Practical Computing

Srikar CV

2024-01-03

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

2 GitHub

- TODO: Add Intro

Hello! If you have worked on multiple notebooks in JupyterLab, and you want to save them in one secure, easy-to-access location, putting each notebook into a different folder can be quite cumbersome. It's much easier to store them all together in a single easy-to-access location, known as a *repository*. If you'd like to make your file storage easier and more efficient than before, this tutorial is *definitely* for you! Just open your computer, read the steps and get started!

2.1 Step 1: Create a new repository

Open the Github Desktop App and select 'Create new repository' as shown in the picture below.

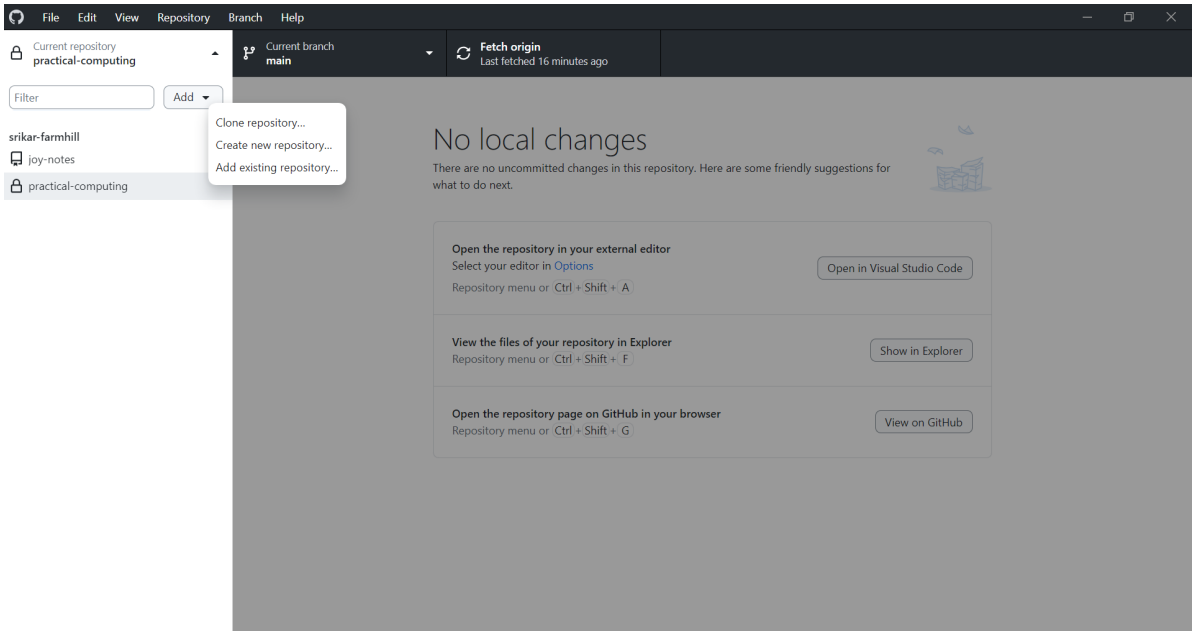


Figure 2.1: image.png

2.2 Step 2: Give your Repository a Name and a Description

Once you've completed the previous step, look at the image below.

