Pyodide: scientific Python compiled to WebAssembly

Roman Yurchak

November 11 - 15, 2020

Agenda

Introducing Pyodide

Pyodide UIs and frontend Python apps

Use case: deploying machine learning models

Current challenges and outlook

Introducing Pyodide

What is WebAssembly?

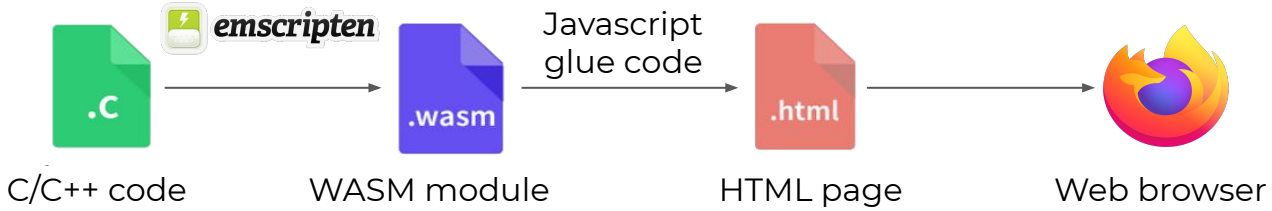
WebAssembly (or WASM) is a binary instruction format for a stack-based virtual machine,

- Initially implemented for browsers
- can also be executed in standalone environments

Features

- portable
- near native performance
- sanboxed

Build pipeline



Pyodide project

CPython 3.8 and the scientific Python stack, compiled to WebAssembly

CPython

+

NumPy pandas

SciPy * matplotlib

* (experimental)

...

WASM

+ micropip

Pure python wheels from PyPi

+ Python / Javascript

type conversions

Project repository: github.com/iodide-project/pyodide

PYODIDE WA

Related projects

Several other projects also allow to run Python in the browser ...

pypy.js

PyPy for Python 2 compiled to asm.js
(No longer maintained)

Brython

Python 3 Javascript implementation + parts of stdlib

RustPython

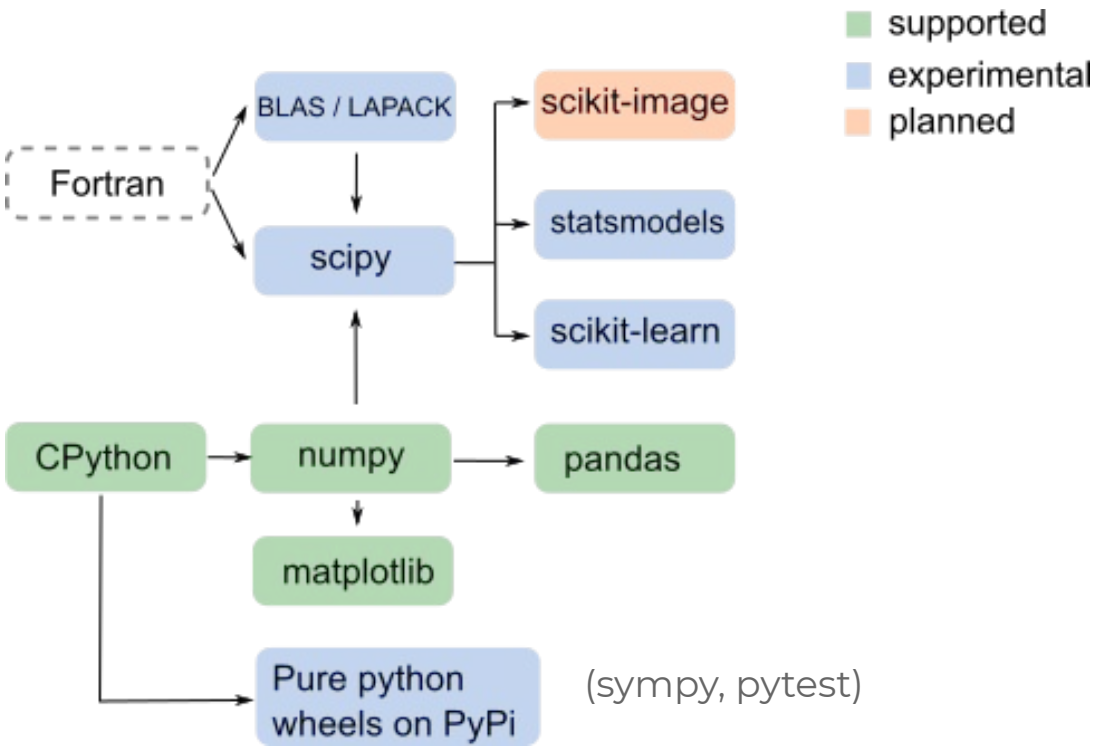
A Python 3 interpreter written in Rust

micropython-ports-wasm

A WASM MicroPython port (experimental)

Supported Python packages in pyodide

that use C extensions



Micropip and Python wheels

Pure Python packages

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

- from PyPi or arbitrary location.
- rudimentary dependency resolution

Examples

See PEP 427:

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

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

Might still need to use the pyodide build system, to apply patches (e.g. unsupported modules)

Python packages with C extensions

Need to use the pyodide build system (write a meta.yaml, similar to conda-forge).

