



# Data Mining

MDSAA

Practical Class #1

Fall Semester 2023-2024

João Fonseca - [jpfonseca@novaims.unl.pt](mailto:jpfonseca@novaims.unl.pt)

Farina Pontejos – [fponcejos@novaims.unl.pt](mailto:fponcejos@novaims.unl.pt)

Please download the lab materials in the meantime

Check Moodle for instructions

# About Us

João Fonseca

## **Academic background:**

- Graduation in Economics (2016 – NOVA SBE)
- MSc in Management (2019 – NOVA SBE)
- MSc in Information Management (2019 – NOVA IMS)
- PhD in Information Management (2023 – NOVA IMS)

## **Research work:**

- Research in Remote Sensing, Natural Language Processing and Machine Learning methods (synthetic data generation and active learning)
- Research in Responsible AI and Explainable AI

# About Us

## Farina Pontejos

### **Academic background:**

- Graduation in Mining Engineering (2011)
- MSc in Data Science & Advanced Analytics (2023 – NOVA IMS)
- PhD in Information Management (since 2023 [Ongoing] – NOVA IMS)

### **Professional Experience:**

- Mining Engineer
- Web Developer
- IT Consulting (Systems Analysis, ERP implementation)

# Resources

- Bibliography
- Data Mining Github repo:
  - <https://github.com/fpontejos/Data-Mining-23-24>
- Class slides and Jupyter Notebooks
- Google, Stack Overflow, documentations, Github and YouTube

# Resources

## Recommended DataCamp courses

Use the **invite link** (on Moodle) to get 6 months free access to all DataCamp features.

Make sure to **use your university email address** to sign up.

### Introduction to Python

<https://app.datacamp.com/learn/courses/intro-to-python-for-data-science>

### Intermediate Python

<https://app.datacamp.com/learn/courses/intermediate-python>

# Our working environment

- We will be using Anaconda: Currently one of the most popular Python distributions.
- Sets up a data science oriented working environment in Python
- It installs a set of libraries (for now, think of libraries as programming tools – like a toolbox in a woodshop)
- But it can be used for many different purposes (all it takes is installing the necessary libraries)

# Anaconda

[www.anaconda.org](http://www.anaconda.org)

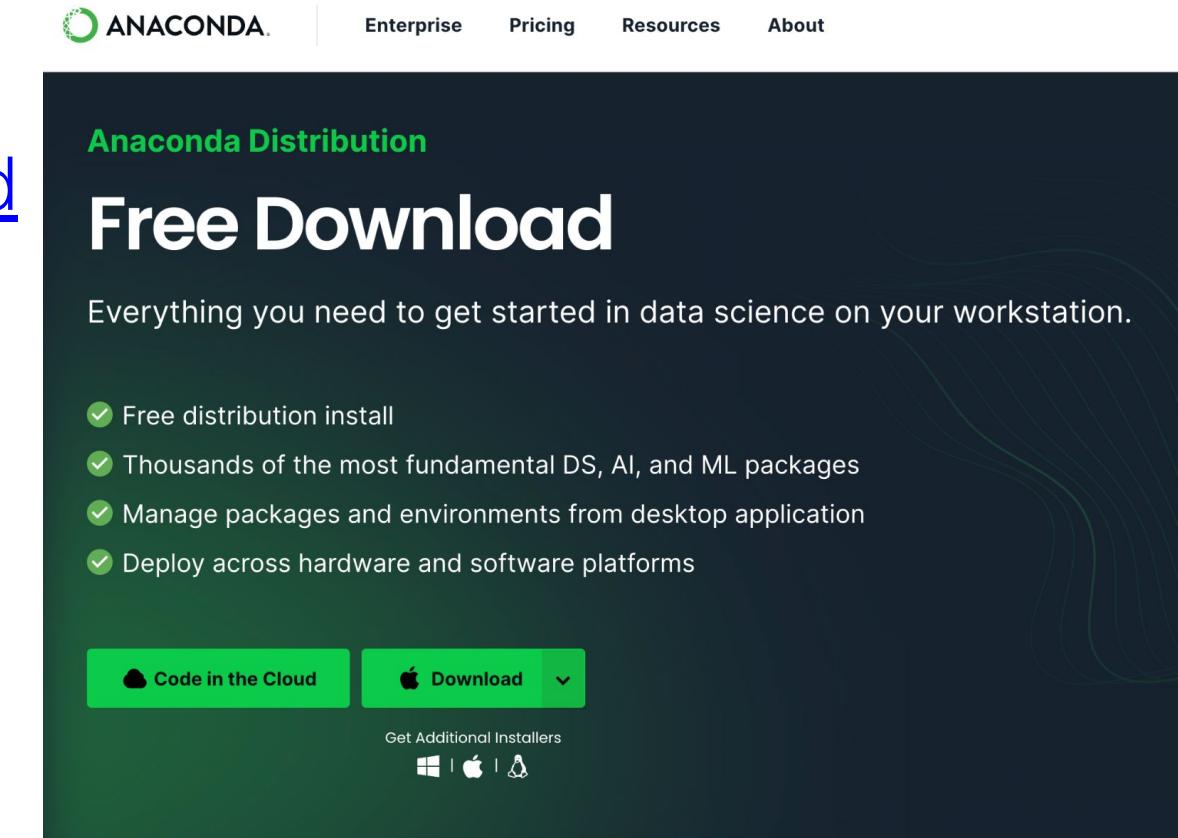
- one of the most popular Python distributions for Data Science
- manages your packages and environments.
- reduce future issues dealing with the various libraries you will be using.
- comes with most of the main libraries for data manipulation
  - Pandas
  - Numpy
  - Matplotlib
  - Scipy
  - ...
- easy to use and install



# In the meantime: Download + Install Anaconda Navigator

Download:

<https://www.anaconda.com/download>



The screenshot shows the Anaconda Distribution website. At the top, there is a navigation bar with the Anaconda logo, followed by links for Enterprise, Pricing, Resources, and About. Below the navigation bar, the text "Anaconda Distribution" is displayed in green, and "Free Download" is prominently shown in large white letters. A subtext below it reads "Everything you need to get started in data science on your workstation." To the right, there is a list of four features with checkmarks: "Free distribution install", "Thousands of the most fundamental DS, AI, and ML packages", "Manage packages and environments from desktop application", and "Deploy across hardware and software platforms". At the bottom, there are two buttons: "Code in the Cloud" and "Download" (with a dropdown arrow). Below these buttons, there is a link "Get Additional Installers" and icons for Windows, Mac, and Linux.

# Install Anaconda Navigator

**Please read the documentation applicable to your system:**

<https://docs.anaconda.com/free/anaconda/install/>



When [installing Anaconda](#), you have the option to “Add Anaconda to my PATH environment variable.” *This is not recommended because it appends Anaconda to PATH.* When the installer appends to PATH, it does not call the activation scripts.

# Virtual Environments

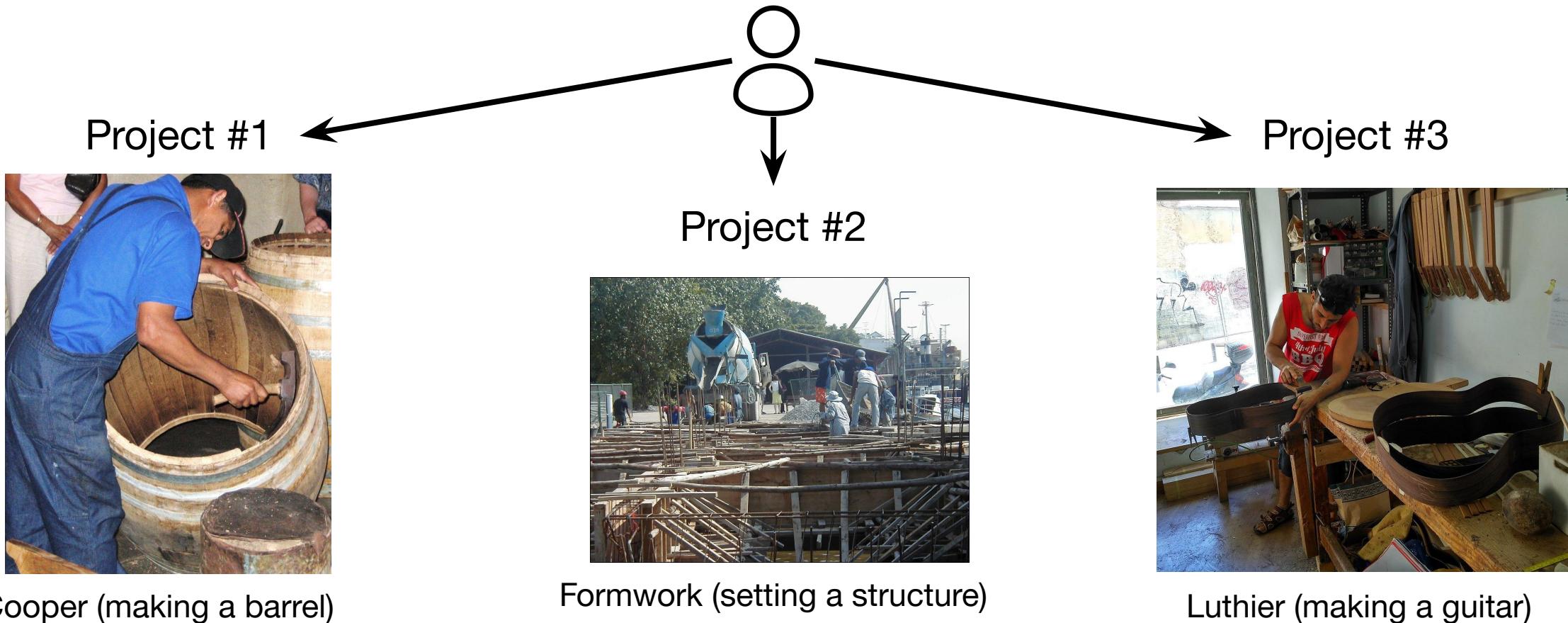
<https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/environments.html>

<https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>

- Isolated spaces that contain per-project dependencies (specific collection of installed conda packages)
- Using conda to manage environments:
  - Create, export, list, remove, and update environments
  - Switching or moving between environments (conda activate)
  - You can also share an environment file
- You can also use pip to manage environment

# In other words

you may need to use different “versions” of the same types of tools, or entirely different “environments”



# In other words

Suppose you will build a Caravel (a Portuguese ship from the 15th century).

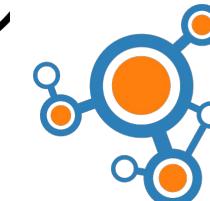
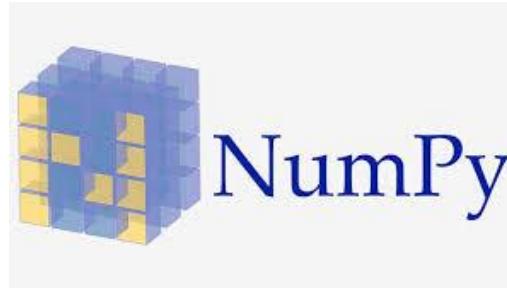
However, you are also a Cooper!  
To build the Caravel the way they did back then, you will need a specific set of tools, much more rudimentary than the ones you will have at your own workshop.



# In other words

- You will need to get them first (i.e., “download” them)
- However, you should not mix these tools with the ones you already have!
  - They are not appropriate to build barrels, and the ones you already have are not appropriate to build Caravels
  - They will create clutter in your workshop (unnecessarily keeping unused packages)
  - You will have duplicate tools with different (version conflicts)
  - Other carpenters may want to build their own replica of your project (reproducibility)
  - If you are working with other carpenters, they will need to use the same types of tools you are using (collaboration)
- These tools (i.e., libraries) and their versions should be specified in the project’s requirements (including the Python version)!

