JupyterLab Code Snippets Documentation Documentation

Release latest

Jan 28, 2023

Getting Started

JupyterLab Code Snippets empowers you to write code more rapidly. Try it on Binder.

CHAPTER 1

Overview

1.1 JupyterLab Code Snippets empowers you to write code more rapidly

Do you find yourself typing in the same code blocks over and over again? Are you tired of clicking through tabs online for starter code or browsing old notebook files to find the right import statements? Save, reuse, and share code snippets using JupyterLab Code Snippets! In the past, you would spend more time scouring the internet than actually working. Searching old code for snippets is like searching for your lost keys; it's frustrating!

JupyterLab Code Snippets allows you to create and store code snippets that can be inserted into any JupyterLab workspace and save you from the hassle of typing repetitive code. Code snippets are pieces of code or individual cells that are frequently used. Simply browse or search snippets in the Snippets panel to use wherever you need in JupyterLab.

Using JupyterLab code snippets is easy! Simply open the snippet explorer and browse through the provided predefined code snippets. Still can't find what you need? Simply highlight code and right-click to save as a snippet or drag cells into the panel. Find snippets by utilizing the search, filter, and preview features. For a fresh start, select the plus icon in the snippet panel to construct a completely new, custom snippet. Also, edit your snippet quickly at any time by clicking the edit icon on the snippet. To use your snippets, you can drag and drop a code snippet to insert it as a cell or press the insert icon on the snippet to inject the code where your cursor is located within the JupyterLab workspace.

"I've grown accustomed to opening an old notebook and searching through it in order to find the necessary import statements every time I start a new project. But, with the JupyterLab Code Snippets extension, I saved a lot of time by storing them as snippets and inserting them directly into my workflow." - amazed customer

This new extension unlocks the exciting ability to seamlessly incorporate code snippets. You can now store the frequently used code as a snippet and use it quickly. Save time, save energy, and get down to business with JupyterLab Code Snippets!

CHAPTER 2

Installation

2.1 Requirements

JupyterLab >= 3.0

2.2 Install with jupyter

```
jupyter labextension install jupyterlab-code-snippets
jupyter lab clean
jupyter lab build
```

2.3 Install with pip

```
pip install jupyterlab-code-snippets
jupyter lab clean
jupyter lab build
```

2.4 Uninstall

jupyter labextension uninstall jupyterlab-code-snippets

pip uninstall jupyterlab-code-snippets

CHAPTER 3

Changelog

3.1 v2.1.1

- Made font size of preview configurable in settings

3.2 v2.1.0

- Made snippets globally accessible.
- Improved search : can search for code, language, and name in search bar.
- Expanded tags : now can filter by language tag and by snippet tag.
- Added ability to download snippets to local directory.
- Made right click to open the code snippet contextMenu possible anywhere on code snippet in addition to the three dots.
- Made possible to create or move a snippet while snippets are filtered.
- Set the language of a new snippet as the kernel language by default.
- Improved snippet creation process by making description optional and allowing uppercase and spaces in snippet names.
- Made minor UI and bug fixes.

3.3 v2.0.1

- Fixed the issue with creating the snippet from scratch or editing the snippet in the editor.

3.4 v2.0.0

- Fixed the issue with the keyboard shortcut by making it user-configurable.
- Fixed broken behavior of drag and drop.
- Made the UI more consistent.
- Added a new feature to rename the code snippet in the code snippet panel easily by double-clicking its name.
- Improved the search feature with fuzzy search.
- Added a function to save the entire cell(s) as a code snippet without highlighting code.
- Added multi-cell saving by right clicking and saving as a code snippet.
- Changed preview height to match the height of code snippet box at the maximum.

3.5 v1.0.4

- Fixed the issue with deleting a snippet while searching or filtering
- Fixed the issue with adding a snippet while searching or filtering

3.6 v1.0.3

- Fixed the issue with keyboard shortcut to create a code snippet with highlighted text

3.7 v1.0.2

- Fixed editing or deleting errors during filtering or searching
- Fixed a name inlinement with language icon

3.8 v1.0.1

- Insert, copy, and edit icons are removed and replaced by a single icon with dropdown
- Replaced language from text to icon
- Added a keybinding for saving highlighted code as code snippet

3.9 v1.0.0

- Added minimap-like preview
- Added plus icon to create a code snippet from scratch
- Added code snippet tags for filter functionality
- Fixed drag image and confirmation message
- Added delete dialog

3.10 v0.1.1

- Editing a snippet

3.11 v0.1.0

- Creating, deleting,and using snippet to JupyterLab workspace
- Undo/Redo deletion of snippet
- Searching and previewing Snippets
- Moving snippet within explorer

CHAPTER 4

User Experience

4.1 Add snippet

- Drag code cell(s) to panel.
- Highlight code and save as snippet.
- Click plus button on panel to create a custom snippet.
- Select cell(s) and use context menu to save all code within selection as snippet.
- Select cell(s) and use keyboard shortcut (cmd-shift-A or ctrl-shift-A) to save all code within selection as snippet.

4.2 Find snippet

- Preview snippet on hover.
- Filter snippets by tags in “Filter by Tags” dropdown.
- Search a snippet’s name or language in the search bar.

4.3 Edit snippet

- Click 3 dots (or right click) and select “Edit snippet” in dropdown.

4.4 Use snippet

- Drag and drop snippet using 6 dot drag icon (visible on hover) to notebook.
  - Will put snippet code into a cell upon insertion.

- Click 3 dots (or right click) and select "Insert snippet" to insert snippet where cursor was on notebook.
- Click 3 dots (or right click) and select "Copy to clipboard" to copy snippet and paste it wherever desired.

4.5 Download snippet

- Click 3 dots (or right click) and select "Download snippet" and input the relative path to download. If the path is not given, it downloads it to the current directory.
- Easy way to share snippets with others.

4.6 Delete snippet

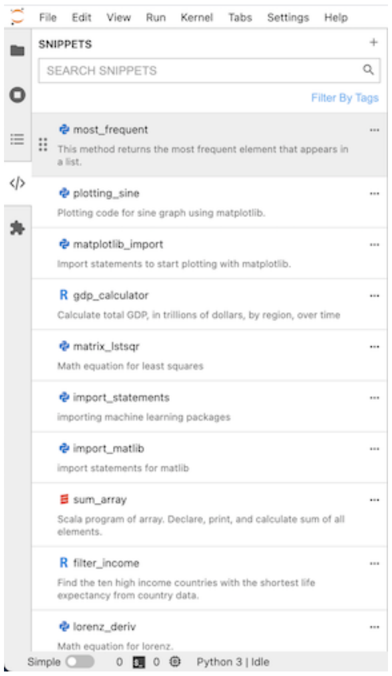
- Click 3 dots (or right click) and select "Delete snippet" and confirm delete in dialog.

