Notebooks are a simple, fun and efficient way to create business reports or play with data science libraries such as scikit-learn or pandas. They are also a great tool for interactive visualisation of data. We have been using Jupyter notebooks at Nexedi as part of Wendelin out-of-core Big Data project. Using Jupyter, we could increase the productivity of engineers in charge of producing reports about the structural health of wind turbines in Gemany. Using Jupyter, we could also provide a tool to analyse and visualise sales trends of a trading company in China. We are currently integrating Jupyter-Lab as the default IDE of SlapOS Edge Computing software.

Jupyter has been of tremendous help at Nexedi. We will keep on using it, especially Jupyter-Lab for server-based software development. But Jupyter has two major drawbacks: it does not scale and it is not ubiquitous. It does not scale because serving Jupyter notebooks to thousands or millions of users requires powerful servers and complex software setup, which tends to break the kind of efficiency or simplicity one can experience as a single user. It is not ubiquitous because one has to install Jupyter to use it, and most users today no longer want to install anything and instead expect everything to be free.

Mozilla Iodide: HTML5 notebook

Along came Iodide, a project originated by Mozilla and lead byHamilton Ulmer and Brendan Colloran.

Iodide is an HTML5 and Javascript notebook that works similar to Jupyter. But instead of processing data on server side, it processes data directly in the browser. Iodide supports dynamic loading of virtually any Javascript library, including plot.y and mpld3 for visualisation and numjs for linear algebra. It also supports natively ndarray data structures which are commonly used in Javascript (see for example scijs).

Iodide is distributed as a collection of static files that can be hosted on a low-cost Content Delivery Network (CDN). There is no need of any kind of application server to deploy Iodide and disseminate it to million users.

The absence of server side component is what makes Iodide so different and so much better than Jupyter for most applications. With Iodide, a multinational corporation can distribute to all its employees a business reporting framework packed with the best of A.I. libraries without virtually no investment in any form of license or infrastructure.

Pyodide: lightspeed data science programming

Although Iodide is written in Javascript, it can load extensions and support other languages. Pyodide is an extension for Iodide initiated by Michael Droettboom that provides a python interpreter compiled in Web Assembly (WASM). Pyodide comes with a growing number of libraries for visualisation and data sciences. Thanks to Web Assembly, those libraries run at near native speed inside a Web browser.

It is therefore possible to run within the same runtime - the Web Browser - both Javascript and Python code which share the same ndarray data structures.

A surprising consequence of this dual headed runtime is that it slashes drastically the time it takes to develop interactive data analysis and visualisation Web based applications. During our initial evaluation of Iodide, we tried to develop a simple application which we had previously developped using two other frameworks: Bokeh and Wendelin. Here are our results.

Comparing data science frameworks

	Wendelin 1.0	**Bokeh**	**Pyodide**	**Wendelin + Iodide**
Time to develop	2 months	2 weeks	2 days	2 days
Works offline	yes	no	yes	yes
Support arbitrary library JS without python wrapper	yes	no	yes	yes
Supports out-of-core arrays	yes	no	no	yes
Supports data management rules	yes	no	no	yes
Distributed computation and exabytes storage	yes	no	no	yes

In both cases, the assigned developer was using the framework for the first time. Before Iodide existed, the choice was simple: Wendelin provided enterprise grade features that are needed to industrialise a data collection and processing project whereas Bokeh provided rapid development. It was one or the other.

We found that with Iodide, we were able to provide even faster development than Bokeh. The reason is simple: a Javascript function in Iodide can call a Python function which can itself call a Javascript function. All the extra layers or wrappers that had to be considered between Python and Javascript in both Wendelin or Bokeh were gone. Developing interactive applications that combine Python and Javascript becomes instant in Iodide.

It is also much easier to integrate Iodide to Wendelin than Bokeh or other frameworks based on an application server. With RenderJS, Iodide can become a UI gadget component, just like other existing components (graphs, spreadsheets, etc.). This is because one of the key concepts of Iodide is to have no dependency to any application server and to be based only

on static files.

Nexedi supports Iodide

Nexedi therefore decided to invest in Iodide so that it becomes the standard rapid application development environment of the next generation Wendelin platform by combining benefits of Wendelin and of Iodide in the same environment. Iodide will also become a standard reporting tool of OfficeJS HTML5 suite with close integration to ERP5 open source ERP/CRM.

A senior developer, Roman Yurchak, was sponsored by Nexedi to contribute to the core of of Pyodide. This includes adding dynamic module loading (from arbitrary Web URL), automating Numpy test coverage and porting scikit-learn.

Meanwhile, Richard Szczerba, a young developer has been finalising and extending the work of Laurent Sebellin (Nexedi GmbH) to integrate Iodide into OfficeJS. OfficeJS Iodide can now save notebooks in ERP5 or Dropbox, stream data from remote storages and - of course - operate offline.

This investment will keep on in 2018 and 2019 thanks to funding secured from France's FUI public research fund. Any developer or intern is welcome to join Nexedi for short to long time and contribute to the development of Iodide or Pyodide (write to jobs@nexedi.com).

OfficeJS Iodide Notebook

OfficeJS Iodide Notebook is a new member of the OfficeJS appstore that provides a simple way to use, edit and manage, online or offline, multiple Iodide notebooks stored locally or on remote online storages.

OfficeJS is an HTML5 appstore that includes an OpenXML compatible office suite (text, spreadsheet, presentation), an HTML5 compatible office suite (text, spreadsheet, illustration, imaging) and a few applications for daily use (expense tracking, bookmarks, PDF, music player, etc.).

