

ABILITÀ INFORMATICHE

(3 CFU)

Link moodle: <https://moodle2.units.it/course/view.php?id=11690>

Teams code: 0oixxrp

Main goals of the course:

- know the basics of Linux commands, bash and Python languages,
- become familiar with different working environments (i.e. shell, Git repository, Jupiter notebook),
- write simple scripts

Calendar

SEPTEMBER 2023

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	Valentini ²⁶ Bertocco	27	28	29	30

OCTOBER 2023

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1	2	Bertocco ³ Valentini	4	5	6	7
8	9	Bertocco ¹⁰ Valentini	11	12	13	14
15	16	Bertocco ¹⁷ Valentini	18	19	20	21
22	23	Valentini ²⁴ Bertocco	25	26	27	28
29	30	Valentini ³¹ Bertocco				

NOVEMBER 2023

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
			1	2	3	4
5	6	Valentini ⁷	8	9	10	11
12	13	Valentini ¹⁴ Bertocco	15	16	17	18
19	20	Valentini ²¹	22	23	24	25
26	27	Valentini ²⁸ Bertocco	29	30		

DECEMBER 2023


SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
					1	2
3	4	Bertocco ⁵	6	7	8	9
10	11	Bertocco ¹²	13	14	15	16
17	18	19	20	21	22	


Timeslots: 15:15 → 16:30 break 16:45 → 18:00

Contacts and exam

Dr. Bertocco Sara


Office:


 Astronomical Observatory of Trieste
via G.B. Tiepolo 11, 34143 - Trieste

 Please, always send an email to
sara.bertocco@inaf.it
to schedule a meeting

Dr. Valentini Milena

Office:

 Astronomical Observatory of Trieste
via G.B. Tiepolo 11, 34143 - Trieste
Office: 1301 (third floor)
Phone: 040 3199243

 Please, always send an email to
milena.valentini@units.it
to schedule a meeting

Final exam:

Set of exercises and problems to be solved and sent for evaluation.

Learning material

Text books, bibliography and useful resources

- <https://www.rigacci.org/docs/biblio/online/sysadmin/toc.htm>
- <https://www.tldp.org/LDP/abs/html/>
- <http://www.ee.surrey.ac.uk/Teaching/Unix/>
- <https://github.com/gtaffoni/Learn-Python/blob/master/Lectures/ShellLecture01.pdf>
- <https://github.com/gtaffoni/Learn-Python/blob/master/Lectures/ShellLecture02.pdf>
- https://github.com/bertocco/bash_lectures

Learning material

Text books, bibliography and useful resources

- Numerical Python in Astronomy and Astrophysics - A Practical Guide to Astrophysical Problem Solving (*Authors: W. Schmidt and M. Völschow*)
- Think Python, 2nd Edition - How to Think Like a Computer Scientist (*Author: A. B. Downey*)
- How to Think Like a Computer Scientist (<https://openbookproject.net/thinkcs/python/english3e/index.html>)
- Python Scripting for Computational Science (*Author: H. P. Langtangen*)
- Parallel Programming with Python (*Author: J. Palach*)
- <https://www.python.org/>
- <https://github.com/sarusso/ProgrammingLab>
- <https://moodle2.units.it/course/view.php?id=7455>

Intro

Lecture 1: Basics, working environments, Anaconda and Jupyter Notebook

Lecture 2: Bash and Linux command line introduction

Lecture 3: Advanced bash

Lecture 4: Bash scripting

Lecture 5: Essentials of Git. Python - introduction, main concepts, errors, variables, scripts

Lecture 6: Python - operators, conditions, functions

Lecture 7: Python - strings, lists, tuples, dictionaries

Lecture 8: Python - data structures, how to read/write from/a file, input/output, input from command line

Lecture 9: Python - classes, module import, libraries, arrays, objects, try/except, generators

Lecture 10: Python exercise and main ideas of parallelization with mpi4py

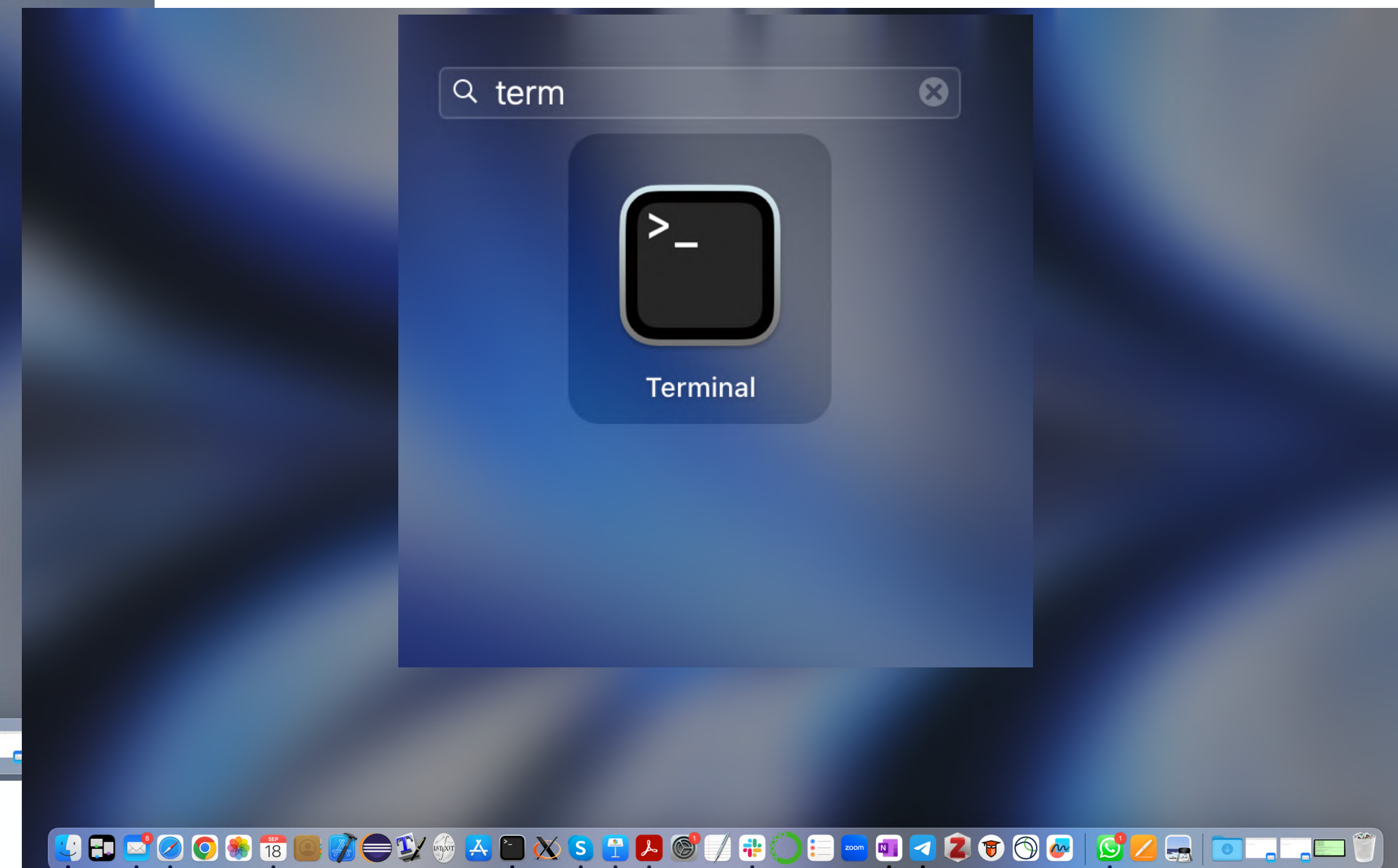
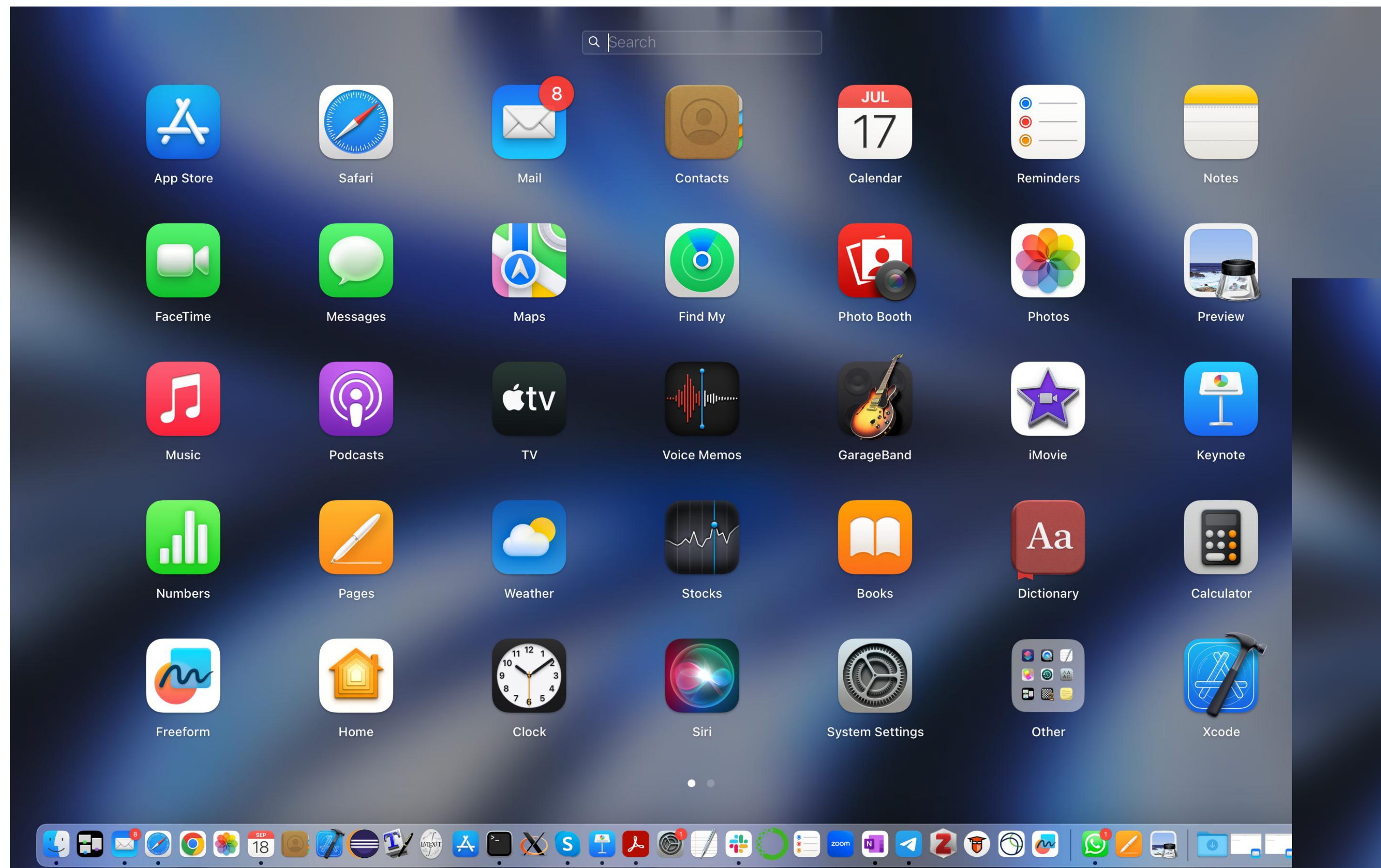
Lecture 11: Bash and Python exercises

Lecture 12: Bash and Python exercises

The command line tool

OS: Mac

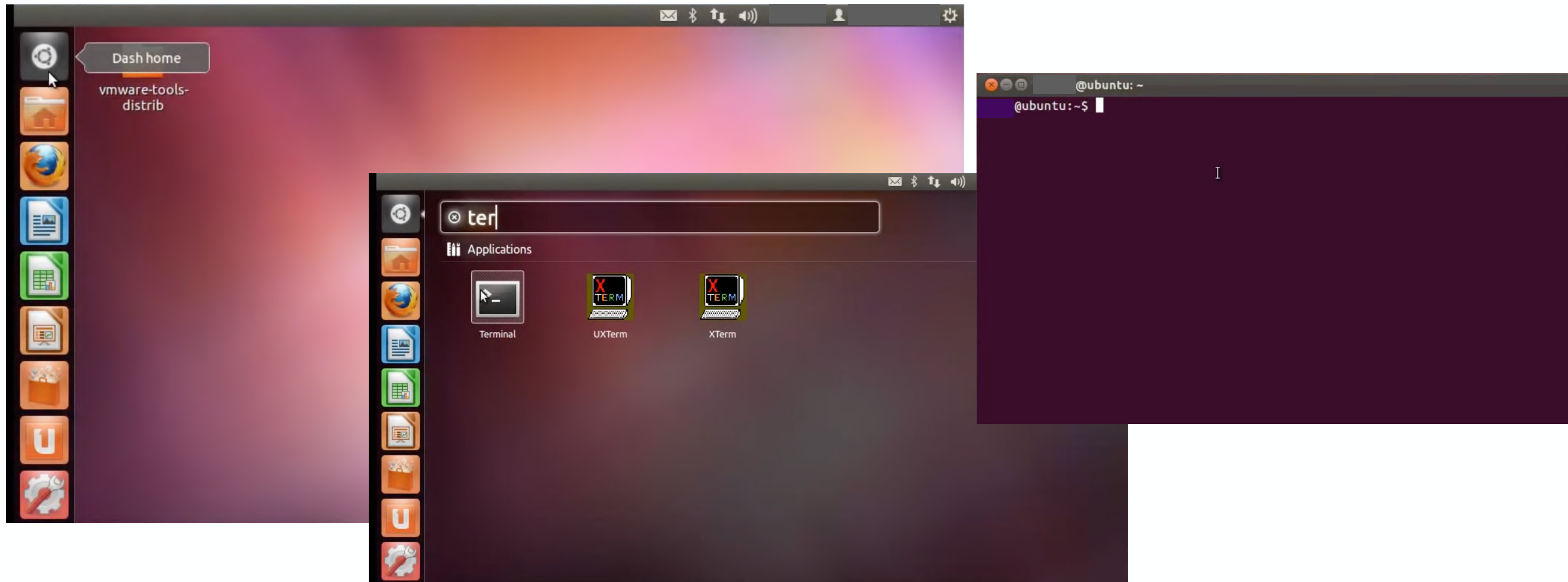
Look for the Terminal among the applications