CHAPTER 5

Features

5.1 Code Snippet Explorer

On the left side bar of the JupyterLab, you can click the snippet icon (</>) to open/close the code snippet explorer that contains all the saved snippets as below. You can scroll/resize code snippet explorer, order the snippets with drag and drop, and manage and search the snippets.



most_frequent
This method returns the most frequent element that appears in a list.

plotting_sine
Plotting code for sine graph using matplotlib.

matplotlib_import
Import statements to start plotting with matplotlib.

gdp_calculator
Calculate total GDP, in trillions of dollars, by region, over time

matrix_lstsqr
Math equation for least squares

import_statements
importing machine learning packages

import_matlib
import statements for matlib

sum_array
Scala program of array. Declare, print, and calculate sum of all elements.

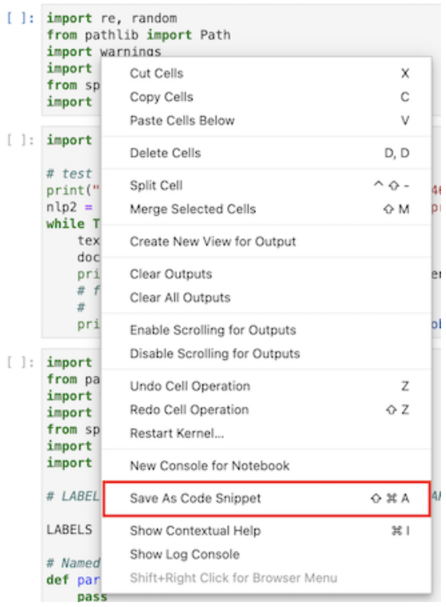
filter_income
Find the ten high income countries with the shortest life expectancy from country data.

lorenz_deriv
Math equation for lorenz.

5.2 Creation of a New Snippet

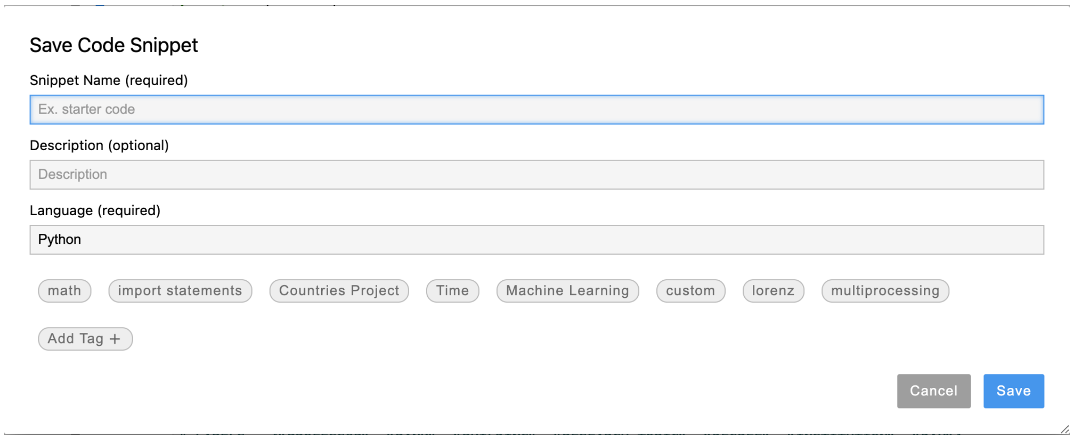
There are several ways to create a new snippet.

1. Highlight lines of code and right click (or use keyboard shortcut (cmd-shift-A or ctrl-shift-A)) to save the code as a snippet. Or, you can select cell(s) and right click (or use keyboard shortcut (cmd-shift-A or ctrl-shift-A)) to save content as a snippet. Then, you will see the "Save As Code Snippet" option in JupyterLab context menu as below:



```
[ ]: import re, random
     from pathlib import Path
     import warnings
     import
     from sp
     import

[ ]: import

     # test
     print("
     nlp2 =
     while T
         tex
         doc
         pri
         # f
         #
         pri

[ ]: import
     from pa
     import
     import
     from sp
     import
     import

     # LABEL

     LABELS

     # Named
     def par
         pass
```

Clicking the option will open the dialog as below where you can input name, description, language, and tags of the snippet.



Save Code Snippet

Snippet Name (required)

Ex. starter code

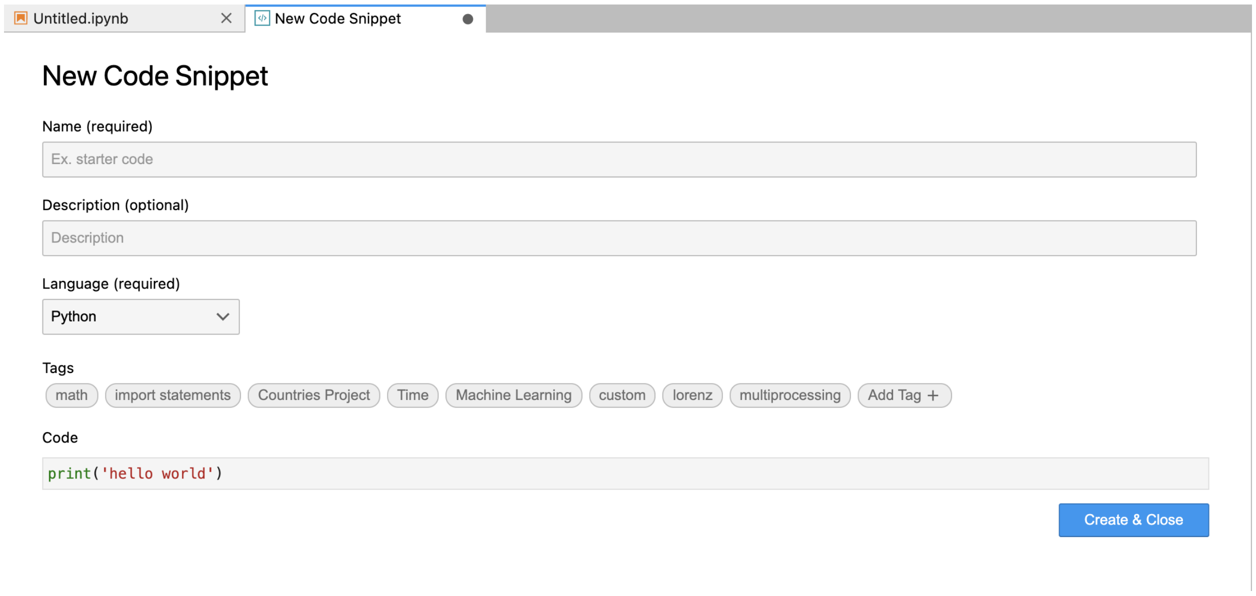
Description (optional)

Description

Language (required)

Python

2. More easily, just drag notebook cell(s) to the code snippet explorer on the left to create a new snippet. It will also open the dialog as above for your inputs.
3. Hit the plus button next to the search bar (refer to *Search of Code Snippets*) to create a new snippet from scratch.



