

Spyder 4 ... the python IDE for science

Post 2020-05-24 by [Dan Patterson \(retired\)](#) Original Post by [Dan Patterson](#) 12-Dec-2019

Spyder

Contents

Spyder	1
Version	1
Theme choices	2
Preference options	3
Graphics options	4
File/project and script navigation	5
Help documentation and presentation	6
Editing tips	7
Finding stuff	8

Details as I go. Everything is there to assist you from initial project thought to final application.

Right now... just the pics and a few tips.

An attachment of the first image, as well, if you want to explore in more detail.

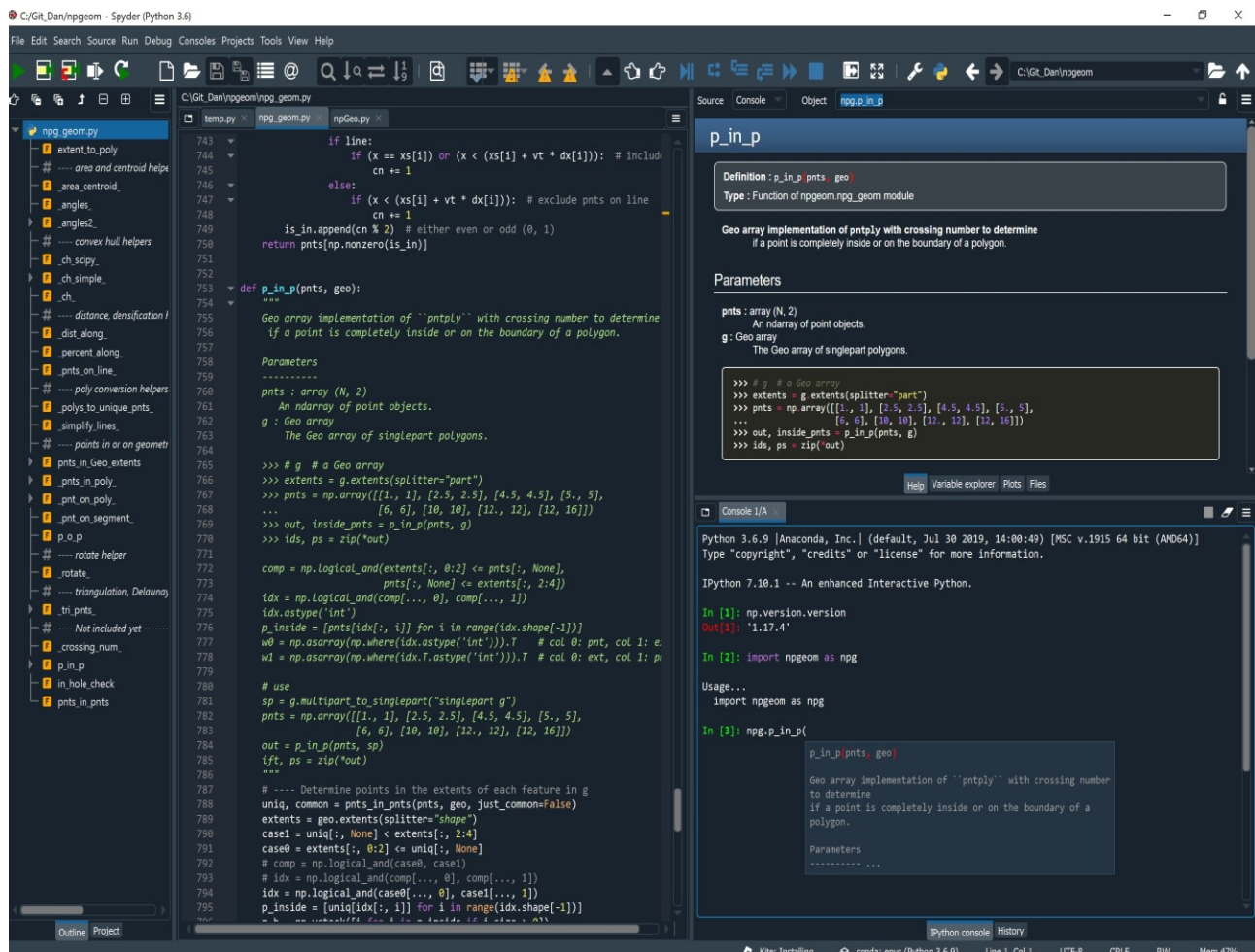
Version

Spyder 4's current version and changelog can be tracked at...

[Spyder changelog on GitHub](#)

Spyder 4 ... the python IDE for science

Theme choices



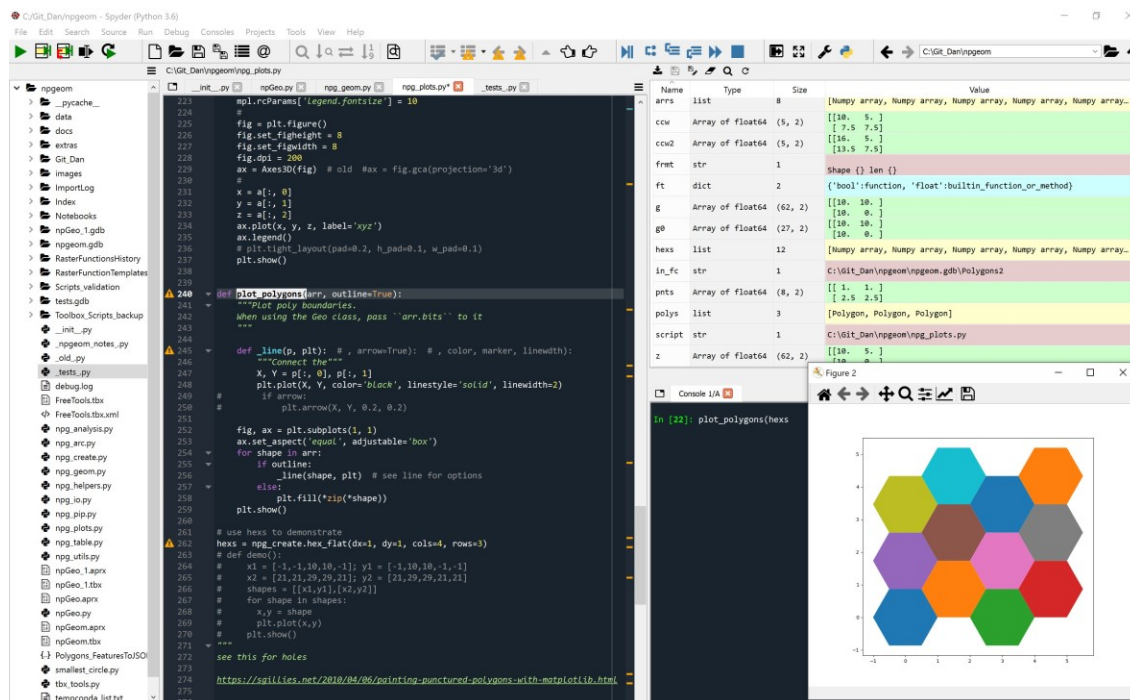
There are a variety of ways to layout and style the IDE. A full dark theme is above.

Or you can split the themes and have different ones for the editor and console.