The command line tool

OS: Linux/Unix

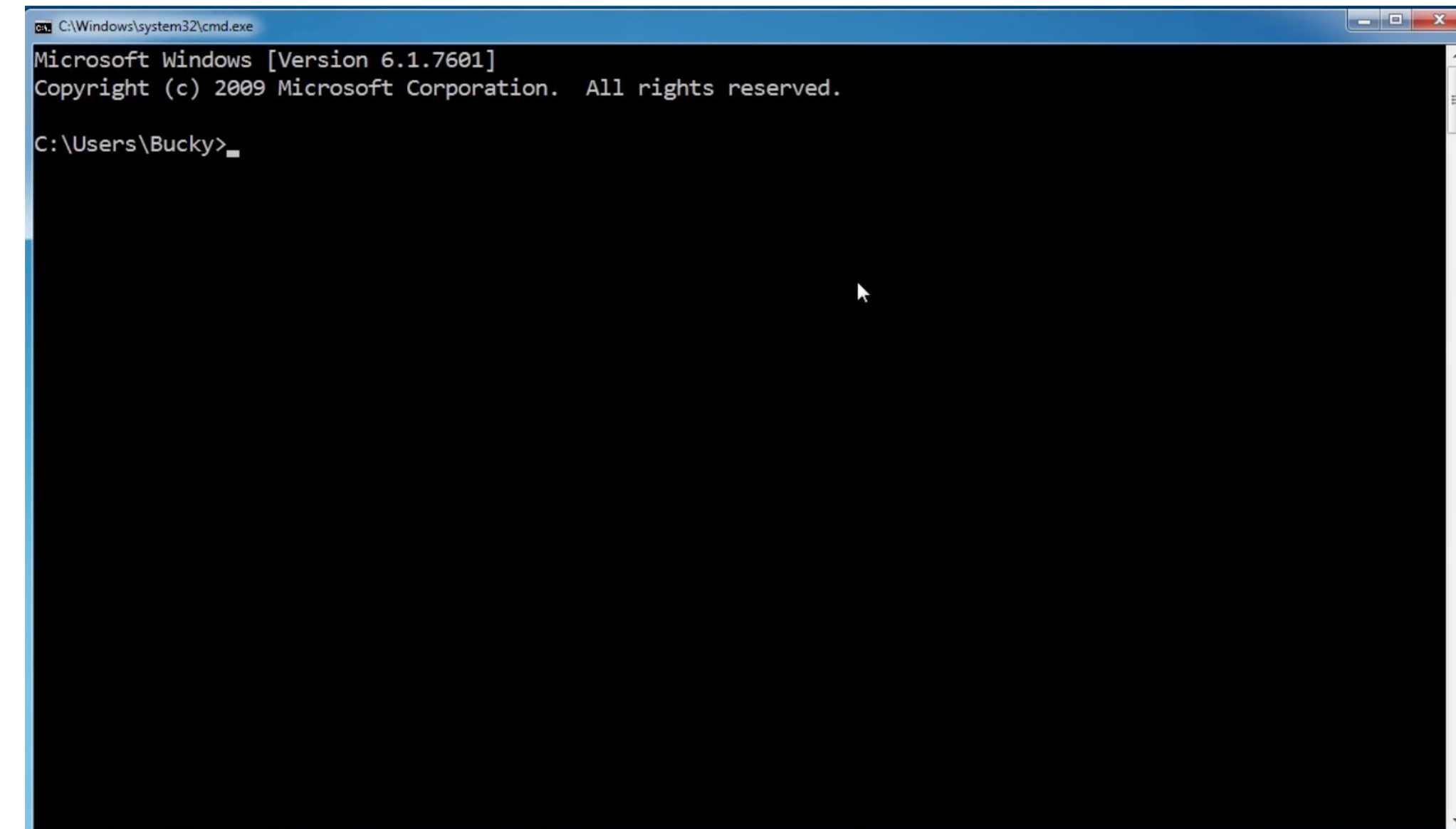
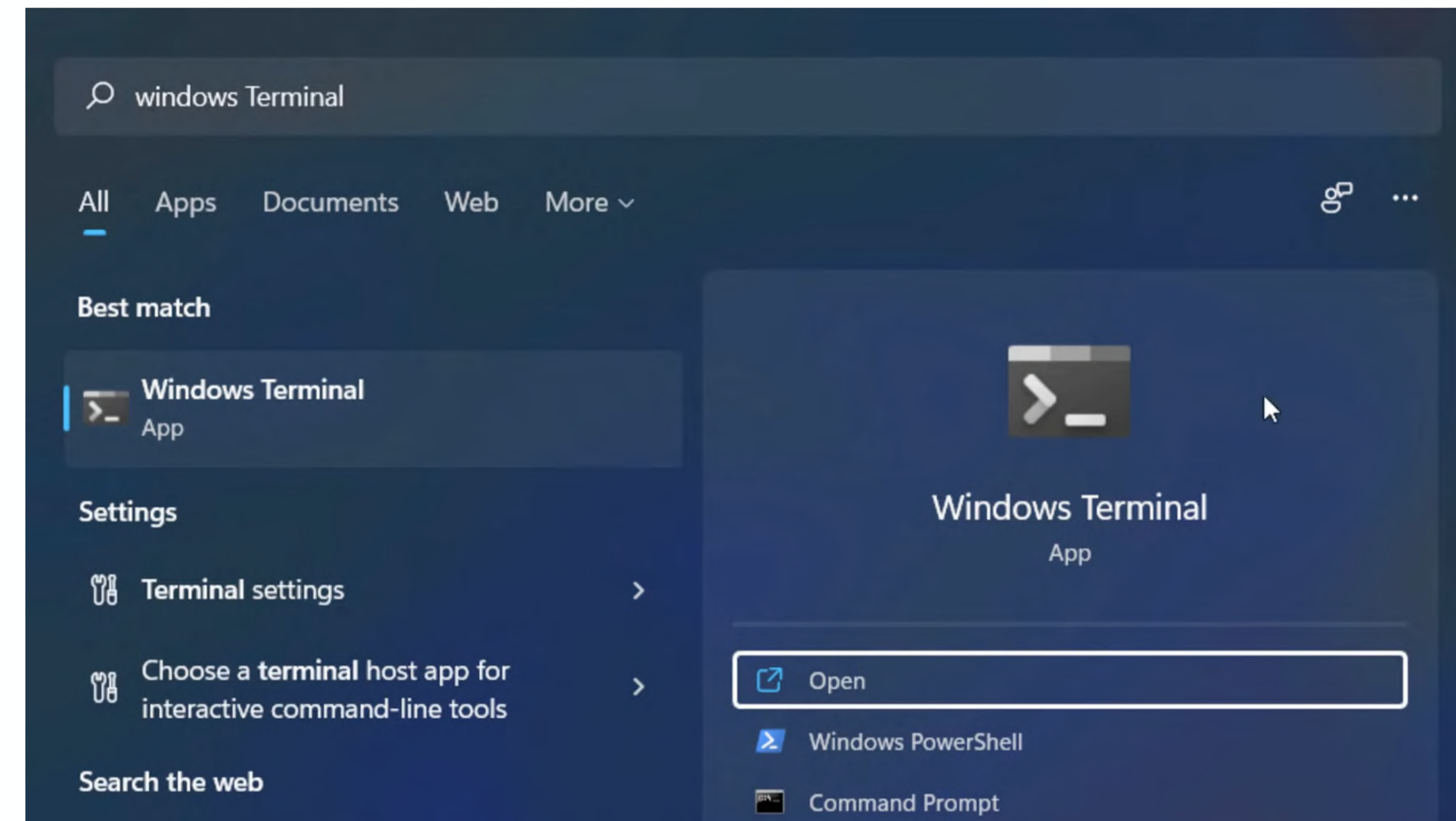
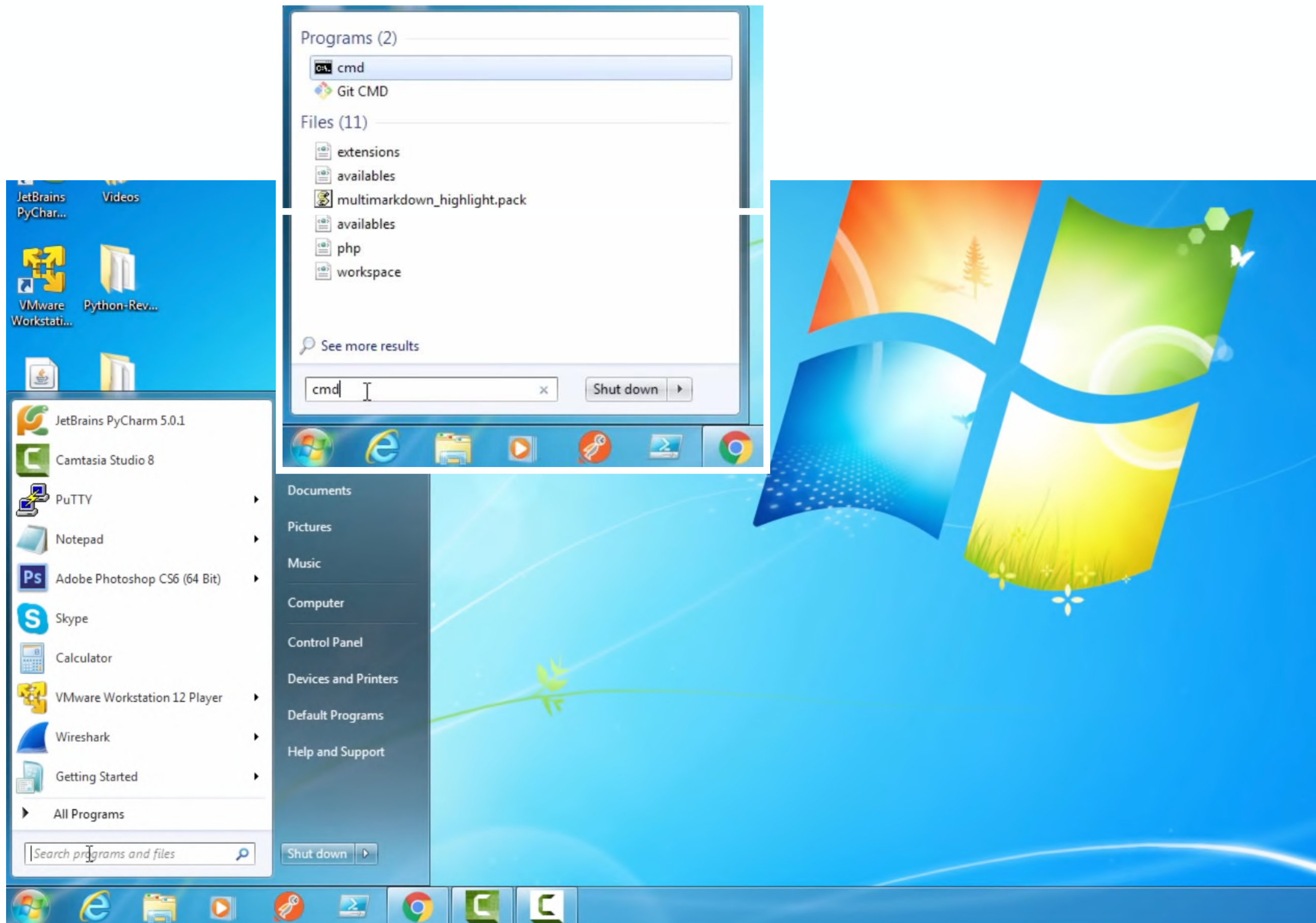
Look for the terminal among the applications or use the shortcut Ctrl+Alt+T



The command line tool

OS: Windows

Look for the command prompt or for the terminal among available programs and applications



About Unix

Vocabulary

UNIX usually refers to a specific OS developed at the end of 1960s

Unix is commonly used to refer to a class of OSes derived/developed from UNIX

Unix-like systems are OSes which behave close to Unix OSes (not fully compliant with UNIX specifics)

Linux is a family of Unix OSes.

Ubuntu is one among the several Linux distributions.

Why learning Unix pays off

- open-source
- several infrastructures are Unix/Linux-based
- supercomputers and machine for HPC are Unix-based
- allows you to better understand how an OS really works
- several available programming tools

The shell

A shell is a key software component of an OS that allows the user to interact with it via command line.

Unix provides several shells, which differ one another for their complexity, specifics of language scripting and peculiar features. Among available shells: Bourne shell (sh), Bourne Again shell (bash), C shell (csh), Korn shell (ksh), Z shell (Zsh)

The shell

A shell is a key software component of an OS that allows the user to interact with it via command line.

Unix provides several shells, which differ one another for their complexity, specifics of language scripting and peculiar features. Among available shells: Bourne shell (sh), Bourne Again shell (bash), C shell (csh), Korn shell (ksh), Z shell (Zsh)

The bash shell is common, quite flexible and provided as default in the majority of Linux distributions.

To verify that you're using bash

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo $SHELL  
/bin/bash  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

command

variable

Command line

Command, general form:

command [flags] [argument1] [argument2] ...

example:

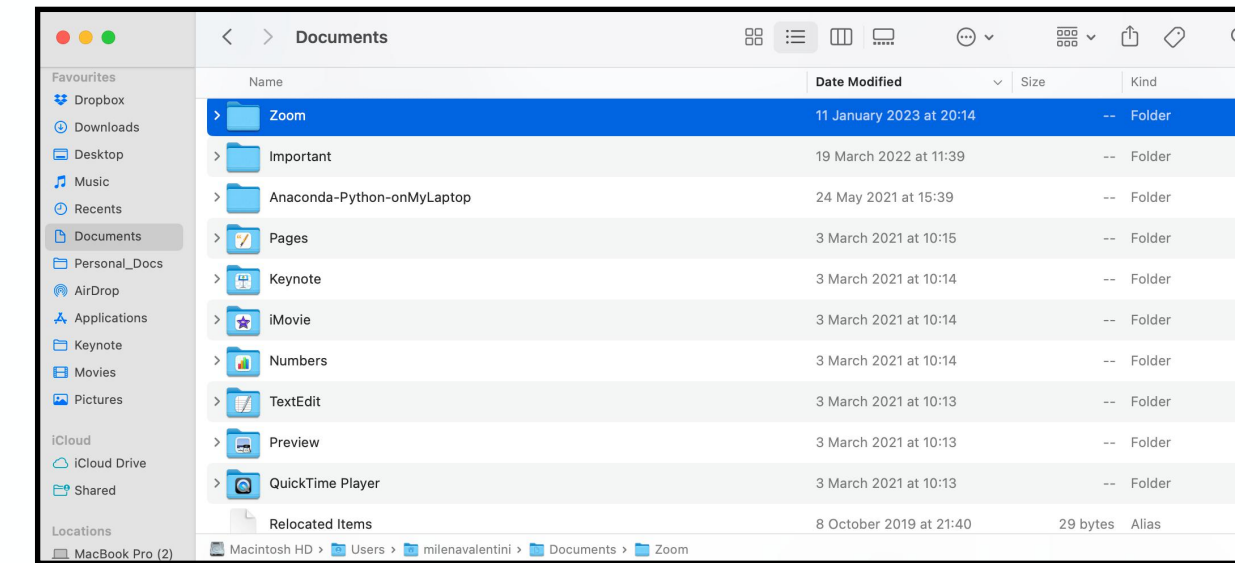
ls -l -a (or: ls -la)

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls  
file_1.txt      file_2.dat      script_1.py     script_2.bash  
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

Useful working tools

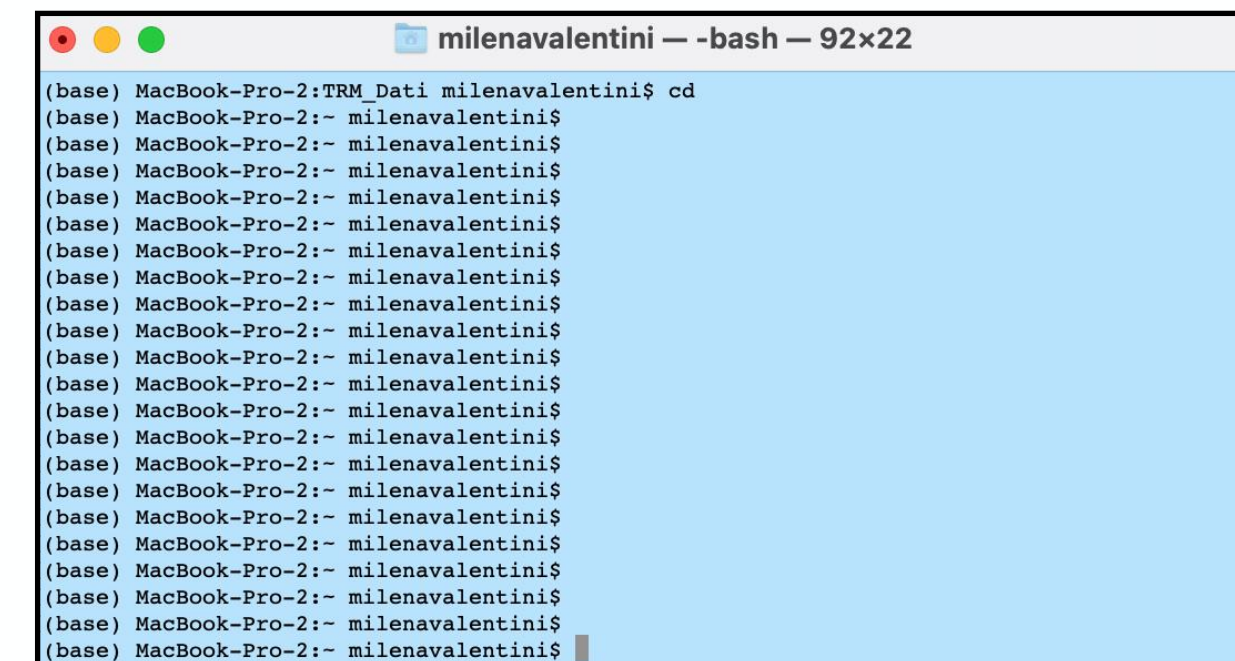
File browser or manager:

program of an Operative System (OS) which provides you with a user interface to manage folders and files



Shell/terminal/console/command line prompt:

interface to interact with the computer via command line without relying on graphical user interfaces



Text / code editor:

tool to edit code and file content

The text editor

vi: Visual Editor is the default editor that comes with the UNIX OS.

