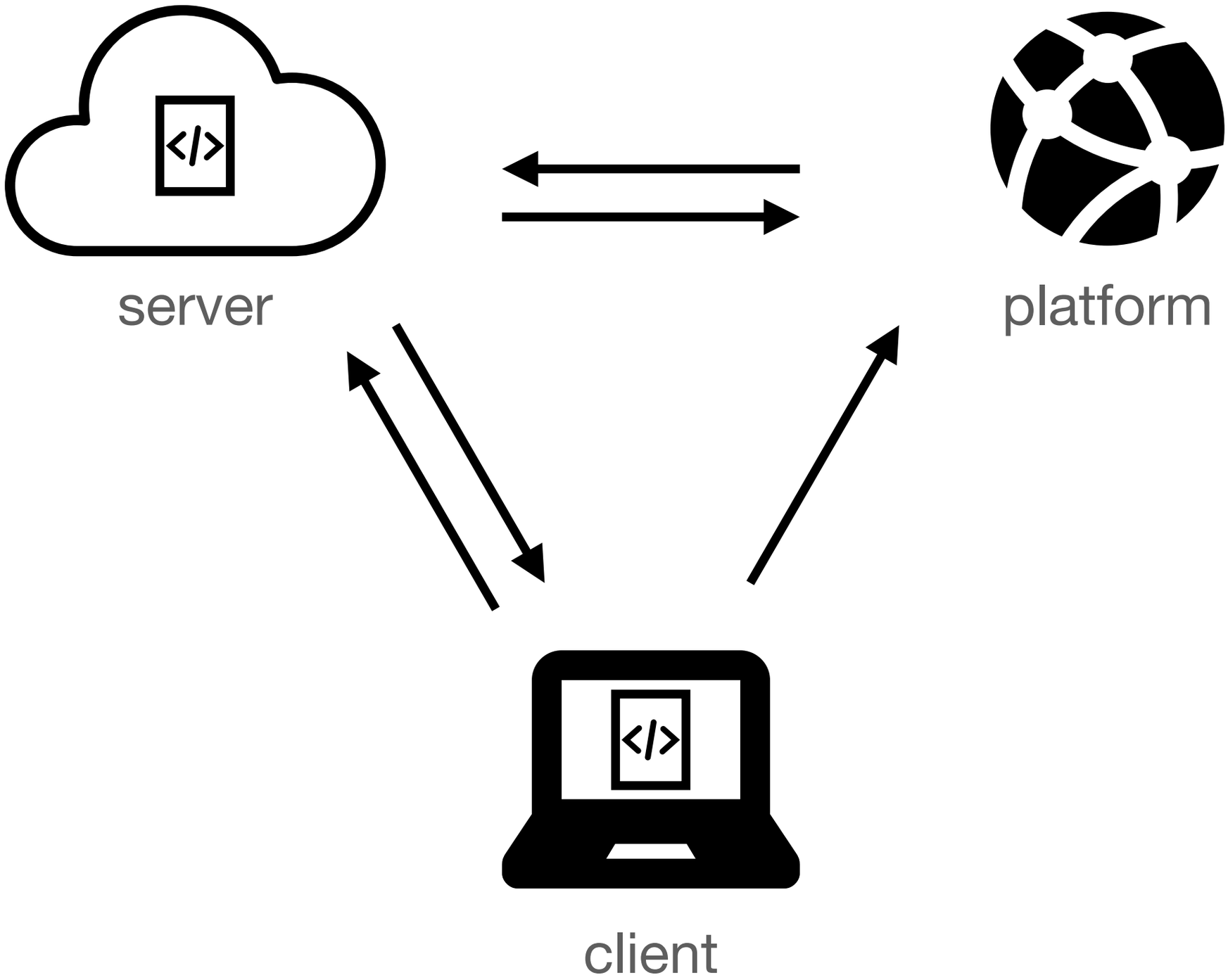


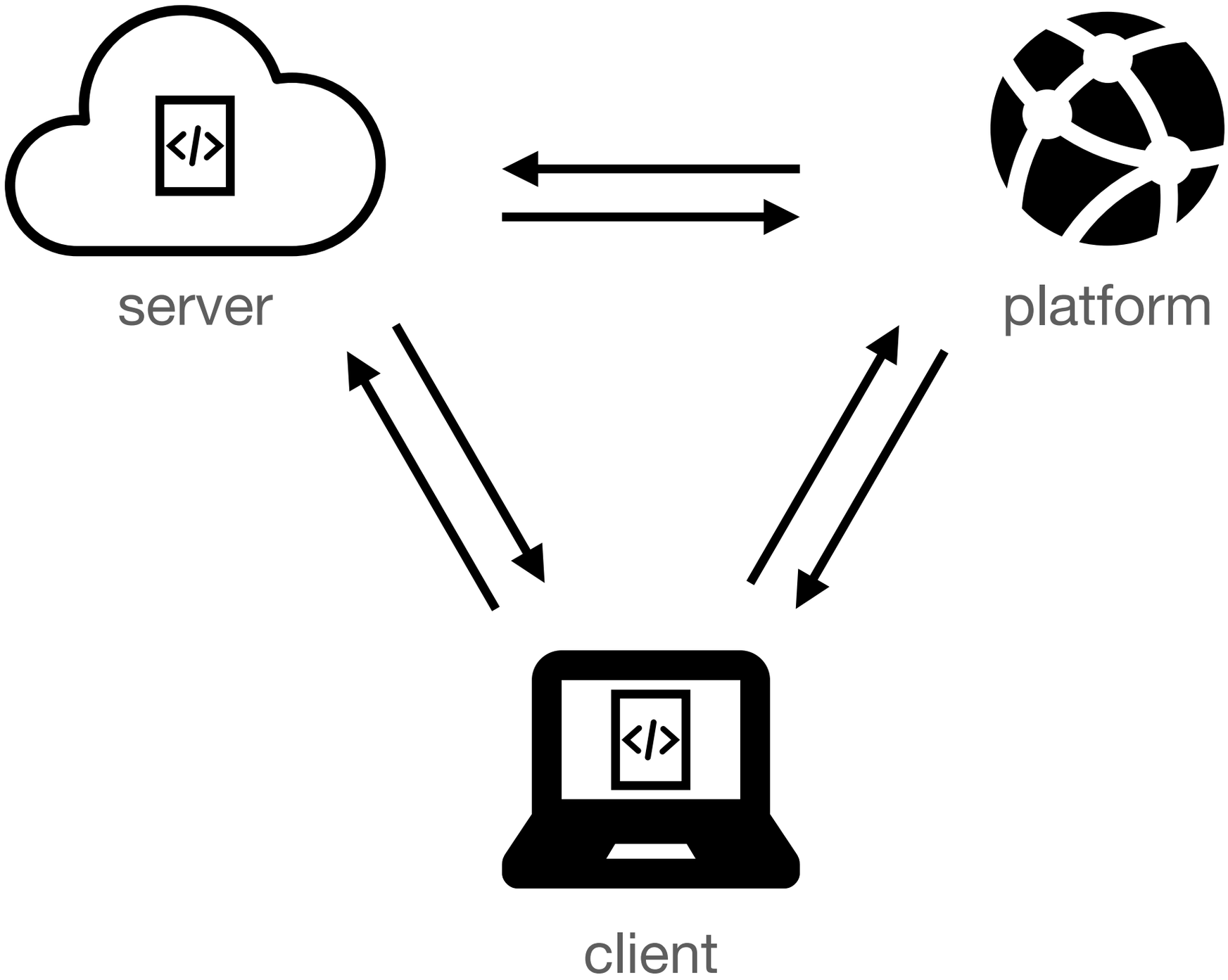
client

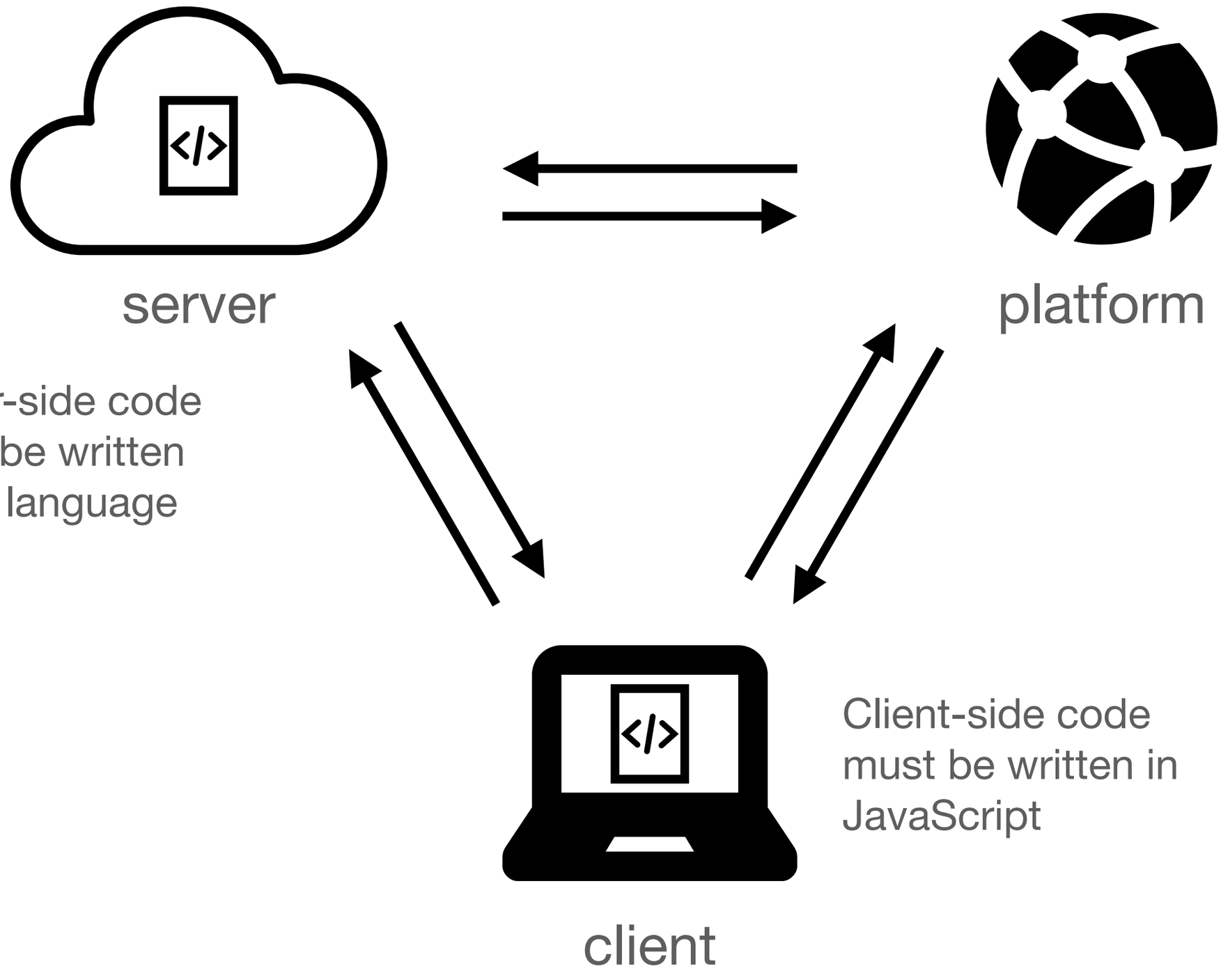

server
platform
client


server
platform
client


server
platform
client


server
platform
client