# Python Packages



# Git and GitHub

<https://guides.github.com/activities/hello-world/>

<https://docs.github.com/en/github/getting-started-with-github>

- What is GitHub?
  - . Code hosting platform for version control and collaboration
- What is Git?
  - . At the heart of GitHub is an open source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer.
- Why Git and GitHub?
  - . Optional: You can use Git and GitHub for collaborating and version control in your projects.
  - . Also we have a GitHub repository with all the practical class contents:
  - . <https://github.com/fpontejos/Data-Mining-23-24>

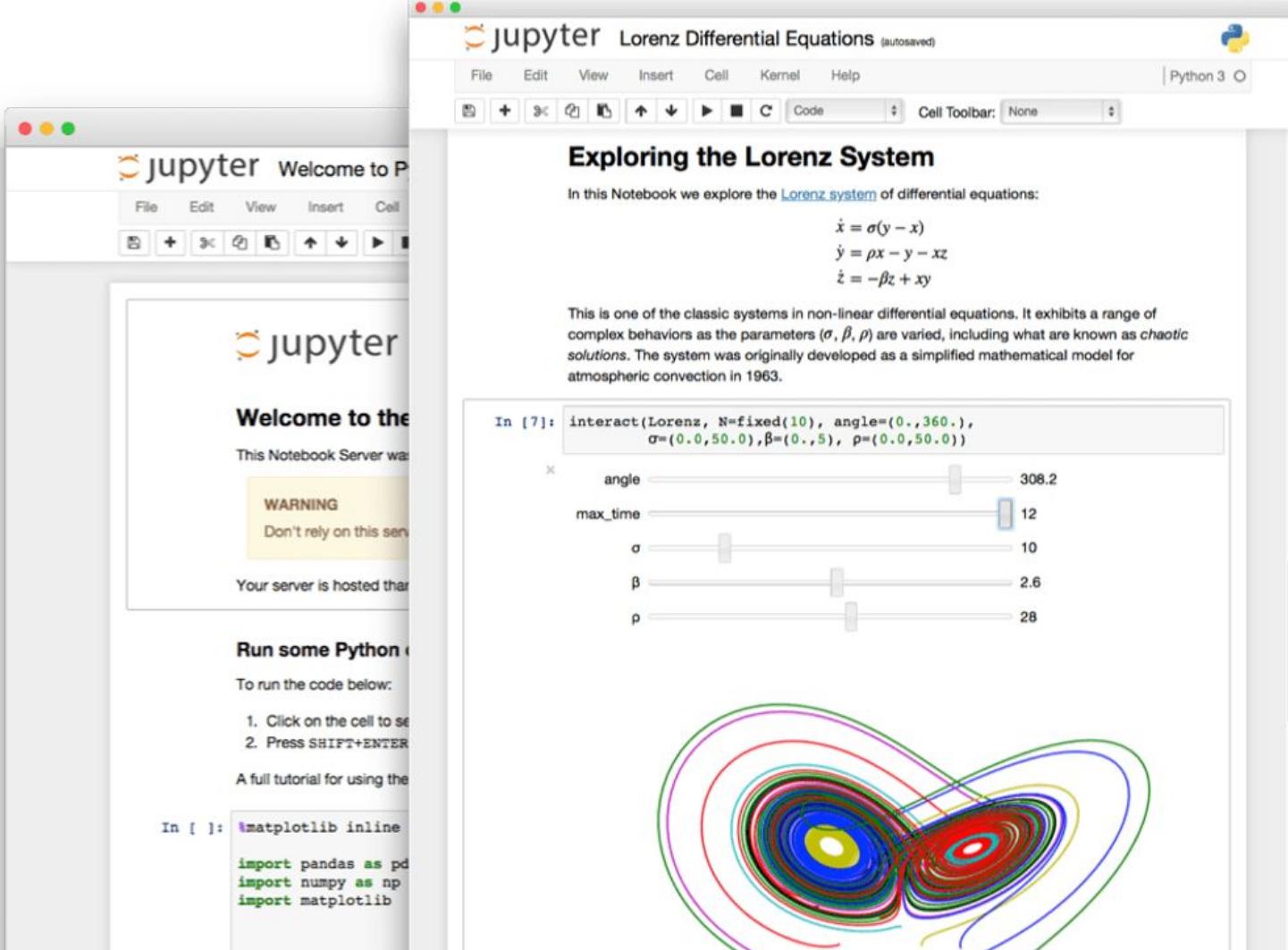
# Main ways to access Python

- Python Shell and IPython
  - An interactive environment for writing and running code
- Jupyter Notebooks
  - A notebook that weaves code, data, prose, equations, analysis, and visualization
  - A tool for prototyping new code and analysis
  - A method for creating a reproducible workflow for scientific research
- IDE (Integrated Development Environment):
  - A software that helps you build code

# Jupyter notebooks

<https://jupyter.org/>

We will be using  
Jupyter notebooks for  
our practical sessions.



The screenshot shows two Jupyter Notebook windows side-by-side. The left window is the 'Welcome to the Jupyter Notebook' page, which includes instructions on how to run code and a sample cell containing Python code for importing matplotlib and pandas.

The right window displays the 'Exploring the Lorenz System' notebook. It features a title, a brief description of the Lorenz system, and three differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Below the equations is a text block explaining the system's behavior and history. A code cell labeled 'In [7]' contains the command `interact(Lorenz, N=fixed(10), angle=(0.,360.), sigma=(0.0,50.0),beta=(0.,5.), rho=(0.0,50.0))`. To the right of the cell are five sliders for parameters: angle (308.2), max\_time (12), sigma (10), beta (2.6), and rho (28). At the bottom is a 3D plot of the Lorenz attractor, a complex, fractal-like shape composed of three interlocking spirals.

# Text Editors

- Another method to write python scripts is using text editors
- Some popular text editors:
  - Vim (Linux terminal text editor)
  - Atom (popular open source editor)
  - Sublime Text (popular proprietary text editor)
  - Notepad ++ (Windows only)
- Usually highly customizable

The screenshot shows the Atom Text Editor interface. On the left, there's a sidebar titled "Project" displaying a file tree for a "FlaskApp" project. The tree includes "FlaskApp", ".DS\_Store", "\_\_pycache\_\_", "static", "support", "templates" (containing ".DS\_Store", "config\_page.html", "dashboard.html", "header.html", "homepage.html"), and "update\_manager.py". The main area has tabs for "Project", "\_\_init\_\_.py", "header.html", and "homepage.html". The "homepage.html" tab is active, showing an empty HTML template. The "header.html" tab is also visible. The "Project" tab shows the file tree. The bottom status bar shows the file name as "\_\_init\_\_.py", line count as 0, character count as 0, and time as 6:13. It also displays the file encoding as UTF-8, the current branch as Python master, and other repository statistics like 4893 files and 6 updates.

```
from flask import Flask, render_template, request, url_for, redirect, flash
from werkzeug.exceptions import BadRequest
import update_manager
import os
# import time

app = Flask(__name__)

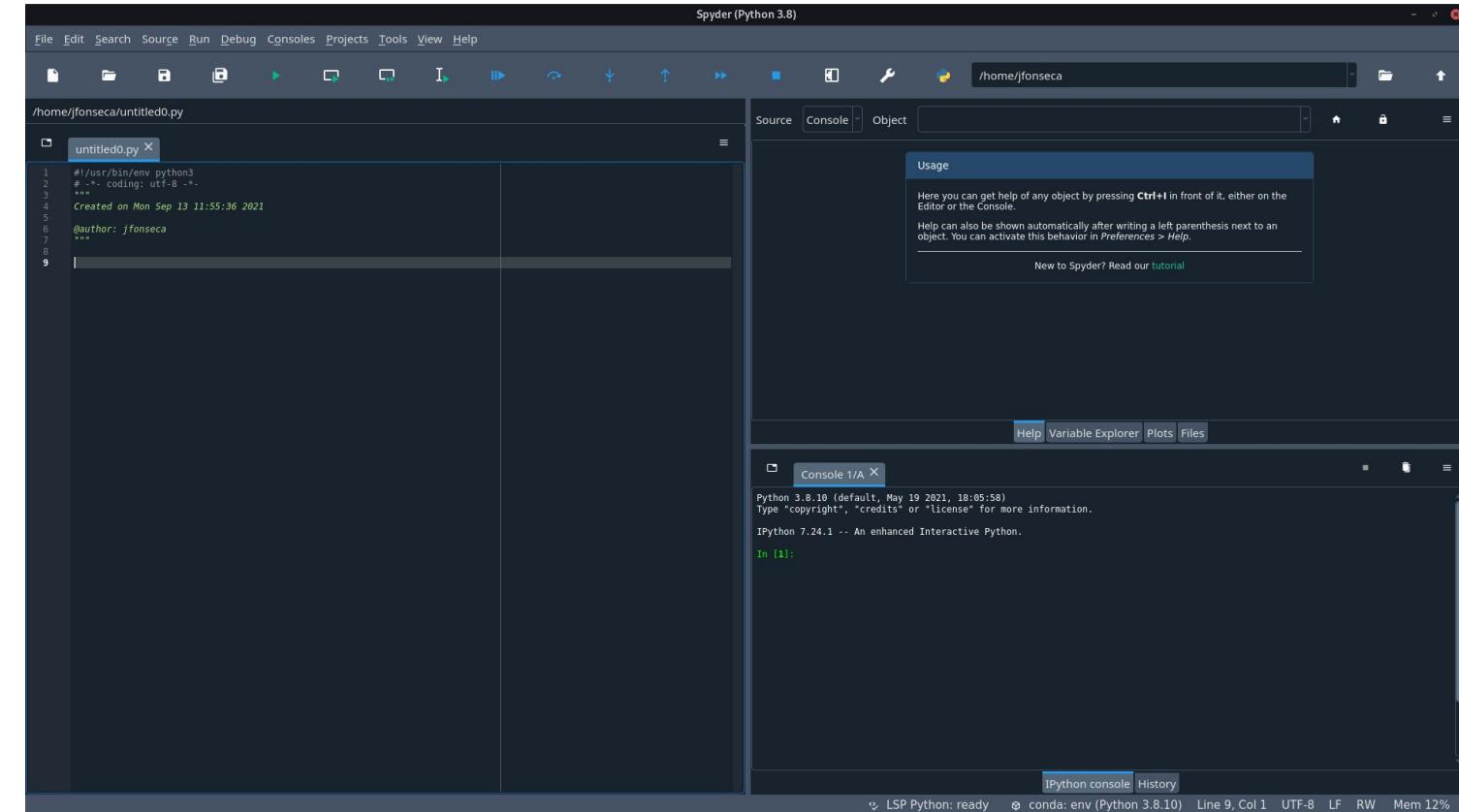
@app.route('/', methods=['GET', 'POST'])
def homepage():
    pagetype = 'home'
    title = 'Welcome to the pre-alpha SMC GUI/Dashboard'
    paragraph = ['Hi there, this is a GUI under development for my social media crawler project!', '', 'So many features to come!']
    #####
    #values for keywords set in /support
    kw_settings=open('support/keywords_config', 'r')
    kws=kw_settings.readlines()
    keyword_1=kws[0]
    keyword_2=kws[1]
    keyword_3=kws[2]
    kw_settings.close()
    #####
    #getting active keyword
    if request.method == "POST":
        active_keyword = request.form['nav_keyword']
        with open('support/active_keyword', 'w') as kw_filter:
            kw_filter.write(active_keyword)
        #time.sleep(6)
        with open('support/active_keyword', 'r') as kw_filter:
            header_keyword=kw_filter.readline()
    #####
    return render_template('homepage.html', pagetype=pagetype,
                           keyword_1=keyword_1,
```

Atom Text Editor

# Integrated Development Environment (IDE)

- Popular IDE's:
  - Spyder
  - PyCharm
  - VSCode
  - Rodeo
- Anaconda comes with Spyder and VSCode

Usage of IDE and/or Text editor (and which ones to use) comes down to personal preference



# Data Mining Project

- Form groups on Moodle (**up to 3 students**)
- Project guidelines will be released on Moodle
- Anonymized data from a real-world scenario
- Your goal is to develop a **Customer Segmentation** in such a way that it will be possible for the Marketing Department to better understand all the different Customers' Profiles.
- **This will be discussed in more detail later.**

# Let's get started!

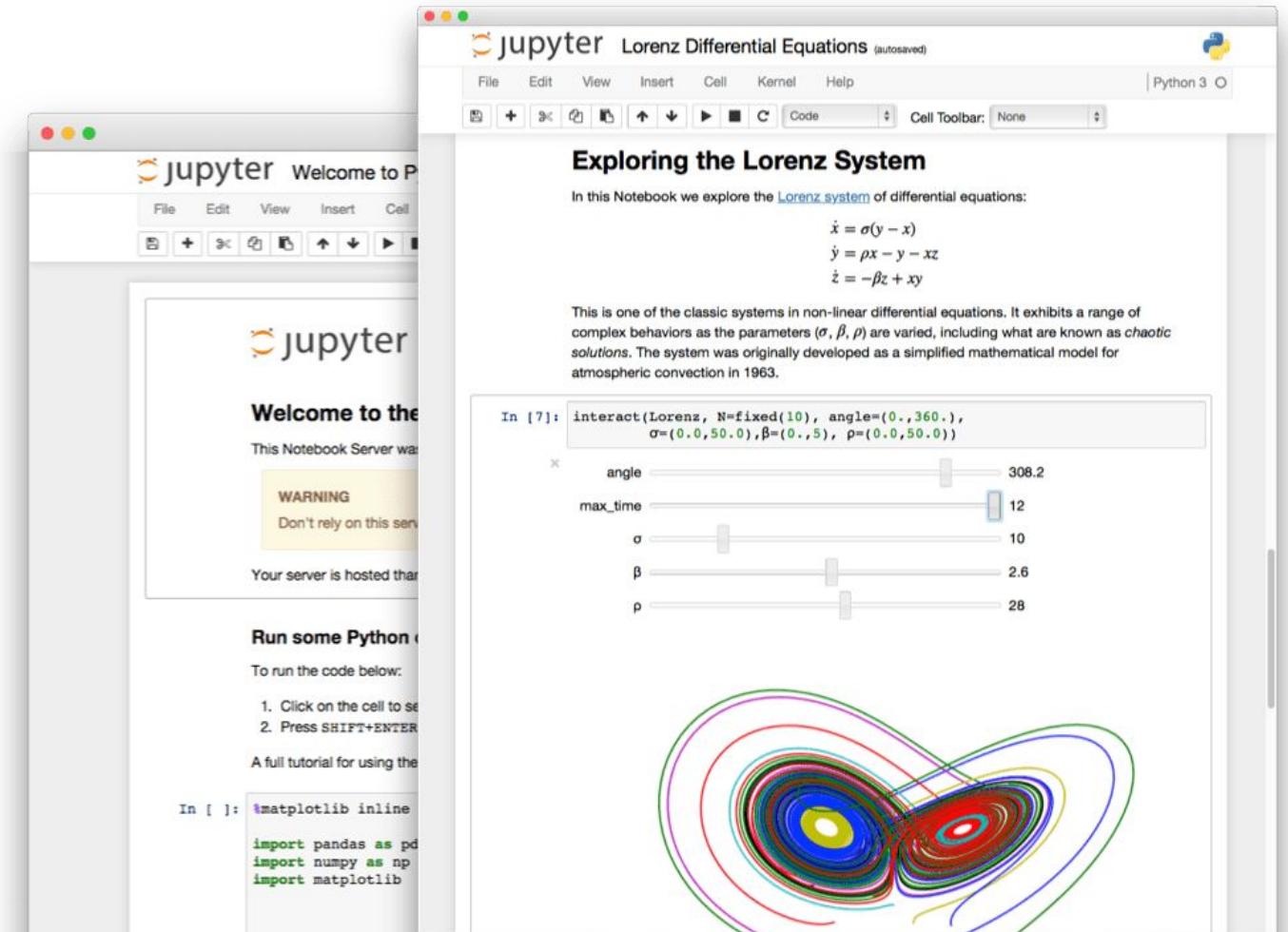
Next: Setting up our tools

# The Jupyter Notebook

<http://jupyter.org/>

Let's try it out!