New Code Snippet

Name (required)

Ex. starter code

Description (optional)

Description

Language (required)

Python

Tags

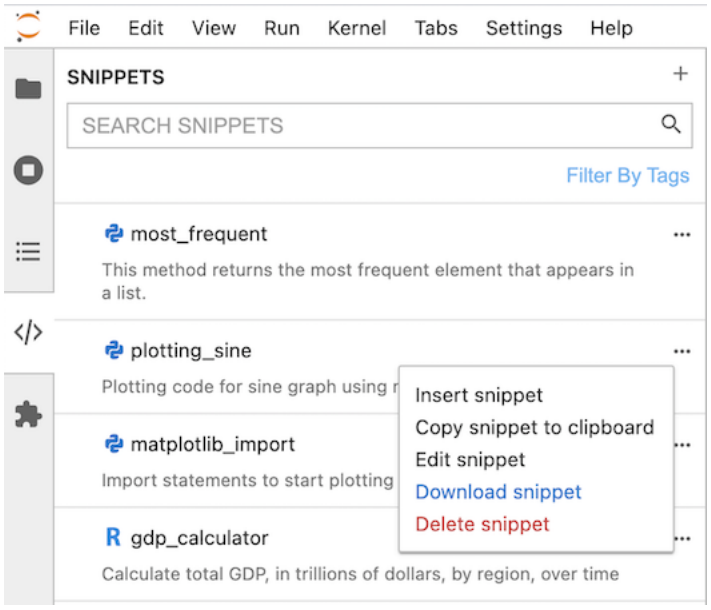
math | import statements | Countries Project | Time | Machine Learning | custom | lorenz | multiprocessing | Add Tag +

Code

```
print('hello world')
```

5.3 Use of Code Snippets

Click the three dots or right click anywhere on the snippet you want to use. It will open code snippet menu as below to insert, copy, edit, download, or delete the snippet.



most_frequent

This method returns the most frequent element that appears in a list.

plotting_sine

Plotting code for sine graph using r

matplotlib_import

Import statements to start plotting

gdp_calculator

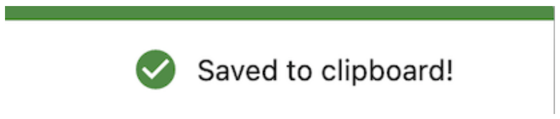
Calculate total GDP, in trillions of dollars, by region, over time

5.3.1 Insert

Click the insert option to insert. It will warn you if the language is different from the kernel langauge. Additionally, on hover with six dots on the left (refer to *Search of Code Snippets*), drag snippet into any JupyterLab workspace to insert the snippet!

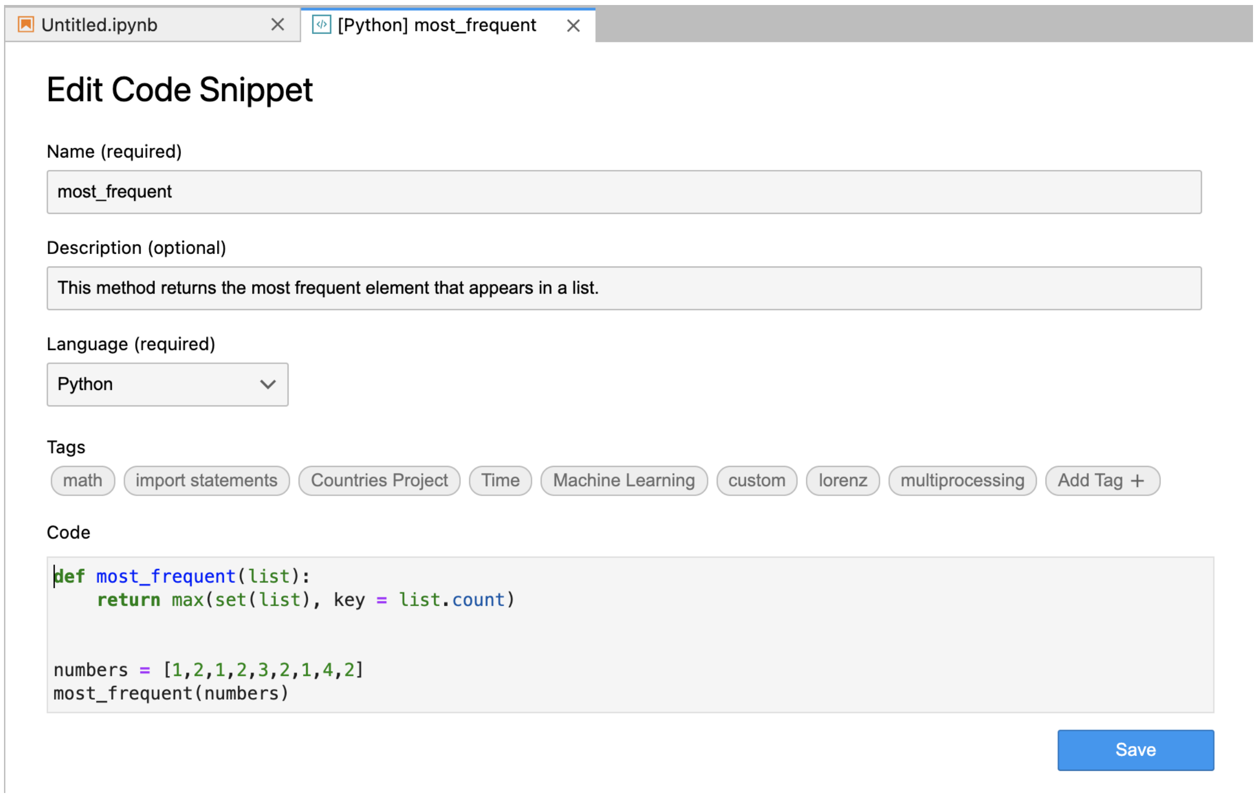
5.3.2 Copy

Click the copy option to copy. It will create a message on the bottom right corner as below.


Saved to clipboard!

5.3.3 Edit

Edit the saved snippets by clicking the edit option. It will open a code snippet editor as below.



Edit Code Snippet

Name (required)

most_frequent

Description (optional)

This method returns the most frequent element that appears in a list.