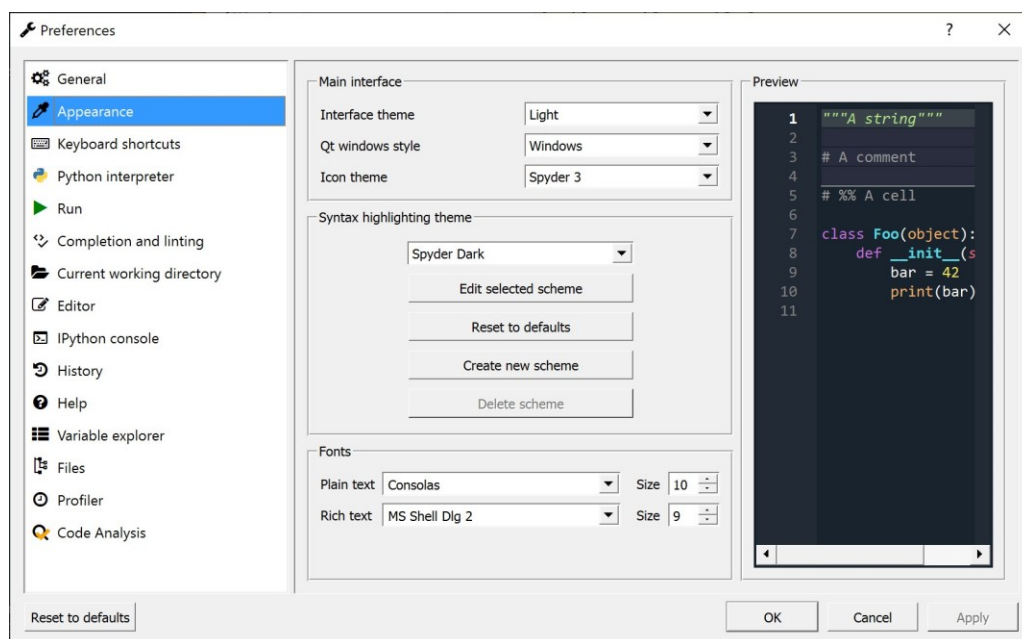
Spyder 4 ... the python IDE for science

The image below shows a lighter theme for the Variable explorer and the file explorer.

Separate (floating) or in-pane graphics available using direct access to Matplotlib.



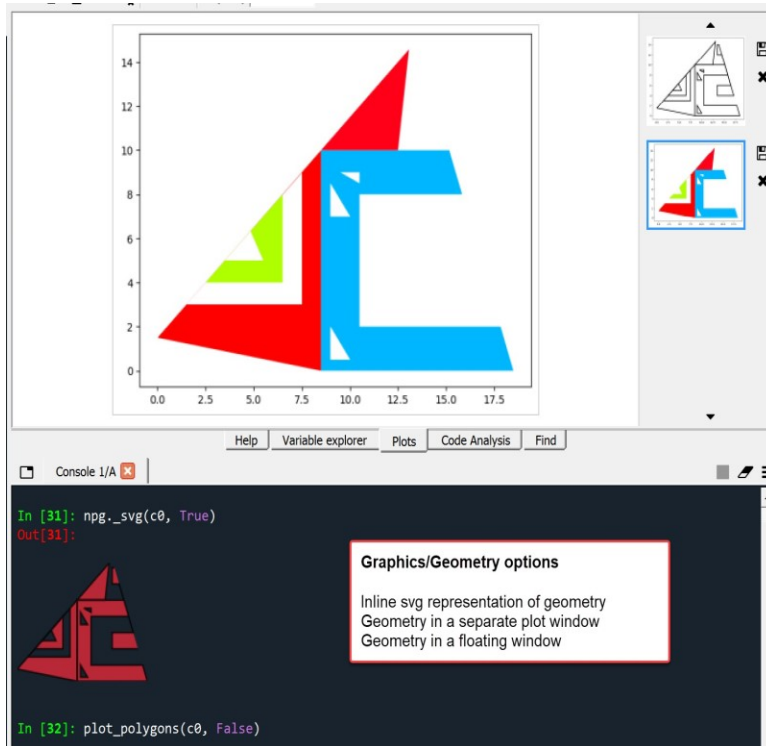
Preference options



Graphics options

There is a new Plots window, or you can set your graphics to automatic to get a separate matplotlib graph window. From there you can interactively alter the graphic to suit your needs.