- Install and Open Anaconda Navigator
- Start Jupyter Notebook



# Setting up our tools

Two options:

- 1. With GUI (Anaconda Navigator)**
2. Command line (miniconda) (skip to slide 64)

# Setting up our tools

Download lab files:

- `environment_dm.yml`
- `requirements_pip.txt`

Download installers:

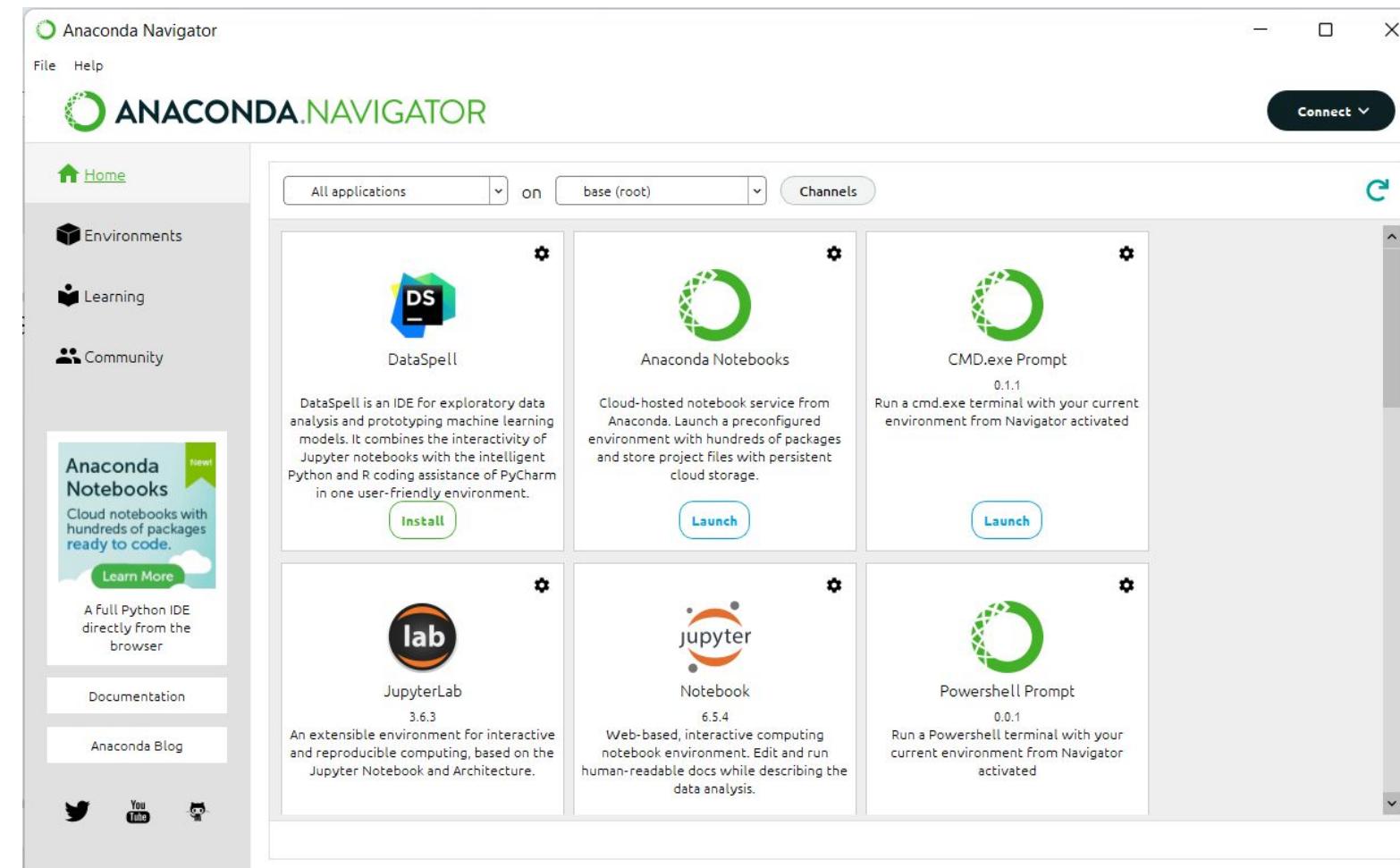
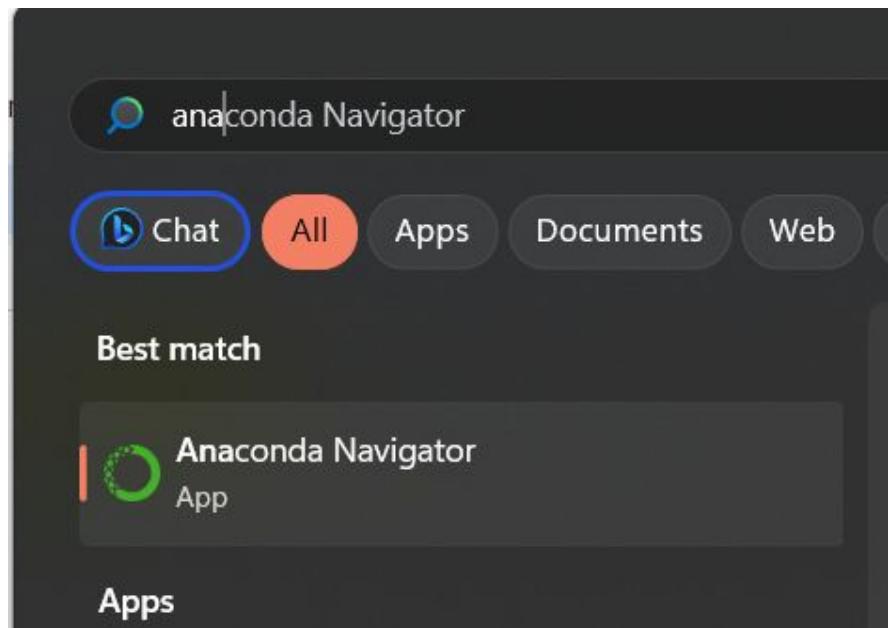
- Instructions on Moodle

If you downloaded a zip file, unzip it to your **Downloads** folder

- This makes it easier to find later

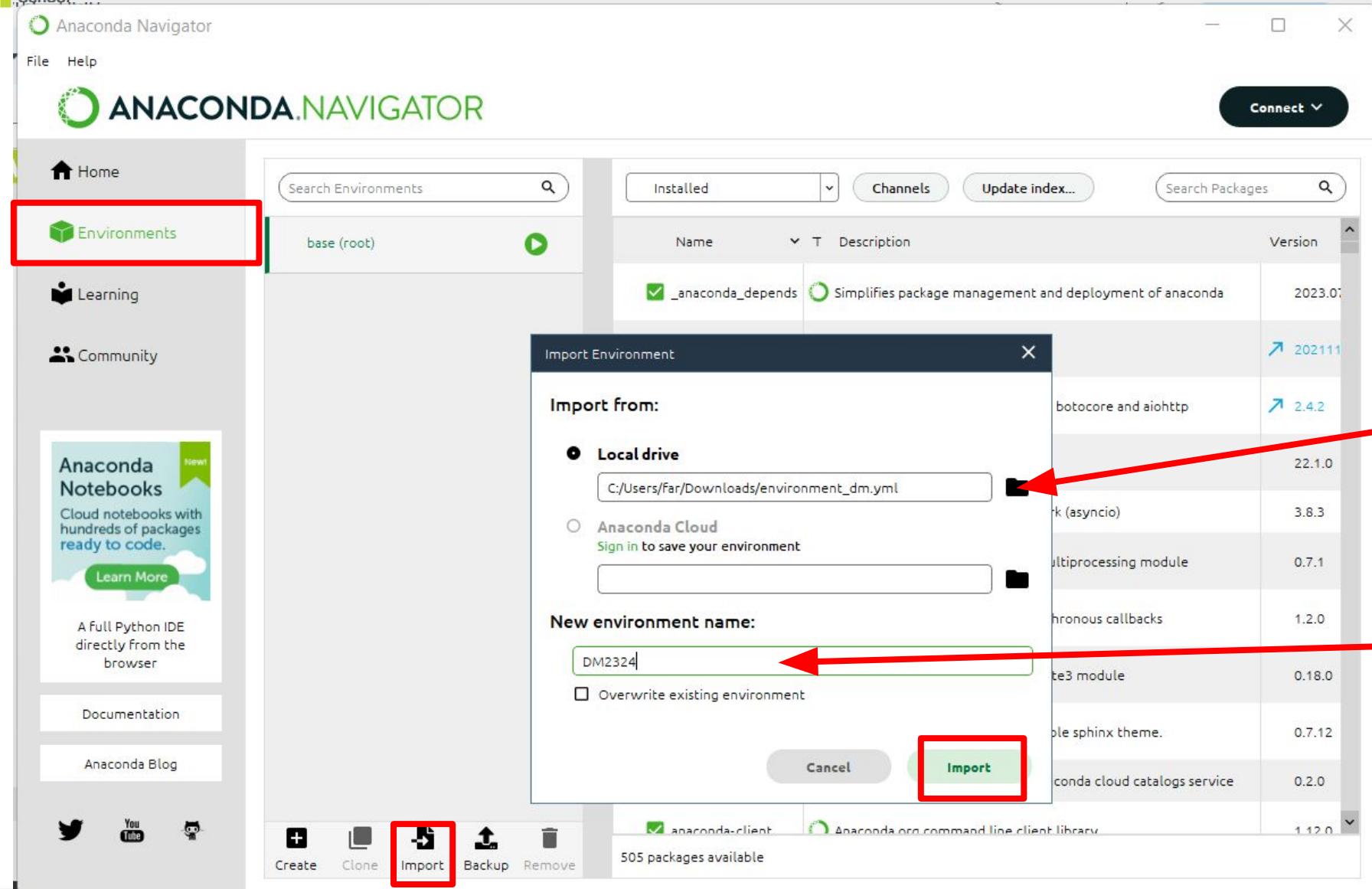
# Setting up our tools

Load Anaconda Navigator



# Import environment

!!! This may take some time !!!



Where you have the file:  
`environment_dm.yml`

Name the environment:  
`DM2324`

# Import environment: Done

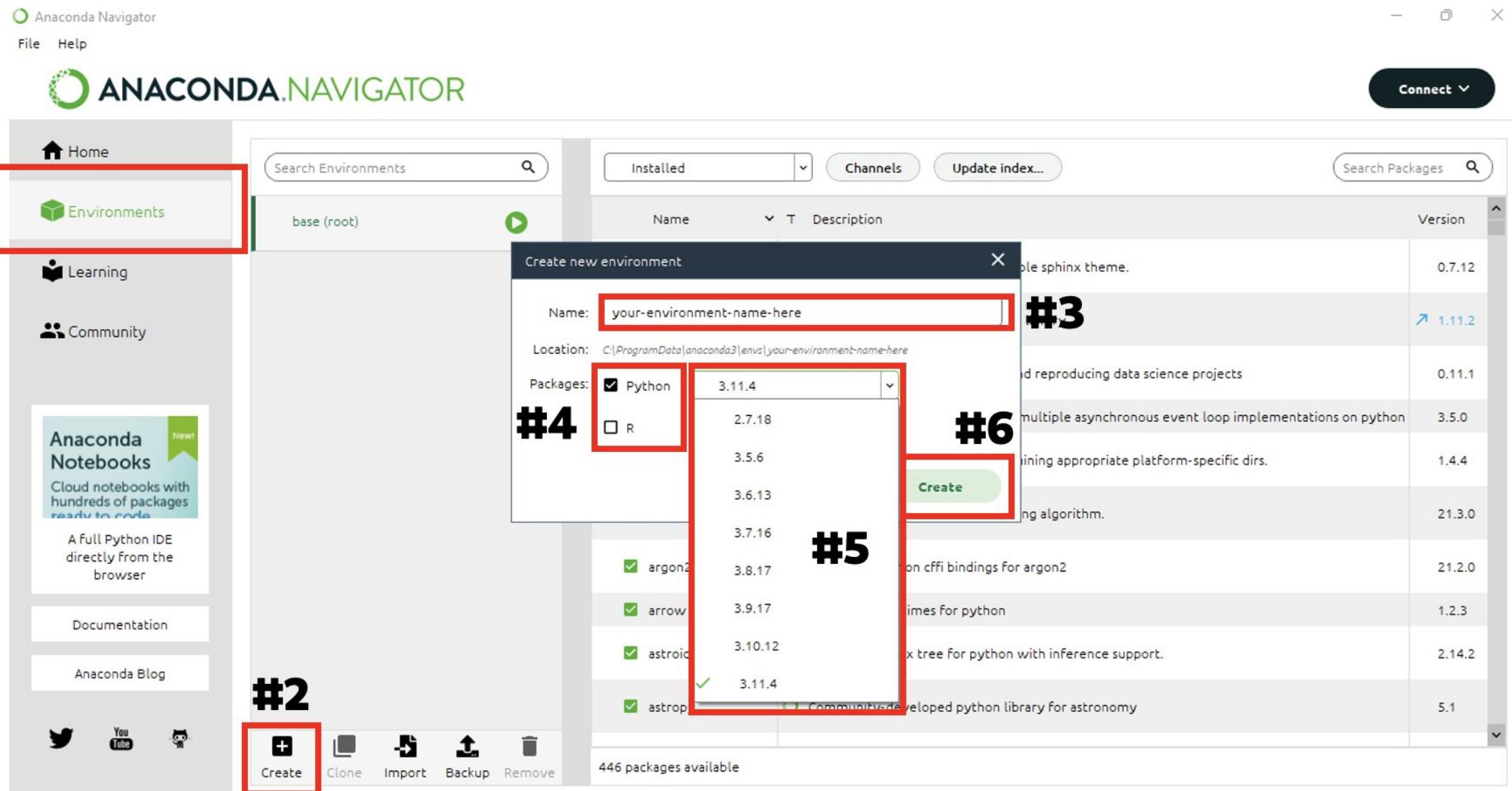
The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with links to Home, Environments, Learning, and Community. A prominent callout box highlights the 'Anaconda Notebooks' section, which says 'Cloud notebooks with hundreds of packages ready to code.' Below it, there are links for 'Learn More', 'A full Python IDE directly from the browser', 'Documentation', and 'Anaconda Blog'. At the bottom of the sidebar are social media icons for Twitter, YouTube, and GitHub.