Distributed via pyodide-cdn2.iodide.io (.js & .data files). Can be loaded with pyodide.loadPackage (or micropip).

Python ↔ Javascript type conversions

Bidirectional type conversions,

- between native types when possible (float, str, list, ..)
- otherwise proxy objects are used with a number of supported operators (getattr, setattr, __call__, ...)

Using Javascript from Python

Allows to access DOM or any JS object in the the global (window) namespace

```
import js
js.document.title = 'New window title'
```

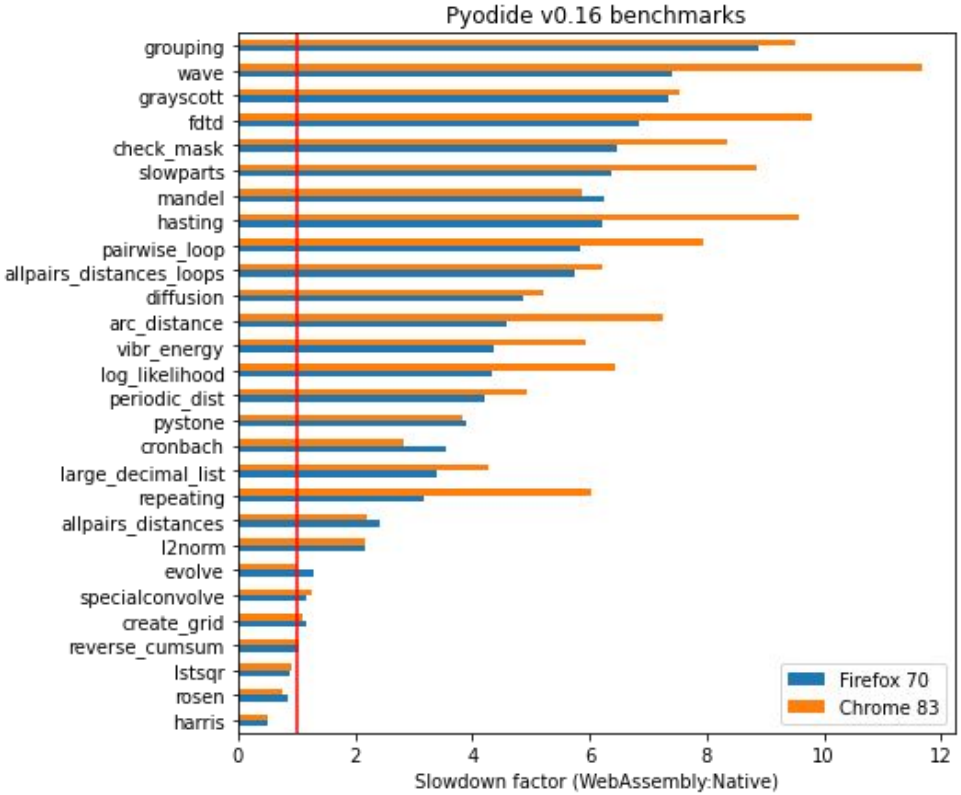
Using Python from Javascript

A Python object (in global scope) can be brought over to Javascript

```
var sys = pyodide.pyimport('sys');
```

Fore more details: pyodide.readthedocs.io/en/latest/type_conversions.html

Performance



Firefox: 4-8 slower for pure Python, 1-2 times slower for C-ext.
Significant speed improvements for WASM in Chrome last year.