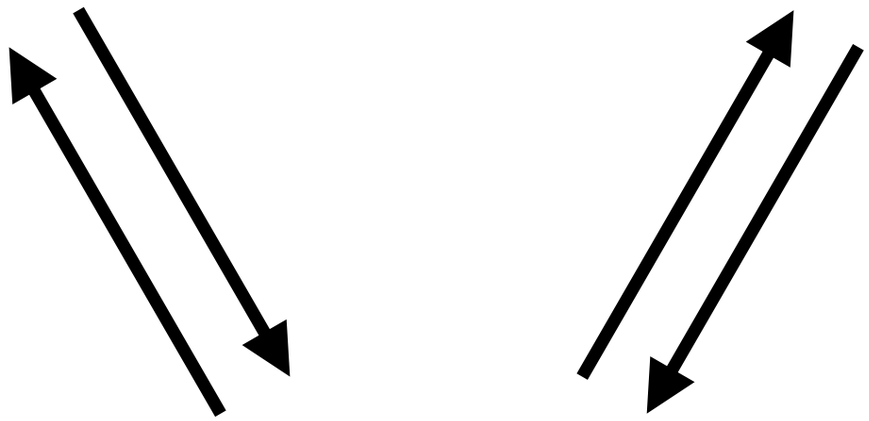

server
platform
client

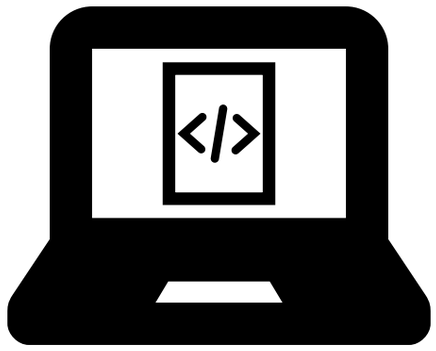

server
platform
Server-side code could be written in any language
client
Client-side code must be written in JavaScript

Cognition.

