
M348 JUPYTERLITE DESKTOP APPLICATION

M348 Module Team

The Open University

23rd Oct, 2024

Developed by  The Open
University

1 Introduction

The *M348 JupyterLite Desktop Application* is a self-contained executable package that publishes a powerful JupyterLab environment from your computer that you can access using your web browser.

All code is executed within the browser, and no files need to be loaded from external websites. This means that the application can be run without any access to a network connection.

Files can be mounted into the JupyterLab environment from your computer, edited in the browser, and saved back to your computer.

Data files located in the M348-24J directory your home directory can also be *read* from code running in the browser.

If you have an issues or problems running the application, or any comments or feedback about how we might improve it, please pass them on to the module team via your tutor.

2 Installation and Test

Notes for installing and testing the M348 JupyterLite desktop installation on Windows computers.

2.1 Installation on Windows

The *M348 JupyterLite Desktop Application* is provided as a single executable .exe application (m348-jupyterlite.exe) that requires no installation and that should be run from the desktop.

Double clicking on the .exe file will launch a terminal that runs a start-up script. *It may take up to a minute for the script to run, during which time the terminal will remain empty, Figure 1. When the script does run, it will display various outputs that log its progress.*

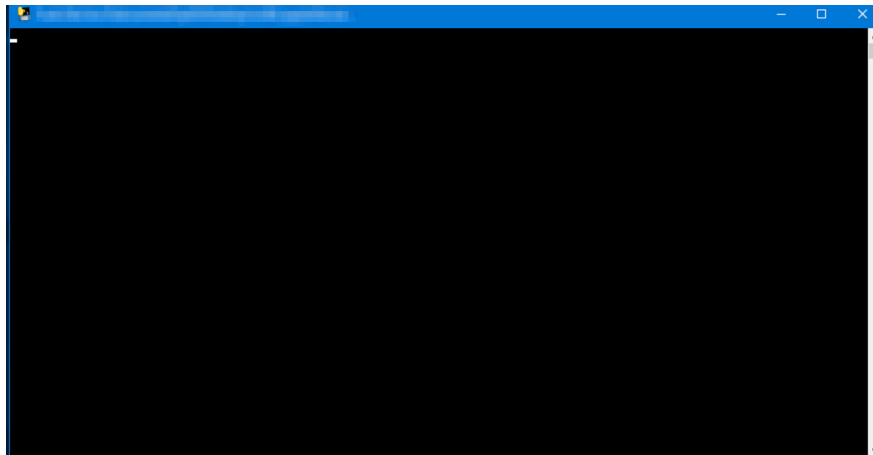


Figure 1: Screenshot of a blank terminal

As part of the startup sequence, the startup script will check to see if an M348-24J directory appears at the top level of the user's home directory (for example, C:\Users\<username>\M348-24J on Windows, or /Users/<username>/M348-24J on a Mac). Data files contained in this directory can be *read* from the notebooks, but not written to. The script will also attempt to write an installation test data file (jl_distro_data_test.csv) into that directory.

You will be prompted to enter a port number, Figure 2. In the first instance you should accept the default value (8348) by hitting Enter.