Pyodide UIs and frontend Python apps

User Interfaces for pyodide (1)

Pyodide REPL

A simple JS console

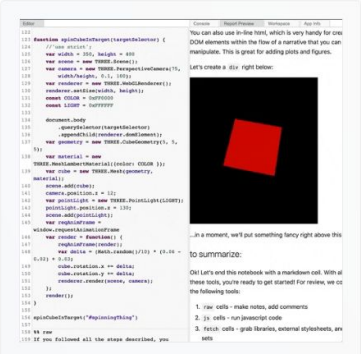
```
Welcome to the Pyodide terminal emulator
>>> from js import document
>>> document
[object HTMLDocument]
```

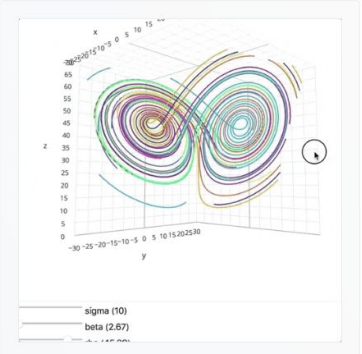
Iodide

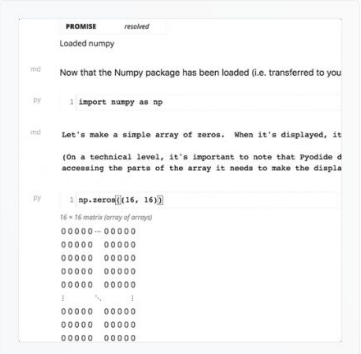
Literate scientific computing and communication for the web

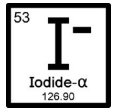
alpha.iodide.io

(no longer actively developed)





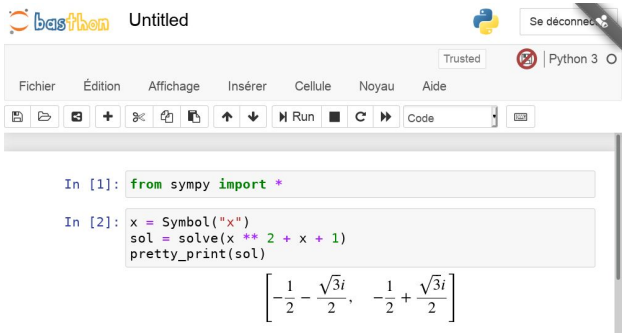




User Interfaces for pyodide (2)

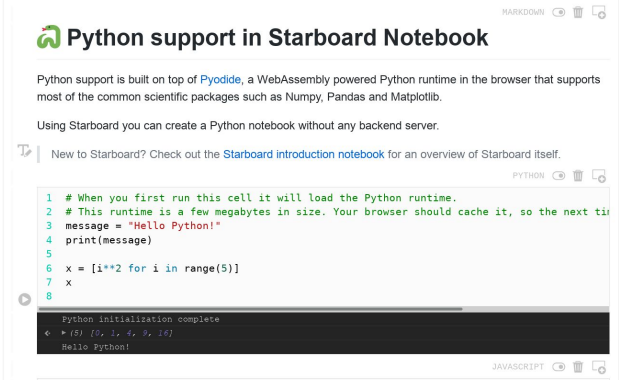
Basthon

Static version of jupyter notebook
notebook.basthon.fr
(first version in French)



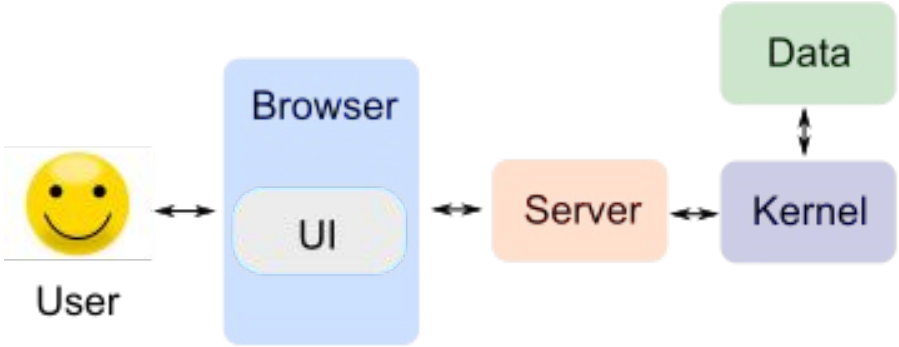
Starboard Notebook

The shareable in-browser notebook
starboard.gg/#python

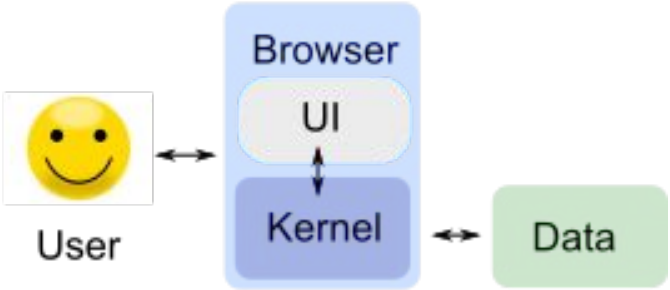


Architecture

Application with a backend server



Application with only static files



Frontend web apps in Python

Usability

Still 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
- *Example*: currently 50k-80k downloads/month for the main pyodide package