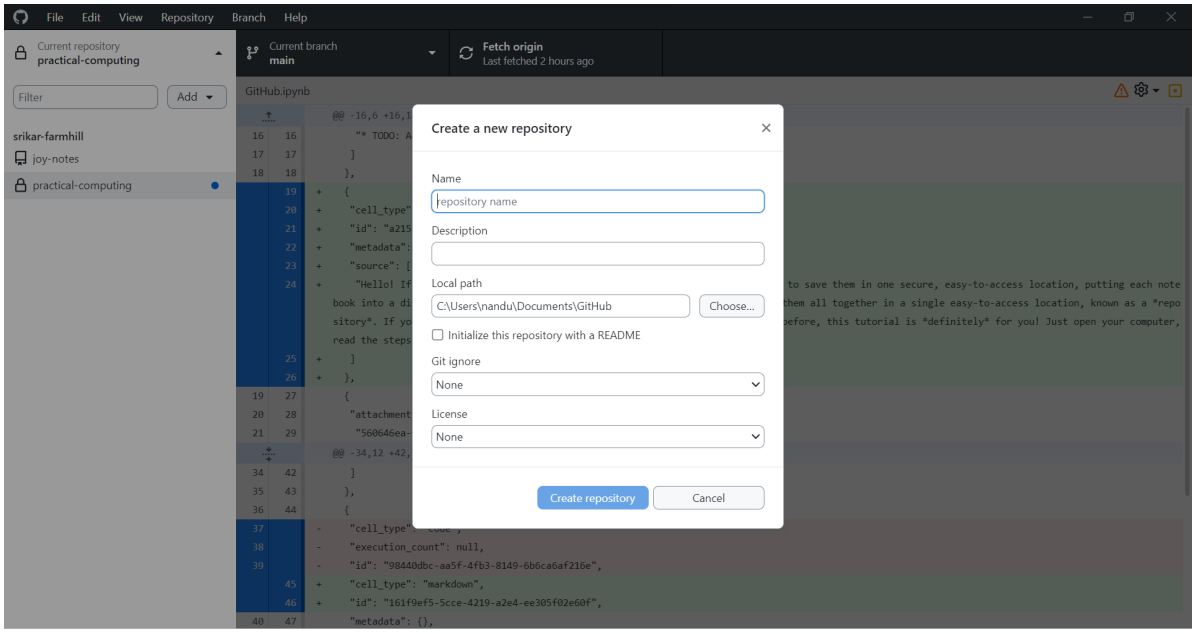


Figure 2.2: image.png

One of the most important steps, if not *the* most important step, in creating any repository is giving it a name and a description. A name makes your repository unique, making it stand out from the rest. A description gives your repository a clear purpose as another means of identification.

You can't create a repository unless you give it a name. You can leave out the description, but this is not a wise course of action, since the resulting repository will be difficult for you to find.

3 Step 3: Initialise your Repository

After you've named your repository and given it an accurate description, look back at the image in Step 2. You should notice a check box labeled 'Initialise this repository with a README'. Click the check box so that it shows a tick mark.

This instruction is given to you to follow because ticking the box enables you to easily edit files kept in the repository without necessarily having to open the GitHub Desktop app. A README initialisation helps you edit such files from your browser as well, using the github.com website.

4 Step 4: Processing and Access Control

Take one last look at the image in Step 2. You should find, after executing Steps 1 to 3, that only two options are left untouched. These are 'Git ignore' and 'Licence'.

'Git ignore' denotes the code language that the repository accepts and allows you to code in. With this option, Python is the recommended code of chocie, although you can select other types of code instead if you are familiar with them.

'Licence' is a safety option. It controls who can access your repository and how. In order to keep your repository safe, you must select a Licence option. However, the option you select must be viable both for you and for others, in case you want your friends to work along with you on your repository.

Alright, now you've folowed the four simple steps above, you can now click the 'Create Repository' button shown at the very bottom of the image in Step 2. Once you've done that, your repository is all set up and ready to go!

5 Lassajous Curves

Lassajous Curves are interesting mathematical curves that are generated using:

$x = \sin(at + \delta)$

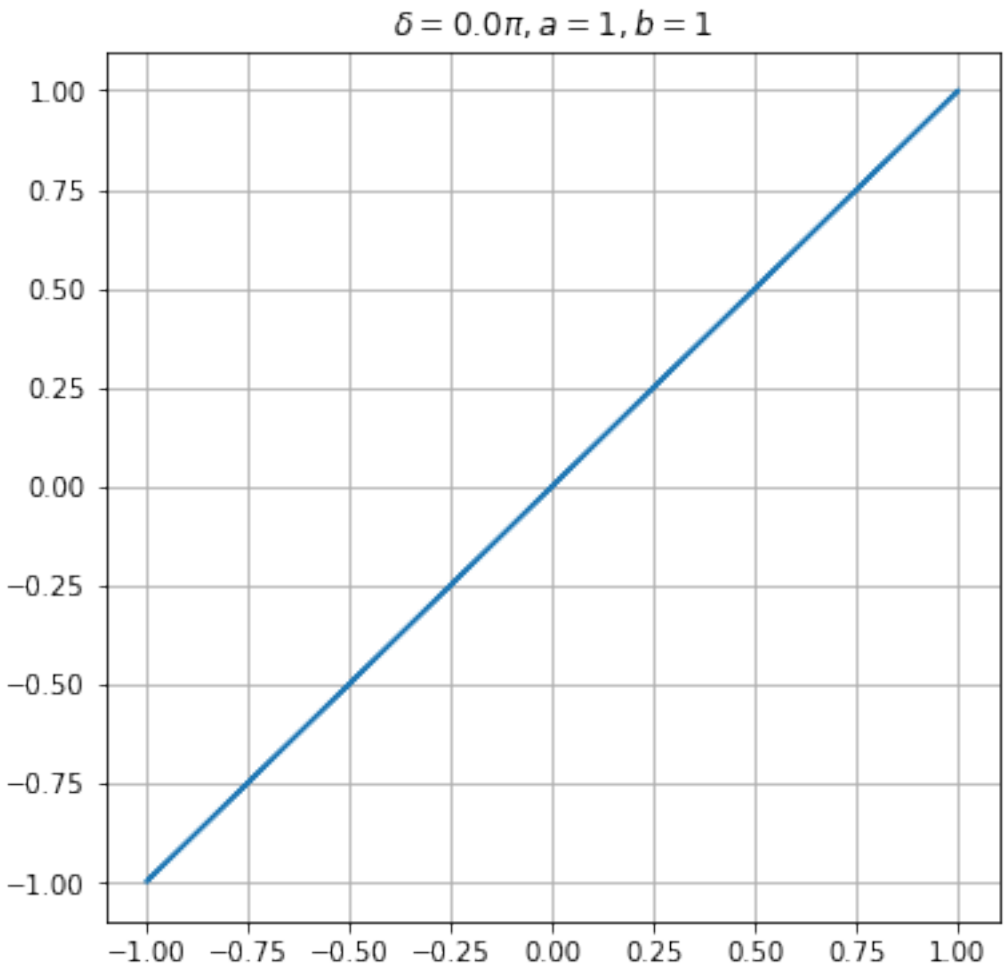
$y = \sin(bt)$

As an example, consider the below function when creating a lassajous curve:

```
import numpy as np
import matplotlib.pyplot as plt
```