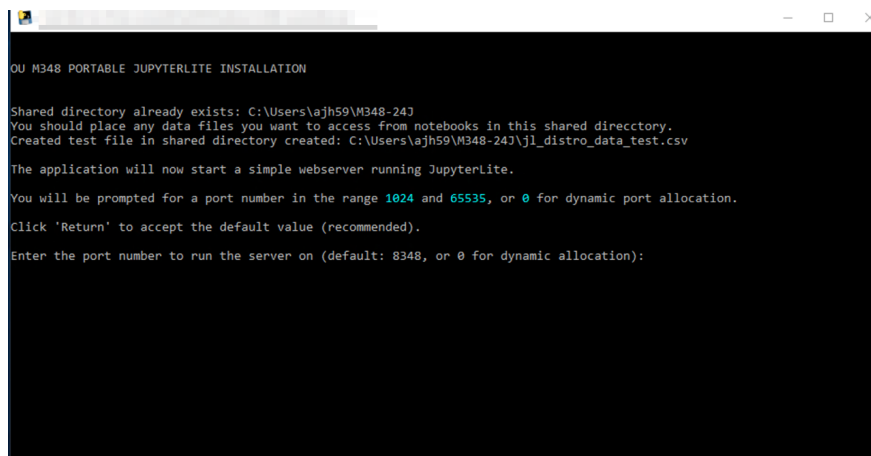
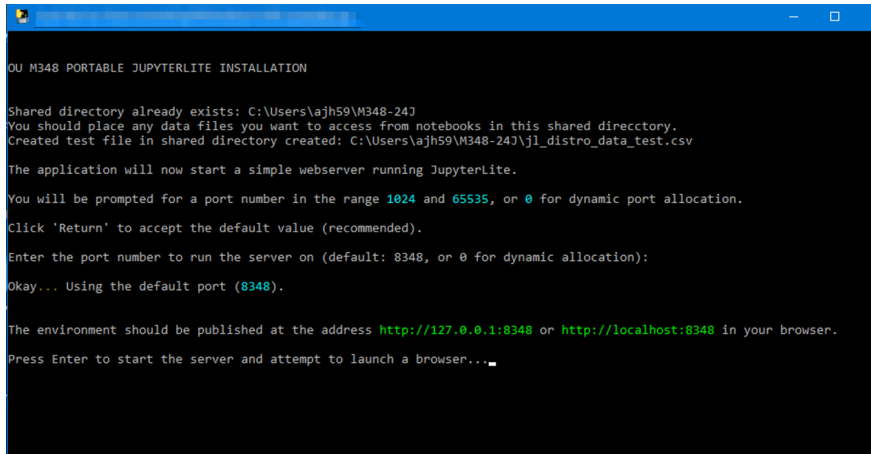


Figure 2: Windows terminal screen showing port number prompt request, with default 8348.

A port will be allocated based on your response, and the port number being used will be reported back to you. Make a note of the web address the application will be published to.

As prompted, click Enter again to start the application webserver, Figure 3.



```

M348 PORTABLE JUPYTERLITE INSTALLATION

Shared directory already exists: C:\Users\ajh59\M348-24J
You should place any data files you want to access from notebooks in this shared directory.
Created test file in shared directory created: C:\Users\ajh59\M348-24J\jl_distro_data_test.csv

The application will now start a simple webserver running JupyterLite.

You will be prompted for a port number in the range 1024 and 65535, or 0 for dynamic port allocation.
Click 'Return' to accept the default value (recommended).

Enter the port number to run the server on (default: 8348, or 0 for dynamic allocation):
Okay... Using the default port (8348).

The environment should be published at the address http://127.0.0.1:8348 or http://localhost:8348 in your browser.
Press Enter to start the server and attempt to launch a browser...

```

Figure 3: Windows terminal screen showing port allocation, awaiting server start

You should now be able to access the JupyterLite environment in your browser. By default the application will be published to <http://localhost:8348> or <http://127.0.0.1:8348> (these both refer to the same location).

A browser should automatically open the application at the appropriate URL. If the browser does not open, use the location provided previously.

If you used a port number other than the default (8348), or if you specified the port as 0 to dynamically assign a port, you should use the port number you specified, or the dynamically allocated port number, when accessing the environment.

2.1.1 In Case of Startup Issues

If the application does not run on the default port, close the terminal window and try rerunning the application. This time, enter 0 to use a dynamically allocated port number.

2.2 Installation testing

Once the application has been launched and accessed in the browser, you should be presented with a view of a JupyterLab environment.

If you have used the default port value (8348) or a port value other than 0, the application should have automatically opened into browser using the Jupyter Notebook view.

Two notebooks are bundled into the environment: `M348-styling-test.ipynb` and `M348-code-tests.ipynb`.

In the local JupyterLab file browser, double click on the `M348-styling-test.ipynb` notebook to open it. It should render a notebook containing coloured cells and an embedded image, Figure 4.

Double click on the `M348-code-tests.ipynb` notebook to open it. From the Run menu, select *Run All Cells*.

Hopefully, each cell will run correctly (indicated by a green block in left hand margin of each cell). *Note the test to read a written CSV data file may fail. Everything else should pass.*

2.2.1 Problems With Reading Data Files

A `JUPYTERLITE_PATH` R variable should be defined that identifies the location that the R kernel believes JupyterLab is running on.

