Python in the browser with Pyodide

Roman Yurchak

2025/04/07

About me

Roman Yurchak

ML engineer and founder at Symerio

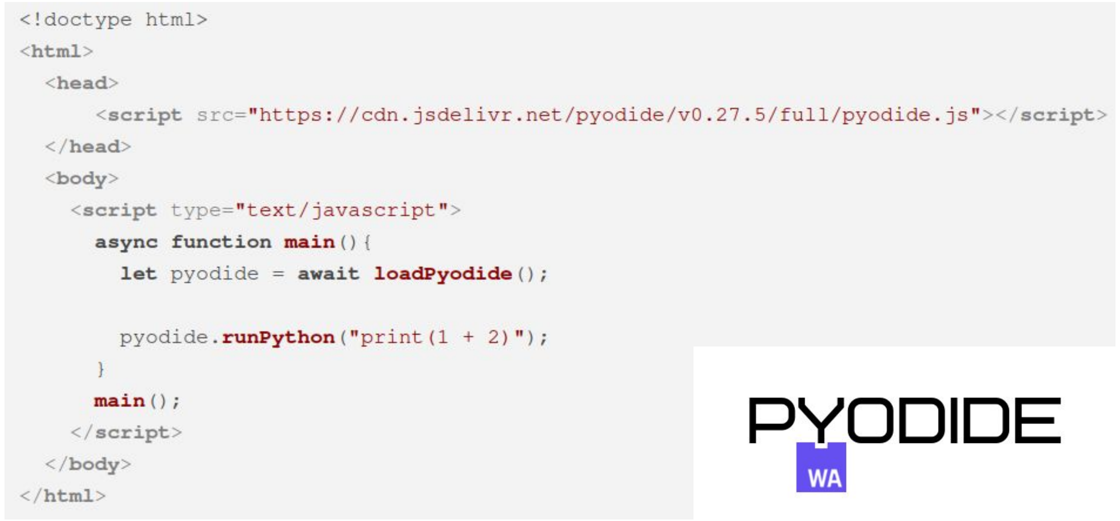
Background in computational physics.

Previously core developer at Pyodide, also scikit-learn

Python in the browser with Pyodide

A Python distribution for the browser and Node.js based on WebAssembly.

- CPython compiled to WebAssembly/Emscripten
- Can install Python packages (including numpy, scipy, ..)
- Javascript / Python bridge



```
<!doctype html>
<html>
  <head>
      <script src="https://cdn.jsdelivr.net/pyodide/v0.27.5/full/pyodide.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      async function main(){
        let pyodide = await loadPyodide();

        pyodide.runPython("print(1 + 2)");
      }
      main();
    </script>
  </body>
</html>
```

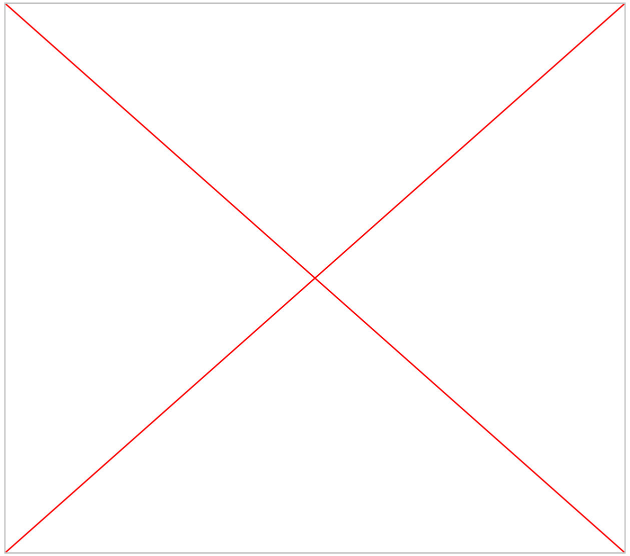
WEBASSEMBLY

PYODIDE

Python simulation in the browser

Example of running a simulation of the 2D Ising model (ferromagnetism in statistical mechanics) in the browser

- Code in Python already exists
- We want to easily share its output
- Using interactive visualization



Source: austen.uk/post/online-demos-with-pyodide/

Agenda

1. Python in the browser with Pyodide

2. Applications: education, scientific computing etc

Python in the browser with Pyodide: an overview

What is WebAssembly?