The vi editor is a full screen editor and has two modes of operation:

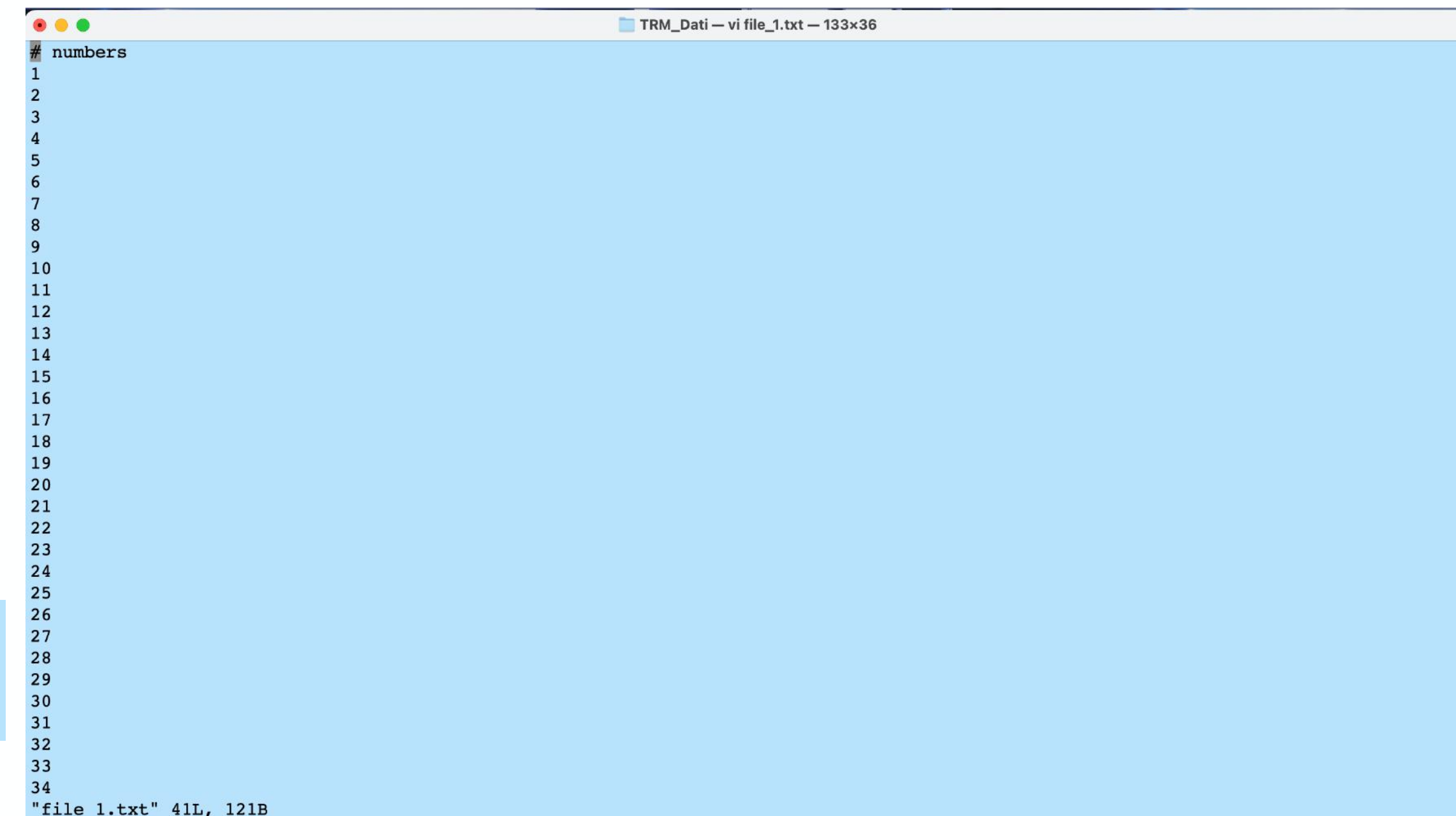
1. — *Command mode*: commands produce actions to be taken on the file, and
2. — *Insert mode*: entered text is written into the file.

In the command mode, every character typed is a command;
the *i* character typed in the command mode makes the vi editor enter the insert mode.

In the insert mode, typed characters are added to the text in the file.
Press the escape key to exit the insert mode.

Several websites where useful manuals can be used, e.g.:
<https://www.cs.colostate.edu/helpdocs/vi.html>
https://vimdoc.sourceforge.net/html/doc/usr_toc.html (vim)

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ vi file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



```
TRM_Dati - vi file_1.txt - 133x36  
# numbers  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
"file_1.txt" 41L, 121B
```


The text editor

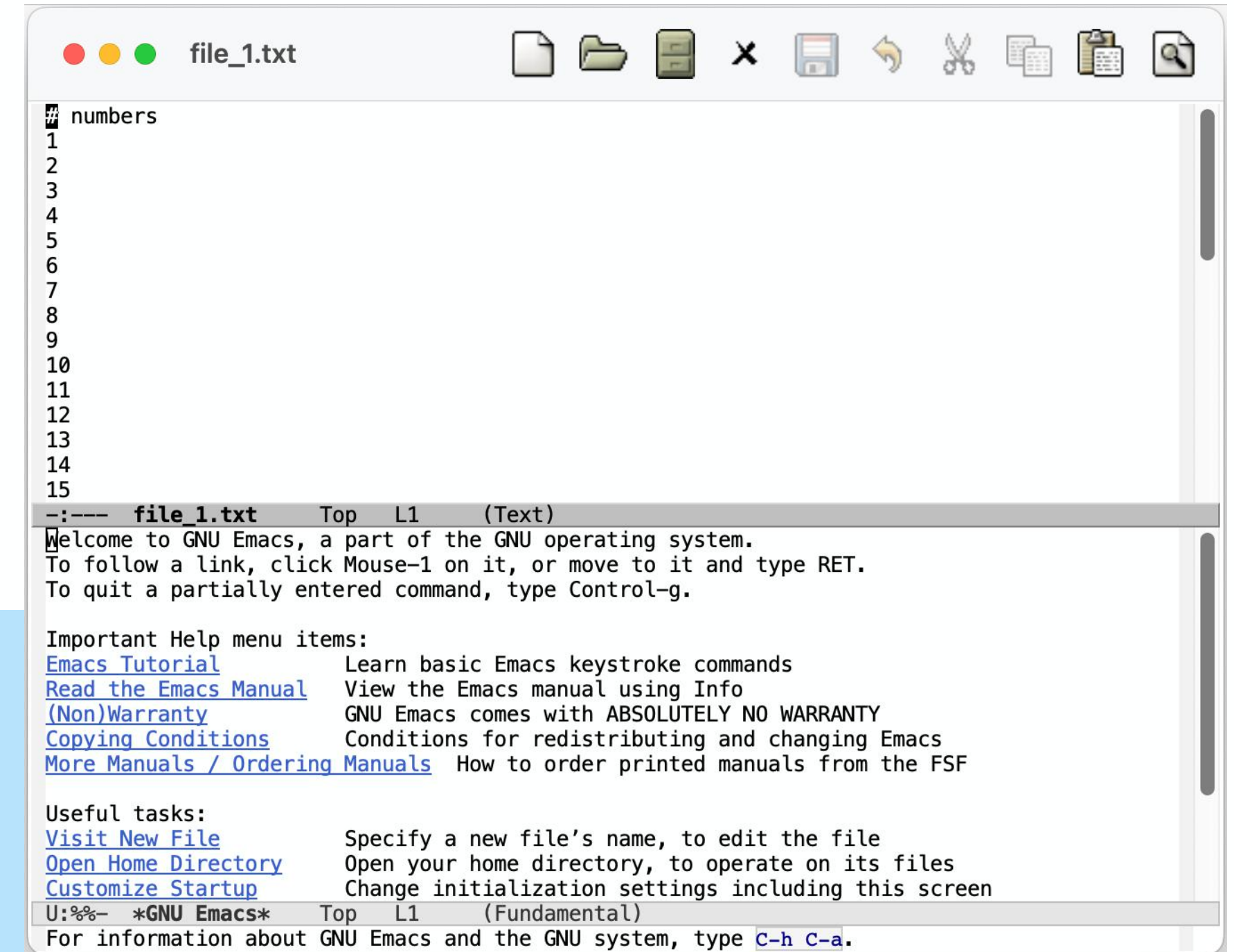
Emacs: it is the advanced, extensible, customizable, self-documenting editor by GNU.

You can follow the instructions to download and install it here: <https://www.gnu.org/software/emacs/download.html>

For instance, for users with a Mac OS:

```
[(base) MacBook-Pro-2:TRM_Dati milenaValentini$ sudo xcodebuild -license accept
[(base) MacBook-Pro-2:TRM_Dati milenaValentini$ brew install --cask emacs
Running `brew update --auto-update`...
==> Homebrew collects anonymous analytics.
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
No analytics have been recorded yet (nor will be during this `brew` run).

==> Downloading https://emacsformacosx.com/emacs-builds/Emacs-29.1-1-universal.dmg
==> Downloading from https://emacsformacosx.com/download/emacs-builds/Emacs-29.1-1-universal.dmg
##### 100.0%
==> Installing Cask emacs
==> Moving App 'Emacs.app' to '/Applications/Emacs.app'
==> Linking Binary 'Emacs' to '/opt/homebrew/bin/emacs'
==> Linking Binary 'ctags' to '/opt/homebrew/bin/ctags'
==> Linking Binary 'ebrowse' to '/opt/homebrew/bin/ebrowse'
==> Linking Binary 'emacsclient' to '/opt/homebrew/bin/emacsclient'
==> Linking Binary 'etags' to '/opt/homebrew/bin/etags'
==> Linking Manpage 'ctags.1.gz' to '/opt/homebrew/share/man/man1/ctags.1.gz'
==> Linking Manpage 'ebrowse.1.gz' to '/opt/homebrew/share/man/man1/ebrowse.1.gz'
==> Linking Manpage 'emacs.1.gz' to '/opt/homebrew/share/man/man1/emacs.1.gz'
==> Linking Manpage 'emacsclient.1.gz' to '/opt/homebrew/share/man/man1/emacsclient.1.gz'
==> Linking Manpage 'etags.1.gz' to '/opt/homebrew/share/man/man1/etags.1.gz'
📦 emacs was successfully installed!
[(base) MacBook-Pro-2:TRM_Dati milenaValentini$ emacs file_1.txt
```



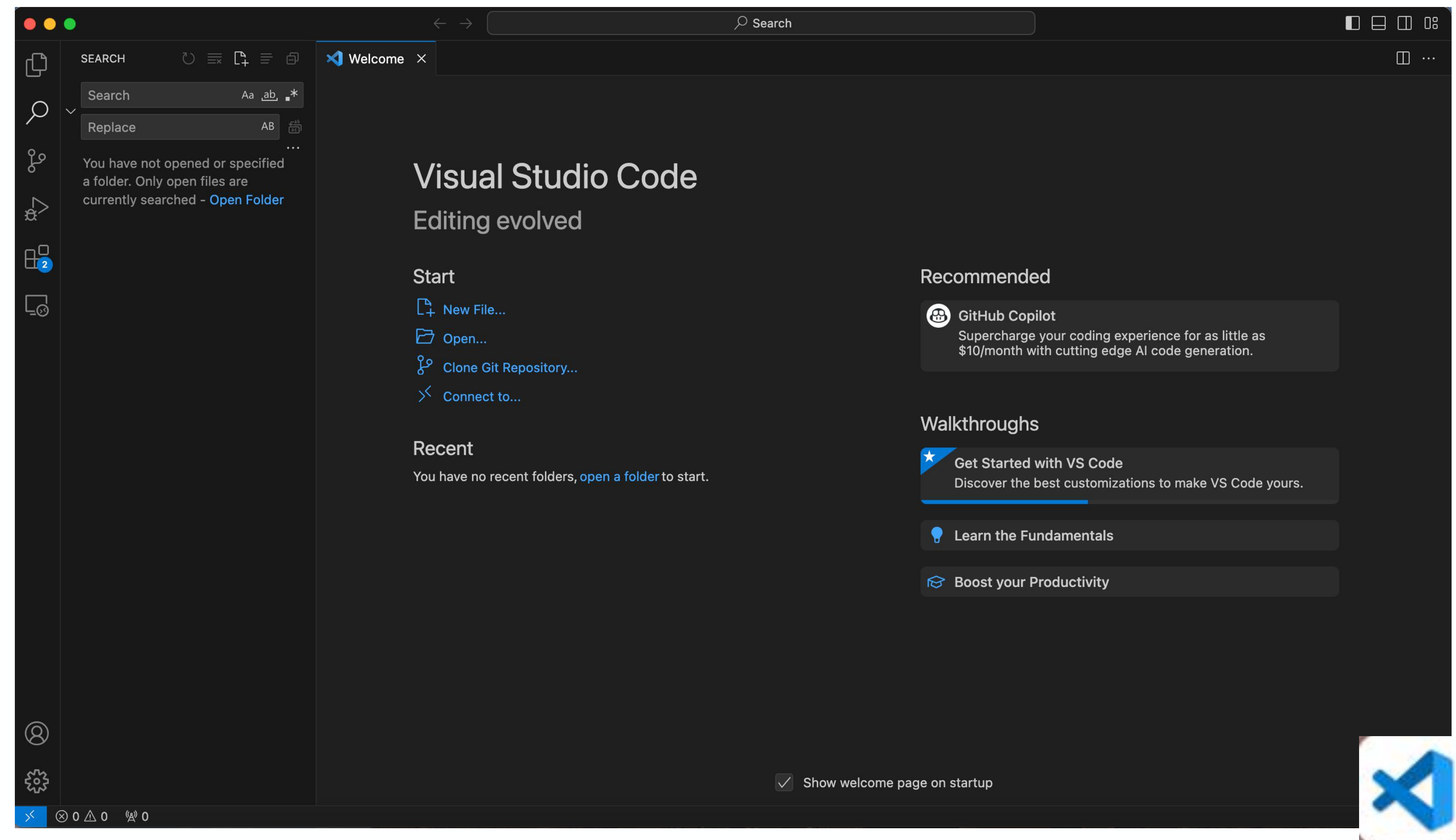
How to use emacs: https://www.gnu.org/software/emacs/manual/html_node/emacs/index.html

The text editor

Visual Studio Code is a powerful source code editor which runs on your desktop.

It is available for Windows, macOS and Linux <https://code.visualstudio.com/docs/?dv=osx>

It comes with built-in support for e.g. JavaScript and has several extensions for other languages and runtimes (such as C++, Java, Python...)



Setting up the working environment

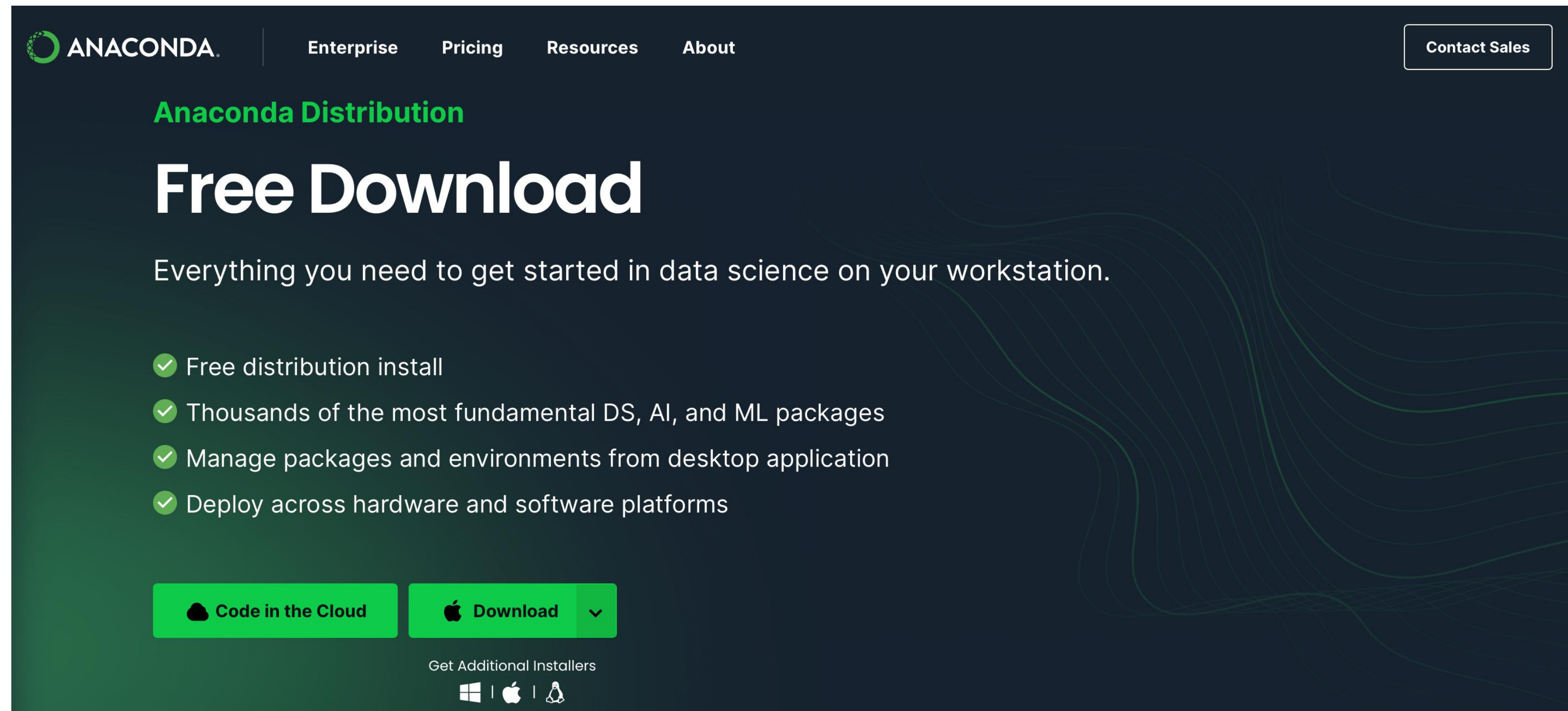
Anaconda is an open-source package and environment management system that runs on Windows, macOS, and Linux.

Conda quickly installs, runs, and updates packages and their dependencies. It also easily creates, saves, loads, and switches between environments on your local computer.

It was created for Python programs, but it can package and distribute software for any language.