The main area has tabs for 'Search Environments' and 'base (root)'. A red box highlights the environment 'DM2324' in the list, which includes a play button icon. The central part of the interface shows a table of installed packages:

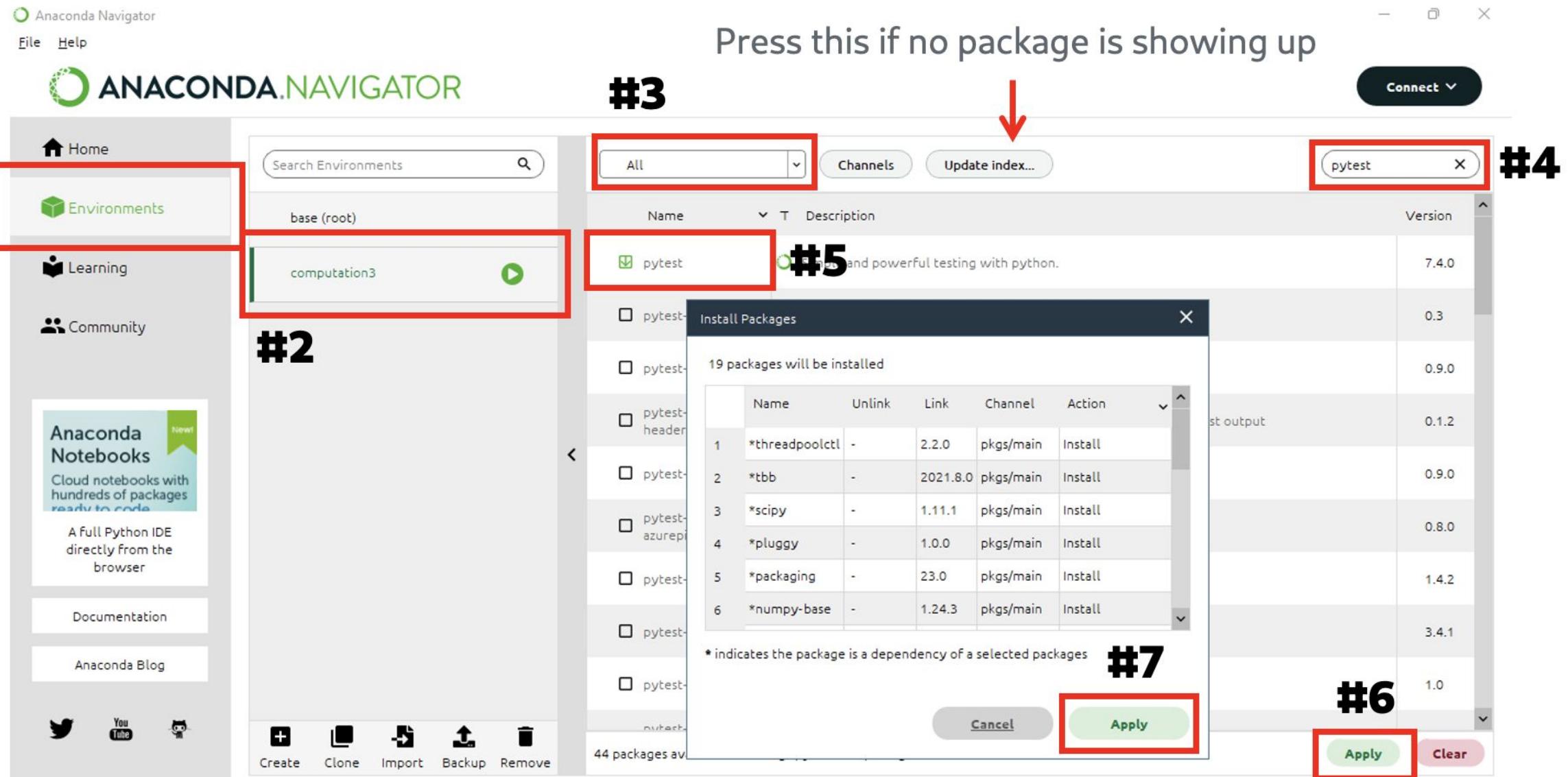
Name	Description	Version
anyio	High level compatibility layer for multiple asynchronous event loop implementations on python	4.0.0
aom	Alliance for open media video codec	3.5.0
argon2-cffi	The secure argon2 password hashing algorithm.	23.1.0
argon2-cffi-bindings	Low-level python ffi bindings for argon2	21.2.0
arrow	Better dates & times for python	1.2.3
asttokens	The asttokens module annotates python abstract syntax trees (asts) with the positions of tokens and text in the source code that generated them.	2.2.1
async-lru		2.0.4
attrs	Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).	23.1.0
babel	Utilities to internationalize and localize python applications	2.12.1
backcall	Specifications for callback functions passed in to an api	0.2.0

At the bottom of the main pane, it says '301 packages available'. Along the bottom edge of the interface are buttons for Create, Clone, Import, Backup, and Remove.

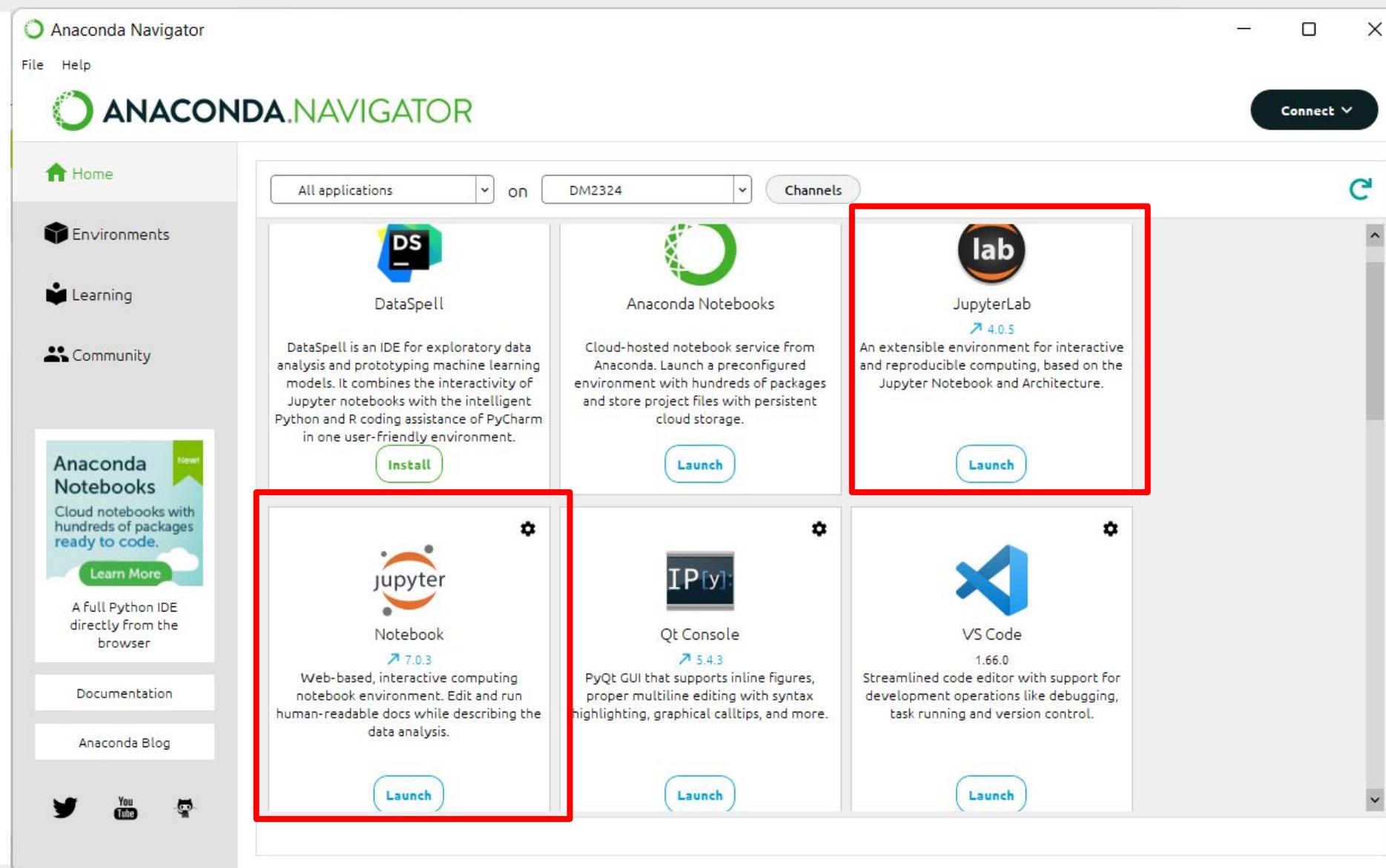
# You can also create an environment from scratch



# You can also install packages one by one



# Test loading Jupyter notebook



# Test loading Jupyter notebook

The screenshot shows a Jupyter Notebook interface running locally at `localhost:8888/tree`. The title bar includes standard browser controls (back, forward, search, etc.) and a user profile icon.

The main area has a header with the Jupyter logo and a navigation menu with links to File, View, Settings, and Help. Below the menu, there are two tabs: "Files" (selected) and "Running".

A central message in large red text reads: "Look for the lab files you downloaded".

The left sidebar shows a file tree with the root directory `/`. The tree lists several standard system directories like `anaconda3`, `Desktop`, `Documents`, `Downloads`, `Favorites`, `Links`, `Music`, `OneDrive`, `Pictures`, `Saved Games`, `scikit_learn_data`, `Searches`, and `Videos`.

The right side of the interface displays a table of files:

Name	Last Modified	File Size
<code>anaconda3</code>	1 hour ago	
<code>Desktop</code>	last year	
<code>Documents</code>	3 months ago	
<code>Downloads</code>	11 months ago	
<code>Favorites</code>	35 minutes ago	
<code>Links</code>	last year	
<code>Music</code>	last year	
<code>OneDrive</code>	last year	
<code>Pictures</code>	12 months ago	
<code>Saved Games</code>	last year	
<code>scikit_learn_data</code>	3 months ago	
<code>Searches</code>	last year	
<code>Videos</code>	last year	