A binary instruction format for a stack-based virtual machine

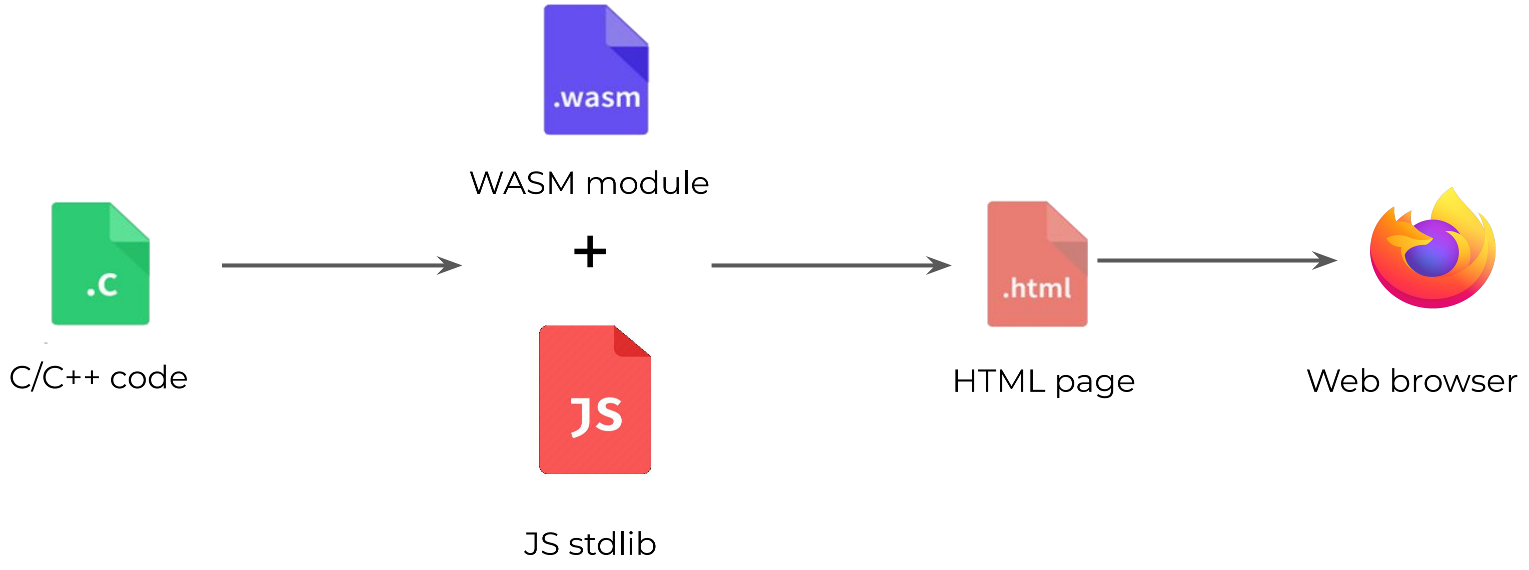
- Portable
- Small code size
- Secure
- No standard APIs or syscalls, only an import mechanism

- Implemented in browsers
- Can also be executed in non-web environments

https://webassembly.org/

The emscripten build toolchain

Emscripten is a complete compiler toolchain targeting WebAssembly



https://emscripten.org/

Pyodide Components

CPython + **Python / Javascript** Foreign function interface → WASM + Javascript stdlib

emscripten

NumPy, pandas, SciPy, matplotlib …

+ **micropip** Pure python wheels from PyPi

Pyodide was created by Michael Droettboom in 2018 at Mozilla

Upstream CPython WASM work

Since 2018 Pyodide was building CPython with many patches.

In 2022 work started on adding WASM build targets in CPython upstream.

Lots of improvements and fixes in Python 3.11

- Upstreaming of Pyodide patches
- Contributing Emscripten fixes
- More of CPython test suite passes

Official Tier 3 support for WASM/Emscripten since Python 3.11

PEP 783: Emscripten Packaging (draft)
 - necessary for wasm/Emscripten wheels on PyPI

Thanks to Hood Chatham, Christian Heimes, Brett Cannon, and Ethan Smith.

Related projects

A number of other projects also allow to run Python in the browser:

- **Brython**: Python 3 javascript implementation + parts of the stdlib
- **pypy.js**: PyPy compiled to asm.js (no longer maintained)
- **RustPython**: using the Rust toolchain to build for WASM

And more recently also,

- **Emscripten-forge**: Build wasm/emscripten packages with conda/mamba/boa
- **CoWasm**: Collaborative WebAssembly for Servers and Browsers. Built using Zig.

For practical usage, compatibility and access to the package ecosystem is critical.

