

Anaconda Code BETA

Anaconda Code empowers you to write Python code and run it locally, directly within Excel. This gives you flexibility and control over the Python environment in your workbook, allowing you to add and remove packages as needed, all while keeping code and data securely within your workbook.

Initializing Anaconda Code

Note

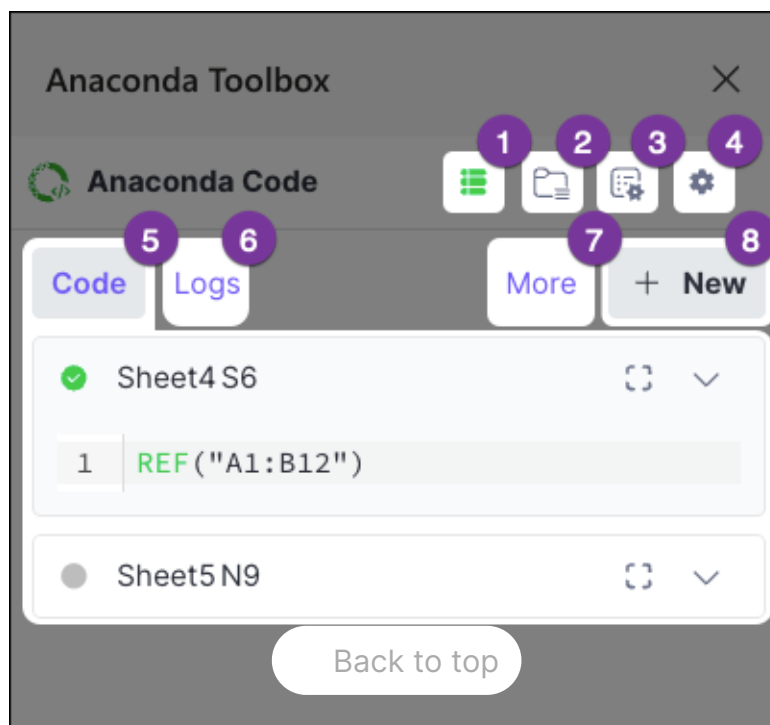
Anaconda Code is included in the [Anaconda Toolbox installation](#).

Anaconda Code is powered by [PyScript](#), our open-source platform for running Python in the browser. When you first launch Anaconda Code, set up and run your PyScript Python environment using the following steps:

1. Click **Enable PyScript**.
2. Once PyScript is enabled, sign in to Anaconda Cloud.

Understanding Anaconda Code

Let's take a look at the different elements within Anaconda Code using the **Dashboard** tab for reference:



1. Dashboard

[Create and run Python code](#) and view script logs

2. Environment

[Manage the packages and Pyodide version](#) for your coding environment

3. Imports and Definitions

[Customize the code](#) that affects all code in your workbook

4. Settings

Modify the [default settings for running code](#)

5. Code

View and [edit the code](#) throughout your workbook

6. Logs

View errors and print statements

7. More

Quickly confirm your code's [cell linking](#) states

8. New code

Create new code to run in your workbook

Running Python code

Begin writing Python code in a cell using the following steps:

1. From the Dashboard, click + **New**, then select a cell where you want to insert your code.
2. Enter your Python code in the code editor. If you want to reference a range of data from your spreadsheet in your code, click @ **Link Range** and select the desired data range.

Using the **REF** function


The **REF** function returns a list of lists and can be used in the following ways:

Function	Use case	Notes
<code>to_df(REF(<EXCEL_RANGE>))</code>	Create a DataFrame	<code>to_df</code> assumes your data has headers
<code>to_array(REF(<EXCEL_RANGE>))</code>	Create a NumPy array	<code>to_array</code> assumes all data is of the same type

You can change the behavior of `to_df()` and `to_array()` from the [Imports and Definitions](#) tab.

3. Set the [cell linking and output options](#).

Note

Toggle between **isolated** and **linked** modes by clicking the plug  button.

4. Click **Save and Run**.


Your code runs in the designated cell.

Editing Python code

Do not edit your code in the cell itself; instead, modify and re-run your code directly in Anaconda Code.

Note

An Anaconda.cloud account is required for users to edit shared code.

1. From the Dashboard, click the expand  button to open the edit view.
2. Adjust your code, then click **Run**.

Managing the environment

Anaconda Code hosts a single, self-contained environment, which manages the back-end software packages that enable you to run Python code within your Excel workbook. You can manage software packages within this environment to extend Python's processing, visualization, and analytical capabilities, and even select the version of [Pyodide](#) (the WASM engine used by PyScript) that you want to run Python.

Note

You can make changes to your environment at any time; however, like with all software projects, altering the environment changes the way the underlying code is interpreted and can cause unintended complications.