Pavlovia

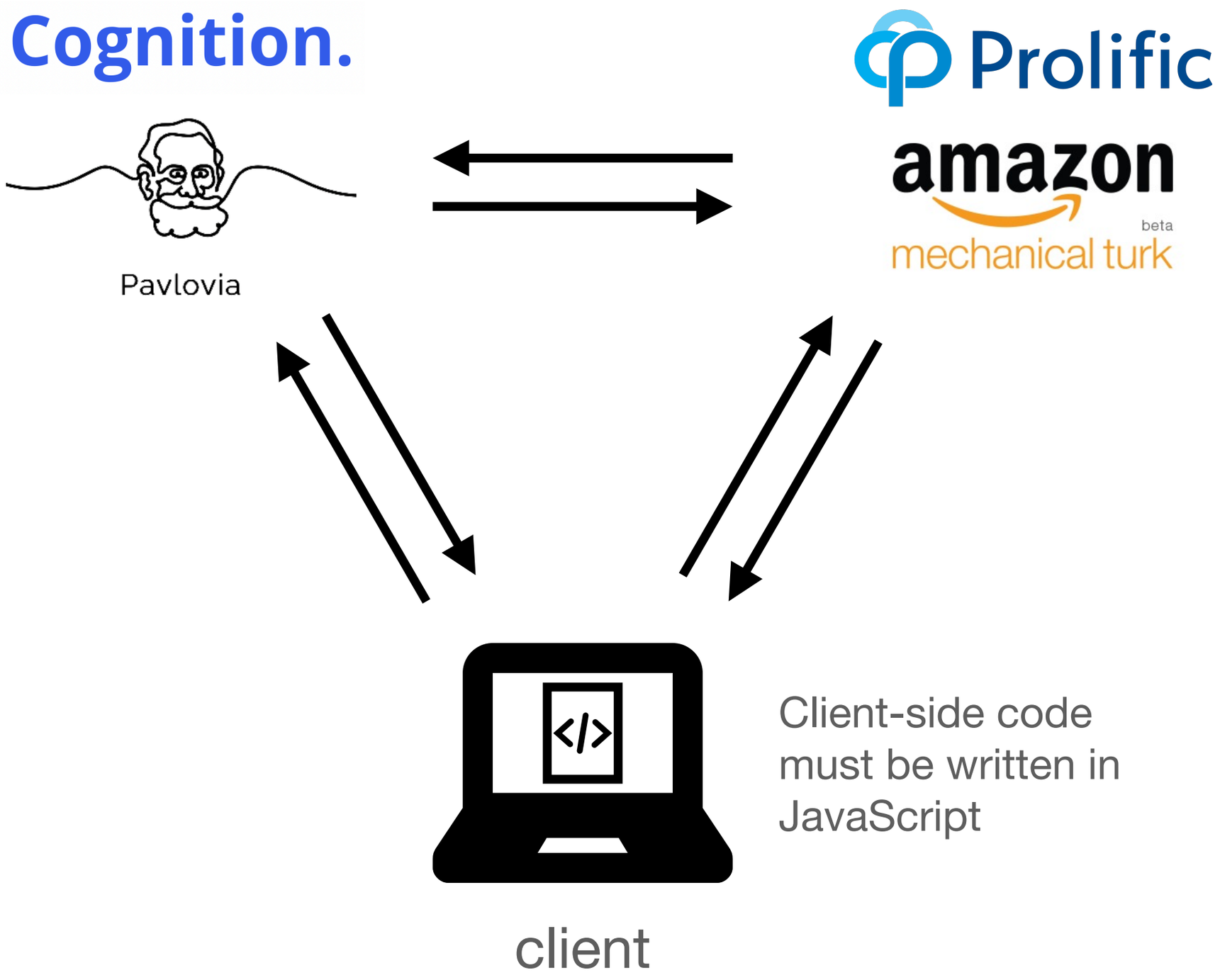
platform





Client-side code must be written in JavaScript

client



client

Client-side code must be written in JavaScript

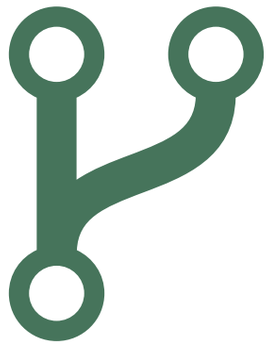

jspsych

Scientific Programming

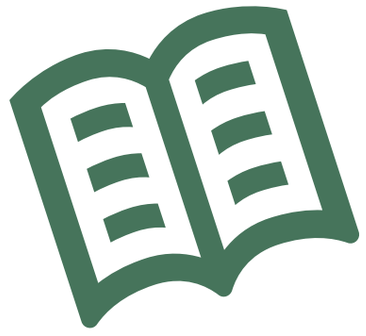
A Crash Course

Class 8

Scripting



Version control



Documentation

Open source

Unit testing

Virtual environments

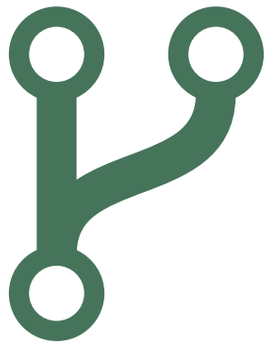
Code review

Scripting

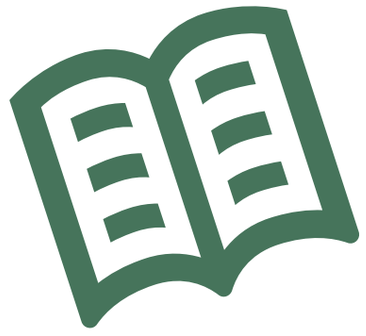
Scripting

- Write reproducible scripts and pipelines
- Avoid "semi-scripting"
- Not just helpful for others, but also for your future self