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

Download, install it and make sure your \$PATH environment variable is updated to include Anaconda

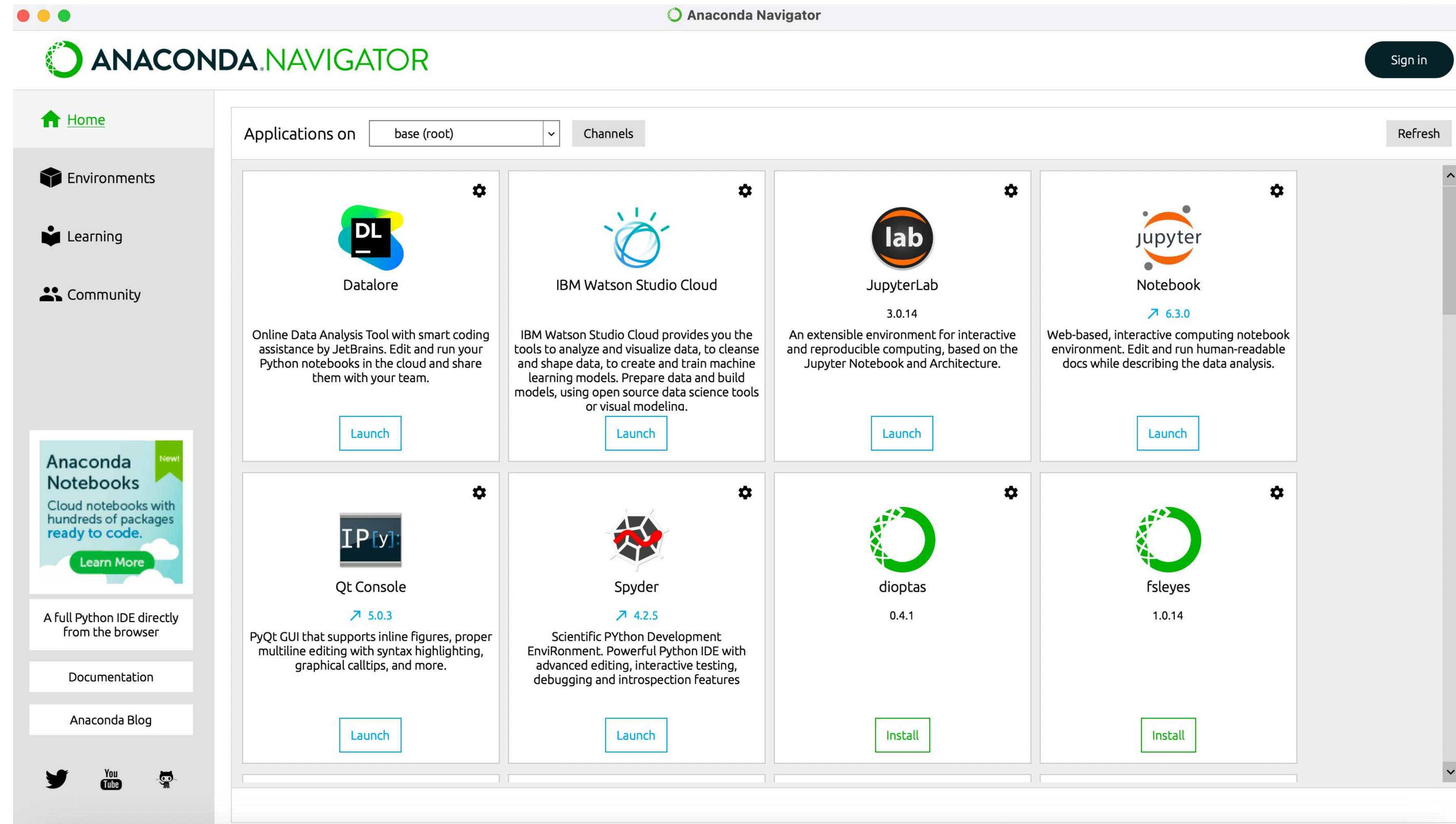


The screenshot shows the Anaconda website's download page. The header includes the Anaconda logo, navigation links for Enterprise, Pricing, Resources, and About, and a Contact Sales button. The main heading is "Anaconda Distribution" followed by "Free Download" in large white text. Below this, a sub-heading reads "Everything you need to get started in data science on your workstation." A list of four benefits is provided, each with a green checkmark: "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 the buttons, it says "Get Additional Installers" with icons for Windows, macOS, and Linux.

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

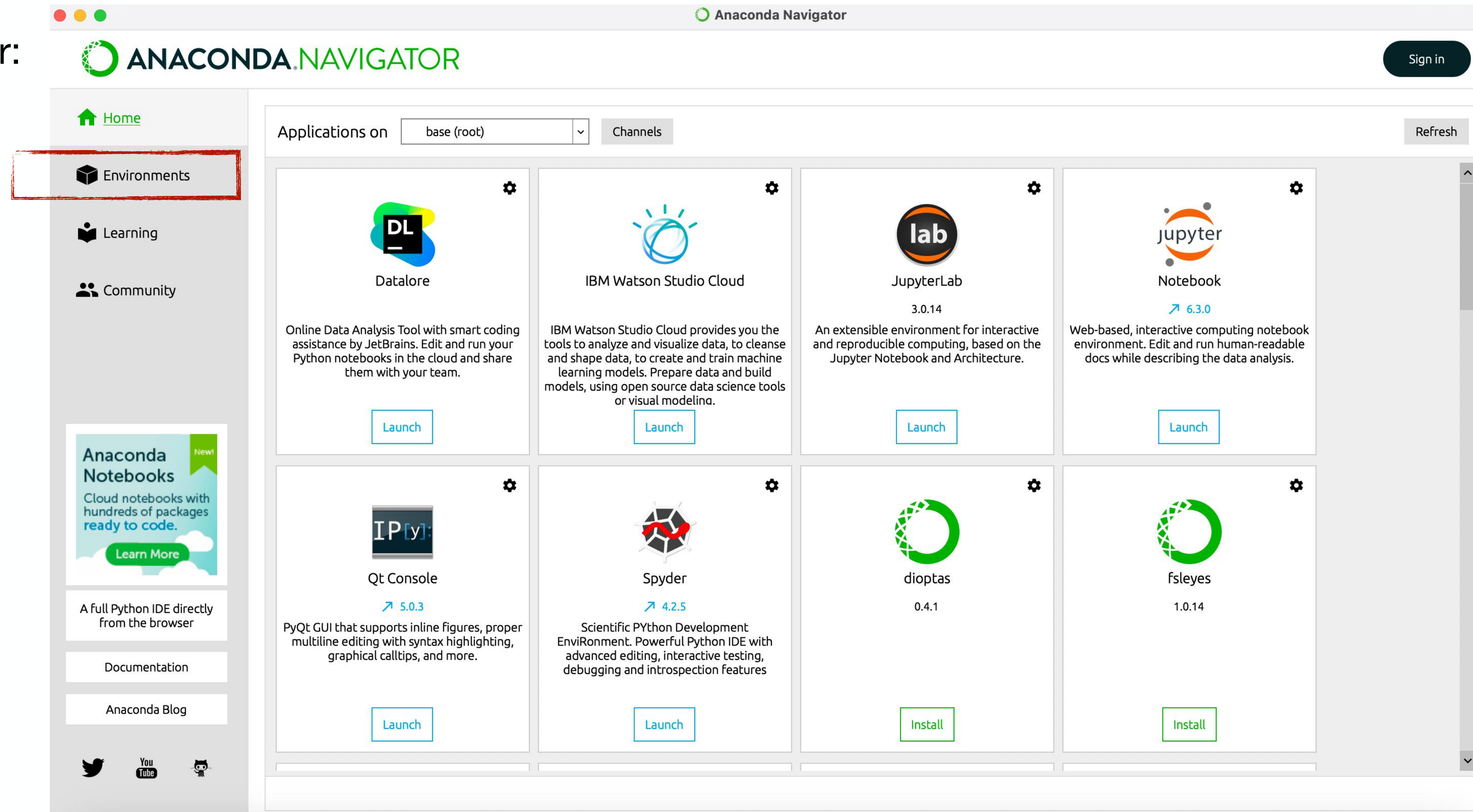


Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:



Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:

Create a new environment:

The screenshot displays the Anaconda Navigator application window. The top bar includes the Anaconda Navigator logo and a 'Sign in' button. The left sidebar contains navigation options: Home, Environments, Learning, and Community. The main area is divided into two sections. The left section, titled 'Search Environments', lists existing environments: 'base (root)', 'Jupyter_', 'spyder_', and 'spyder_'. A red box highlights this list, and the text 'Already existing environments' is overlaid below it. The right section, titled 'Installed', shows a list of installed packages with columns for Name, Description, and Version. A red box highlights the 'Create' button in the bottom left corner of the interface.

Name	Description	Version
✓ _ipyw_jlab_nb_ex...	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
✓ alabaster	Configurable, python 2+3 compatible sphinx theme.	0.7.12
✓ anaconda	Simplifies package management and deployment of anaconda	2021.05
✓ anaconda-client	Anaconda cloud command line client library	1.7.2
✓ anaconda-project	Tool for encapsulating, running, and reproducing data science projects	0.9.1
✓ anyio	High level compatibility layer for multiple asynchronous event loop implementations on python	2.2.0
✓ appdirs	A small python module for determining appropriate platform-specific dirs.	1.4.4
✓ applaunchservices	Simple package for registering an app with apple launch services to handle uti and url	0.2.1
✓ appnope	Disable app nap on os x 10.9	0.1.2
✓ appscript	Control applescriptable applications from python.	1.1.2
✓ argh	The natural cli.	0.26.2
✓ argon2-cffi	The secure argon2 password hashing algorithm.	20.1.0
✓ asn1crypto	Python asn.1 library with a focus on performance and a pythonic api	1.4.0
✓ astroid	A abstract syntax tree for python with inference support.	2.5
✓ astropy	Community-developed python library for astronomy	4.2.1

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:

Create a new environment:

The screenshot displays the Anaconda Navigator application window. The main interface is divided into a left sidebar with navigation options (Home, Environments, Learning, Community) and a main content area. The 'Environments' tab is active, showing a list of environments: 'base (root)', 'Jupyter_', 'spyder_', and another 'spyder_'. A 'Create new environment' dialog box is open in the foreground, with the following details:

- Name: JupyterNotebook_test
- Location: /Users/milenaValentini/opt/anaconda3/envs/JupyterNotebook_test
- Packages: Python 3.8, R

The background interface shows a table of installed packages with columns for Name, Description, and Version. The 'Create' button at the bottom left of the environment list is highlighted with a red box.

Name	Description	Version
base (root)		
Jupyter_		
spyder_		
spyder_		
base (root)		
_ipyw_jlab_nb_ex...	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
alabaster	Configurable, python 2+3 compatible sphinx theme.	0.7.12
anaconda-client	Client for the Anaconda ecosystem	2021.05
anaconda-core	Core components of anaconda	1.7.2
anaconda-lstic	Integrating data science projects	0.9.1
anaconda-nb-extensions	Asynchronous event loop implementations on python	2.2.0
anaconda-nb-frontend	Appropriate platform-specific dirs.	1.4.4
anaconda-nb-view	Apple launch services to handle uti and url	0.2.1
anaconda-nb-widgets		0.1.2
appnope	Disable app nap on os x 10.9	1.1.2
appscript	Control applescriptable applications from python.	0.26.2
argh	The natural cli.	20.1.0
argon2-cffi	The secure argon2 password hashing algorithm.	1.4.0
asn1crypto	Python asn.1 library with a focus on performance and a pythonic api	2.5
astroid	A abstract syntax tree for python with inference support.	2.5
astropy	Community-developed python library for astronomy	4.2.1

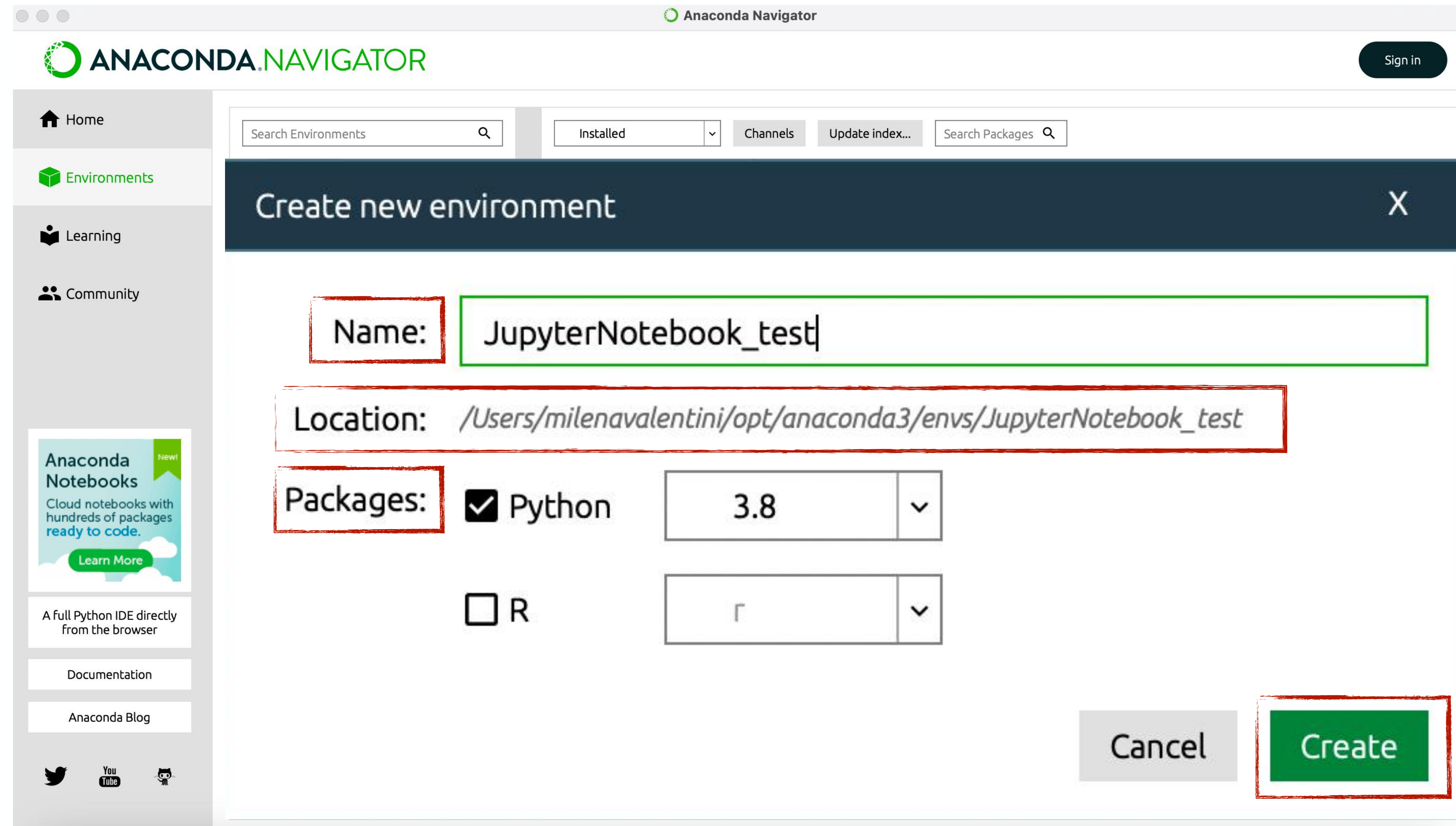
Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:

Create a new environment:



Setting up the working environment with Anaconda

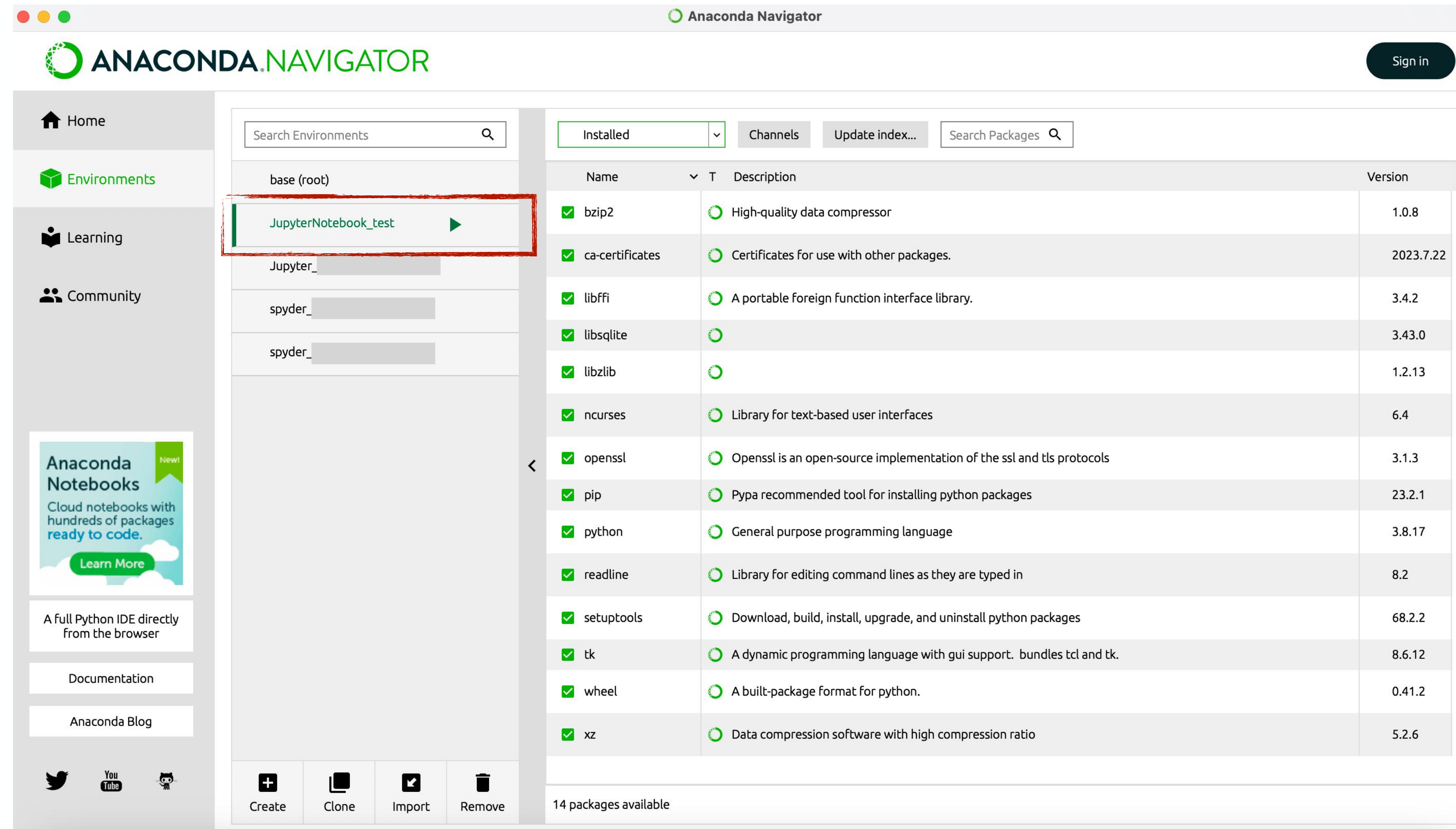
To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:

Create a new environment:

Here is the new environment:



The screenshot shows the Anaconda Navigator application window. The 'Environments' tab is active, displaying a list of environments. A new environment named 'JupyterNotebook_test' is highlighted with a red box. Below the environment list, a table shows the installed packages for this environment.