Pure Python packages with micropip

Installed with **micropip**, if wheels available:

- from PyPI or arbitrary location
- rudimentary dependency resolution

Some packages need to be patched,

- with ongoing effort to upstream fixes

Examples

See PEP 427:

-py3-none-any.whl -> pure Python wheel

-cp38-manylinux1_x86_64.whl -> Linux wheel (not compatible with pyodide)

Packages with binary extensions

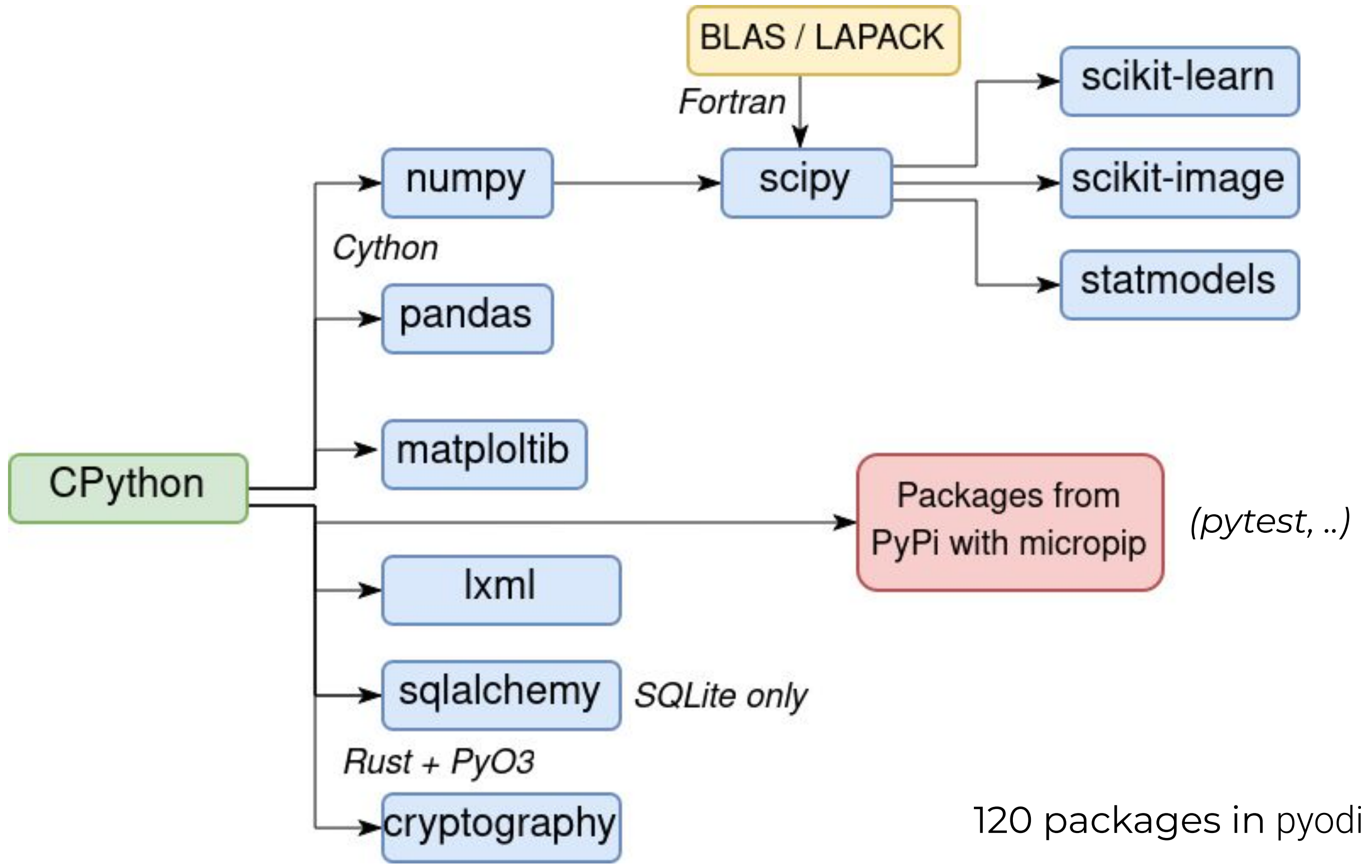
Need to use the Pyodide build system (write a **meta.yaml**, similar to conda)

- A cross-compilation setup, now building wheels
- Recent support for **pypa/build** for build isolation
- Additional post-processing: unvendoring tests as separate packages
- Still some way to a wheel standard for WASM, before their support on PyPI
  - No stable ABI in Emscripten

Wheels distributed via JSDelivr.

There are also other more conda / conda-forge oriented initiatives (**emscripten-forge**).