# Load the lab01\_setup.ipynb notebook file

The screenshot shows a Jupyter Notebook interface with the title "jupyter lab01\_setup Last Checkpoint: 1 hour ago". The notebook contains a single section titled "Test if packages were installed correctly". Below the section title, there is a series of code cells. The first cell imports numpy and pandas. The second cell imports datasets from sklearn. The third cell imports matplotlib.pyplot and seaborn, and sets %matplotlib inline. It also configures the figure format to 'retina'. The fourth cell sets the seaborn style. The fifth cell prints the matplotlib version. The sixth cell ignores warnings. The seventh cell sets the logging level for matplotlib to WARNING.

```
[ ]: import numpy as np
[ ]: import pandas as pd

[ ]: from sklearn import datasets

[ ]: import matplotlib.pyplot as plt
[ ]: import seaborn as sns

%matplotlib inline

# for better resolution plots
%config InlineBackend.figure_format = 'retina'

# Setting seaborn style
sns.set()

[ ]: from matplotlib import __version__ as mplver
[ ]: print(mplver)

[ ]: import warnings
[ ]: warnings.filterwarnings('ignore')

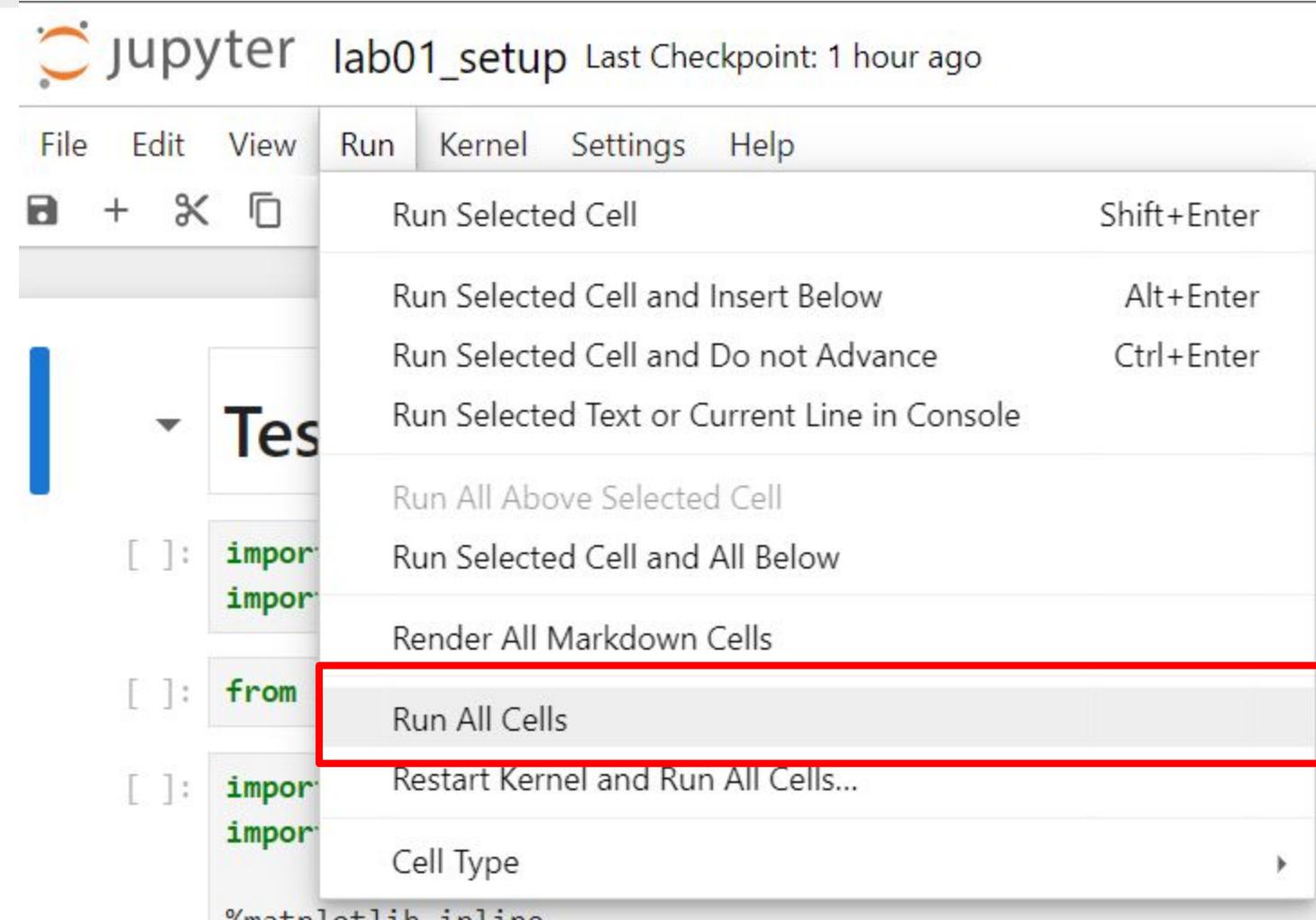
[ ]: import logging
[ ]: logging.getLogger('matplotlib').setLevel(logging.WARNING)
```

# Don't worry about understanding the code at this point

Right now we just want to make sure that all the packages we need are installed and work correctly

# Run all cells

You will reach a part in the notebook that will give an error.  
We still need to install some additional packages.



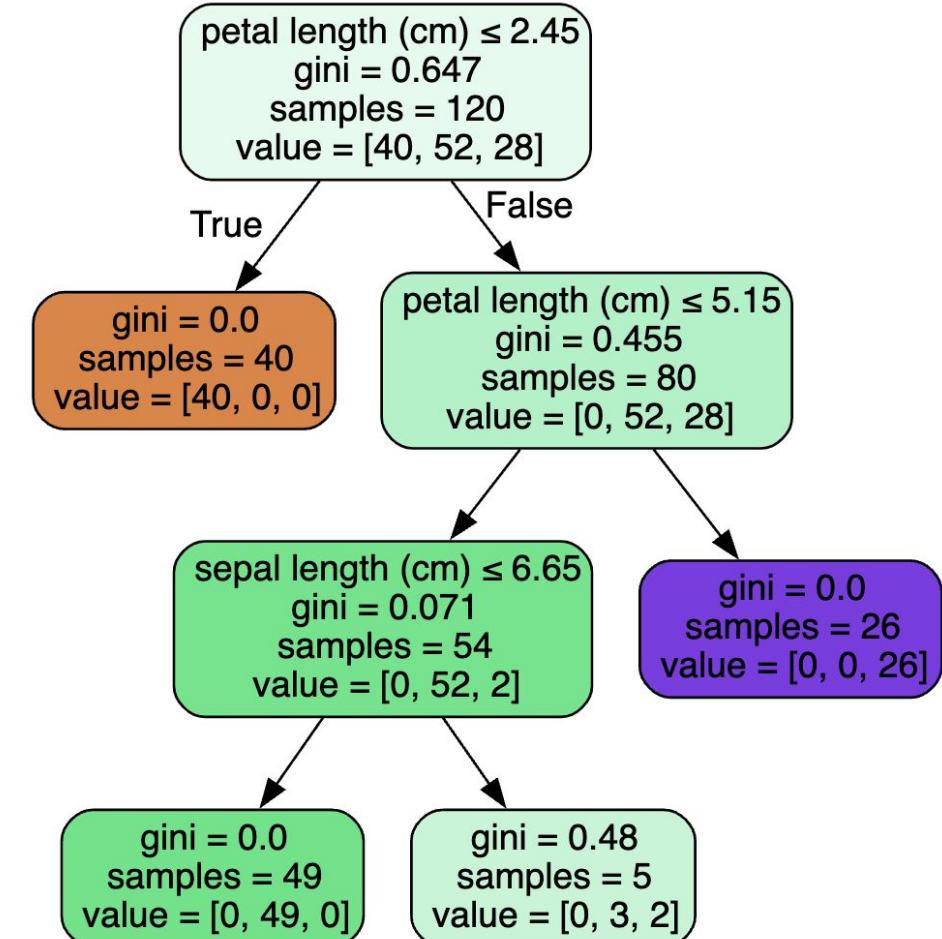
# Install other packages

Some of these packages may require more steps to install depending on your system

# Install other packages: graphviz

The graphviz package allows us to visualize graphs, trees, networks etc.

We will use it to visualize the results of a Decision Tree Classifier.



# Install other packages: graphviz

Mac users:

(assuming you have homebrew installed; if not, install homebrew first)

Open Terminal (Applications > Utilities > Terminal)

```
brew install graphviz
```

# Install other packages: graphviz

Ubuntu/Debian users

Open Terminal

```
sudo apt-get update && sudo apt-get install graphviz
```

# Install other packages: graphviz

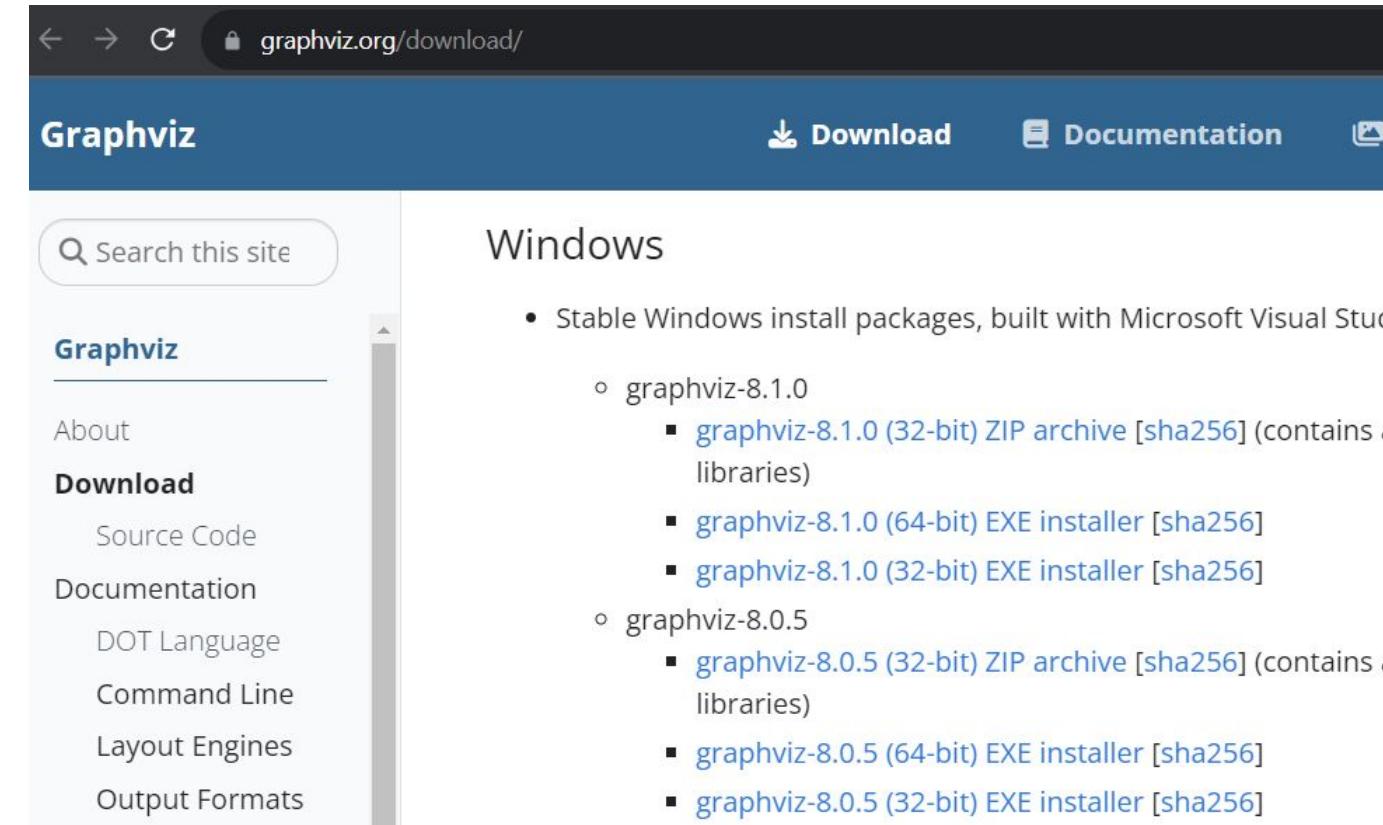
Windows users:

<https://stackoverflow.com/questions/35064304/runtimeerror-make-sure-the-graphviz-executables-are-on-your-systems-path-after-installing>

# Install other packages: graphviz

Windows users:  
download graphviz from

<https://graphviz.org/download/>



The screenshot shows a web browser displaying the Graphviz download page at [graphviz.org/download/](https://graphviz.org/download/). The page has a blue header with the Graphviz logo and navigation links for Download and Documentation. A sidebar on the left contains links for Graphviz, About, Download (with sub-links for Source Code, Documentation, DOT Language, Command Line, Layout Engines, and Output Formats), and a search bar. The main content area is titled "Windows" and lists stable Windows install packages. It includes sections for graphviz-8.1.0 and graphviz-8.0.5, each with links to 32-bit and 64-bit ZIP archives and EXE installers, along with their corresponding SHA256 checksums.