Language (required)

Python

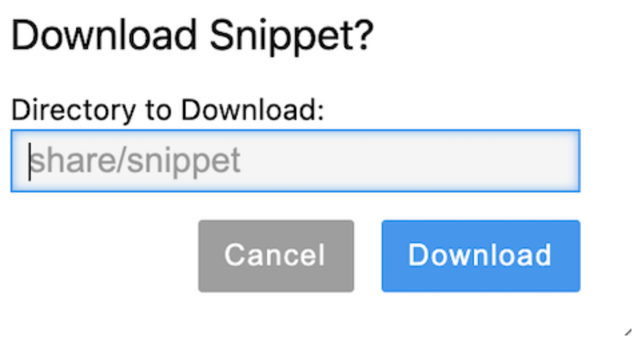
Tags

Code

```
def most_frequent(list):
    return max(set(list), key = list.count)


numbers = [1,2,1,2,3,2,1,4,2]
most_frequent(numbers)
```

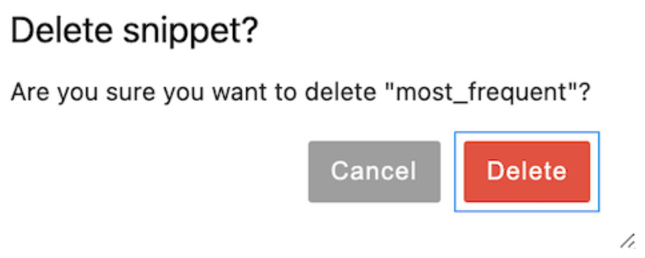
5.3.4 Download

Download snippets and share them with others by clicking the download option. It will open a box as below to input **relative path** to download the snippets.



5.3.5 Delete

Delete snippets by clicking delete option. It will open a warning box as below.

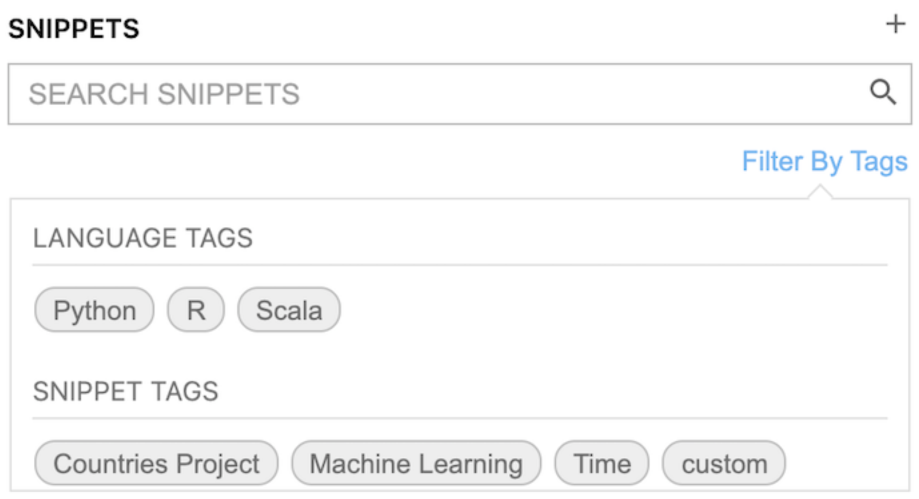


Delete snippet?

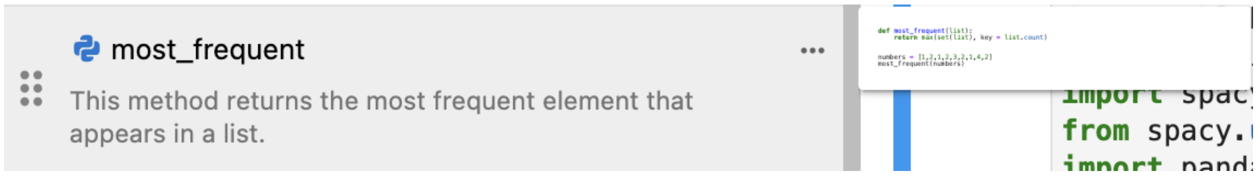
Are you sure you want to delete "most_frequent"?

5.4 Search of Code Snippets

With the filter box as below, **search** snippets with name, language, and code with search bar or **filter** them with language/filter tags!



Furthermore, hover over snippets to see the preview of each snippet.



most_frequent

This method returns the most frequent element that appears in a list.

```
def most_frequent(list):
    return max(set(list), key = list.count)

numbers = [1,2,1,2,3,2,1,4,2]
most_frequent(numbers)
```

```
import spac
from spacy.
import pand
```

5.5 Change Font Size of Snippet Preivew

In the JupyterLab Settings -> Advanced Settings Editor, users can set the size of code in snippet preview to see part of its content clearly.

CHAPTER 6

Transition to 2.1.0

The 2.1.0 update has made snippets globally accessible across notebooks by saving them at the JupyterLab Settings API endpoint.

One side effect of this change is that snippets created in the previous version of this extension will not be supported. The /snippets folder will continue to be available if previously created in a project folder.

In addition, the searching/tag feature has been expanded as well.