Name	Description	Version
✓ bzip2	High-quality data compressor	1.0.8
✓ ca-certificates	Certificates for use with other packages.	2023.7.22
✓ libffi	A portable foreign function interface library.	3.4.2
✓ libsqlite		3.43.0
✓ libzlib		1.2.13
✓ ncurses	Library for text-based user interfaces	6.4
✓ openssl	Openssl is an open-source implementation of the ssl and tls protocols	3.1.3
✓ pip	Pypa recommended tool for installing python packages	23.2.1
✓ python	General purpose programming language	3.8.17
✓ readline	Library for editing command lines as they are typed in	8.2
✓ setuptools	Download, build, install, upgrade, and uninstall python packages	68.2.2
✓ tk	A dynamic programming language with gui support. bundles tcl and tk.	8.6.12
✓ wheel	A built-package format for python.	0.41.2
✓ xz	Data compression software with high compression ratio	5.2.6

14 packages available

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

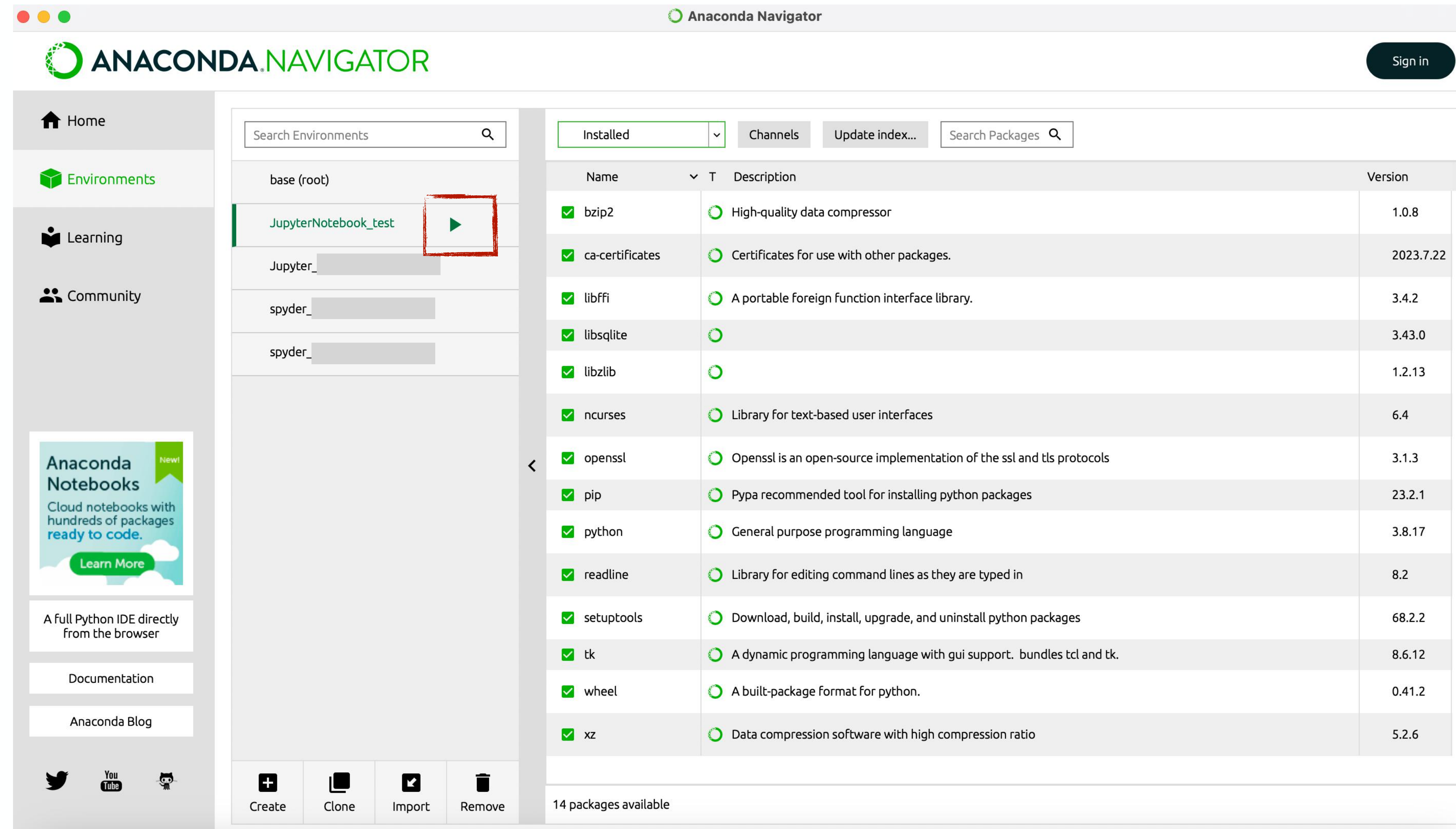
Launch Anaconda-Navigator:

Select Environments:

Create a new environment:

Here is the new environment:

The green arrow tells you that the new environment is active



The screenshot shows the Anaconda Navigator interface. The 'Environments' tab is selected in the left sidebar. The main panel displays a list of environments: 'base (root)', 'JupyterNotebook_test', 'Jupyter_', 'spyder_', and 'spyder_'. The 'JupyterNotebook_test' environment is highlighted with a green bar, and a green play button is visible next to it, indicating it is the active environment. Below the environment list, there are buttons for 'Create', 'Clone', 'Import', and 'Remove'. On the right side, the 'Installed' tab is selected, showing a list of installed packages with their names, descriptions, and versions.

Name	Description	Version
✓ bzip2	High-quality data compressor	1.0.8
✓ ca-certificates	Certificates for use with other packages.	2023.7.22
✓ libffi	A portable foreign function interface library.	3.4.2
✓ libsqlite		3.43.0
✓ libzlib		1.2.13
✓ ncurses	Library for text-based user interfaces	6.4
✓ openssl	Openssl is an open-source implementation of the ssl and tls protocols	3.1.3
✓ pip	Pypa recommended tool for installing python packages	23.2.1
✓ python	General purpose programming language	3.8.17
✓ readline	Library for editing command lines as they are typed in	8.2
✓ setuptools	Download, build, install, upgrade, and uninstall python packages	68.2.2
✓ tk	A dynamic programming language with gui support. bundles tcl and tk.	8.6.12
✓ wheel	A built-package format for python.	0.41.2
✓ xz	Data compression software with high compression ratio	5.2.6

14 packages available

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

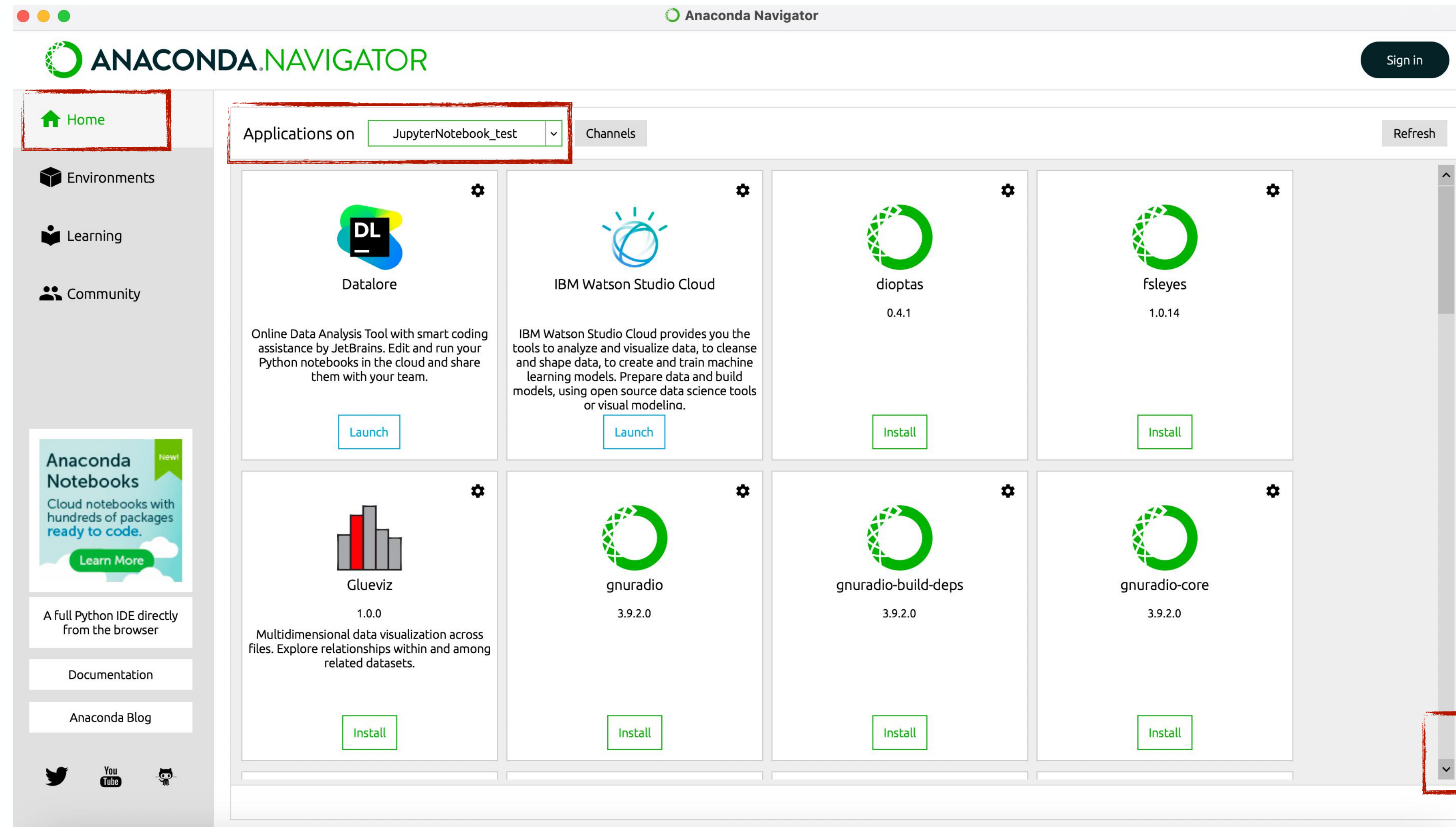
Select Environments:

Create a new environment:

Here is the new environment:

The green arrow tells you that the new environment is active

Select the applications to be installed in the environment among available ones



Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

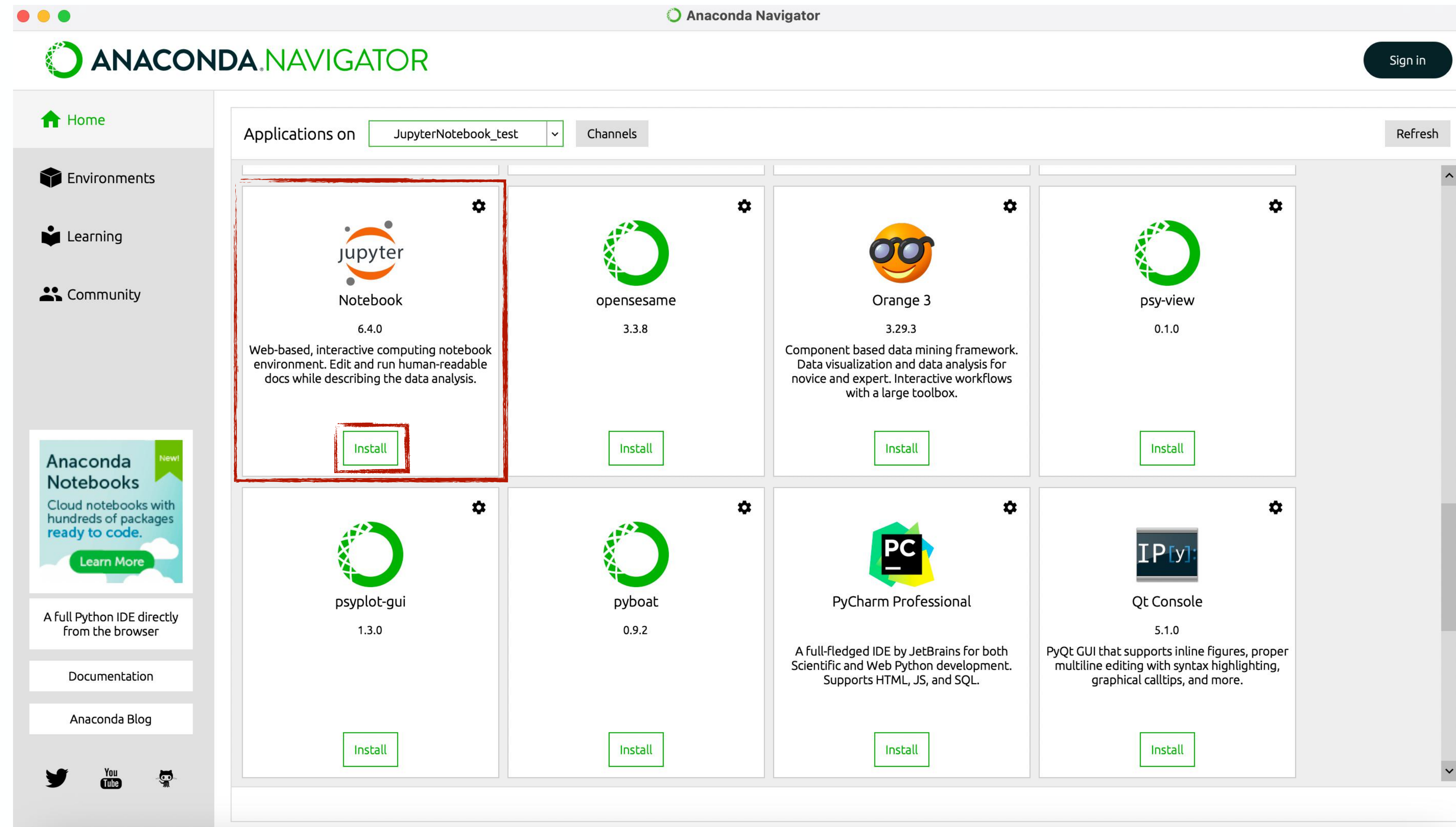
Select Environments:

Create a new environment:

Here is the new environment:

The green arrow tells you that the new environment is active

Select the applications to be installed in the environment among available ones



Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

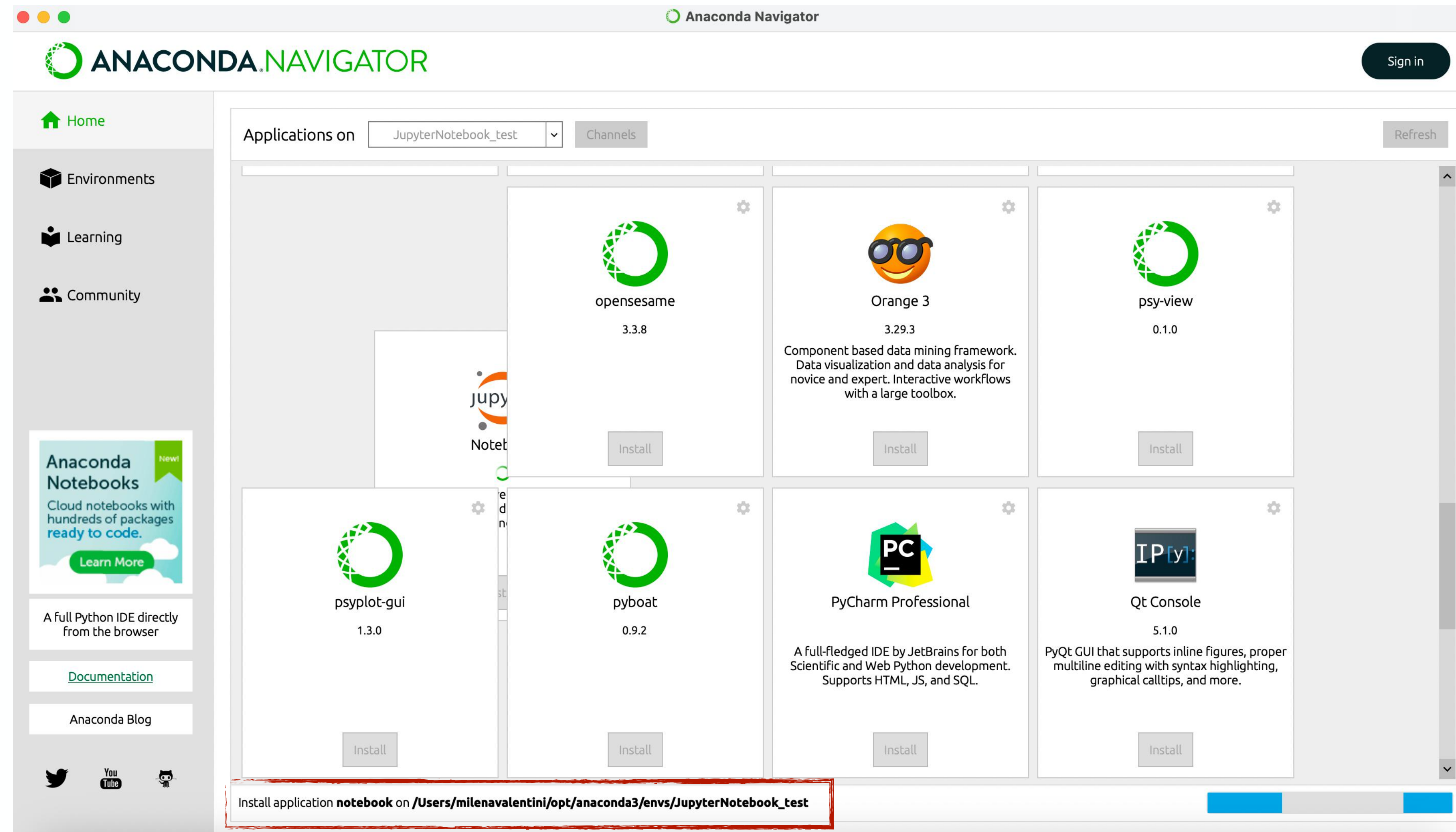
Select Environments:

Create a new environment:

Here is the new environment:

The green arrow tells you that the new environment is active

Select the applications to be installed in the environment among available ones



Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:

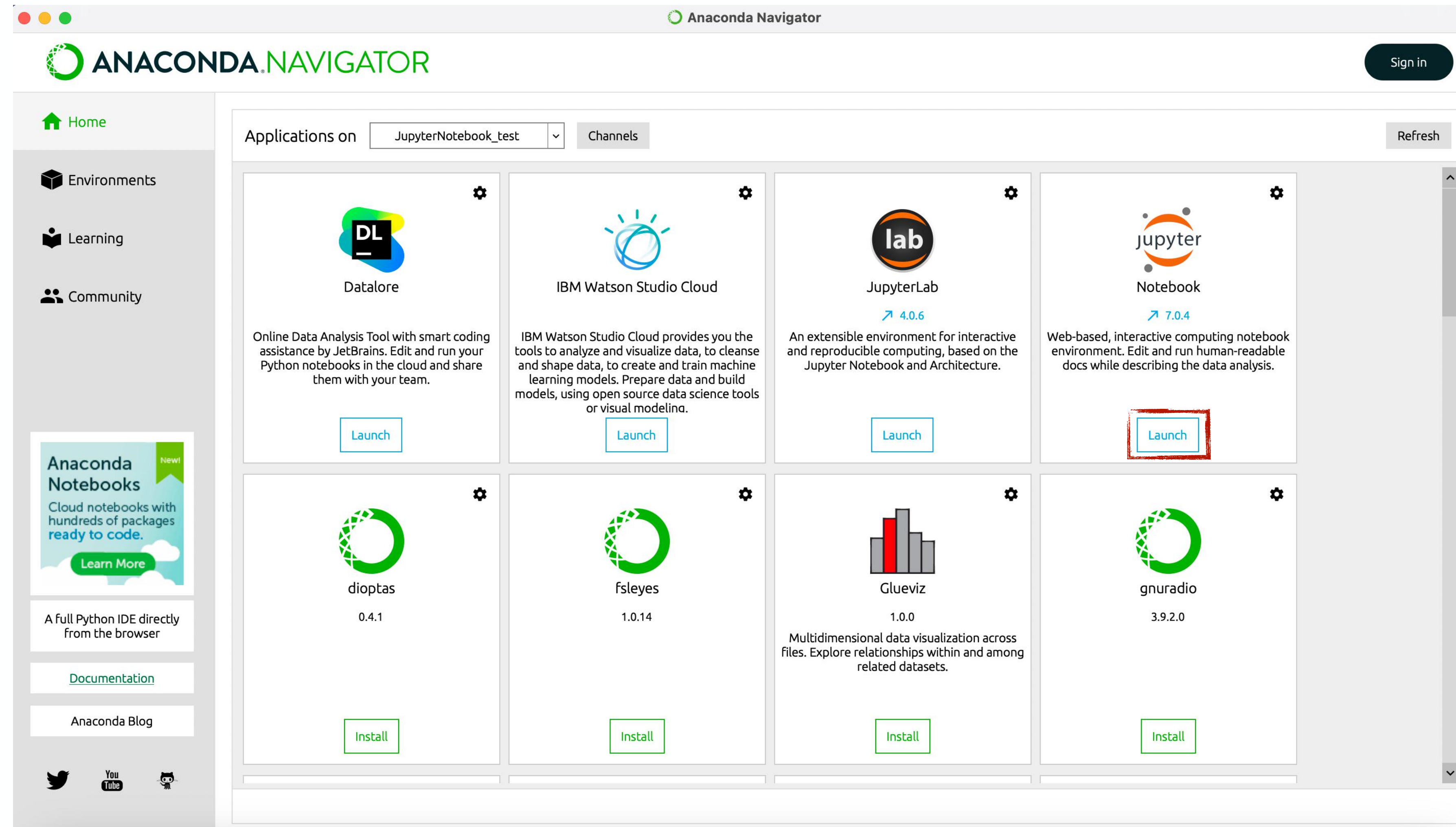
Create a new environment:

Here is the new environment:

The green arrow tells you that the new environment is active

Select the applications to be installed in the environment

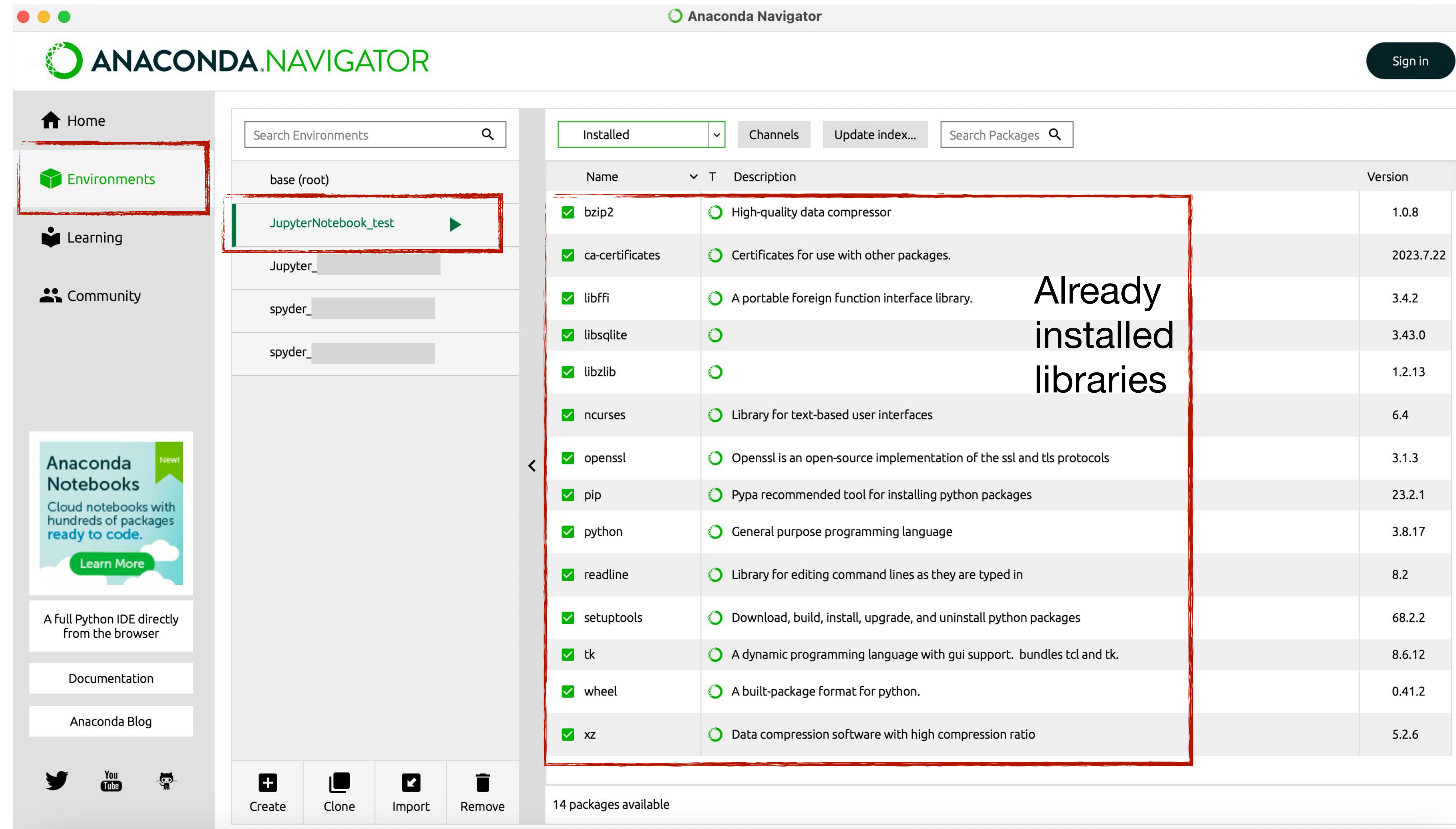
The application has just been installed and can be launched



Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

To install libraries
(instead of applications)
within a given environment:



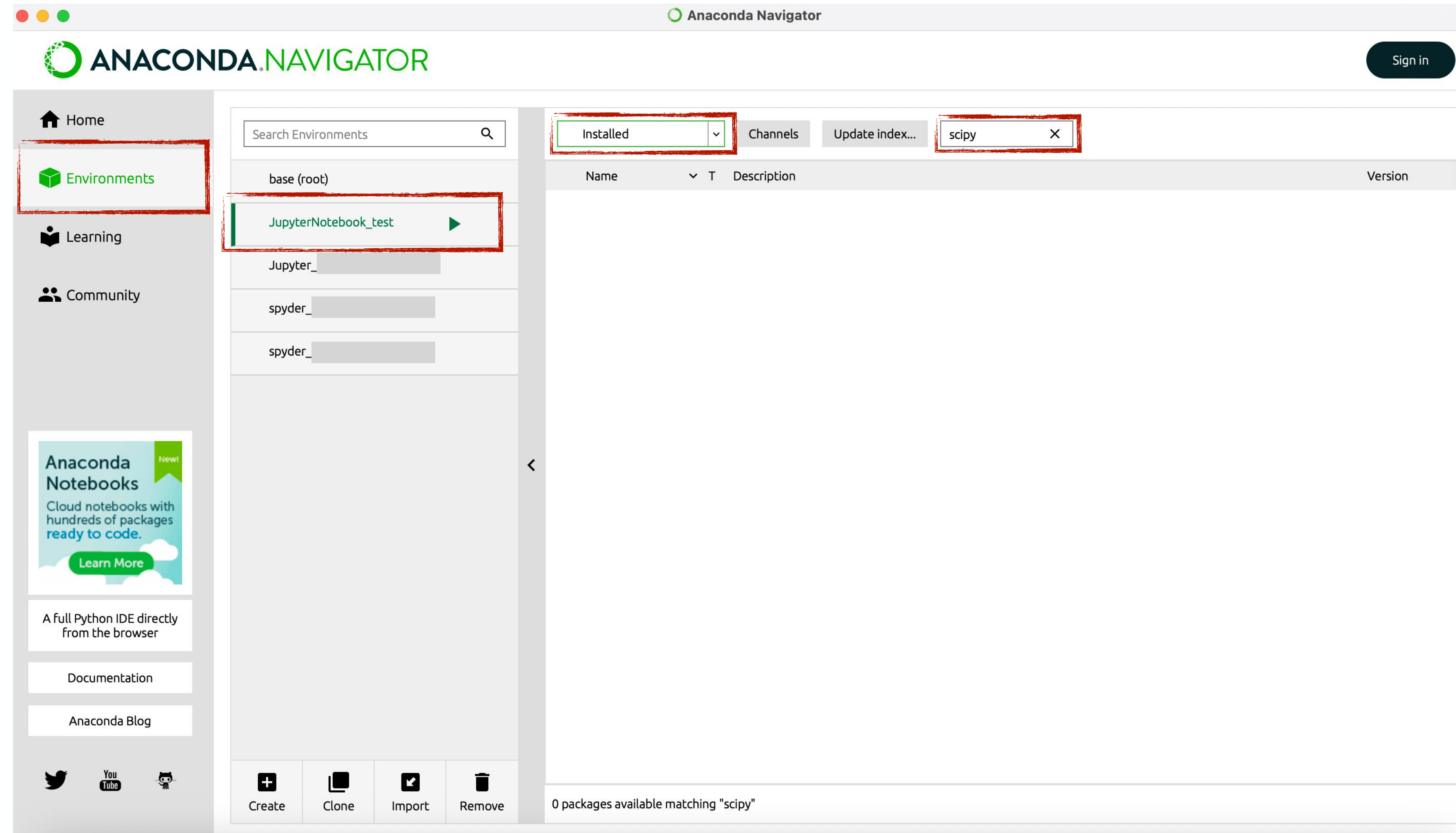
The screenshot shows the Anaconda Navigator interface. On the left sidebar, the 'Environments' tab is highlighted with a red box. In the center, the 'JupyterNotebook_test' environment is selected, also highlighted with a red box. On the right, a table lists installed packages, with a red box around it and the text 'Already installed libraries' overlaid.