Scripting



Version control



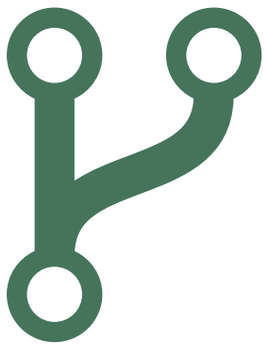
Documentation

Open source

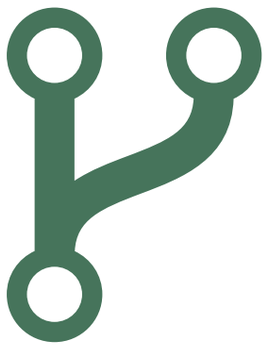
Unit testing

Virtual environments

Code review

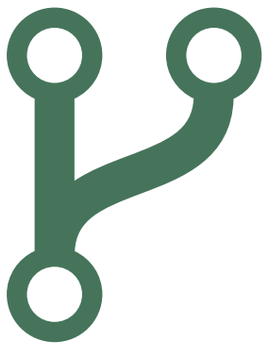


Version control



Version control

```
analysis_script.py
```



Version control

analysis_script_final_v2_New+DavidesComments_versionC.py

Version control

Repository

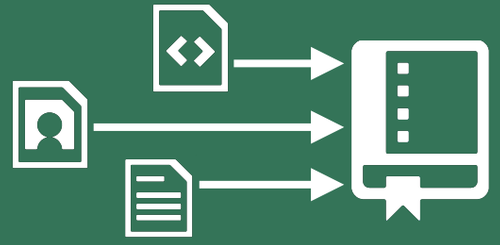
Commit

Branch

Fork

Repository

Repository



Commit

Commit

Commit

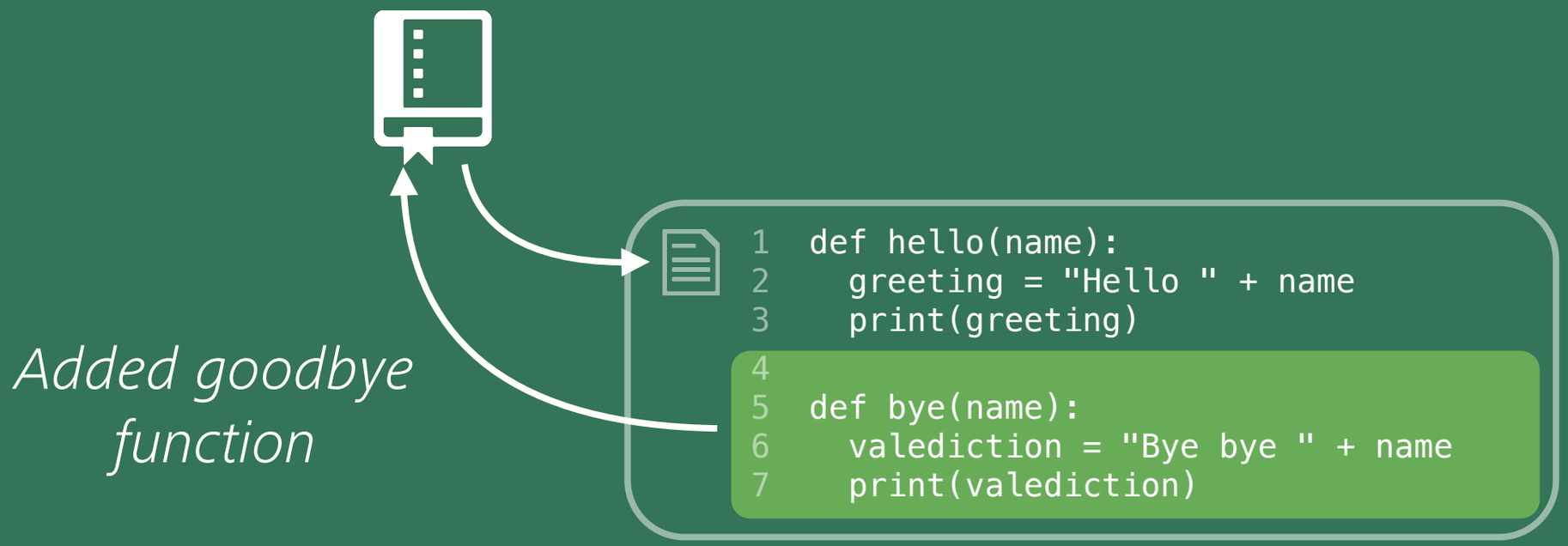
```
def hello(name):
  greeting = "Hello " + name
  print(greeting)

def bye(name):
  valediction = "Bye bye " + name
  print(valediction)
```

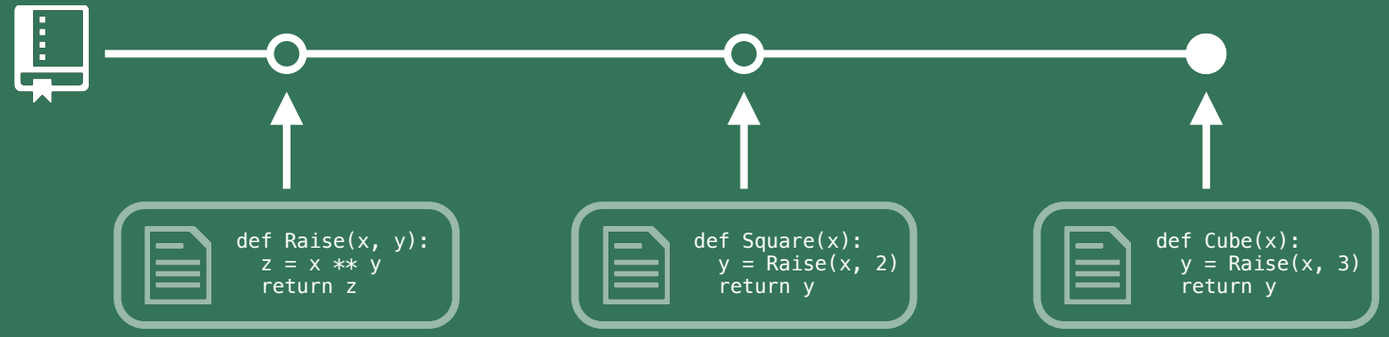
Commit


1 def hello(name):
2 greeting = "Hello " + name
3 print(greeting)
4
5 def bye(name):
6 valediction = "Bye bye " + name
7 print(valediction)

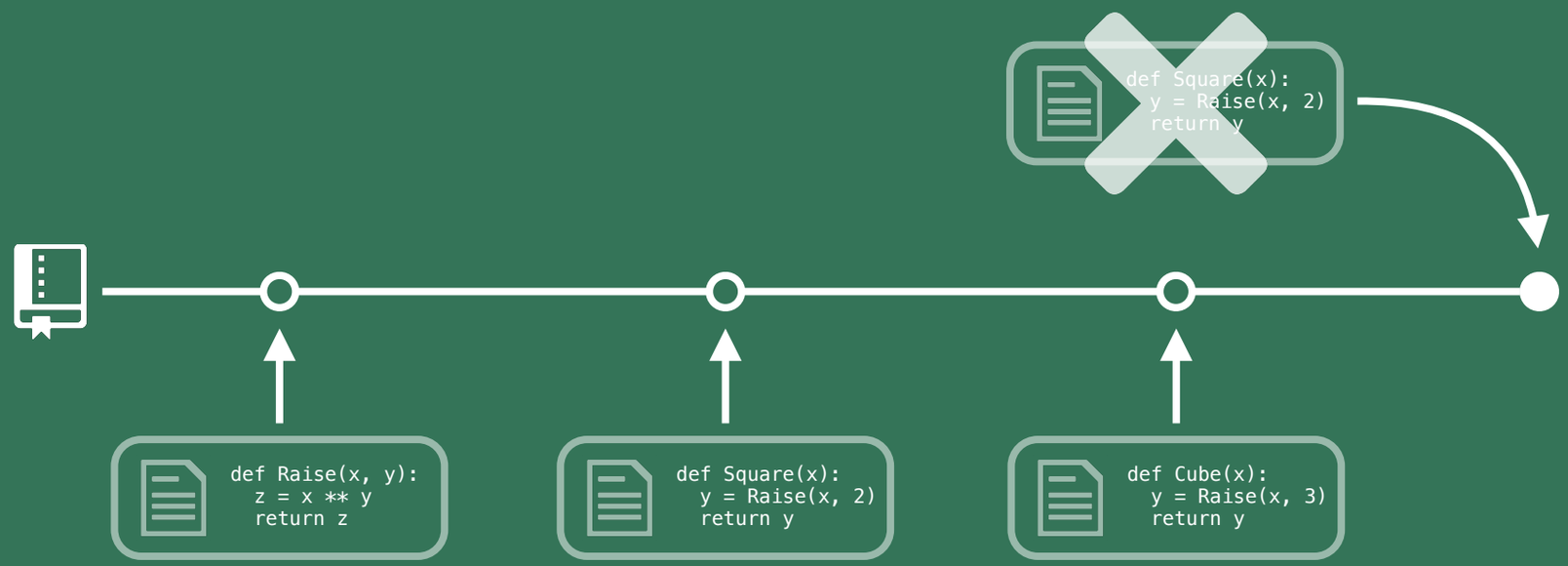
Commit


1 def hello(name):
2 greeting = "Hello " + name
3 print(greeting)
4
5 def bye(name):
6 valediction = "Bye bye " + name
7 print(valediction)
Added goodbye function