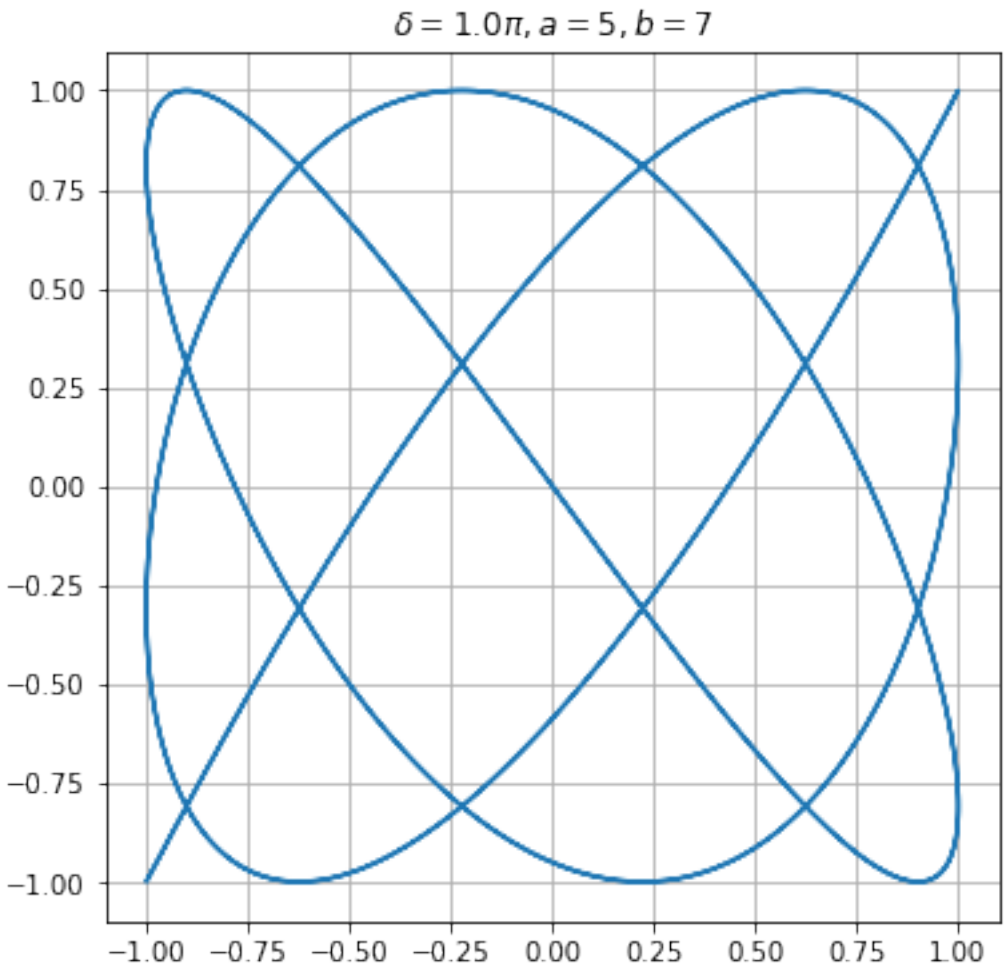
```
def lassajous(a, b, delta):
    t = np.linspace(0, 2*np.pi, 1000)
    x = np.sin(a*t+delta)
    y = np.sin(b*t)

    plt.figure(figsize=(6, 6))
    plt.plot(x, y)
    plt.grid()

    # show delta as a fraction of pi
    delta_fraction = delta/np.pi
    plt.title(rf"$\delta={delta_fraction}\pi, a={a}, b={b}$")
```

```
lassajous(1, 1, 0)
```



Let's try even higher values of delta, a and b and see what happens.

```
lassajous(5, 7, np.pi)
```



Cool, no?

For further exploration, you may want to spend some time playing with the below interactive widget. it features sliders with which you can manipulate a, b and delta as you wish. This in turn helps you find out different kinds of lassajous curve, which you may never have imagined! Have fun!

```
from ipywidgets import interact, IntSlider, FloatSlider

def lassajous_curve(a, b, delta):
    lassajous(a, b, delta*np.pi)

interact(lassajous_curve,
         a=IntSlider(1, min=1, max=6),
```

```
        b=IntSlider(1, min=1, max=6),
        delta=FloatSlider(0, min=0, max=2, step=0.25, readout_format="0.2f"))
```

```
interactive(children=(IntSlider(value=1, description='a', max=6, min=1), IntSlider(value=1, d
```

```
<function __main__.lassajous_curve(a, b, delta)>
```

The results you see may vary widely depending on how you change delta, a and b. Sometimes, changing delta may have no visible effect, but the effect of changing delta is usually quite noticeable. By the way, did you notice thta, if a=b, then you get a straight, slanting line? You may also get a shape of some sort which, if you look carefully, follows the trend of such a line.

6 Summary

In summary, this book has no content whatsoever.

References

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.