Thanks to service worker technology powered by CribJS, OfficeJS HTML5 applications can operate both online and offline (only on latest IOS). This is how OfficeJS Iodide Notebook can operate entirely offline.

Thanks to storage abstraction powered by JIO, OfficeJS HTML5 applications can store data locally inside the browser (IndexedDB), remotely onto online storages (Dropbox, Google Drive, WebDAV, ERP5, etc.) and synchronise both. This is how OfficeJS Iodide Notebook can store and retrieve the notebook's jsmd text to and from a wide variety of storage without depending on any application server or changing Iodide's code (currently, the last run of any cell is stored in localStorage by Iodide).

And thanks to RenderJS, a lightweight component framework, OfficeJS applications run fast on slow devices (low-end smartphone, ARM based chromebooks, etc.) and can integrate well with other frameworks (Angular, REACT, etc.). RenderJS is the framework that provides to OfficeJS Iodide Notebook the ability to add and display a list of notebooks that support full text search.

All OfficeJS applications are implemented as a collection of static assets (HTML, CSS, JS, etc.) that can be encapsulated into a ZIP file and hosted on any static HTTPS server. No application server is needed. Same applies to OfficeJS Iodide Notebook.

Building OfficeJS Iodide Notebook

The build process involves the following steps:

- build Iodide on a remote Linux virtual machine together with various python libraries required (see https://www.erp5.com/erp5-HowTo.Build.Pyodide for a complete description);
- combine into a single zip file Iodide static files built previously together with JIO, RenderJS and a manifest file used for Progressive Web Applications (PWA);
- upload the resulting zip file into OfficeJS appstore;
- wait for publication validation.

The complete OfficeJS Iodide Notebook consists of about 130 files for a total of 40 MB of assets. It was tested on a load end ARM based laptop with reasonable performance.

Iodide vs. Jupyter Benchmark

We then decided to run some tests on Iodide to ensure that it can do the job in a real business context.

To compare Iodide and Jupyter, we selected a python based Jupyter notebook that retrieves sales records from an ERP (ERP5), processes them and visualises results with matplolib. This report is used in a daily business situation to analyse best sales in a trading company and display trends.

We thus created an equivalent notebook using Iodide and Pyodide. We used JIO to retrieve sales record from ERP5 and Plotly.js to visualise results. The rest of the code, mainly in python, is similar to the original Jupyter notebook.

We achieved in rather short time to make a Pyodide notebook that provides same results as the oiginal Jupyter notebook.:

Here are our observations:

- Running Pyodide requires a lot of RAM (ex. 8 GB) for typical business data processing, else the web browser can crash;
- We could not use chinese characters because the xlrd library failed to read them properly when importing the data into pandas (we had to convert data first to CSV);
- Initialising pandas took 3 minutes on Chrome while it took 10 seconds on Firefox;
- graph visualisation is based on seaborn library which is not available (yet) in Pyodide since it has scipy as a prerequisite (we used matplotlib instead);
- dynamic loading of python modules is possible by calling something like `pyodide.loadPackage('https://foo/bar/numpy.js'` where numpy.js is a Javascript representation of a python module.

Dynamic loading of python modules from a URL is really a useful feature since anyone can convert existing python code into a Javascript representation and publish it somewhere on the Web. There is thus no longer any need to rebuild Pyodide to extend it. This feature, sponsored by Nexedi, is now described in Pyodide's documentation.

JIOdide: Data Access Jedi

Interfacing Iodide with data sources automatically has been a kind of challenge and a dilemma until now:

- anything would be possible through a server based data access proxy, but then Iodide would lose its unique "static asset only" nature;
- or data would have to be published over https in an ad-hoc way.

We solved this dilemma in OfficeJS by integrating JIO as part of our standard Iodide build. JIO is a library that provides a unified way to local and remote data sources from the Web browser:

- create data;
- read data;
- write data;
- query data.

It applies to both data (ex. files as in a file system) and records (ex. lines in a relational database) through a unified API.

Thanks to JIO, we were able to eliminate the need of exporting data to CSV/Excel. We could instead to load data straight from our ERP (ERP5) through simple JIO calls.

The illustration bellow shows an example of accessing ERP5 data from Iodide:

This data can then be turned into a graph:

JIO can operate without any server side proxy or adapter. It can be extended to support more data sources (ex: Google Drive, Amazon S3, Qiniu,etc.). Its only (important) limitation is that it requires good support of cross-origin resource sharing (CORS), a standard feature of HTTP protocol that some cloud providers still refuse to support. Without CORS, a simple HTTP proxy is a must.

Daily Iodide at Nexedi

A few programmers at Nexedi started using Iodide and Pyodide as a scratchpad to test snippets of js and python code.

Iodide was introduced as a playground for third party system integrators to discover JIO library and to access ERP5 records from Javascript or python applications.

Iodide notebooks are saved and synchronised in Nexedi's ERP5 instance, just like any other corporate document.

We hope to increasingly use Iodide to generate business reports for our own use or for our customers. Thanks to RenderJS, we plan to integrate Iodide into our business applications as iframes in the same way as we do currently with complex spreadsheets.

Future improvements

Based on our current use of Iodide and Pyodide, we would like to see or contribute to the following improvements:

- support of scikit-learn (ongoing);
- generic and extensible import / export API in iodide (rather than ?gist= type URLs) so that we can directly plug JIO into Iodide's core;
- generic and extensible python module loading API so that it is possible load python modules from different kinds of Web based repositories;
- Iodide API for read access to JSMD and Tools, with a callback function to inform if the state has been changed in any way so that it is possible to inform the user that they have unsaved changes.

Article Contributors: Richard Szczerba, Sven Franck, Valentin Benozillo, Jean-Paul Smets.