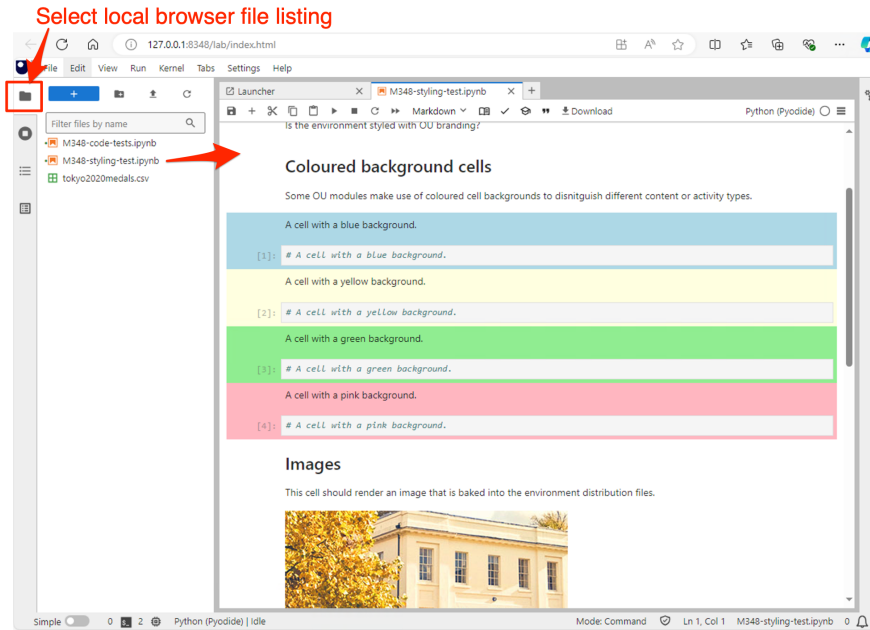


Figure 4: Screenshot of JupyterLab UI showing rendered styling text notebook with coloured background cells and displaying an embedded image

View the value of the variable by running a code cell with the content: `JUPYTERLITE_PATH`, Figure 5.

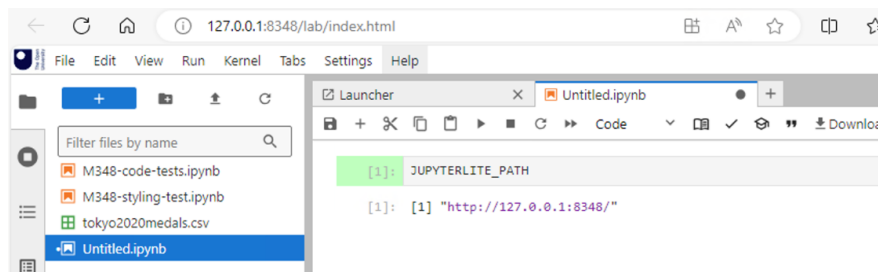


Figure 5: Screenshot showing a code cell displaying the value of the `JUPYTERLITE_PATH` variable. The part of the URL in the browser location bar up to the `/lab` element matches the `JUPYTERLITE_PATH` value, so all is good.

If the value of the path up to the `/lab` part of the address *does not* match the URL in the browser, you will need to set it manually. In such a case, set the `JUPYTERLITE_PATH` value to the value copied from the browser location bar. For example, if the environment is running at `http://127.0.0.1:8348/lab/index.html`, run the following in a code cell:

```
JUPYTERLITE_PATH <- "http://127.0.0.1:8348/lab/index.html"
```

2.3 In Case of Further Issues

If you encounter any other issues with running the desktop application, using JupyterLite, or working with the notebooks, please raise them with your tutor or the M348 module team.

3 User Guidance

Notes for using the M348 JupyterLite desktop installation on Windows computers.

These guidance notes cover:

- starting the JupyterLite server;
- accessing the JupyterLite environment in a browser
- sharing Jupyter notebook files from your computer into the application
- sharing data files from your computer into the application

3.1 Installation on Windows

See the *Installation on Windows* subsection in the *Installation and Test* section.