Supported Python packages in Pyodide



120 packages in pyodide/packages/ …

Foreign function interface (JS ↔ Python)

Using Javascript from Python

A Javascript object in global scope can be imported into Python

```
from js import setTimeout
setTimeout(f, 100)
```

Using Python from Javascript

A Python object in global scope can be accessed from Javascript

```
let sum = pyodide.globals.get("sum");
sum([1, 3, 4]); // 8
```

- Automatic conversion of simple native types (float, str, int, ...,)
- Other types are proxied

For more details: pyodide.org/en/stable/usage/type-conversions.html

Example: Python utils from JavaScript

```
const functools =
pyodide.pyimport("functools");
functools.reduce((x,y) => x*y, [1,2,3,4]);


const math = pyodide.pyimport("math");
math.lcm(4, 6, 13); # Least common multiple
```

Example: random.sample

From Javascript:

```
const random = pyodide.pyimport("random");
random.sample(
   pyodide.toPy(['red', 'blue']),
   5
).toJs();
```

Examples: fetch API from Python

```python
from js import fetch

response = await fetch("example.com", method="GET",
redirect="error")
text = await response.text()
```

Emscripten Host Environment

Features

- 32 bit architecture
- (Javascript) In memory Filesystem
- System calls implemented in Javascript

Limitations

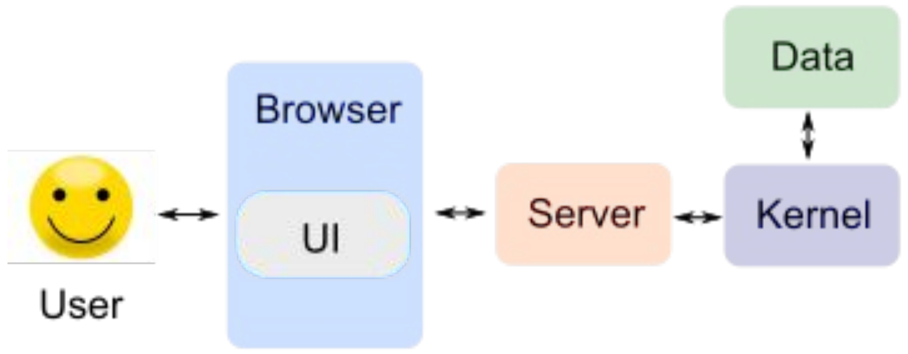
- No subprocess, no threading (theoretically possible, significant work needed)
- No sockets
- Not all syscalls are implemented in Emscripten
- Difficult to use traditional I/O

Some use cases

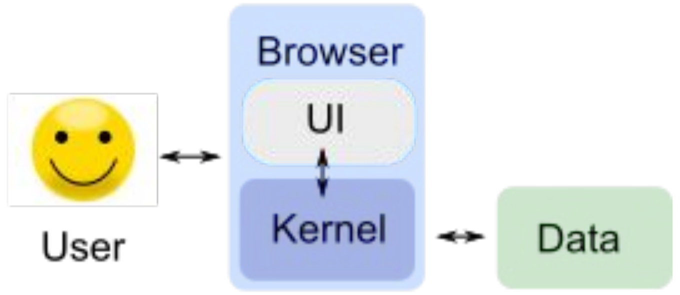
Interactive computing
Education
Machine learning

Client-only Architecture

Application with a backend server



Application with only static files



Client-only Web Apps in Python

Usability

No Python installation needed, just open a web page

Scalability

Serving static files is easy, scales well to a large number of users

- No need for extensive backend infrastructure / maintenance effort

Packages only downloaded once, then cached in the browser

Client-only Web Apps in Python

Privacy

All calculations run locally, no data sent to a remote server

- Good for users
- Good for developers (less GDPR related paperwork)

See: “Analyzing sensitive data at scale doesn’t have to be a headache” by Tambe Tabitha

www.socialfinance.org.uk/blogs/analysing-sensitive-data-scale-doesn’t-have-be-headache

A growing ecosystem

- **Pyscript**: a framework to create rich Python applications in the browser using HTML
  pyscript.net/ by Anaconda <py>

- **Irydium**: Interactive documents and data visualizations in markdown irydium.dev

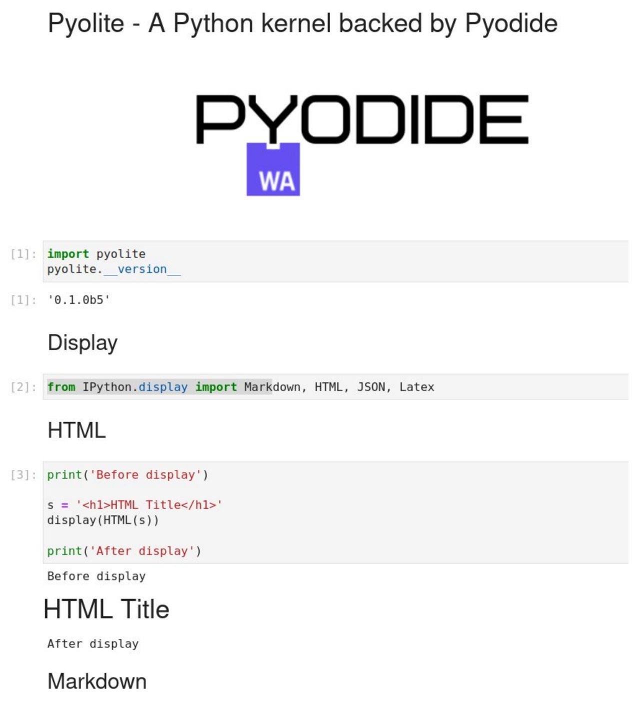
- **React + Pyodide**: using a JavaScript framework in Python
  blog.pyodide.org/posts/react-in-python-with-pyodide/