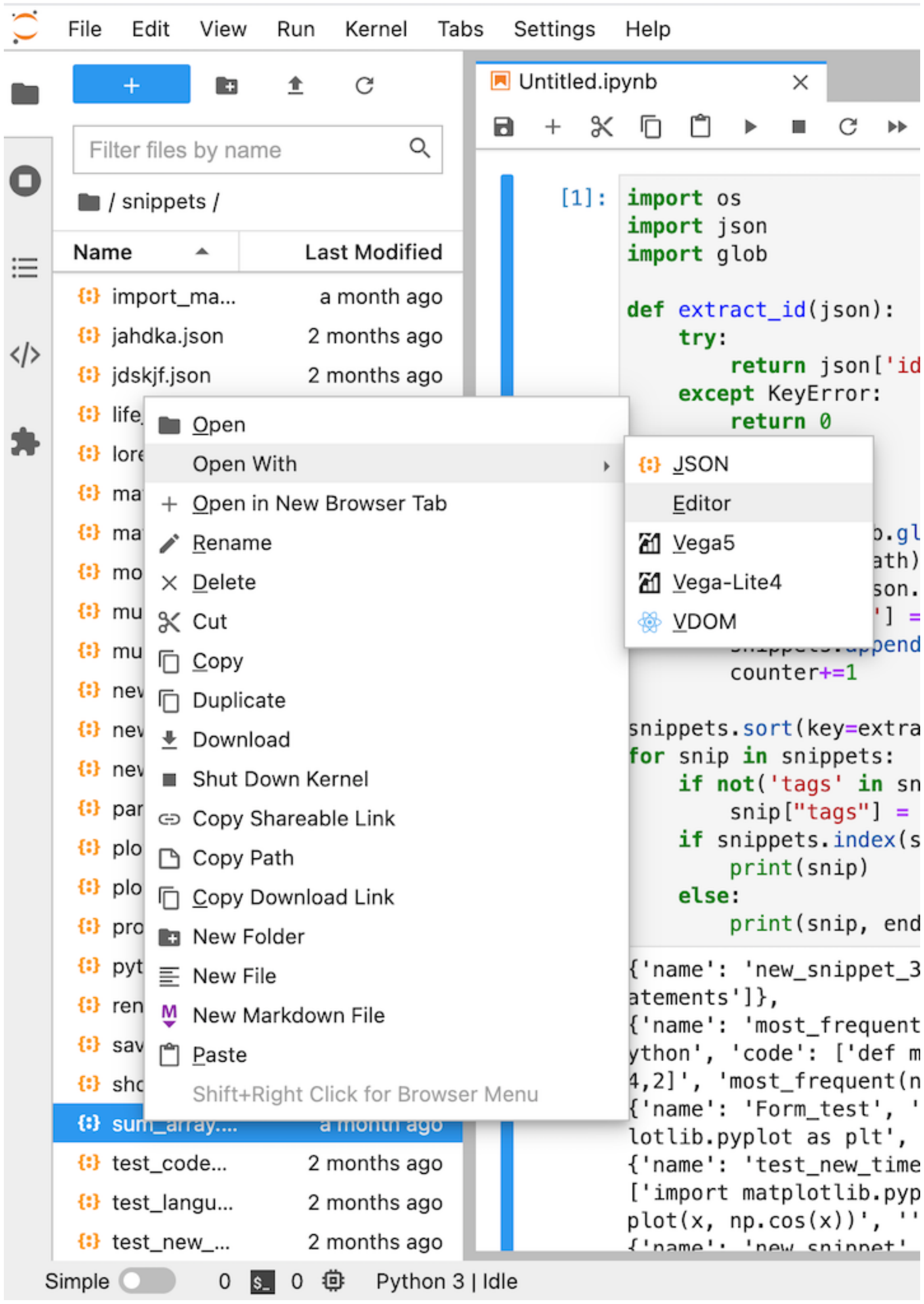
6.1 Transferring Single Snippets

To add old snippets to the new snippet location, simply copy the JSON object and go to: Settings > Advanced Settings Editor > Code Snippet Manager

In the User Preferences panel there will be an array of snippets existing globally (something like "snippets" : [. . .]) where the . . . are the globally accessible snippets.

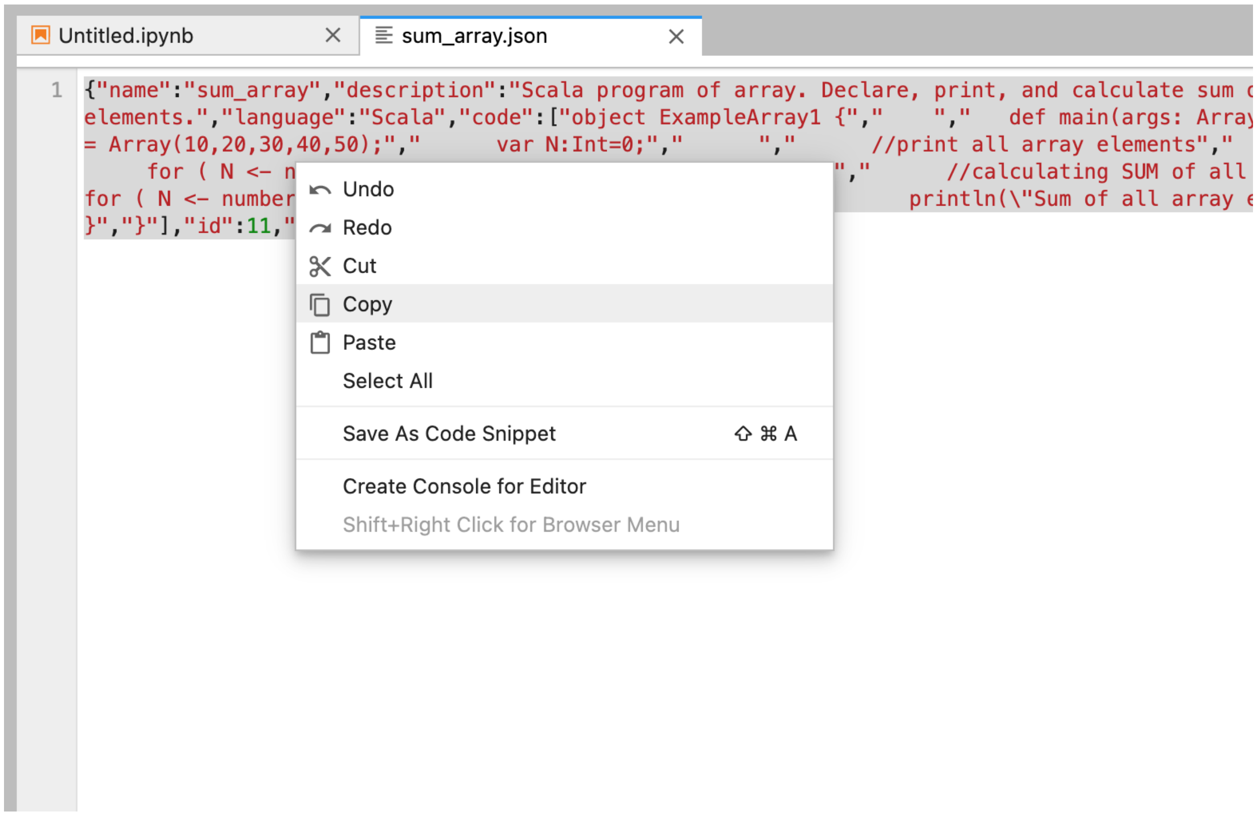
At the end of the existing list, paste the copied JSON object, make sure the id value is sequential to what already is in the list. Save using the save icon in the top right corner. Snippet should appear in the snippets panel!