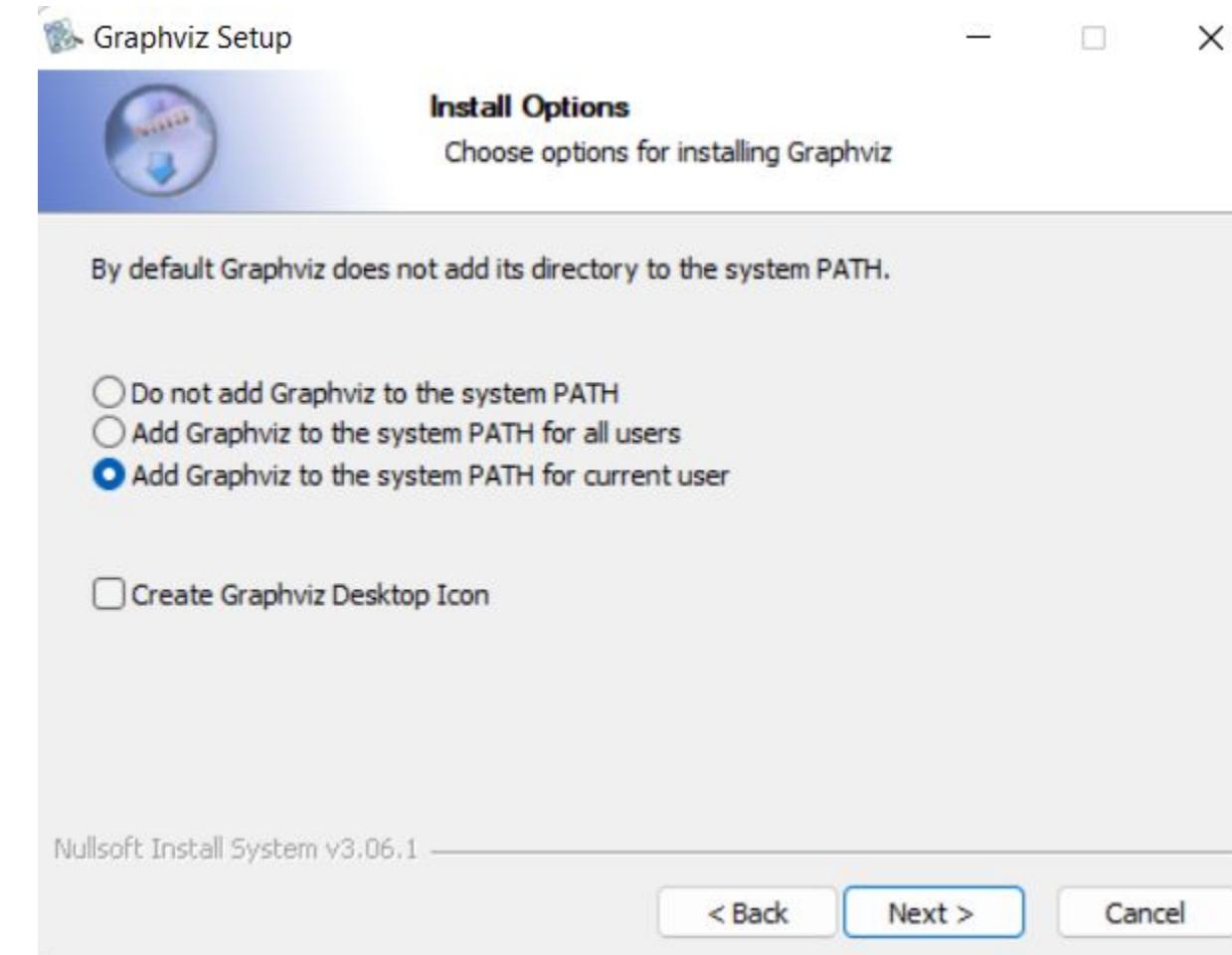
Windows

- Stable Windows install packages, built with Microsoft Visual Studio
  - graphviz-8.1.0
    - [graphviz-8.1.0 \(32-bit\) ZIP archive \[sha256\]](#) (contains libraries)
    - [graphviz-8.1.0 \(64-bit\) EXE installer \[sha256\]](#)
    - [graphviz-8.1.0 \(32-bit\) EXE installer \[sha256\]](#)
  - graphviz-8.0.5
    - [graphviz-8.0.5 \(32-bit\) ZIP archive \[sha256\]](#) (contains libraries)
    - [graphviz-8.0.5 \(64-bit\) EXE installer \[sha256\]](#)
    - [graphviz-8.0.5 \(32-bit\) EXE installer \[sha256\]](#)

# Install other packages: graphviz

Windows users:  
download graphviz from

<https://graphviz.org/download/>



# Install other packages: graphviz

Windows users:

Follow instructions here to add graphviz / dot to PATH

<https://stackoverflow.com/questions/35064304/runtimeerror-make-sure-the-graphviz-executables-are-on-your-systems-path-aft>

# Install other packages: graphviz

Windows users:

(This is step 1, we'll do step 2 later)



26



## Step 1: Install Graphviz binary

### Windows:

1. Download Graphviz from <http://www.graphviz.org/download/>
2. Add below to PATH environment variable (mention the installed graphviz version):
  - C:\Program Files (x86)\Graphviz2.38\bin
  - C:\Program Files (x86)\Graphviz2.38\bin\dot.exe
3. Close any opened Jupyter notebook and the command prompt
4. Restart Jupyter / cmd prompt and test

### Linux:

1. sudo apt-get update
2. sudo apt-get install graphviz
3. or build it manually from <http://www.graphviz.org/download/>

## Step 2: Install graphviz module for python

### pip:

- pip install graphviz

# Install other packages: graphviz

Windows users:

A screenshot of a search engine results page. The search bar at the top contains the query "add path windows". Below the search bar are several category filters: Videos, Command line, Images, Directory, Windows 10, 10 command line, and Python. A horizontal line separates these from the search results. Below the line, it says "About 768,000,000 results (0.42 seconds)". Underneath this, there is a section titled "Direct link to this answer" followed by a numbered list of 8 steps.

add path windows

Videos   Command line   Images   Directory   Windows 10   10 command line   Python

About 768,000,000 results (0.42 seconds)

**Direct link to this answer**

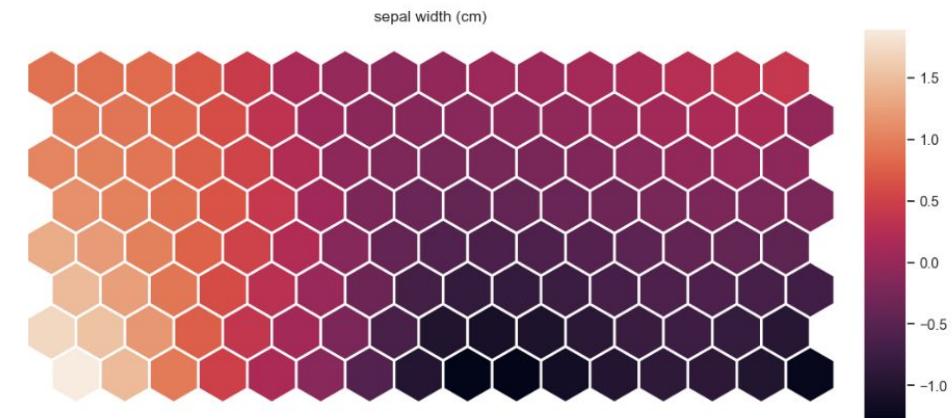
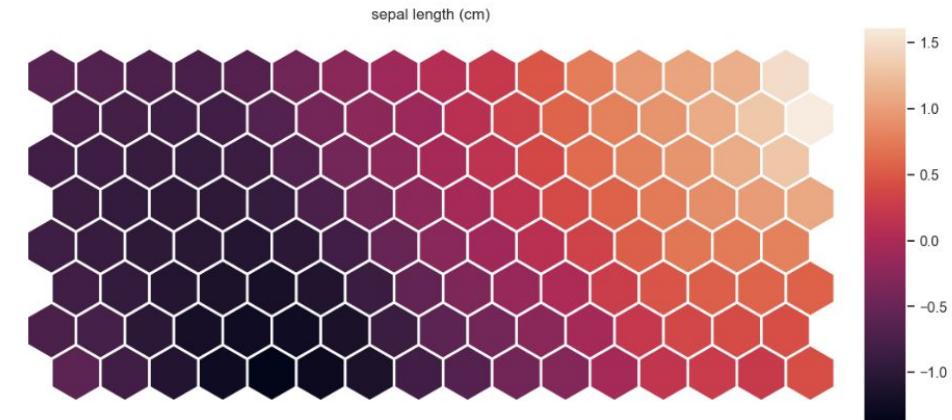
1. Right-click on the Start Button.
2. Select “System” from the context menu.
3. Click “Advanced system settings”
4. Go to the “Advanced” tab.
5. Click “Environment Variables...”
6. Click variable called “Path” and click “Edit...”
7. Click “New”
8. Enter the path to the folder containing the binary you want on your PATH.

# Does everyone have graphviz installed now?

# Install Python packages: sompy

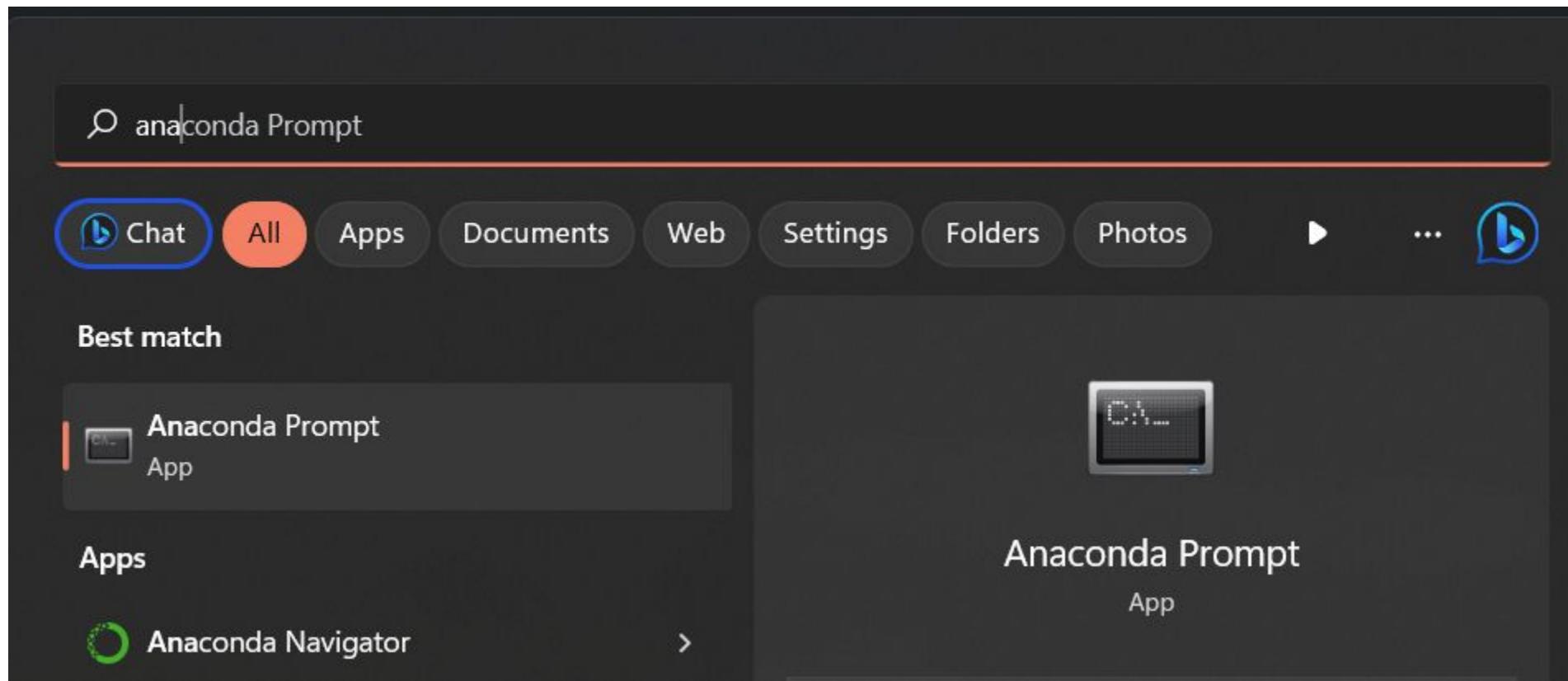
The sompy package allows us to train and visualize Self-Organizing Maps.

```
sns.set()  
view2D = View2D(28, 46, "", text_size=150)  
view2D.show(sm, col_sz=1, what='codebook')  
  
plt.show()
```



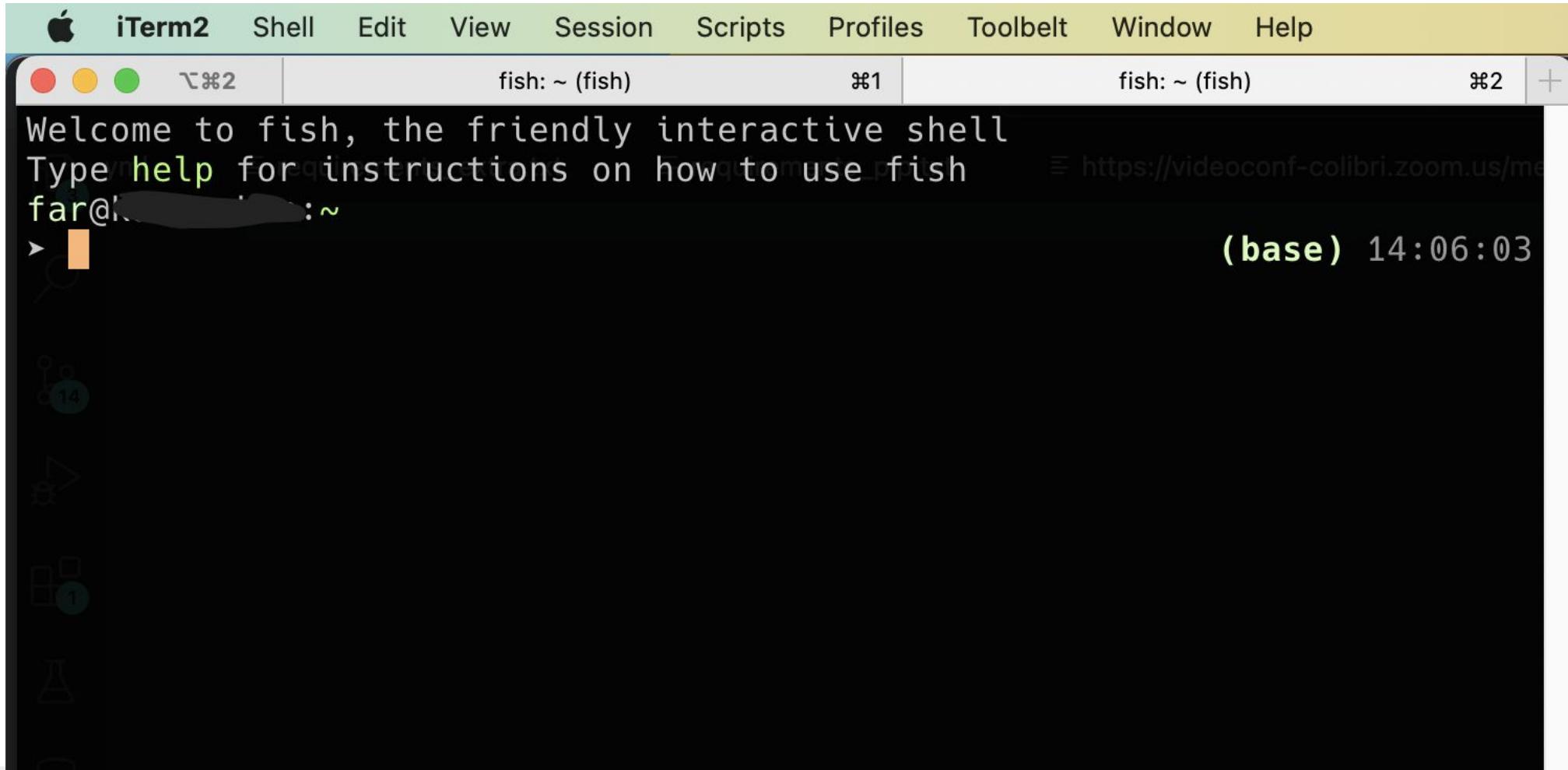
# Install Python packages: graphviz/sompy