*In case of issues, please refer to the **Installation** section.*

3.2 Accessing the JupyterLite Environment

When the application is running, you should be able to access the JupyterLite environment in your browser. By default the application will be published to `http://localhost:8348` or `http://127.0.0.1:8348` (these both refer to the same location).

3.3 Accessing the Module Notebooks

The JupyterLite / Jupyterlab environment provides an environment for working with Jupyter notebook documents, which have the file suffix `.ipynb`.

These documents will be distributed in their own `.zip` file archive. Unzip the file archive and put the notebooks in a location where they can be saved. You may find it convenient to place the notebooks in an M348-25J folder in your user home directory (for example, `C:\Users\<username>\M348-25J` on Windows, or `/Users/<username>/M348-25J` on a Mac).

There are two ways of making the notebooks accessible in the JupyterLite environment:

1. Uploading the notebooks to the JupyterLite environment, described in Subsection 2.2.1 and then manually downloading them again if you want to keep an updated copy on your desktop. *If your browser can save browser sessions, any notebooks you upload to JupyterLite will be saved to browser storage and will be available in future sessions.*
2. Allowing the JupyterLite environment to access a folder on your computer using the *Local Filesystem Access* extension, described in Subsection 2.2.2.

3.3.1 Uploading Notebooks to the JupyterLite Environment

To upload files or zipped file archives, click the up arrow ("Upload Files") icon in the JupyterLab file browser toolbar. *It is recommended that you upload files to the top level JupyterLite directory.*

If you upload a zip file archive, you will need to unzip it in order to access the files it contains.

A zip file uploaded to the top level JupyterLite directory can be unzipped in the following way:

1. Open a new R notebook by clicking on the R (webR) icon under Notebook.
2. In the code cell type the following command: `unzip("filename.zip")` where `filename.zip` corresponds to the name of the zip file. *(To preview a listing of the files contained in the zip file, run `unzip("filename.zip", list=TRUE)`)*

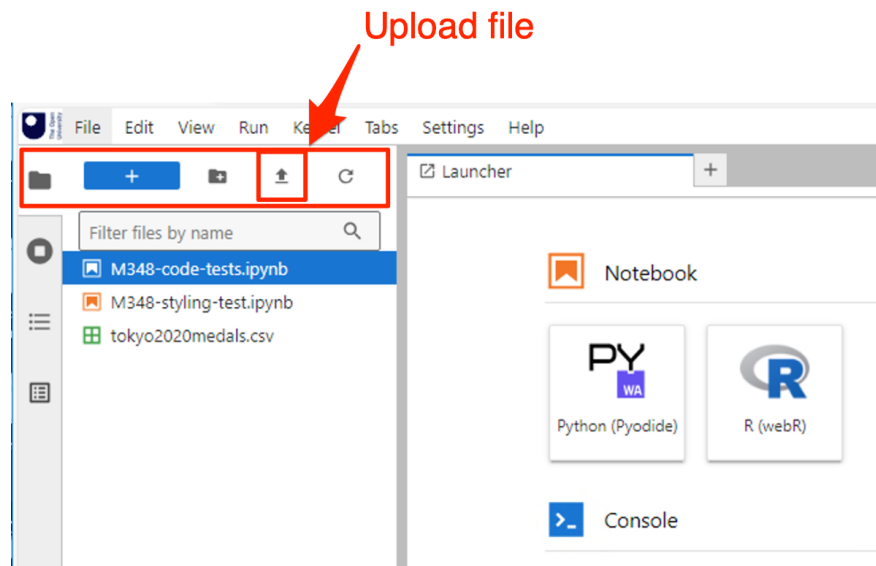


Figure 6: JupyterLab file browser, file upload button (up arrow icon)

3. Run the code cell by clicking on *Run* to unzip the file. The unzipped files should appear in the file list in the sidebar.

If you upload the zip file to a subdirectory in JupyterLite, you will need to set the path to the file. Irrespective of which directory a notebook file is in, R also starts to run in the top-level directory, which had the absolute path `/drive`. If your zip file is in the directory `unit_1`, you need to specify the path, such as `unzip("./unit_1/filename.zip")` (relative path) or `unzip("/drive/unit_1/filename.zip")` (absolute path). The relative path is set relative to the current working directory, which can be identified using `print(getwd())`. The current working directory can be set using `setwd('./unit_1')` (set current working directory relative to current working directory) or `setwd('/drive/unit_1')` (set current working directory using an absolute path).