Open JSON file of snippet to upload:



```
[1]: import os
     import json
     import glob

     def extract_id(json):
         try:
             return json['id
         except KeyError:
             return 0

                                   o.gl
                                   ath)
                                   son.
                                   '] =
               snippets.append
               counter+=1

     snippets.sort(key=extra
     for snip in snippets:
         if not('tags' in sn
             snip["tags"] =
         if snippets.index(s
             print(snip)
         else:
             print(snip, end

     {'name': 'new_snippet_3
     atements']},
     {'name': 'most_frequent
     ython', 'code': ['def m
     4,2]', 'most_frequent(n
     {'name': 'Form_test', '
     lotlib.pyplot as plt',
     {'name': 'test_new_time
     ['import matplotlib.pyp
     plot(x, np.cos(x))', ''
     {'name': 'new_snippet'
```

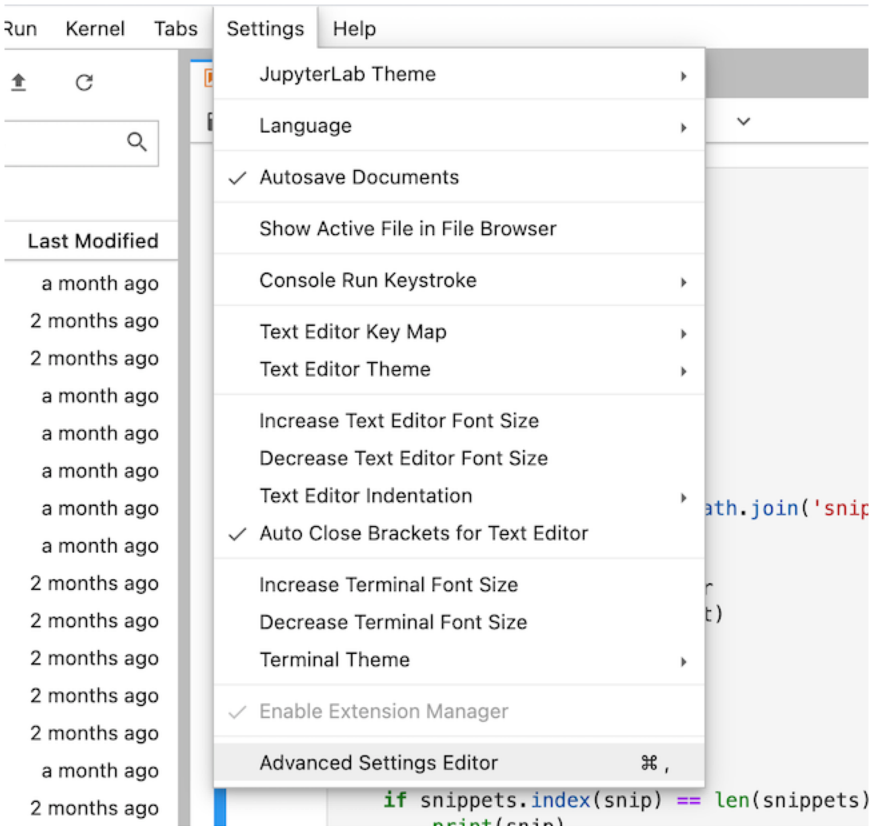
Copy the JSON object:



```
{"name":"sum_array","description":"Scala program of array. Declare, print, and calculate sum o
elements.","language":"Scala","code":["object ExampleArray1 {","    ","    def main(args: Array
= Array(10,20,30,40,50);","      var N:Int=0;","      ","      //print all array elements","
    for ( N <- n                                    ","      //calculating SUM of all
for ( N <- number                                        println(\"Sum of all array e
}","}"],"id":11,"
```

Untitled.ipynb
sum_array.json
Undo
Redo
Cut
Copy
Paste
Select All
Save As Code Snippet ⇧ ⌘ A
Create Console for Editor
Shift+Right Click for Browser Menu

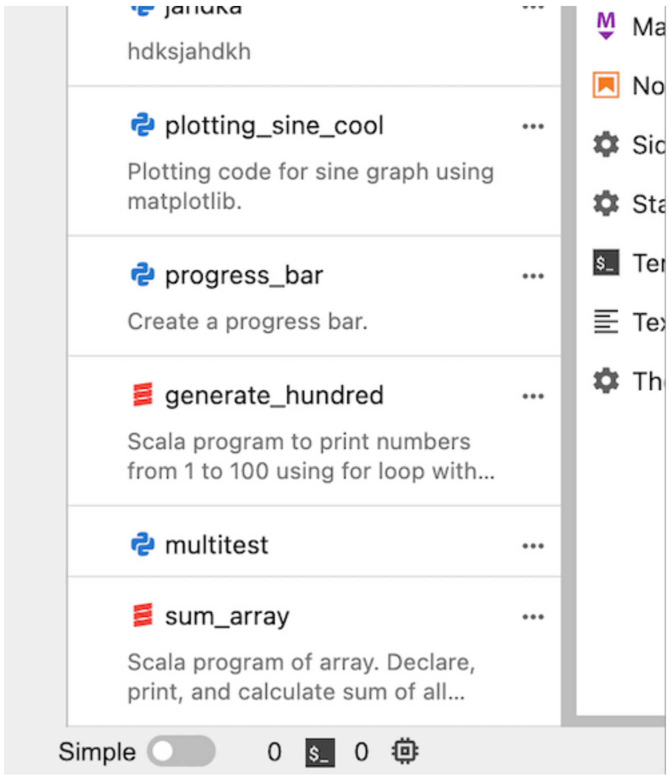
Navigate to settings:



```
ath.join('snip
r
t)
if snippets.index(snip) == len(snippets)
```

Paste JSON object into list of snippets:

After saving, new snippet should appear at the bottom of the snippets list in the snippet panel:


hdksjahdkh
plotting_sine_cool
Plotting code for sine graph using matplotlib.
progress_bar
Create a progress bar.
generate_hundred
Scala program to print numbers from 1 to 100 using for loop with...
multitest
sum_array
Scala program of array. Declare, print, and calculate sum of all...
Simple
0
0

6.2 Transferring Multiple Snippets

To help with converting entire /snippets folder worth of snippets we have developed a python script to help with the transition:

```
import os
import json
import glob

def extract_id(json):
    try:
        return json['id']
    except KeyError:
        return 0

snippets = []
counter = 0
for filepath in glob.glob(os.path.join('snippets', '*.json')):
    with open(filepath) as f:
        content = json.load(f)
        content['id'] = counter
        snippets.append(content)
        counter+=1
```

(continues on next page)

(continued from previous page)

```
snippets.sort(key=extract_id)
print('{"snippets": [\n')
for snip in snippets:
    if not('tags' in snip):
        snip["tags"] = []
    if snippets.index(snip) == len(snippets)-1:
        print(json.dumps(snip, indent=4, sort_keys=True))
    else:
        print(json.dumps(snip, indent=4, sort_keys=True), end=",\n")
print("]\n}\n")
```

This script will concatenate and print out all of the json objects in a /snippets folder in a project. After running the script, copy the output and paste into the User Preferences panel in settings, similar to the single snippet upload.