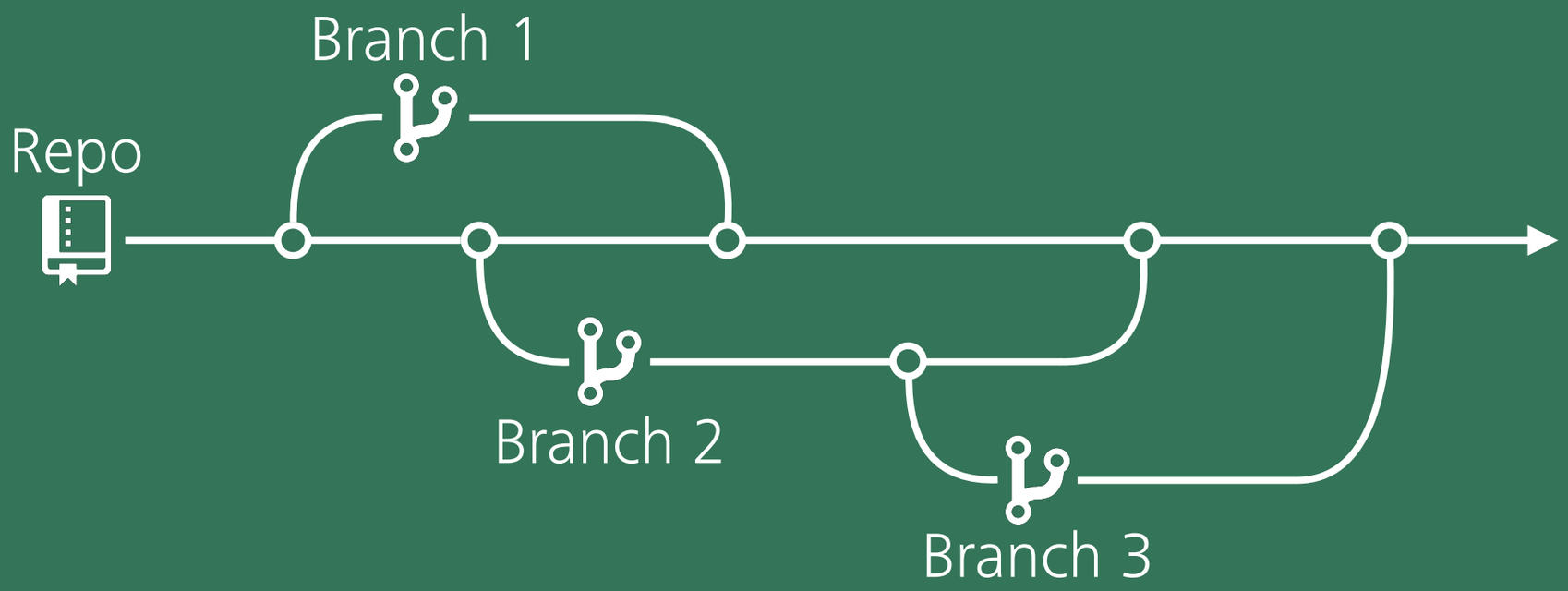
The timeline



The timeline



Branching



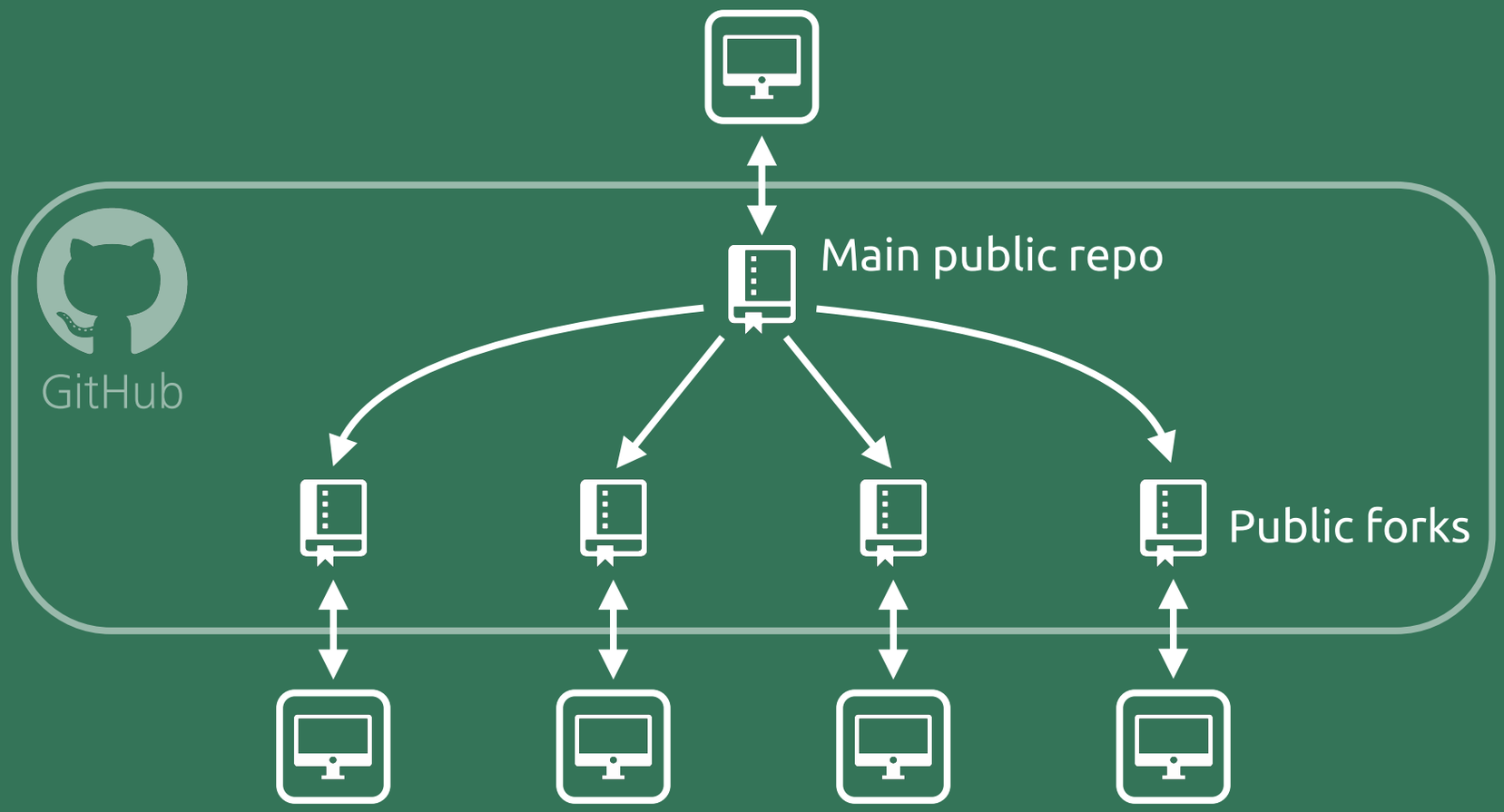
Branching

Develop new features or explore new ideas

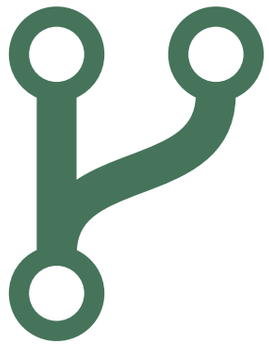
Keep a development version separate from a stable version

Easier to deal with multiple collaborators on a single repo

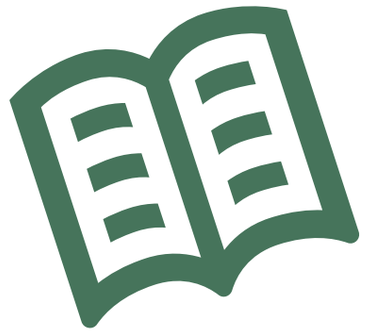
Forking



Scripting



Version control



Documentation

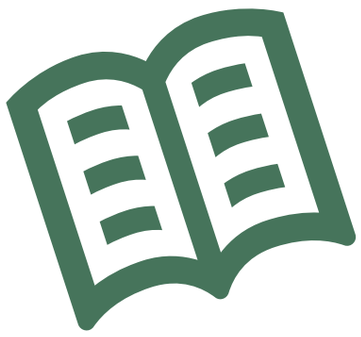
Open source

Unit testing

Virtual environments

Code review

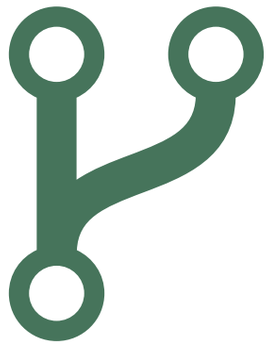
Documentation



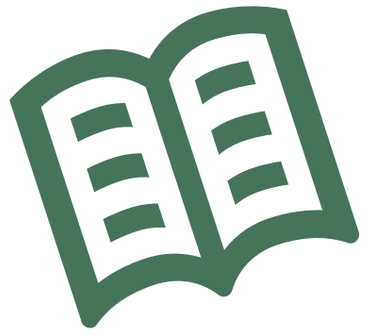
Documentation

- Use clear variable/function names
- Comment your code as appropriate