- **wc-code**: running Python code snippets with HTML tags
  github.com/vanillawc/wc-code

Notebook environments

jupyterlite

jupyterlite.readthedocs.io


Pyolite - A Python kernel backed by Pyodide

PYODIDE WA

```
[1]: import pyolite
     pyolite.__version__

[1]: '0.1.0b5'
```

Display

```
[2]: from IPython.display import Markdown, HTML, JSON, Latex
```

HTML

```
[3]: print('Before display')

     s = '<h1>HTML Title</h1>'
     display(HTML(s))

     print('After display')

     Before display
```

HTML Title

```
After display
```

Markdown

Many other interactive computing projects:

- **Starboard Notebook**: The shareable in-browser notebook starboard.gg/#python
- **Basthon**: Static version of Jupyter notebook notebook.basthon.fr (in French)

Interactive dashboards

voici

https://github.com/voila-dashboards/voici Dashboards with Jupyterlite

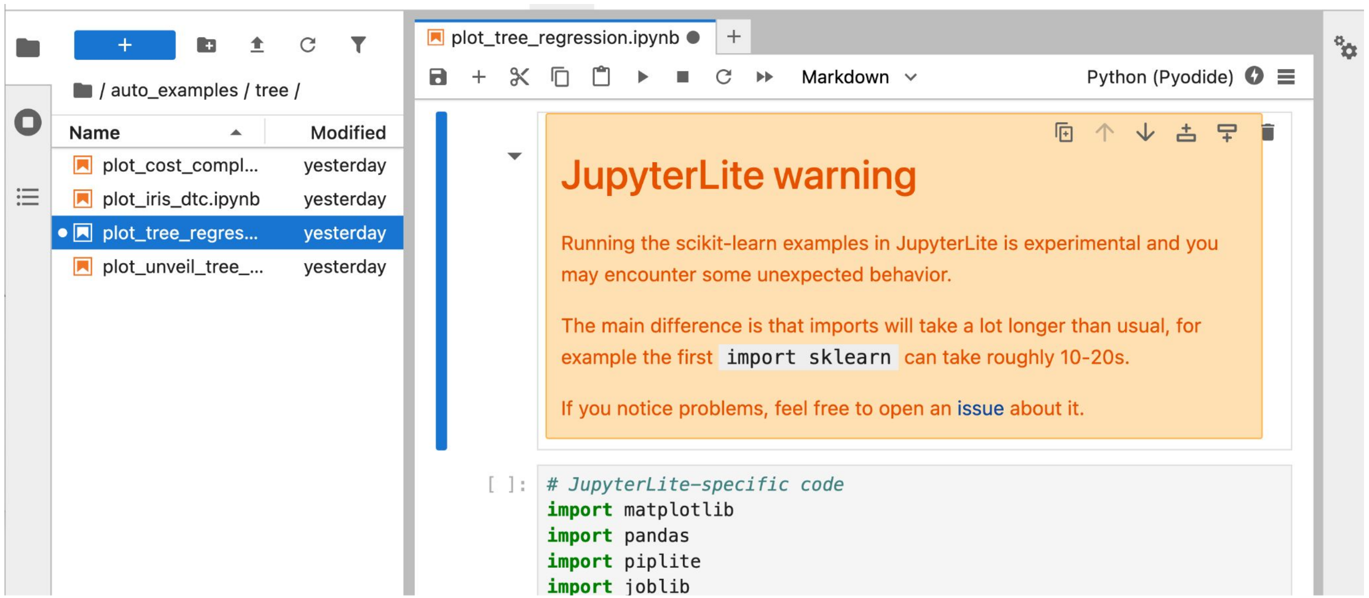
Other solutions:

- Stlite: In-browser Streamlit
- Gradio Lite

Online documentation

Interactive documentation that usage

- **Scikit-learn examples:** https://scikit-learn.org/dev/lite/lab/index.html



/ auto_examples / tree /
Name
Modified
plot_cost_compl... yesterday
plot_iris_dtc.ipynb yesterday
plot_tree_regres... yesterday
plot_unveil_tree_... yesterday
plot_tree_regression.ipynb
Markdown
Python (Pyodide)

JupyterLite warning

Running the scikit-learn examples in JupyterLite is experimental and you may encounter some unexpected behavior.