Choosing a Pyodide version

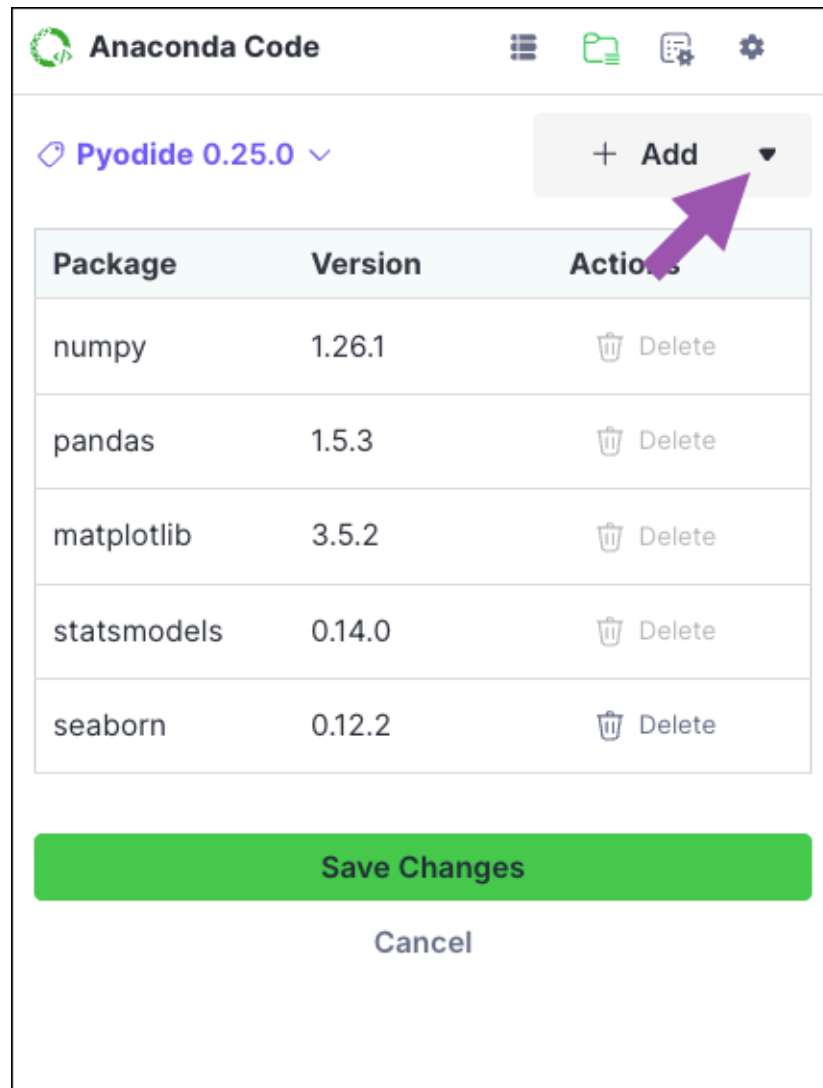
The latest version of Pyodide is used by default for all new spreadsheets. For existing spreadsheets, the versions of Pyodide and packages necessary for your code are pinned to the environment.

You can switch versions of Pyodide using the following steps:

1. From the **Environment** tab, click **Edit**.
2. To switch versions of Pyodide, click the ▾ dropdown beside your current Pyodide version.

Managing software packages

To add new packages, click + **Add**. Alternatively, click the arrow to add from either PyPI or a direct download link to a Python wheel (`.whl`).



Note

Packages that contain compiled code might not be compatible with PyScript's WASM engine. For more information, visit [PyScript.net](https://pyscript.net).


To remove a package, click 🗑 **Delete** beside the package you want to remove.

Customizing code initialization

You can think of Anaconda Code’s **Imports and Definitions** as an initialization file for your code or like the first cell in a Jupyter Notebook. All code in this section is available to all cells, whether they are [run isolated](#) or [linked](#).

To customize your code’s Imports and Definitions:

1. From the **Imports and Definitions** tab, establish the connections to the packages you need to run your code.



Note

You can only `import` from the packages included in the standard Python installation and those listed in the **Environment** tab.

2. Click **Run**.

Modifying workbook settings

While you can adjust the settings for running code in your workbook on a case-by-case basis when creating and editing code, you can also assign default settings from the **Settings** tab.

Cell linking

Mode	Description
Run Isolated	Code runs independently of other cells. Variables declared in previous PyScript cells cannot be referenced. Other cells can reference the return value through the <code>REF</code> function.
Run Linked	PyScript cells run in row major order, comparable to how Python in Excel is executed. Variables declared in one cell can be referenced in cells following the cell that initializes them. When one cell is calculated, all linked cells recalculate in order. Linked cells run left-to-right, top-to-bottom, and can access objects defined in previously linked cells.

Cell output

Output	Description
Excel Values	When outputting a DataFrame, array, list, etc., the values will “spill” to fill the required space. If the spill were to overwrite cells containing data, the cell displays a #SPILL error.
Local Python Object	For certain object types, you can view the contents in a “Card View” by clicking the cell. You can reference this cell and the returned object like you would any other Python object.

Troubleshooting

If you encounter an issue that is not listed here, you can obtain support for Anaconda through the [Anaconda community](#) or by [opening a support ticket](#).

Error installing functions

Cause	>
Solution	>

Was this helpful?