- Write a readme file to help people get started with your project

Scripting



Version control



Documentation

Open source

Unit testing

Virtual environments

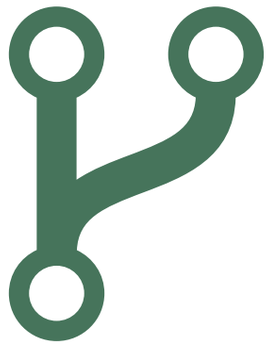
Code review

Open source

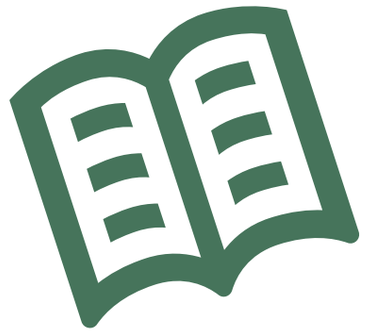
Open source

- Prefer open-source software if possible
- Open code & data: Work transparently and openly right from the start
- Give back to the community if you can

Scripting



Version control



Documentation

Open source

Unit testing

Virtual environments

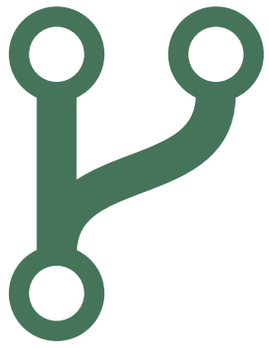
Code review

Unit testing

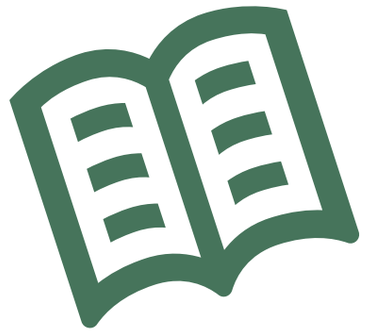
Unit testing

- Write modular code
- Write tests for your code
- Run tests frequently to check things are still working