The main difference is that imports will take a lot longer than usual, for example the first `import sklearn` can take roughly 10-20s.

If you notice problems, feel free to open an issue about it.

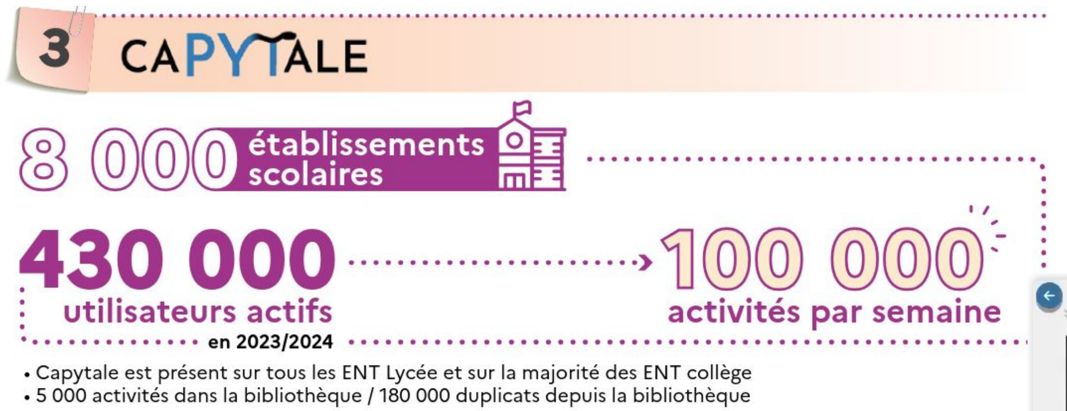
```
[ ]: # JupyterLite-specific code
     import matplotlib
     import pandas
     import piplite
     import joblib
```

Use case: education

- Python now used extensively for education
- Avoid spending time installing Python for students
- Make sure everyone has the same environment
- Privacy preserving (i.e. without using third party services) and without hosting effort

Education: Projet Capytale

Outil de création et le partage d'activités de codage entre enseignants et élèves

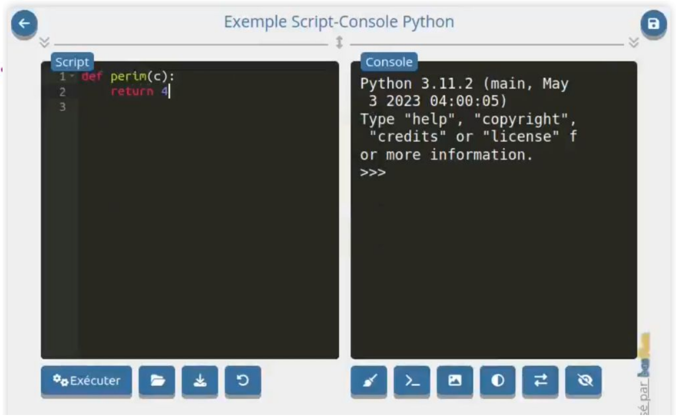

3 CAPYTALE

8 000 établissements scolaires

430 000 utilisateurs actifs → 100 000 activités par semaine

en 2023/2024

- Capytale est présent sur tous les ENT Lycée et sur la majorité des ENT collège
- 5 000 activités dans la bibliothèque / 180 000 duplicats depuis la bibliothèque