3.3.2 Local Filesystem Access

The [jupyterlab-contrib/jupyterlab-filesystem-access](#) extension adds local file system access to the JupyterLab environment (*this may not work in all browsers*).

This extension allows you to select a directory from your local filesystem (which is to say, the files on your own computer) and access that directory from JupyterLite, Figure 7.

From the “Open new folder” button in the *Local File System* tab, open a directory on your computer. *You may be prompted for permission*. Sharing the folder allows the browser to read and open the contents of the folder on your computer in the Jupyter environment, Figure 8.

No files are copied: the browser is just given permission to read to them from that directory.

You should then be able to see files shared from your computer into JupyterLab, Figure 7.

You can also create files in the Jupyter environment and save them back to your computer. For example, in the local file system browser, right click and select *New File*. You may be prompted for permission to write to the shared directory (the browser cannot write outside that directory). The file should appear in the folder on your computer. If you open and edit the file in the Jupyter environment, then save it, it should be saved to your computer.

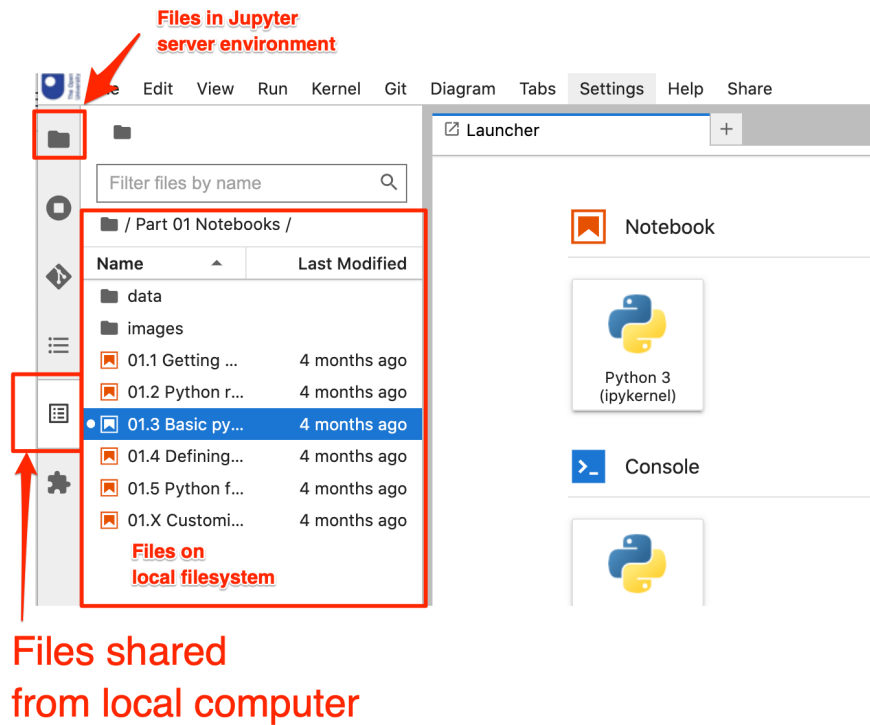


Figure 7: Local file browser.

Screenshot of a sidebar that lists files mounted into the browser from the local file system. *Note that files are shown for a different module.*

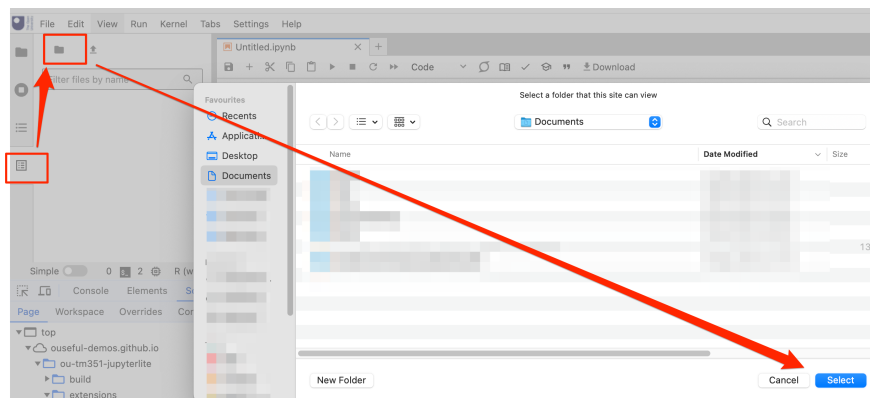


Figure 8: Shared computer filesystem.