Scripting



Version control



Documentation

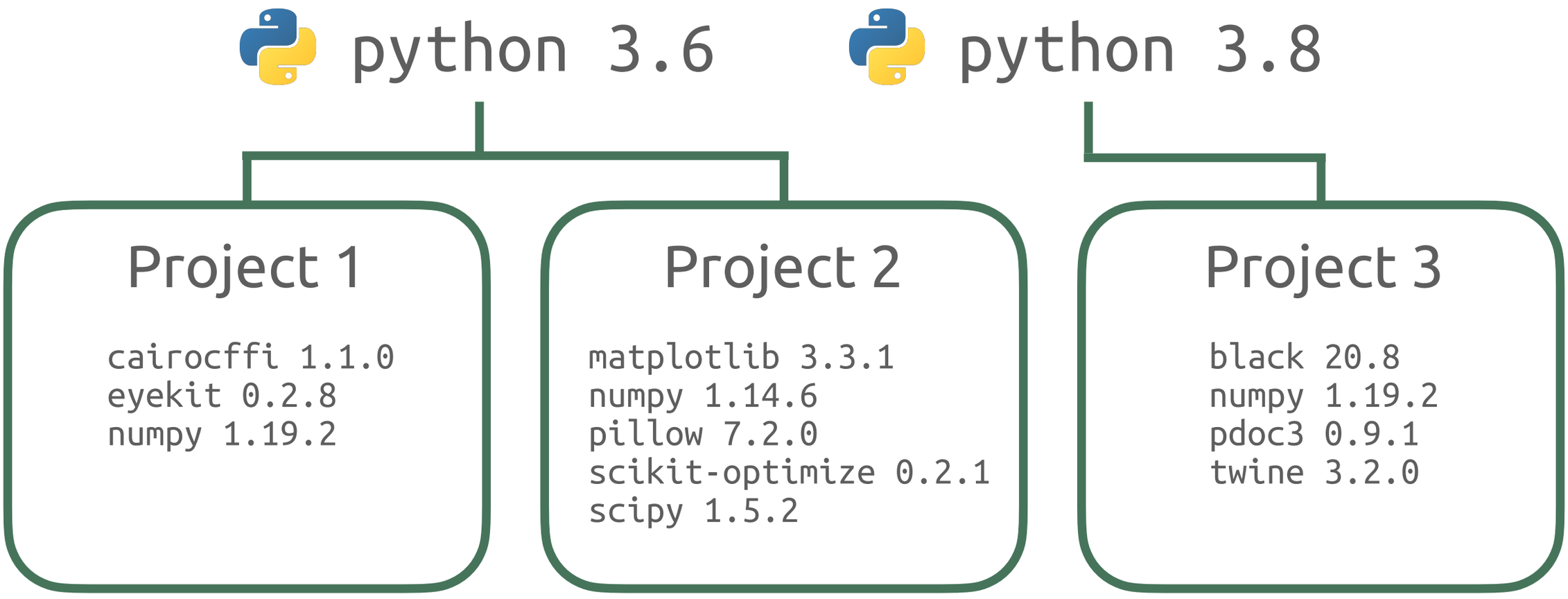
Open source

Unit testing

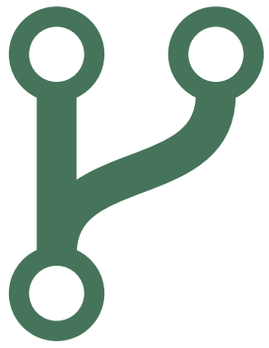
Virtual environments

Code review

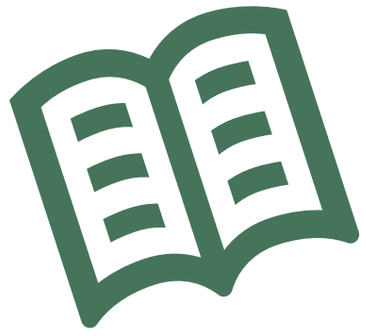
Virtual environments


python 3.6
python 3.8
Project 1
cairocffi 1.1.0
eyekit 0.2.8
numpy 1.19.2
Project 2
matplotlib 3.3.1
numpy 1.14.6
pillow 7.2.0
scikit-optimize 0.2.1
scipy 1.5.2
Project 3
black 20.8
numpy 1.19.2
pdoc3 0.9.1
twine 3.2.0

Scripting



Version control



Documentation

Open source

Unit testing

Virtual environments

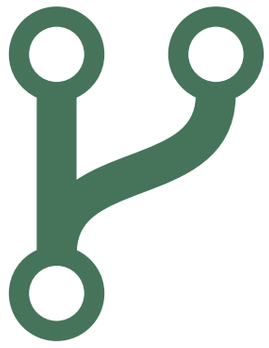
Code review

Code review

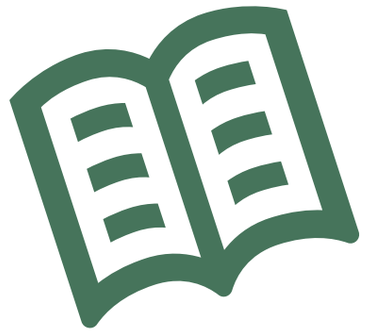
Code review

- Regularly review your code and refactor if necessary
- Meet up with friends and review each other's code