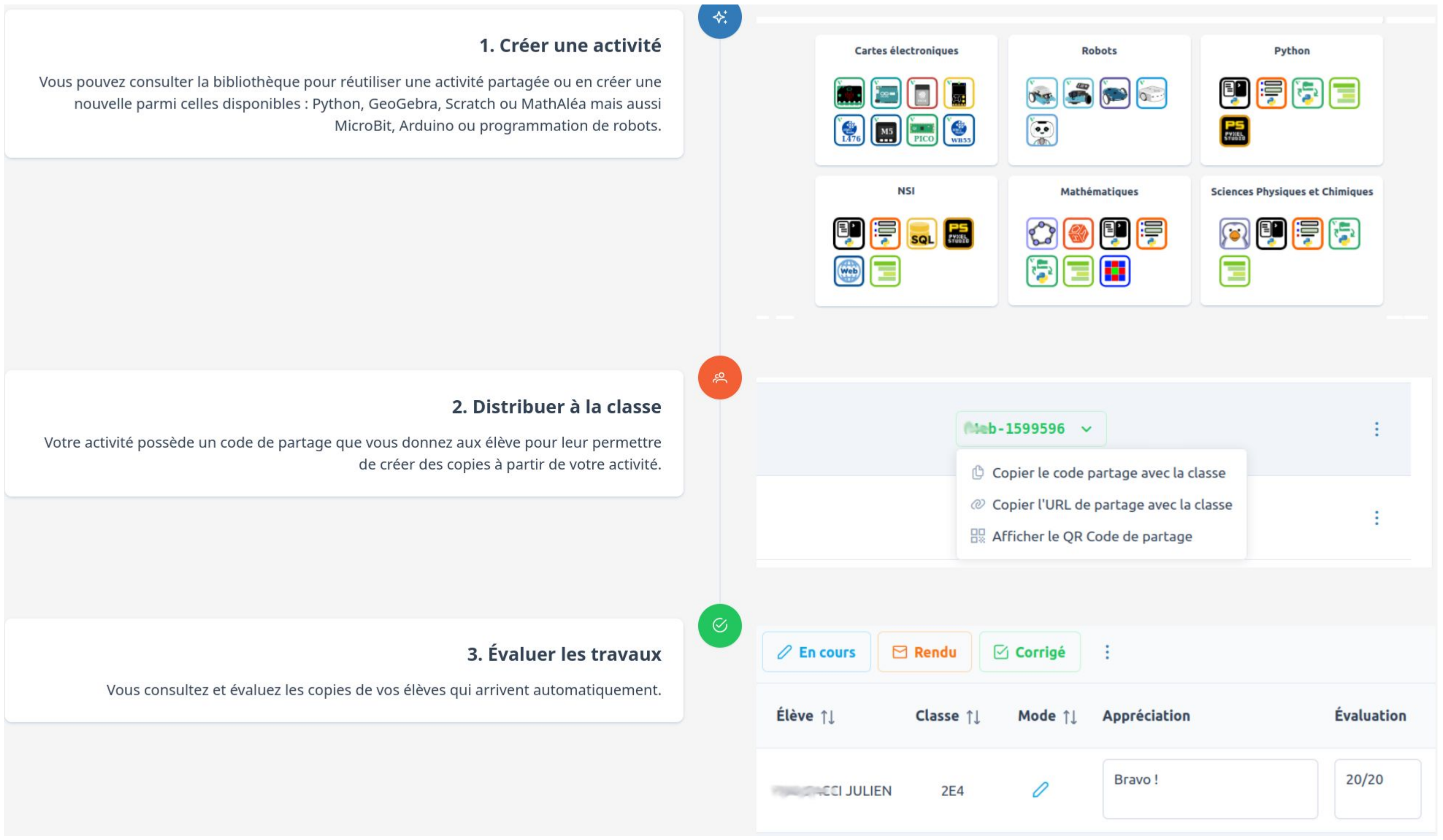


```
def perim(c):
    return 4
```

```
Python 3.11.2 (main, May
 3 2023 04:00:05)
Type "help", "copyright",
 "credits" or "license" f
or more information.
>>>
```

Commun numérique soutenu par le ministère de l'éducation nationale

Education: Projet Capytale



1. Créer une activité

Vous pouvez consulter la bibliothèque pour réutiliser une activité partagée ou en créer une nouvelle parmi celles disponibles : Python, GeoGebra, Scratch ou MathAléa mais aussi MicroBit, Arduino ou programmation de robots.

2. Distribuer à la classe

Votre activité possède un code de partage que vous donnez aux élève pour leur permettre de créer des copies à partir de votre activité.

[redacted]b-1599596
Copier le code partage avec la classe
Copier l'URL de partage avec la classe
Afficher le QR Code de partage

3. Évaluer les travaux

Vous consultez et évaluez les copies de vos élèves qui arrivent automatiquement.

Élève ⇅	Classe ⇅	Mode ⇅	Appréciation	Évaluation
[redacted]CCI JULIEN	2E4	✎	Bravo !	20/20

Machine learning in the browser

Many ways projects allows to run machine

- ONNX
- transformers.js
- Burn

More recently WebGPU support.

Python still useful for pre-processing / post-processing

Usage in LLMs for a Python interpreter

Large Language Models (LLMs) can interact with a Python interpreter to run generated code.

Some use Pyodide to either run in the **browser sandbox** or **run Python from Javascript**.