**Windows:** Open Anaconda Prompt



# Install Python packages: graphviz/sompy

**Mac / Linux:** Open Terminal



# Install Python packages: graphviz/sompy

```
conda activate DM2324          ← Activate the environment we created earlier  
cd Downloads                  ← Navigate to where you have the downloaded files  
pip install -r requirements_pip.txt ← Install packages with pip
```

```
(base) C:\Users\far>conda activate DM2324
(DM2324) C:\Users\far>cd Downloads
(DM2324) C:\Users\far\Downloads>pip install -r requirements_pip.txt
Collecting git+https://github.com/sevamoo/SOMPY (from -r requirements_pip.txt)
  Cloning https://github.com/sevamoo/SOMPY to c:\users\far\appdata\local\temp\tmpqjwzv\pip-req-build-fyxt0w2b'
    Running command git clone --filter=blob:none --quiet https://github.com/sevamoo/SOMPY
      Resolved https://github.com/sevamoo/SOMPY to commit 6aca604b06e5eeaa...
      Preparing metadata (setup.py) ... done
Collecting graphviz (from -r requirements_pip.txt (line 2))
  Downloading graphviz-0.20.1-py3-none-any.whl (47 kB)
    47.0/47.0 kB 593.5 kB/s
```

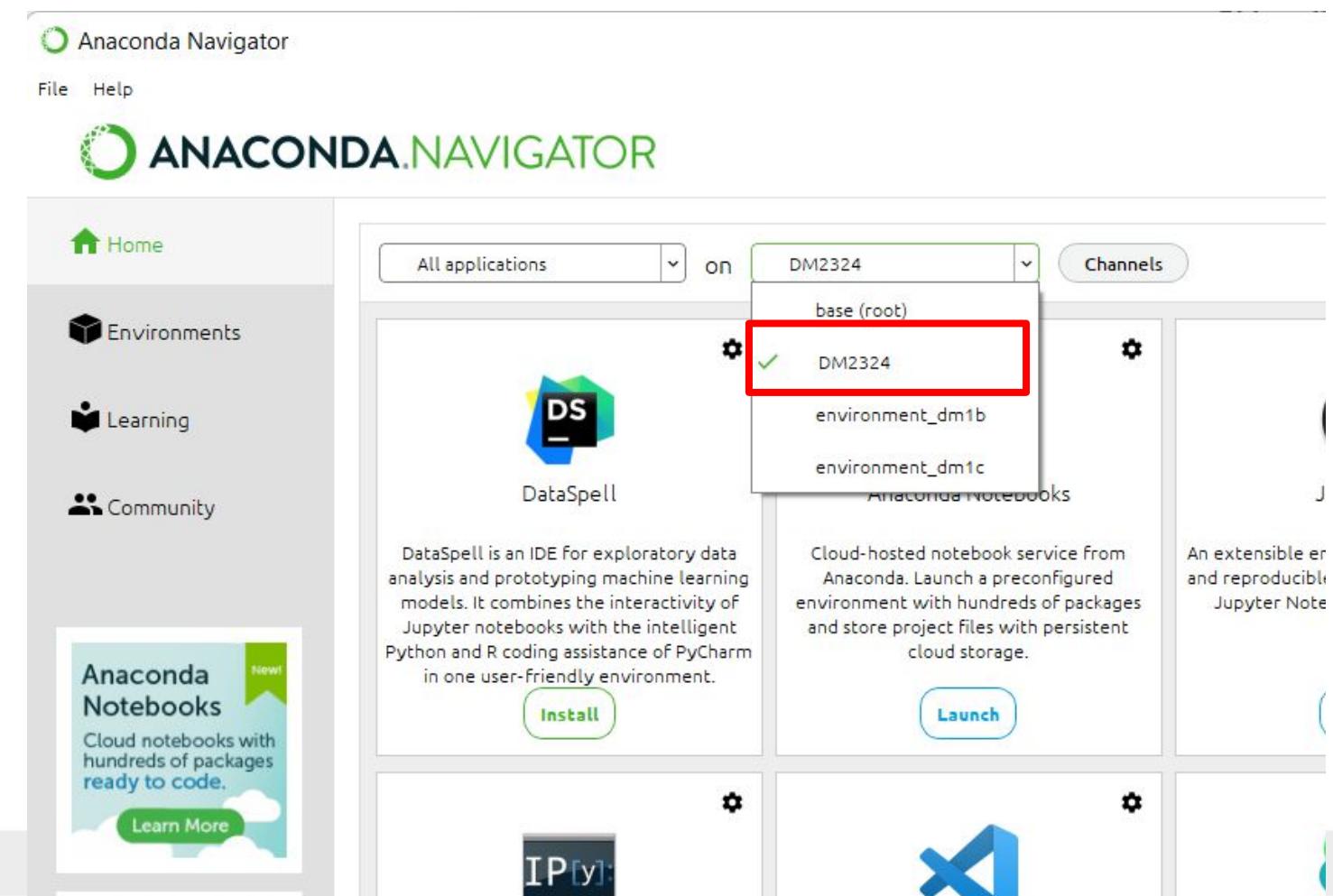
# Test Jupyter notebook

Close all Terminal windows / Anaconda Navigator / Anaconda Prompt

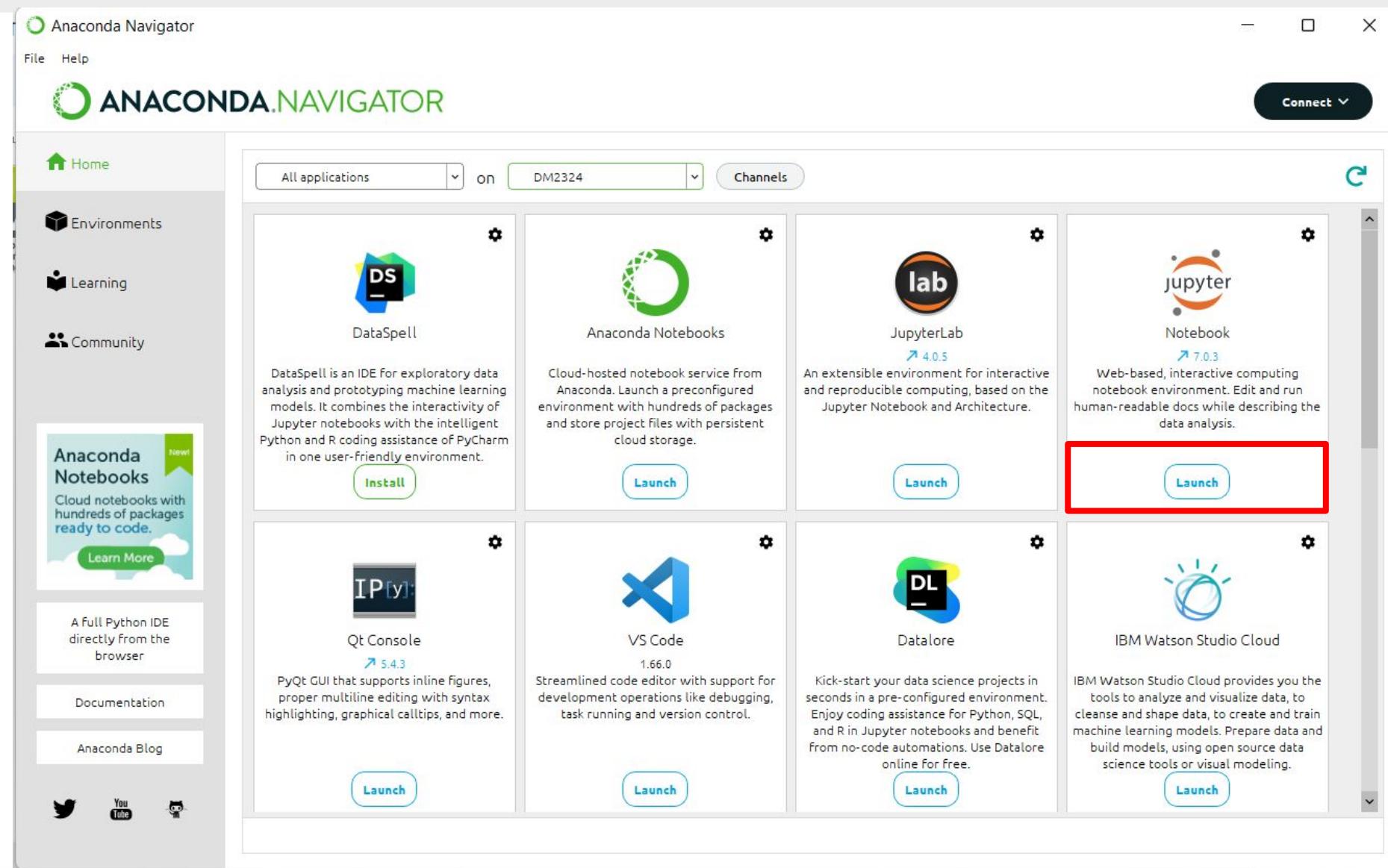
# Test Jupyter notebook

Open Anaconda Navigator

Make sure you select the environment we created



# Test Jupyter notebook



# Test Jupyter notebook

The screenshot shows a Jupyter Notebook interface running on a local server at `localhost:8888/tree`. The title bar includes standard browser controls like back, forward, and search, along with user profile and settings icons.

The main area displays a file tree under the heading "jupyter". The "Files" tab is selected, showing a list of directories: anaconda3, Contacts, Desktop, Documents, Downloads, Favorites, Links, Music, OneDrive, Pictures, Saved Games, scikit\_learn\_data, Searches, and Videos. Each entry includes a checkbox for selection and a timestamp indicating when it was last modified.

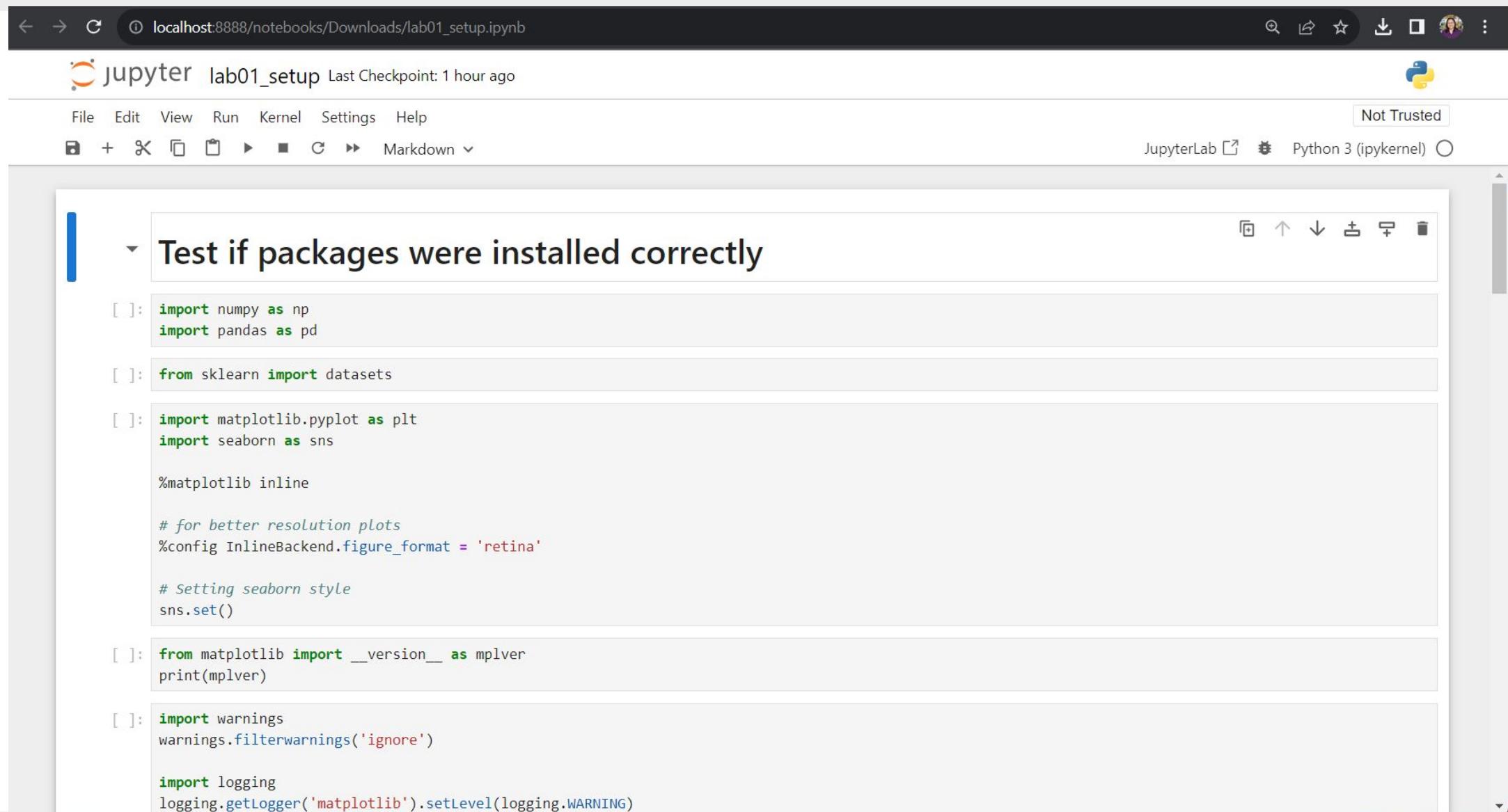
A large, bold, red text message "Look for the lab files you downloaded" is centered over the file list.