You will also note, that svg inline graphics are supported. I wrote a function to display numpy arrays representing geometry objects to get a quick preview without the need to create a featureclass.



File/project and script navigation

The screenshot displays the Spyder 4 IDE interface with three main panels. The left panel (1) shows the 'File access for your project' view, listing the project structure. The middle panel (2) shows the 'Project outline' view, displaying a hierarchical tree of the project's contents. The right panel (3) shows the 'script details for quick navigation' view, displaying the code of the selected file, npGeo.py. The code defines a class Geo and includes various methods and attributes.

1 File access for your project

2 Project outline

3 script details for quick navigation

```
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
class Geo(np.ndarray):
    """
    Geo class
    """
    _name_ = "npGeo"
    _module_ = "npgeom"
    _author_ = "Dan Patterson"
    _hlp_ = hlp
    _doc_ += hlp

    def __new__(cls,
                arr=None,
                IFT=None,
                Kind=2,
                Extent=None,
                Info="Geo array",
                SR=None
                ):
        """See script header for construction notes."""
        arr = np.ascontiguousarray(arr)
        IFT = np.ascontiguousarray(IFT)
        if (arr.ndim != 2) or (IFT.ndim != 2):
            m = "ndim != 2 : {} or IFT.dim != 2 : {}".format(arr.ndim, IFT.ndim)
            print(dedent(m))
            return None
        if (IFT.shape[-1] < 6) or (Kind not in (0, 1, 2)):
            print(dedent(hlp))
```

Help documentation and presentation

From this

```
def plot_polygons(arr, outline=True):  
    """Plot Geo array poly boundaries.  
  
    Parameters  
    -----  
    arr : ndarray or Geo array  
        If the arrays is a Geo array, it will convert it to `arr.bits`.  
    outline : boolean  
        True, returns the outline of the polygon. False, fills the polygon  
  
    References  
    -----  
    `random color generation in matplotlib`  
    <https://stackoverflow.com/questions/14720331/how-to-generate-random-colors-in-matplotlib>`_  
  
    See module docs for general references.  
    """
```

Help documentation supporting a variety of types... numpydocs shown

To this...

plot_polygons

Definition : plot_polygons(arr, outline=True)
Type : Function of __main__ module

Plot Geo array poly boundaries.

Parameters

arr : ndarray or Geo array
If the arrays is a Geo array, it will convert it to *arr.bits*.
outline : boolean
True, returns the outline of the polygon. False, fills the polygon

References

[random color generation in matplotlib.](#)

See module docs for general references.

Help is everywhere.

The example to the right shows what a function docstring looks like in the console and in the help tab.

You can choose between coding styles within the preferences.

The **numpydoc** style used by packages like *scipy*, *matplotlib*, *pandas* to name a few, is shown below for comparison.