Examples

- open-webui
- pydantic-ai
- LangChain.js

Latest developments and outlook

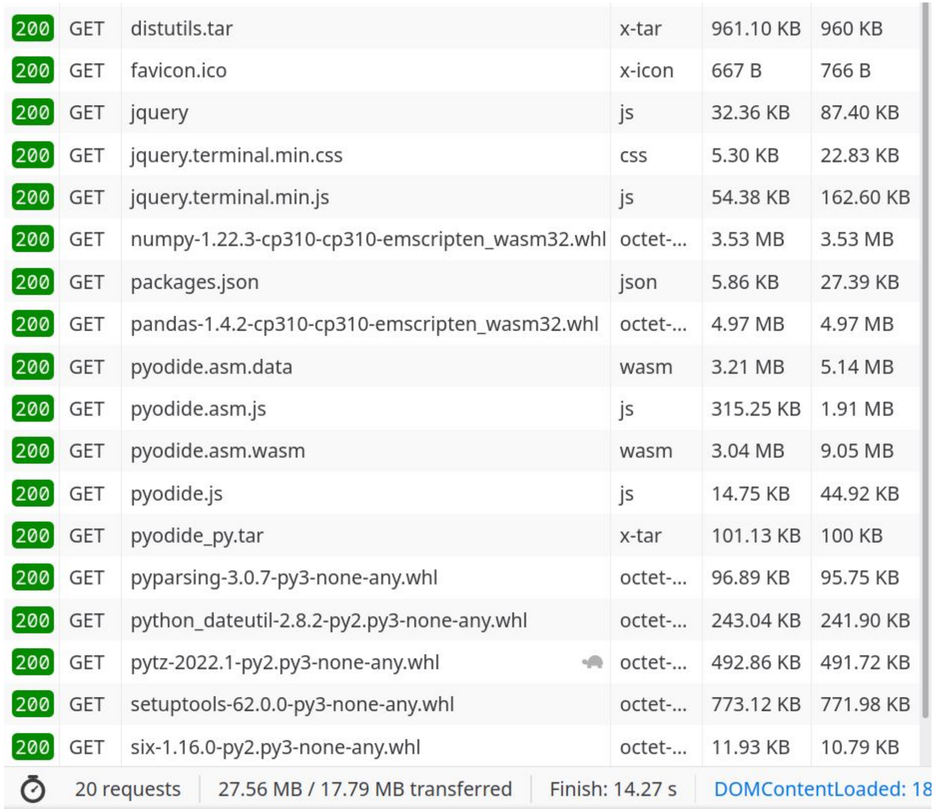
Download sizes for packages

Download size is not an optimisation criterion in the Python ecosystem (unlike for JS)

Historically large packages (e.g. scipy)

Inclusions of test files in the main package (e.g. import numpy.tests)

Example of loading pandas



200	GET	distutils.tar	x-tar	961.10 KB	960 KB
200	GET	favicon.ico	x-icon	667 B	766 B
200	GET	jquery	js	32.36 KB	87.40 KB
200	GET	jquery.terminal.min.css	css	5.30 KB	22.83 KB
200	GET	jquery.terminal.min.js	js	54.38 KB	162.60 KB
200	GET	numpy-1.22.3-cp310-cp310-emscripten_wasm32.whl	octet-...	3.53 MB	3.53 MB
200	GET	packages.json	json	5.86 KB	27.39 KB
200	GET	pandas-1.4.2-cp310-cp310-emscripten_wasm32.whl	octet-...	4.97 MB	4.97 MB
200	GET	pyodide.asm.data	wasm	3.21 MB	5.14 MB
200	GET	pyodide.asm.js	js	315.25 KB	1.91 MB
200	GET	pyodide.asm.wasm	wasm	3.04 MB	9.05 MB
200	GET	pyodide.js	js	14.75 KB	44.92 KB
200	GET	pyodide_py.tar	x-tar	101.13 KB	100 KB
200	GET	pyparsing-3.0.7-py3-none-any.whl	octet-...	96.89 KB	95.75 KB
200	GET	python_dateutil-2.8.2-py2.py3-none-any.whl	octet-...	243.04 KB	241.90 KB
200	GET	pytz-2022.1-py2.py3-none-any.whl	octet-...	492.86 KB	491.72 KB
200	GET	setuptools-62.0.0-py3-none-any.whl	octet-...	773.12 KB	771.98 KB
200	GET	six-1.16.0-py2.py3-none-any.whl	octet-...	11.93 KB	10.79 KB

20 requests | 27.56 MB / 17.79 MB transferred | Finish: 14.27 s | DOMContentLoaded: 18

Roadmap

- Keep up with Emscripten releases (fixes, size and performance improvements)
- Emscripten wheels on PyPI
- Support for synchronous I/O and web workers
- Reduce size of packages
- Improve sustainability of the package build system
- Better support for scipy
- Other features: Threading, SIMD, GPU

pyodide.org/en/stable/project/roadmap.html

New contributors are welcome!

Thank you!

github.com/pyodide/pyodide