Scripting



Version control



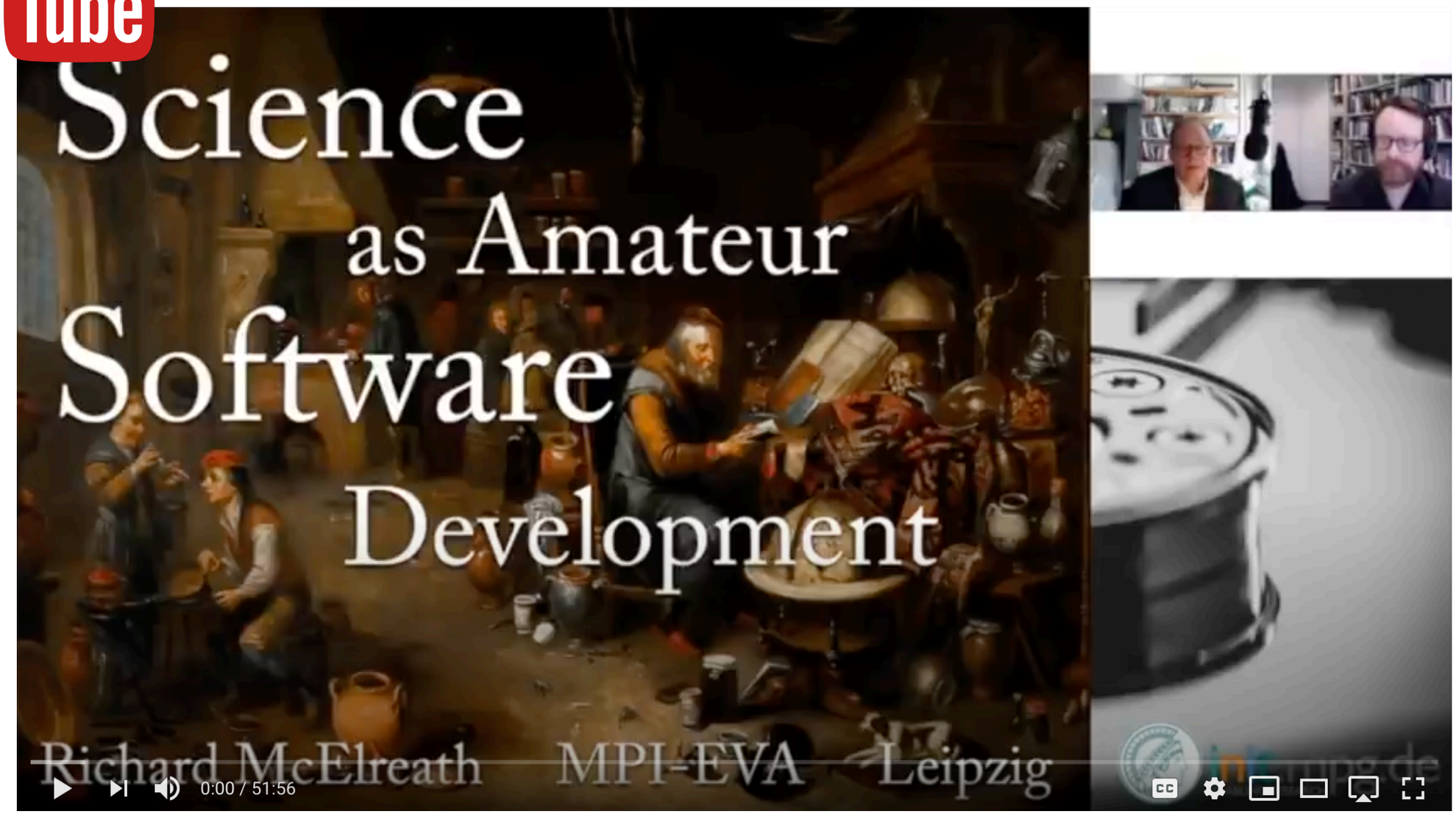
Documentation

Open source

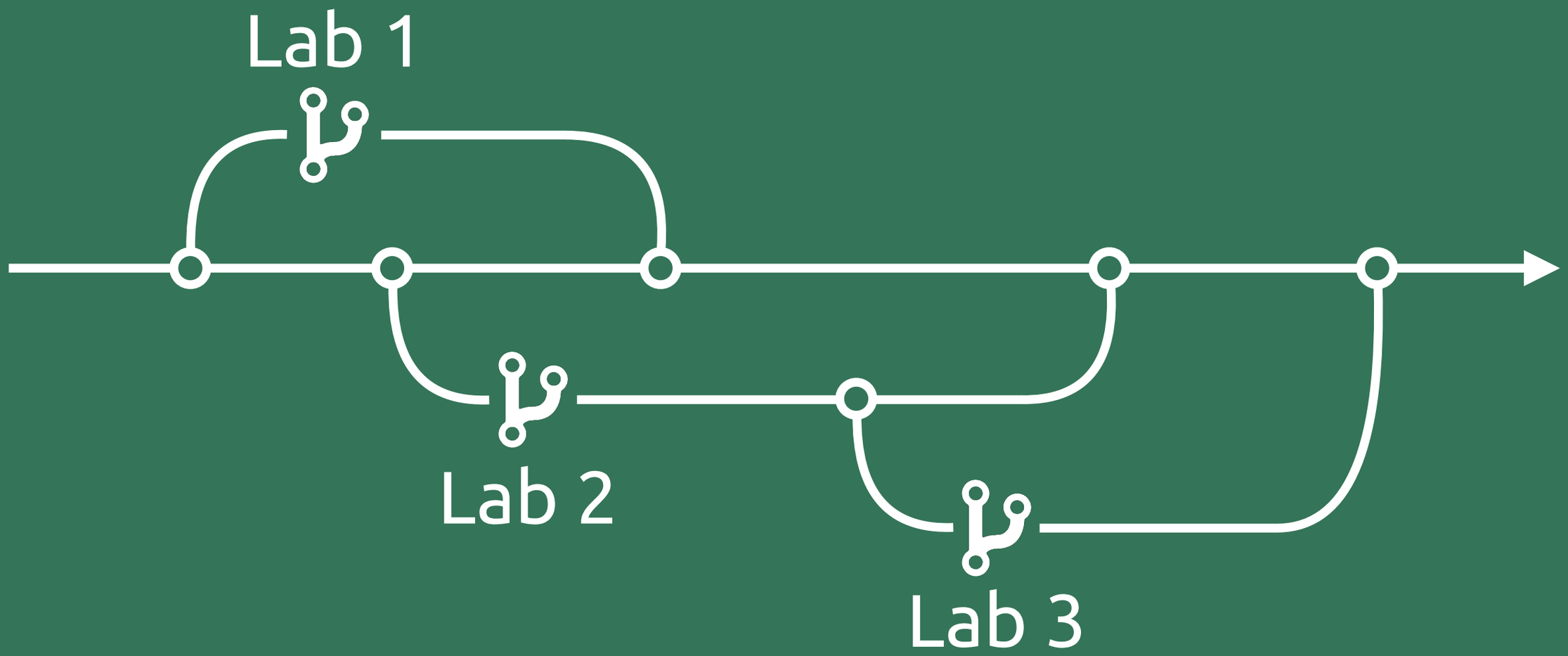
Unit testing

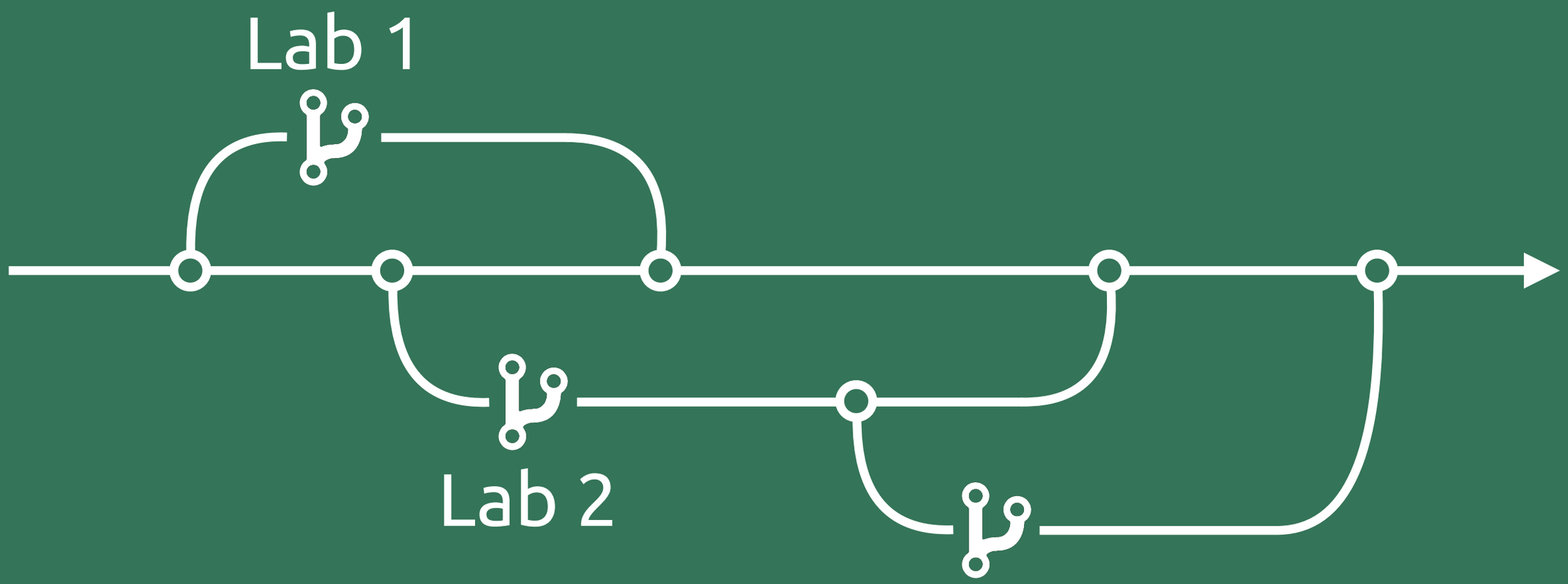
Virtual environments

Code review



https://www.youtube.com/watch?v=zwRdO9_GGhY


Lab 1
Lab 2
Lab 3



Be the change!