Name	Last Modified	File Size
anaconda3	1 hour ago	
Contacts	last year	
Desktop	3 months ago	
Documents	11 months ago	
Downloads	35 minutes ago	
Favorites	last year	
Links	last year	
Music	last year	
OneDrive	last year	
Pictures	12 months ago	
Saved Games	last year	
scikit_learn_data	3 months ago	
Searches	last year	
Videos	last year	

# Don't worry about understanding the code at this point

Right now we just want to make sure that all the packages we need are installed and work correctly

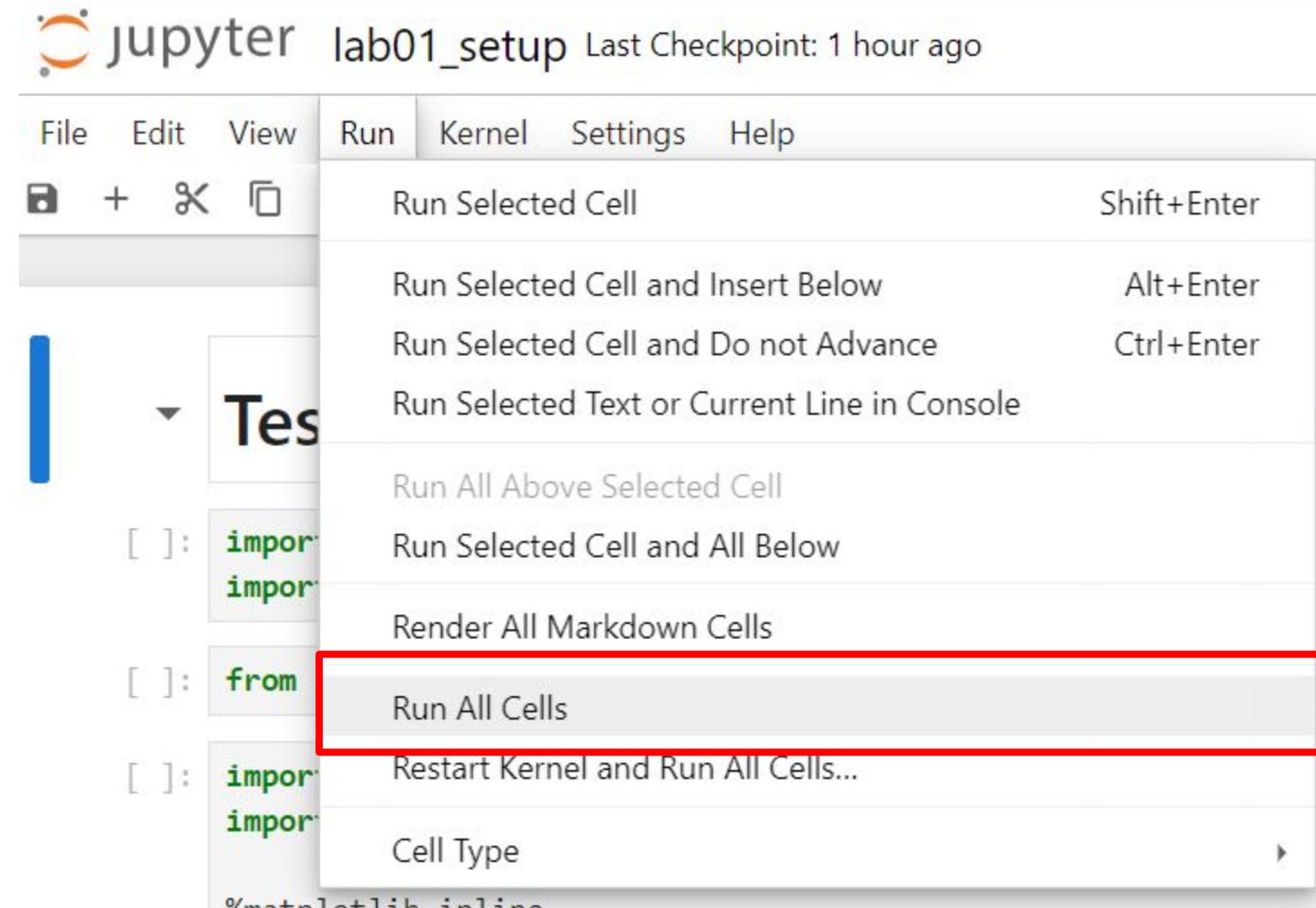
# Load the notebook file



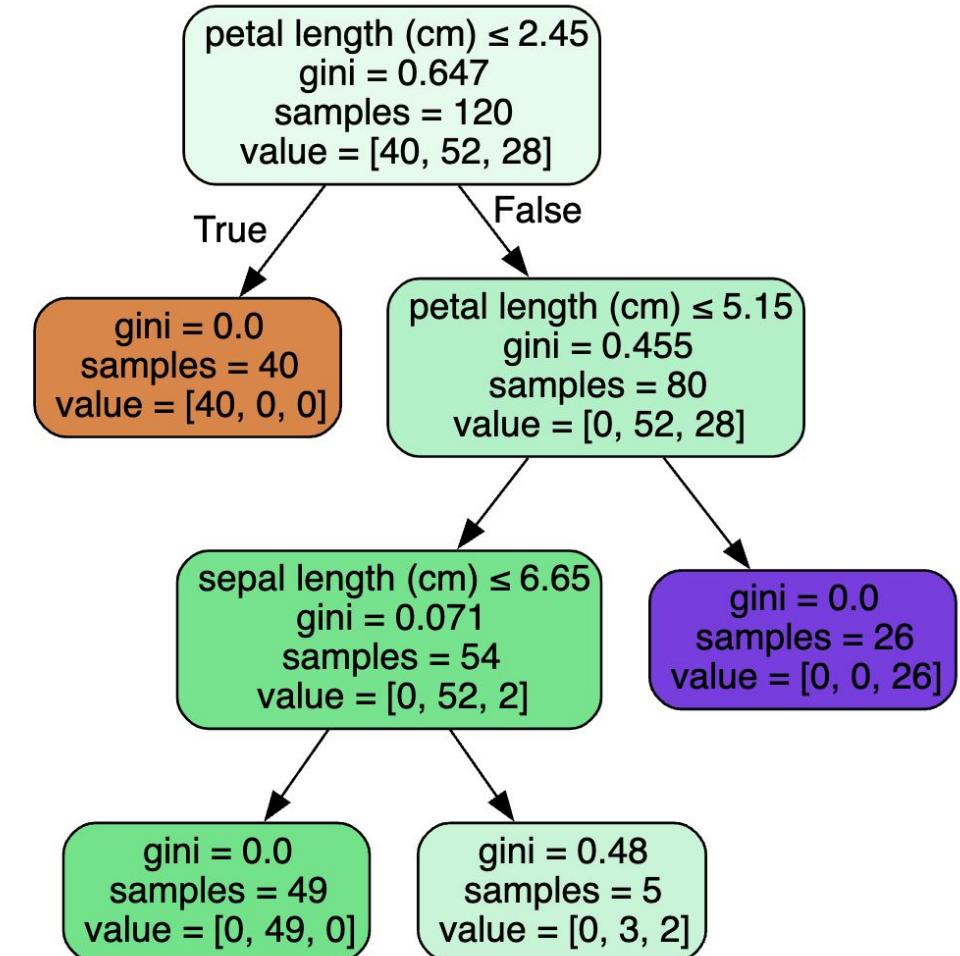
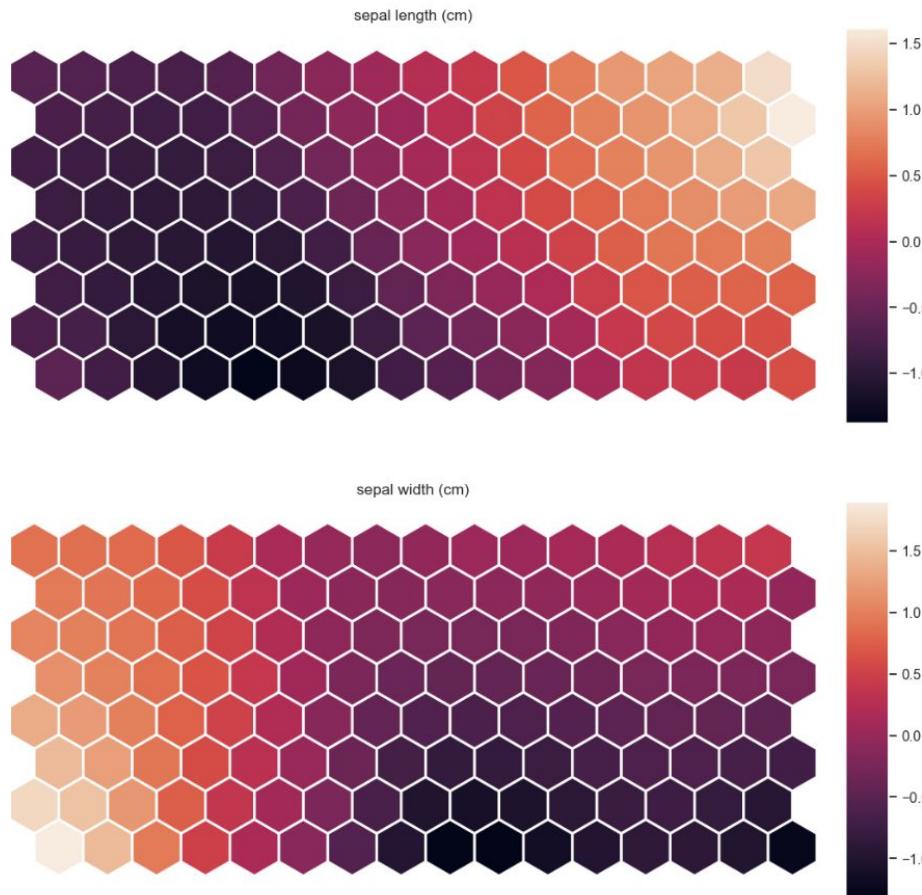
The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost:8888/notebooks/Downloads/lab01\_setup.ipynb
- Header:** jupyter lab01\_setup Last Checkpoint: 1 hour ago, Python 3 (ipykernel)
- Toolbar:** File, Edit, View, Run, Kernel, Settings, Help, Not Trusted, JupyterLab, Python 3 (ipykernel)
- Cells:** The notebook contains several code cells:
  - [ ]: `import numpy as np`  
`import pandas as pd`
  - [ ]: `from sklearn import datasets`
  - [ ]: `import matplotlib.pyplot as plt`  
`import seaborn as sns`  
  
`%matplotlib inline`  
  
`# for better resolution plots`  
`%config InlineBackend.figure_format = 'retina'`  
  
`# Setting seaborn style`  
`sns.set()`
  - [ ]: `from matplotlib import __version__ as mplver`  
`print(mplver)`
  - [ ]: `import warnings`  
`warnings.filterwarnings('ignore')`  
  
`import logging`  
`logging.getLogger('matplotlib').setLevel(logging.WARNING)`

# Run all cells



# You should be able to see these visualizations



Does anyone still have any errors in the  
notebook?

# Installing using command line

You can also use miniconda  
instead of Anaconda Navigator

<https://docs.anaconda.com/free/anaconda/getting-started/distro-or-miniconda/>

# Installing using the command line

Install miniconda

<https://docs.conda.io/projects/miniconda/en/latest/#quick-command-line-install>

# Create environment using the command line

Windows: Open Anaconda Prompt (miniconda3)

Linux/Mac: Open Terminal

```
cd Downloads
```

```
conda env create -f environment_dm.yml
```

```
conda activate DM2324
```

# Test Jupyter notebook + install other packages

Windows: Open Anaconda Prompt (miniconda3)  
Linux/Mac: Open Terminal

```
conda activate DM2324  
jupyter notebook
```

Follow the instructions in previous slides for testing Jupyter notebook and installing additional packages (**from slide 34**)

# Let's get started! (for real)

Next:  
Jupyter notebook  
Distance Matrix

# Questions?

Morada: Campus de Campolide, 1070-312 Lisboa, Portugal

Tel: +351 213 828 610 | Fax: +351 213 828 611

Acreditações e Certificações da NOVA IMS



UNIGIS



Computing  
Accreditation  
Commission



Cofinanciado por