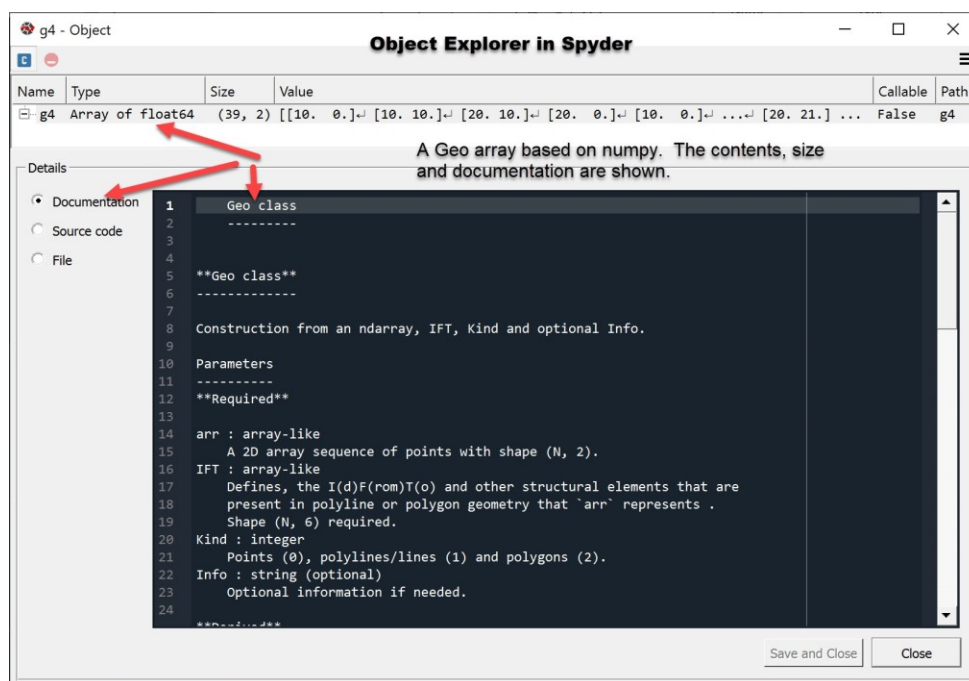
Editing tips

```
In [60]: _test_()
Traceback (most recent call last):
  File "<ipython-input-60-898d2c7336ed>", line 1, in <module>
    _test_()
  File "C:\Git_Dan\npgeom\_tests_.py", line 822, in _test_
    d = npg.fc_data(in_fc)
AttributeError: module 'npgeom' has no attribute 'fc_data'
```

Click on the line number to go to line in the code to fix the error

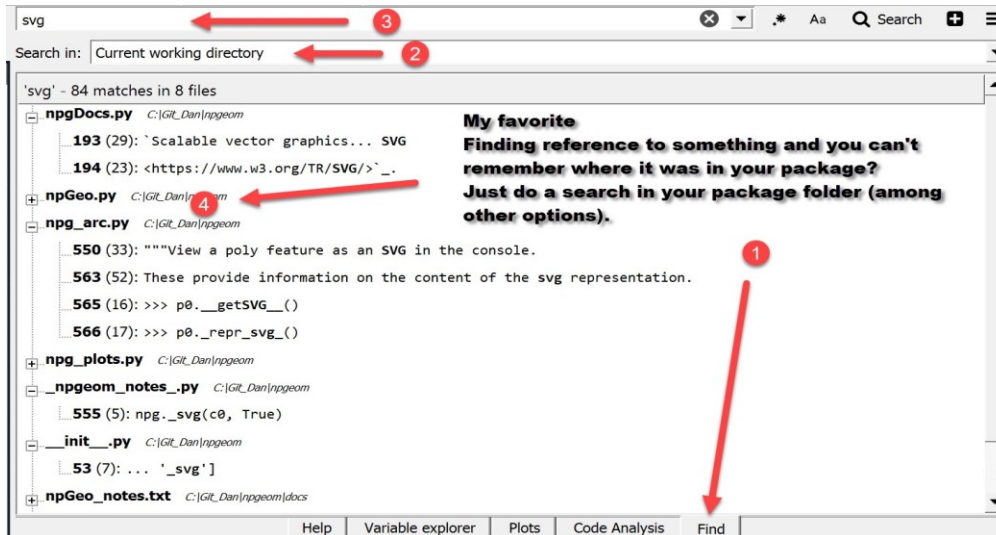
I hate scrolling, so when you get an error, click on the line number.

If it is in an imported script, you can even click on the script name to go there.



The **object explorer** can be used to retrieve information for objects. This is useful for documentation purposes.

Finding stuff



When your package gets large and you are trying to locate something... Find is your friend.

A quick click and you are there to make edits, copy or just read.

Kite can be installed as well

[Kite - AI Autocomplete and Docs for Python](#)

- [spyder_4.png](#) 561.2 KB



Dan Patterson (both 😊)