NOTE: If adding objects to existing list of snippets in user preferences, change *counter* variable to n+1 where n is the ID of the last snippet stored in user preferences.

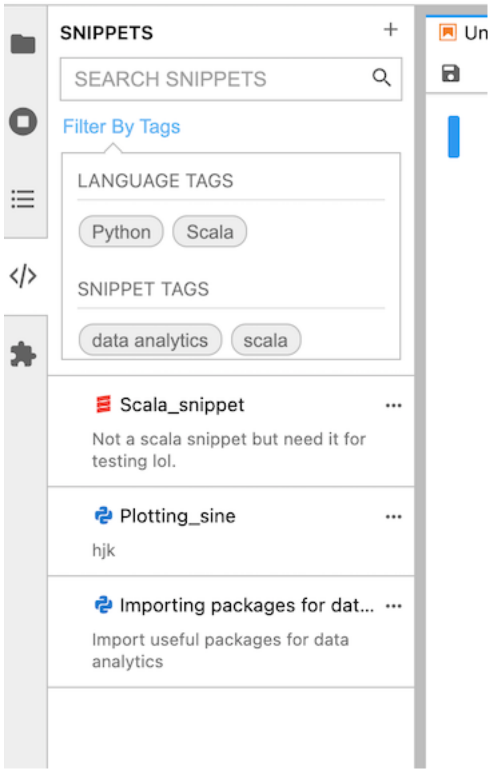
This script will print something like :

```
{"snippets": [

{
   "code": [
      "print(\"hello\")"
   ],
   "description": "",
   "id": 0,
   "language": "Python",
   "name": "new_snippet_3",
   "tags": [
      "import statements"
   ]
},
{
   "code": [
      "def most_frequent(list):",
      "    return max(set(list), key = list.count)",
      "   ",
      "",
      "numbers = [1,2,1,2,3,2,1,4,2]",
      "most_frequent(numbers)   "
   ],
   "description": "This method returns the most frequent element that appears in a␣
↪list.",
   "id": 1,
   "language": "Python",
   "name": "most_frequent",
   "tags": []
}
]
}
```

After generating this dictionary, one can simply delete the current contents of user preferences (Advanced Settings > Code Snippet Manager > User Preferences) and paste this dictionary instead. This will delete the current snippets at the endpoint (which will be default snippets if the extension is freshly updated/installed) and replace them with the old snippets.

NOTE: If adding objects onto an existing list of objects, make sure the ID numbers are all in sequential, ascending order.

6.3 Search and Tag Update



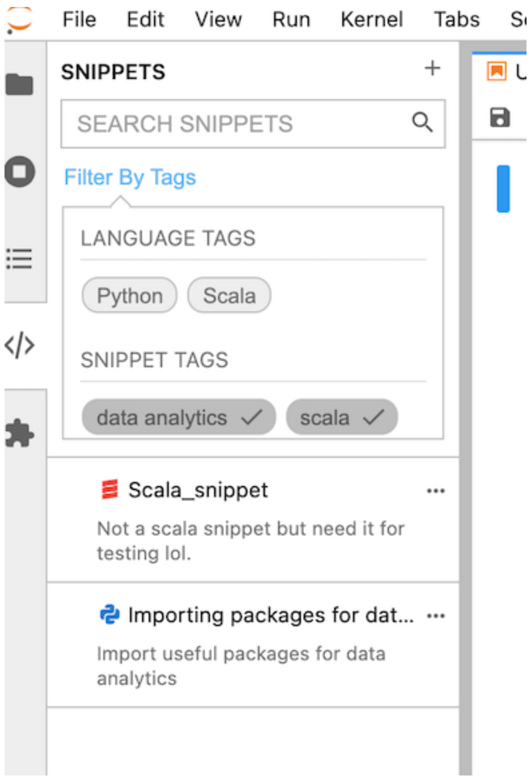
Scala_snippet
Not a scala snippet but need it for testing lol.

Plotting_sine
hjk

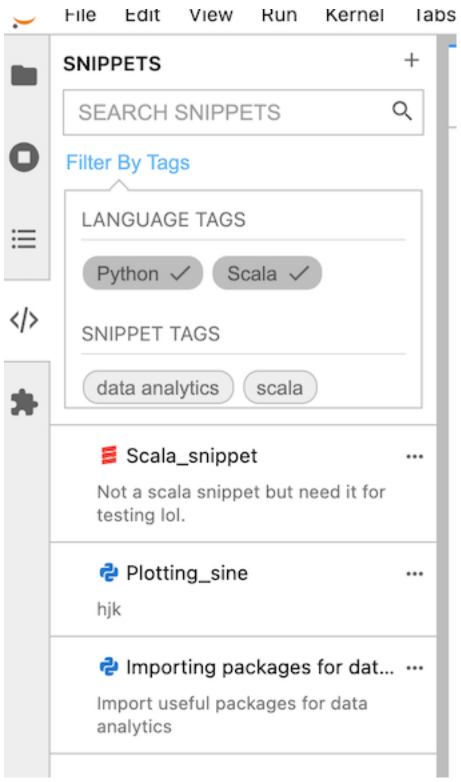
Importing packages for dat...
Import useful packages for data analytics

NOTE 1: Snippet tags function on an OR basis, as in when the "data analytics" tag and the "import statements" tag are selected together, the panel displays any tags that are tagged as import statements OR tagged as "data analytics."

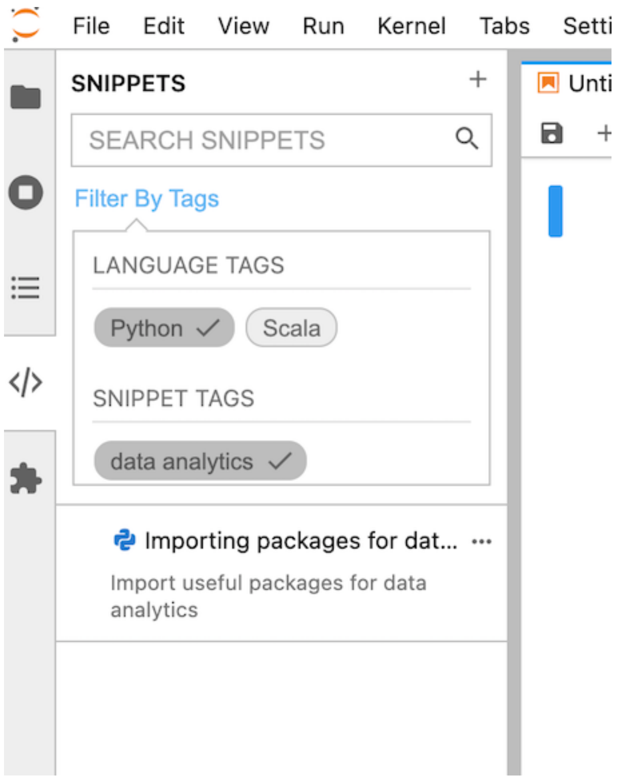
ex) Snippet tags selected together:



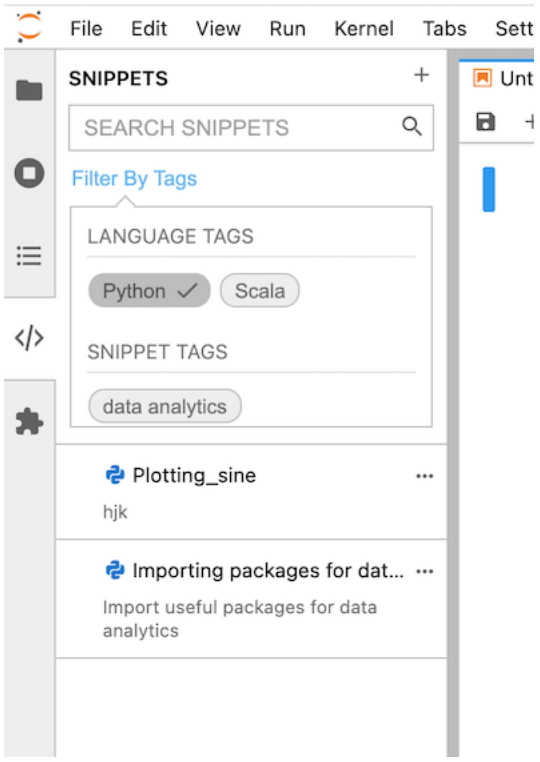
ex) Language tags selected together:



NOTE 2: Language tags and snippets tags have an AND relationship. As in when the “Python” tag and the “data analytics” tags are selected together, only snippets that are both in the language Python AND tagged as data analytics will appear.



NOTE 3: When language tags are selected, only snippet tags in that language will appear for ease of selection.



NOTE 4: If 2 of the same language tags appear in the Language Tags section this indicates that one of the snippets contains a snippet tag that matches the language name. To get rid of the duplicate tag, filter through snippets and untag any snippets that have that Language in their snippet tags.

CHAPTER 7

Codebase Orientation

7.1 Directories

7.1.1 Code Snippets: `snippets/`

This comprises the sample code snippets for the extension. Refer to Code Snippet Metadata to learn more about the content of each code snippet.

7.1.2 Binder setup: `binder/`

This contains an environment specification for `repo2docker` which allows the repository to be tested on mybinder.org. This specification is developer focused. For a more user-focused binder see the JupyterLab demo.

7.1.3 Test: `test/ cypress/`

`test/` contains unit test scripts using jest. And, `cypress/` contains integration test scripts using Cypress.

7.1.4 Design: `design/`

This directory contains gifs or images that show the design perspective of our extension.

7.1.5 Style: `style/`

This directory contains all of the CSS styling used in the extension.

7.1.6 Documentation: `docs/`

This directory contains the Sphinx project for this documentation.

7.2 Files

Description of each file in `src/`

- CodeSnippetContentsService.ts: this contains a wrapper class that uses the functions defined in @jupyter-lab/contentsSerivce.
- CodeSnippetDisplay.tsx: this contains a major React component that defines key code snippets UI and renders code snippets.
- CodeSnippetEditor.tsx : this contains a React component that creates an editor for each code snippet.
- CodeSnippetEditorTags.tsx: this contains a React component that renders code snippet tags in code snippet editor.
- CodeSnippetFilterTools.tsx: this contains a react component that renders a search bar and filter box.
- CodeSnippetInputDialog.ts: this contains a class that defines the content of the custom code snippet form.
- CodeSnippetLanguages.ts: this contains a list of supported languages and their corresponding icons.
- CodeSnippetMenu.ts: this contains a lumino widget that creates dropdown dialog when three dots icon is clicked.
- CodeSnippetMessage.ts: this contains a class that creates a message as a modal window after creating, copying, or downloading snippet.
- CodeSnippetPreview.ts: this contains a lumino widget that creates a preview minimap.
- CodeSnippetService.ts: this contains a wrapper class that handles the backend storage of the code snippets in JupyterLab Settings API.
- CodeSnippetUtilities.ts: this contains a few utility functions that are used across the codebase.
- CodeSnippetWidget.tsx: this contains a Lumino react widget that acts as a container of code snippets.
- index.ts: this contains the activation of our extension.

CHAPTER 8

How to Contribute

8.1 General Guidelines for Contributing

For general documentation about contributing to Jupyter projects, see the Project Jupyter Contributor Documentation and Code of Conduct.

All source code is written in TypeScript. See the Style Guide.

All source code is formatted using prettier. When code is modified and committed, all staged files will be automatically formatted using pre-commit git hooks (with help from the lint-staged and husky libraries). The benefit of using a code formatter like prettier is that it removes the topic of code style from the conversation when reviewing pull requests, thereby speeding up the review process.

You may also use the prettier npm script (e.g. `npm run prettier` or `yarn prettier` or `jlpm prettier`) to format the entire code base. We recommend installing a prettier extension for your code editor and configuring it to format your code with a keyboard shortcut or automatically on save.

8.2 Contributing Installation

```
# Clone the repo to your local environment
# Move to jupyterlab-code-snippets directory

# Install dependencies
jlpm
# Build Typescript source
jlpm build
# Link your development version of the extension with JupyterLab
jupyter labextension install .
# Rebuild Typescript source after making changes
jlpm build
# Rebuild JupyterLab after making any changes
jupyter lab build
```

You can watch the source directory and run JupyterLab in watch mode to watch for changes in the extension's source and automatically rebuild the extension and application.

```
# Watch the source directory in another terminal tab
jlpm watch
# Run jupyterlab in watch mode in one terminal tab
jupyter lab --watch
```

Now every change will be built locally and bundled into JupyterLab. Be sure to refresh your browser page after saving file changes to reload the extension (note: you'll need to wait for webpack to finish, which can take 10s+ at times).

CHAPTER 9

Code Snippet Metadata

This extension uses JupyterLab Contents Service and creates a `snippets/` if it doesn't exist. This is where code snippet json files are stored, following a schema defined below.

```
{
   name: string,
   description: string,
   language: string,
   code: string[],
   id: number,
   tags?: string[]
}
```

This is a sample code snippet json file:

```
{
  "name": "import_statements",
  "description": "Import statements for matlibplot.",
  "language": "python",
  "code": [
        "import matplotlib as mpl",
        "import matplotlib.pyplot as plt"
  ],
  "id": 1,
  "tags":["import statements"]
}
```