Packages only downloaded once, then cached in the browser

Frontend web apps in Python

Privacy

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

Algorithmic Transparency

The app is just an archive with python files/objects

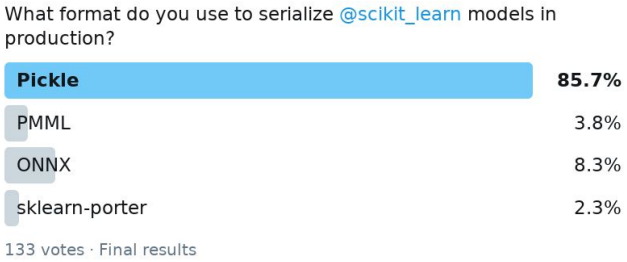
- can be examined on the client side
- can be either and advantage or a disadvantage depending on the use case

Use case: deploying machine learning models

Deploying machine learning models

Classical workflow

1. Train the machine learning (ML) model
2. Serialize model to disk
3. Develop a web service
4. Package in a container (Docker)
5. Deploy on a server



Tools for ML inference with WASM support

ONNX

TensorFlow.js

tract

Fast, small model size but restricted to predefined operators...

Deploying scikit-learn models in pyodide*

Use pickle?

Unsafe, brittle to environment changes **but** portable and non opaque

Steps

1. Créate an environment with the same Python and dependencies versions as pyodide
2. Pickle the model (pickle.dumps) and deserialize it in pyodide (pickle.loads)
3. Run inference from JS

Walk through: github.com/rth/notebooks/tree/master/pyodide/PyDataGlobal2020-demo

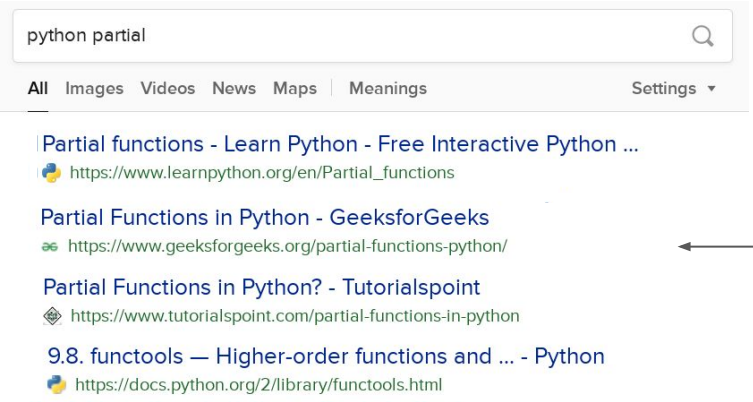
* Very experimental: see next part for the remaining challenges for use in production

Personalized active learning

Not only inference, we could also re-train (or fine tune) models in the browser.

- Smaller, personalized dataset
- Privacy preserving, control over the objective function
- Though, challenging model evaluation

Example



Python 3 stdlib docs, please!

Current challenges and outlook

Download sizes for pyodide packages

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

Historically large packages (e.g. scipy)

- Possibly due the state of Python packaging 10+ years ago

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

- Also sometimes test data

Static imports analysis is challenging

- blanket imports in __init__, conditional imports

Package	Size	Size (brotli compression)
Python/pyodide core	22 MB	7.6 MB
pandas	16 MB	8.2 MB

Significant optimization potential for the package sizes.

Known limitations

Some of the constraints that require workarounds,

WASM VM

- No subprocess, no threading (WIP)
- No sockets (HTTP requests need to be expressed with JS)
- 32 bit architecture
- Not all syscalls are implemented in emscripten

JavaScript

- No int type, only Number or BigInt
- No standard ND array type
- Can use reserved keywords from Python

```
Promise.new(...).then(...).finally(...)
```

→ SyntaxError in Python

Outlook

- Keeping up with emscripten releases (many fixes, size and performance improvements)
- Upstreaming patches
- Re-implement some stdlib modules (e.g. http.client) with Web APIs
- Avoid blocking the main JS thread during calculations
- Dynamic linking of LAPACK and better scipy packaging
- Sustainability of the package build system
- Threading (waiting for wider browser adoption)

New contributors are very welcome!

Both technical and non technical.

PYODIDE WA

Team and partners

Michael Droettboom
William Lachance
Brendan Colloran
Hamilton Ulmer
Teon Brooks

Jan Max Meyer
Madhur Tandon

Roman Yurchak

Pyodide contributors

Partners and related project

Thank you!

https://github.com/iodide-project/pyodide

See PyData Global 2020 program for dates of question sessions.

@RomanYurchak

roman.yurchak@symerio.com