Screenshot showing how to mount a directory from your computer into the JupyterLab UI.

Read and write permissions over the shared directory on the local filesystem are granted to the JupyterLite environment (which runs inside your browser) for only as long as it is open in your browser.

Note that *whilst you can preview the contents of CSV data files listed in the local filesystem access file listing, you **cannot** read from, or write to, these shared data files from R code.*

3.4 Reading and writing CSV data files

You can read and write CSV data files to the JupyterLite file area using the `read.csv(FILENAME)` and `write.csv(DATAFRAME, FILENAME)` commands. If you specify a simple filename, this will be read from or written to that filename in the current working directory. If a file cannot be opened, you will see a “cannot open the connection” error.

The location of the current working directory can be identified using `print(getwd())`. The current working directory can be set using `setwd('./unit_1')` (set current working directory relative to current working directory) or `setwd('/drive/unit_1')` (set current working directory using an absolute path).

If you have mounted files into JupyterLite using the local filesystem access extension, you can *preview* the contents of a CSV file mounted from the desktop in JupyterLite by double clicking on the file in the local filesystem access sidebar listing, **but you cannot read or write files** to the desktop from R code. Instead, you will need to *upload* any data files you want to access from R code to JupyterLite and *download* any data files that you write to if you want to keep a copy of them on your desktop.

3.5 Downloading Files from the JupyterLite Environment

To download a copy of an open notebook, click on the *Download* link in the notebook toolbar. This *should* automatically save the notebook before downloading it, but we advise that you *always* save a notebook before downloading it, just to be sure.

To download a file from the file browser sidebar, right click on the file and then select *Download*, Figure 9.

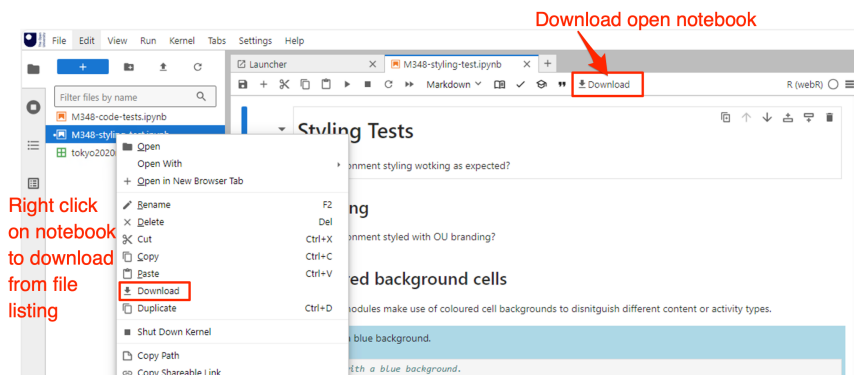


Figure 9: Downloading files from JupyterLab

Screenshot of JupyterLab showing two ways of downloading notebooks: from a Download button in an open notebook; and by right clicking on a notebook in the file browser, then selecting the “Download” menu option.

3.6 Accessing Data Files on the Desktop from the Notebooks

On occasion, you may need to read the contents of a data file on your computer into a notebook. The application is configured so that a notebook can read the contents of CSV data files that are contained in the M348-24J directory inside your personal home directory on the computer using the R `read.csv()` function.

This shared directory **MUST** be at the top level inside your user home directory, for example, `C:\Users\<username>\M348-24J` on Windows, or `/Users/<username>/M348-24J` on a Mac.

Notebooks may also be placed inside this directory, but to edit them in the JupyterLab environment, they will need to be mounted into the browser using the *Local File System* extension described above.