Name	Description	Version
✓ bzip2	High-quality data compressor	1.0.8
✓ ca-certificates	Certificates for use with other packages.	2023.7.22
✓ libffi	A portable foreign function interface library.	3.4.2
✓ libsqlite		3.43.0
✓ libzlib		1.2.13
✓ ncurses	Library for text-based user interfaces	6.4
✓ openssl	Openssl is an open-source implementation of the ssl and tls protocols	3.1.3
✓ pip	Pypa recommended tool for installing python packages	23.2.1
✓ python	General purpose programming language	3.8.17
✓ readline	Library for editing command lines as they are typed in	8.2
✓ setuptools	Download, build, install, upgrade, and uninstall python packages	68.2.2
✓ tk	A dynamic programming language with gui support. bundles tcl and tk.	8.6.12
✓ wheel	A built-package format for python.	0.41.2
✓ xz	Data compression software with high compression ratio	5.2.6

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

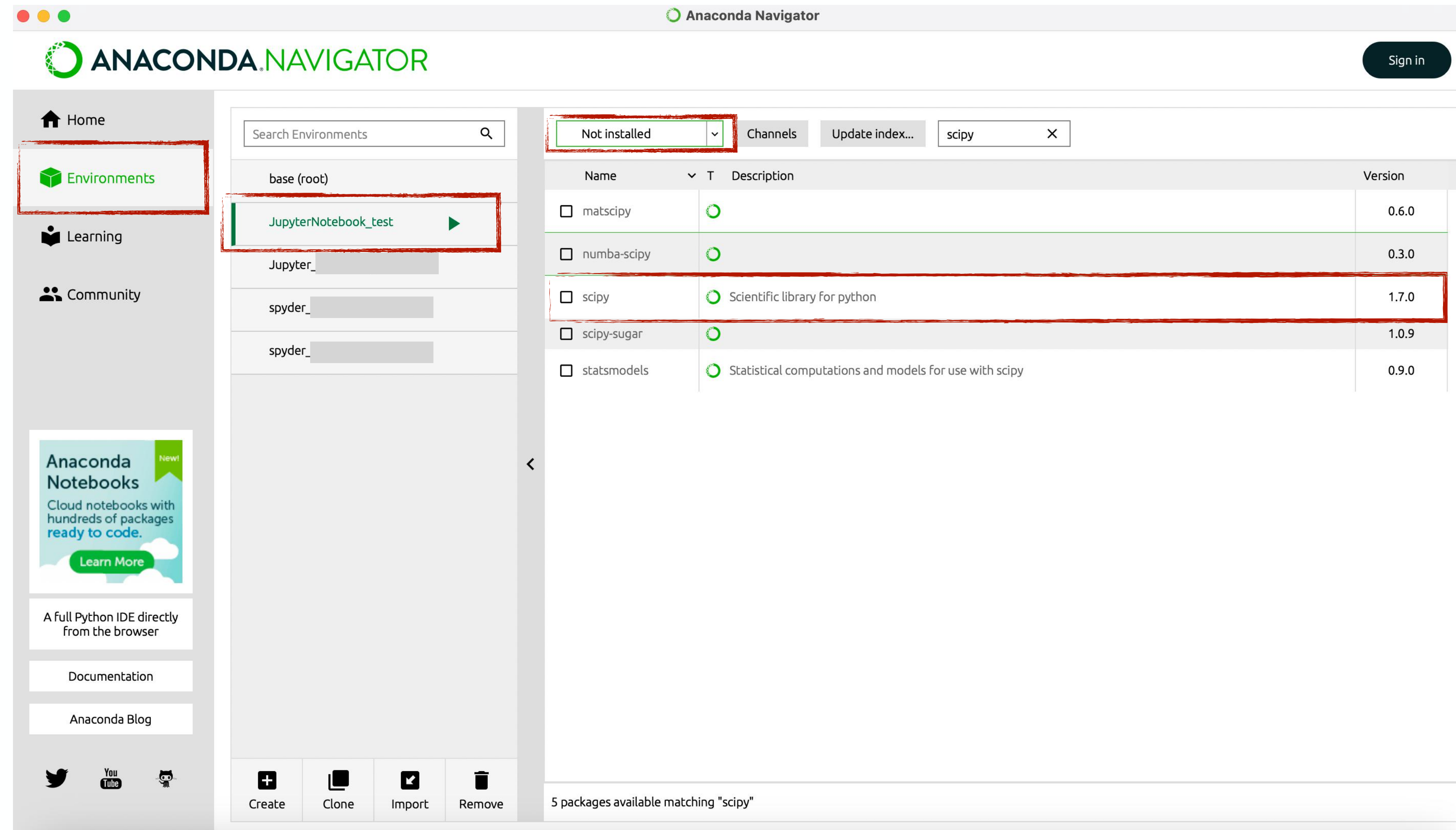
To install libraries
(instead of applications)
within a given environment:



Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

To install libraries
(instead of applications)
within a given environment:



The screenshot shows the Anaconda Navigator interface. The 'Environments' tab is selected in the left sidebar. The main panel displays a list of environments, with 'JupyterNotebook_test' highlighted. The right panel shows a search for 'scipy' in the 'Not installed' state. The search results table is as follows:

Name	T	Description	Version
<input type="checkbox"/> matscipy	○		0.6.0
<input type="checkbox"/> numba-scipy	○		0.3.0
<input type="checkbox"/> scipy	○	Scientific library for python	1.7.0
<input type="checkbox"/> scipy-sugar	○		1.0.9
<input type="checkbox"/> statsmodels	○	Statistical computations and models for use with scipy	0.9.0

At the bottom of the right panel, it states: "5 packages available matching 'scipy'".

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

To install libraries
(instead of applications)
within a given environment:

Select the library to be
installed in the environment

The screenshot displays the Anaconda Navigator application window. On the left sidebar, the 'Environments' tab is highlighted with a red box. The main panel shows a list of environments, with 'JupyterNotebook_test' selected and highlighted by a red box. To the right, a search for 'scipy' is performed, showing a list of packages. The 'scipy' package is selected with a green checkmark. At the bottom right, the 'Apply' button is highlighted with a red box, indicating the installation process.

Name	Description	Version
<input type="checkbox"/> matscipy		0.6.0
<input type="checkbox"/> numba-scipy		0.3.0
<input checked="" type="checkbox"/> scipy	Scientific library for python	1.7.0
<input type="checkbox"/> scipy-sugar		1.0.9
<input type="checkbox"/> statsmodels	Statistical computations and models for use with scipy	0.9.0

5 packages available matching "scipy" 1 package selected

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

To install libraries
(instead of applications)
within a given environment:

Select the library to be
installed in the environment

The screenshot shows the Anaconda Navigator GUI. On the left sidebar, the 'Environments' tab is highlighted with a red box. The main panel displays a list of environments: 'base (root)', 'JupyterNotebook_test' (highlighted with a red box), 'Jupyter_', 'spyder_', and another 'spyder_'. An 'Install Packages' dialog box is open, showing a search for 'scipy'. The dialog lists the following packages to be modified:

Name	Description	Version
		0.6.0
		0.3.0
		1.7.0
		1.0.9
		0.9.0

At the bottom of the dialog, it says 'Solving package specifications' with a progress bar. The 'Apply' button is highlighted with a red box. At the bottom of the main interface, a status bar shows '5 packages available matching "scipy" 1 package selected' and another 'Apply' button is highlighted with a red box.

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

To install libraries
(instead of applications)
within a given environment:

Select the library to be
installed in the environment

The screenshot shows the Anaconda Navigator application window. The 'Environments' tab is selected in the left sidebar. The main panel displays a list of environments, with 'JupyterNotebook_test' highlighted. An 'Install Packages' dialog box is open, showing a list of 10 packages to be installed. The 'Apply' button in the dialog is highlighted with a red box. At the bottom right of the main panel, there is another 'Apply' button, also highlighted with a red box.

Search Environments

base (root)

JupyterNotebook_test

Jupyter_

spyder_

spyder_

Install Packages

10 packages will be installed

Name	Unlink	Link	Channel	Action
*pooch	-	1.7.0	conda-forge	Installed
*numpy	-	1.24.4	conda-forge	Installed
*lvm-openmp	-	16.0.6	conda-forge	Installed
*libopenblas	-	0.3.24	conda-forge	Installed
*liblapack	-	3.9.0	conda-forge	Installed
*libgfortran5	-	13.2.0	conda-forge	Installed

* indicates the package is a dependency of a selected package

Cancel Apply

5 packages available matching "scipy" 1 package selected

Apply Clear

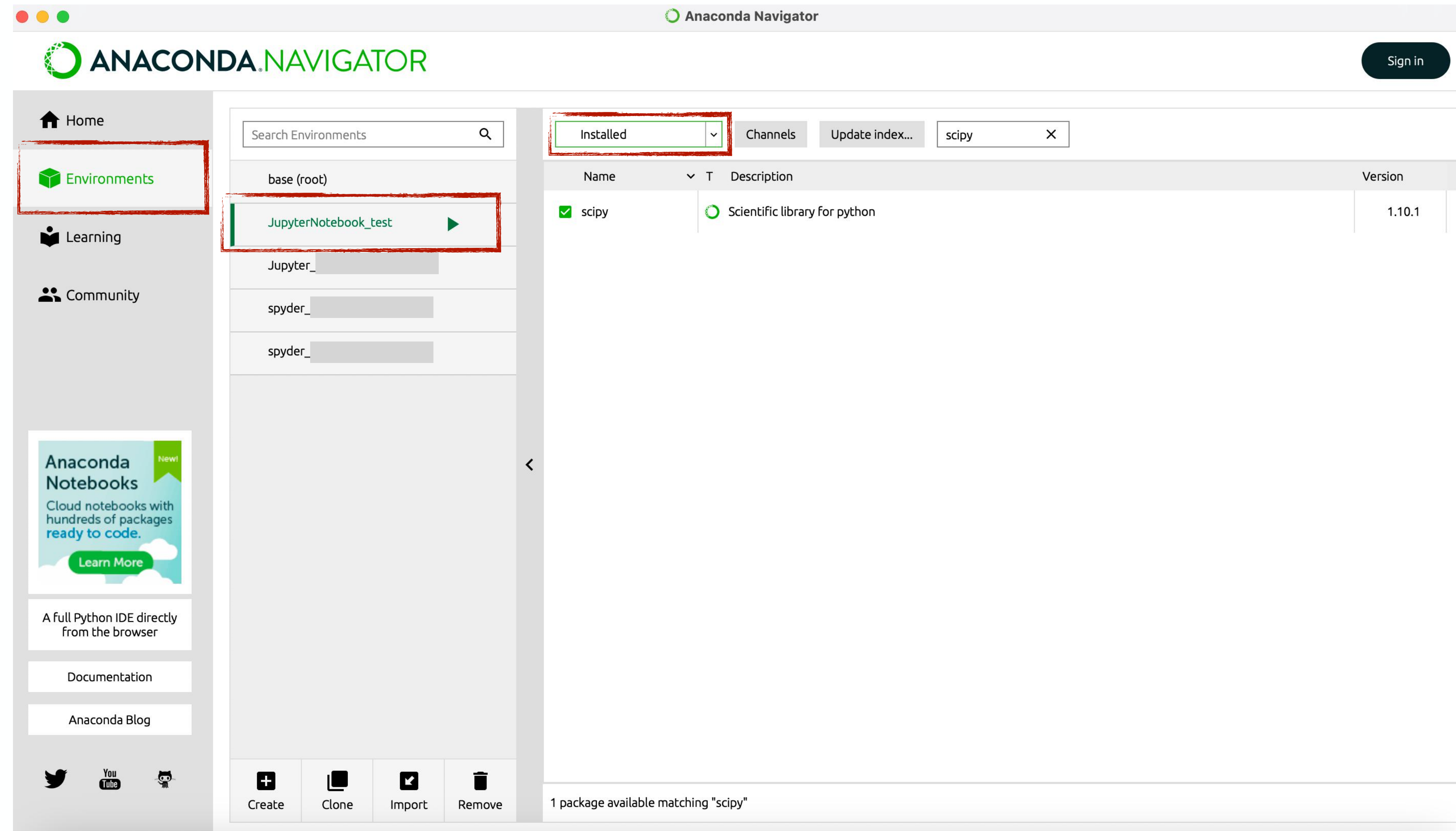
Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

To install libraries
(instead of applications)
within a given environment:

Select the library to be
installed in the environment

The library has just been
installed and can be launched



Setting up the working environment with Anaconda

Let's use Anaconda via shell (i.e. without its graphical user interface):

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ conda
usage: conda [-h] [--no-plugins] [-V] COMMAND ...

conda is a tool for managing and deploying applications, environments and packages.

optional arguments:
  -h, --help            Show this help message and exit.
  --no-plugins          Disable all plugins that are not built into conda.
  -V, --version         Show the conda version number and exit.

commands:
  The following built-in and plugins subcommands are available.

COMMAND
  build                See `conda build --help`.
  clean                Remove unused packages and caches.
  compare              Compare packages between conda environments.
  config               Modify configuration values in .condarc.
  content-trust        Signing and verification tools for Conda
  convert              See `conda convert --help`.
  create               Create a new conda environment from a list of specified packages.
  debug                See `conda debug --help`.
  develop              See `conda develop --help`.
  doctor              Display a health report for your environment.
  env                  See `conda env --help`.
  index                See `conda index --help`.
  info                 Display information about current conda install.
  init                 Initialize conda for shell interaction.
  inspect              See `conda inspect --help`.
  install              Install a list of packages into a specified conda environment.
  list                 List installed packages in a conda environment.
  metapackage          See `conda metapackage --help`.
  notices              Retrieve latest channel notifications.
  pack                 See `conda pack --help`.
```

Setting up the working environment with Anaconda

Let's use Anaconda via shell

Already available environments:

Create a new environment (you can also specify which version of Python you want to use by including the version number after the environment name):

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini]$ conda info --envs
# conda environments:
#
base * /Users/milenavalentini/opt/anaconda3
JupyterNotebook_test /Users/milenavalentini/opt/anaconda3/envs/JupyterNotebook_test
Jupyter_ /Users/milenavalentini/opt/anaconda3/envs/Jupyter_
spyder_ /Users/milenavalentini/opt/anaconda3/envs/spyder_
spyder_ /Users/milenavalentini/opt/anaconda3/envs/spyder_
```

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini]$
[(base) MacBook-Pro-2:TRM_Dati milenavalentini]$ conda create --name TRMD_2023 python=3.8
```

The new environment has been create

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini]$ conda info --envs
# conda environments:
#
base * /Users/milenavalentini/opt/anaconda3
JupyterNotebook_test /Users/milenavalentini/opt/anaconda3/envs/JupyterNotebook_test
Jupyter_ /Users/milenavalentini/opt/anaconda3/envs/Jupyter_
TRMD_2023 /Users/milenavalentini/opt/anaconda3/envs/TRMD_2023
spyder_ /Users/milenavalentini/opt/anaconda3/envs/spyder_
spyder_ /Users/milenavalentini/opt/anaconda3/envs/spyder_
```

Activate it:

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini]$ conda activate TRMD_2023
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$
```

Setting up the working environment with Anaconda

Let's use Anaconda via shell

Packages already available
within the active environment:

```
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$ conda list
# packages in environment at /Users/milenavalentini/opt/anaconda3/envs/TRMD_2023:
#
# Name                                Version                                Build                                Channel
bzip2                                  1.0.8                                  h0d85af4_4                           conda-forge
ca-certificates                        2023.7.22                              h8857fd0_0                           conda-forge
libffi                                  3.4.2                                  h0d85af4_5                           conda-forge
libsqlite                               3.43.0                                  h58db7d2_0                           conda-forge
libzlib                                 1.2.13                                  h8aleda9_5                           conda-forge
ncurses                                 6.4                                     hf0c8a7f_0                           conda-forge
openssl                                 3.1.3                                  h8aleda9_0                           conda-forge
pip                                     23.2.1                                  pyhd8ed1ab_0                         conda-forge
python                                  3.8.17                                  hf9b03c3_0_cpython                   conda-forge
readline                                8.2                                     h9e318b2_1                           conda-forge
setuptools                              68.2.2                                  pyhd8ed1ab_0                         conda-forge
tk                                       8.6.12                                  h5dbffcc_0                           conda-forge
wheel                                   0.41.2                                  pyhd8ed1ab_0                         conda-forge
xz                                       5.2.6                                  h775f41a_0                           conda-forge
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$
```


Setting up the working environment with Anaconda

Let's use Anaconda via shell

Packages already available within the active environment:

As an example of how to install an application:

Install the Jupyter Notebook

```
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$ conda install jupyter
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: /Users/milenavalentini/opt/anaconda3/envs/TRMD_2023
```

```
added / updated specs:
- jupyter
```

```
The following packages will be downloaded:
```

package	build		
dbus-1.13.6	h811a1a6_3	551 KB	conda-forge
icu-69.1	he49afe7_0	12.9 MB	conda-forge
libclang-13.0.1	root_62804_h2961583_3	20.5 MB	conda-forge
libllvm13-13.0.1	h64f94b2_2	25.3 MB	conda-forge
libpq-14.5	h3df487d_7	2.1 MB	conda-forge
mysql-common-8.0.33	h794ff91_4	744 KB	conda-forge
mysql-libs-8.0.33	he48d296_4	1.4 MB	conda-forge
pyqt-5.12.3	py38hca2ab18_4	5.2 MB	conda-forge
qt-5.12.9	h2a607e2_5	87.9 MB	conda-forge
Total:		156.6 MB	

```
The following NEW packages will be INSTALLED:
```

Jupyter Notebook

Let's use Anaconda via shell

Launch it:

```
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$ jupyter notebook
[I 2023-09-22 15:16:08.718 ServerApp] Package notebook took 0.0000s to import
[I 2023-09-22 15:16:10.127 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip c
onfirmation).
[C 2023-09-22 15:16:10.140 ServerApp]

To access the server, open this file in a browser:
  file:///Users/milenavalentini/Library/Jupyter/runtime/jpserver-70912-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/tree?token=84fd8e0bc4e833913be7f0e14d7bbc6a8650cf79f8d4ae03
  http://127.0.0.1:8888/tree?token=84fd8e0bc4e833913be7f0e14d7bbc6a8650cf79f8d4ae03
[I 2023-09-22 15:28:31.982 ServerApp] Saving file at /Untitled.ipynb
```

Jupyter Notebook

Let's use Anaconda via shell

Launch it:

```
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$ jupyter notebook  
[I 2023-09-22 15:16:08.718 ServerApp] Package notebook took 0.0000s to import  
[I 2023-09-22 15:16:10.127 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).  
[C 2023-09-22 15:16:10.140 ServerApp]
```

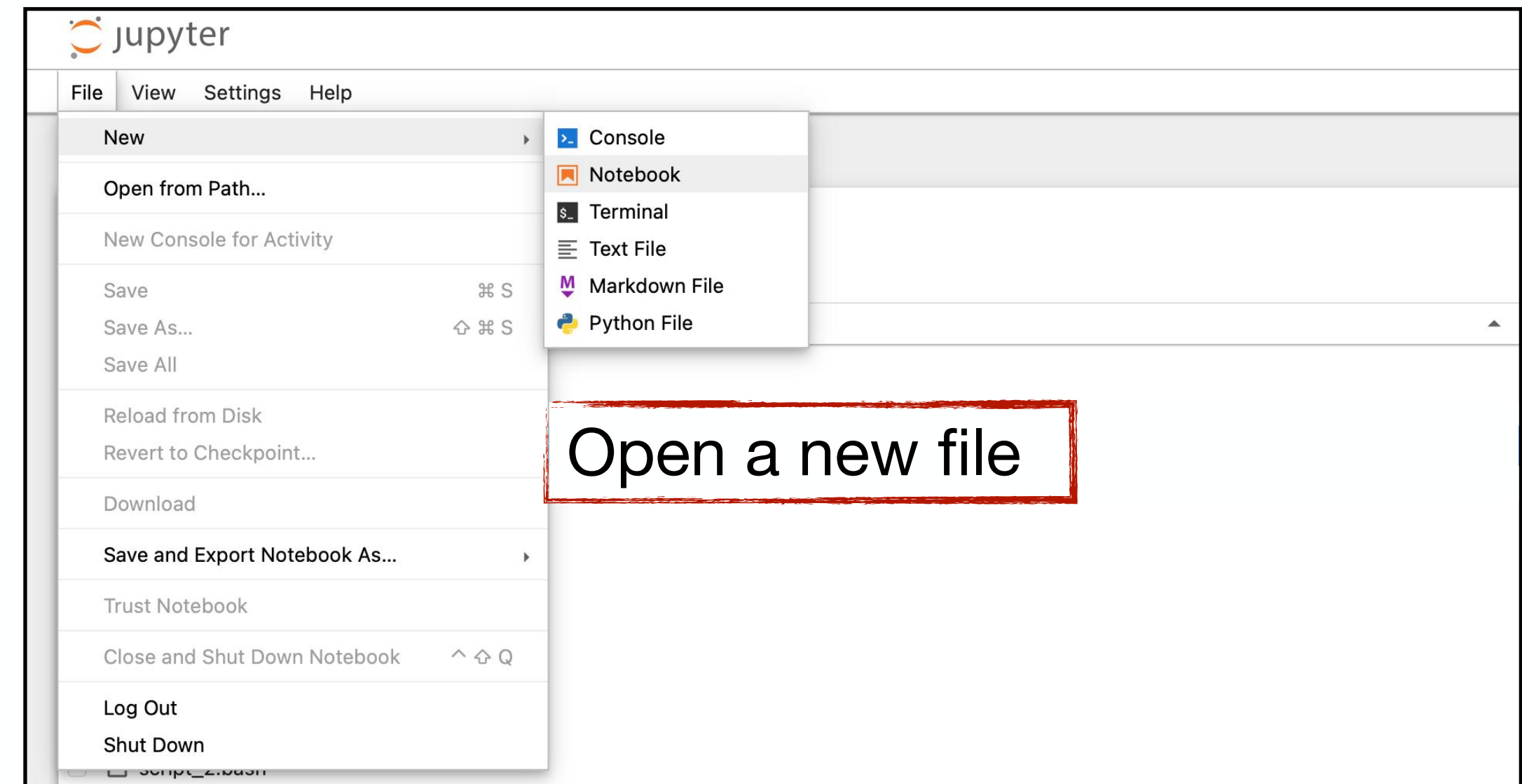
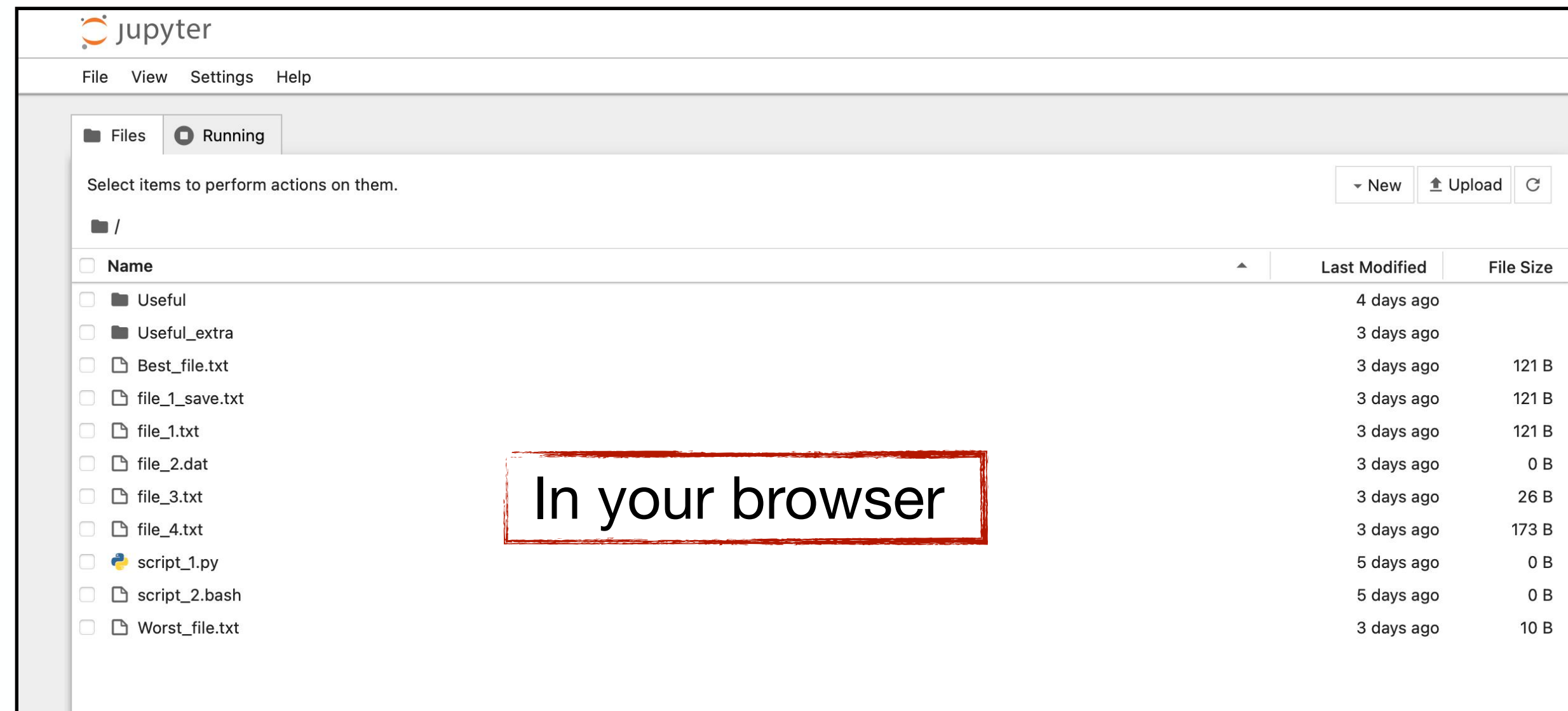
To access the server, open this file in a browser:

```
file:///Users/milenavalentini/Library/Jupyter/runtime/jpserver-70912-open.html
```

Or copy and paste one of these URLs:

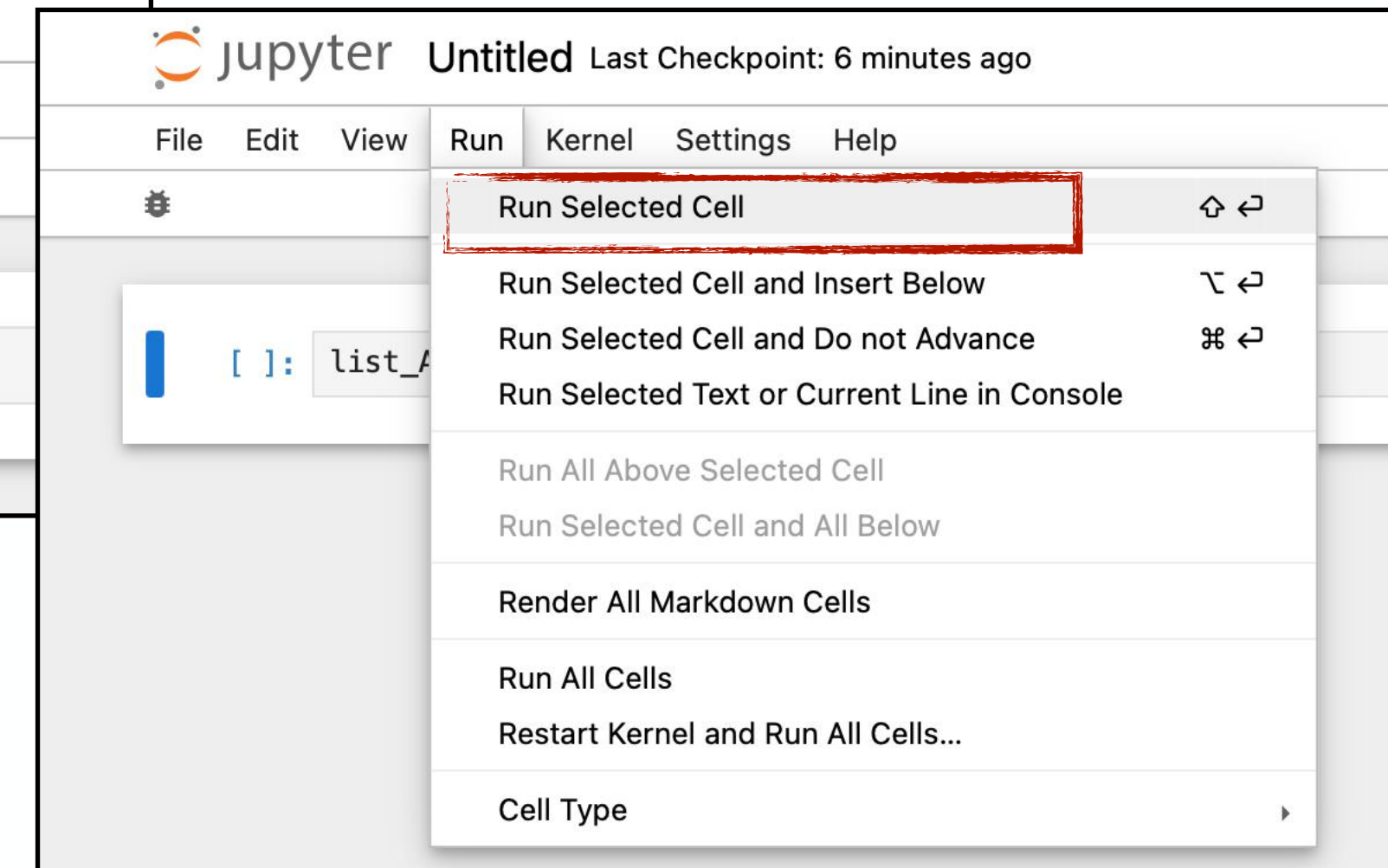
```
http://localhost:8888/tree?token=84fd8e0bc4e833913be7f0e14d7bbc6a8650cf79f8d4ae03
```

```
http://127.0.0.1:8888/tree?token=84fd8e0bc4e833913be7f0e14d7bbc6a8650cf79f8d4ae03
```



Jupyter Notebook

Let's use Anaconda via shell



Jupyter Notebook

Let's use Anaconda via shell

jupyter Untitled Last Checkpoint: 7 minutes ago

File Edit View Run Kernel Settings Help

```
[ ]: list_A=[1,2,3]
```

jupyter Untitled Last Checkpoint: 6 minutes ago

File Edit View Run Kernel Settings Help

- Run Selected Cell ⌘ ↵
- Run Selected Cell and Insert Below ⌘ ↵
- Run Selected Cell and Do not Advance ⌘ ↵
- Run Selected Text or Current Line in Console
- Run All Above Selected Cell
- Run Selected Cell and All Below
- Render All Markdown Cells
- Run All Cells
- Restart Kernel and Run All Cells...
- Cell Type ▶

jupyter Untitled Last Checkpoint: 8 minutes ago

File Edit View Run Kernel Settings Help

```
[4]: list_A=[1,2,3]
```

```
[ ]: print(list_A)
```

jupyter Untitled

File Edit View Run Kernel Settings Help

- Run Selected Cell ⌘ ↵
- Run Selected Cell and Insert Below ⌘ ↵
- Run Selected Cell and Do not Advance ⌘ ↵
- Run Selected Text or Current Line in Console
- Run All Above Selected Cell
- Run Selected Cell and All Below
- Render All Markdown Cells
- Run All Cells
- Restart Kernel and Run All Cells...
- Cell Type ▶

jupyter Untitled Last Checkpoint: 9 minutes ago

File Edit View Run Kernel Settings Help

```
[4]: list_A=[1,2,3]
```

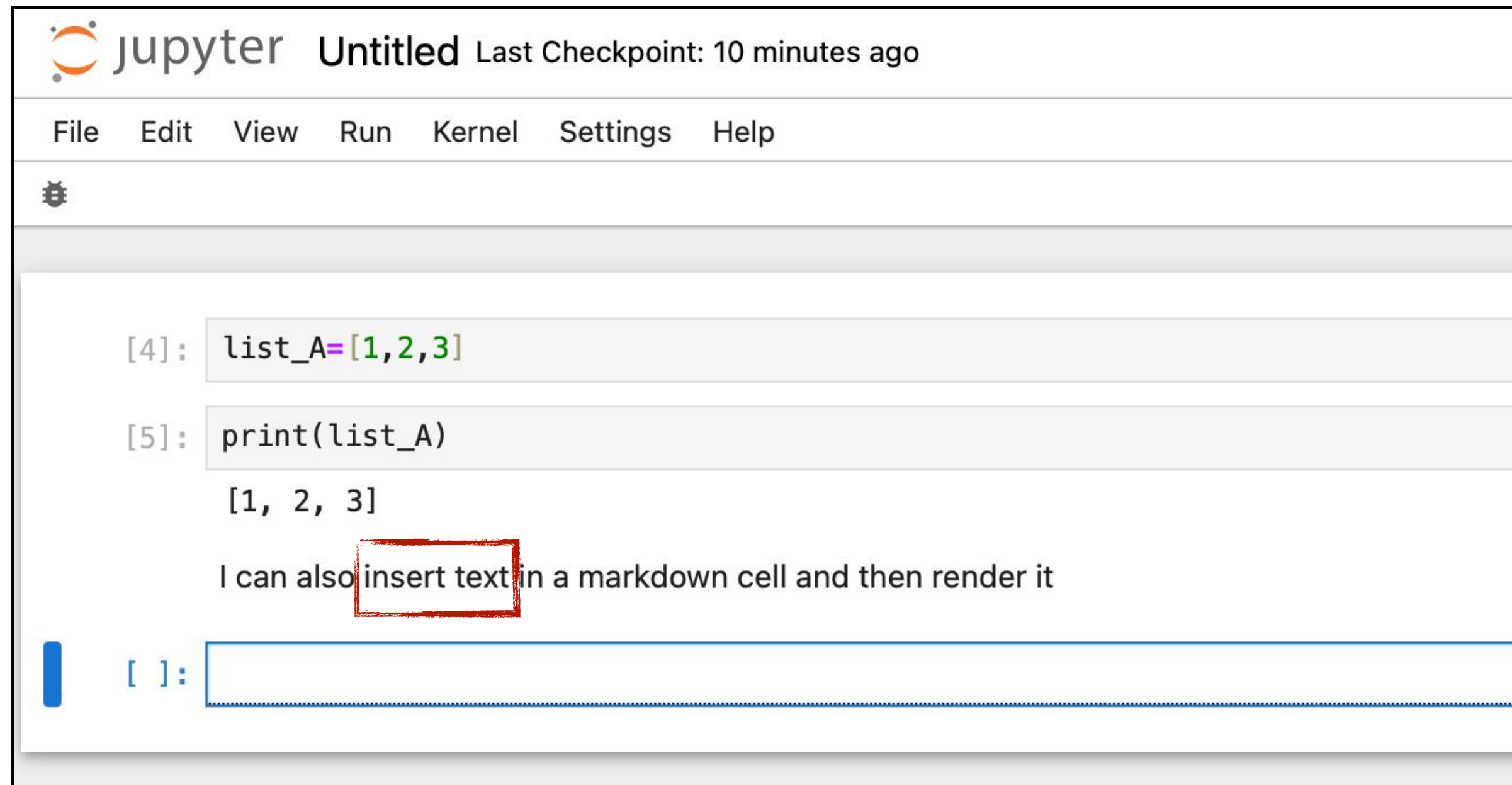
```
[5]: print(list_A)
```

```
[1, 2, 3]
```

Jupyter Notebook

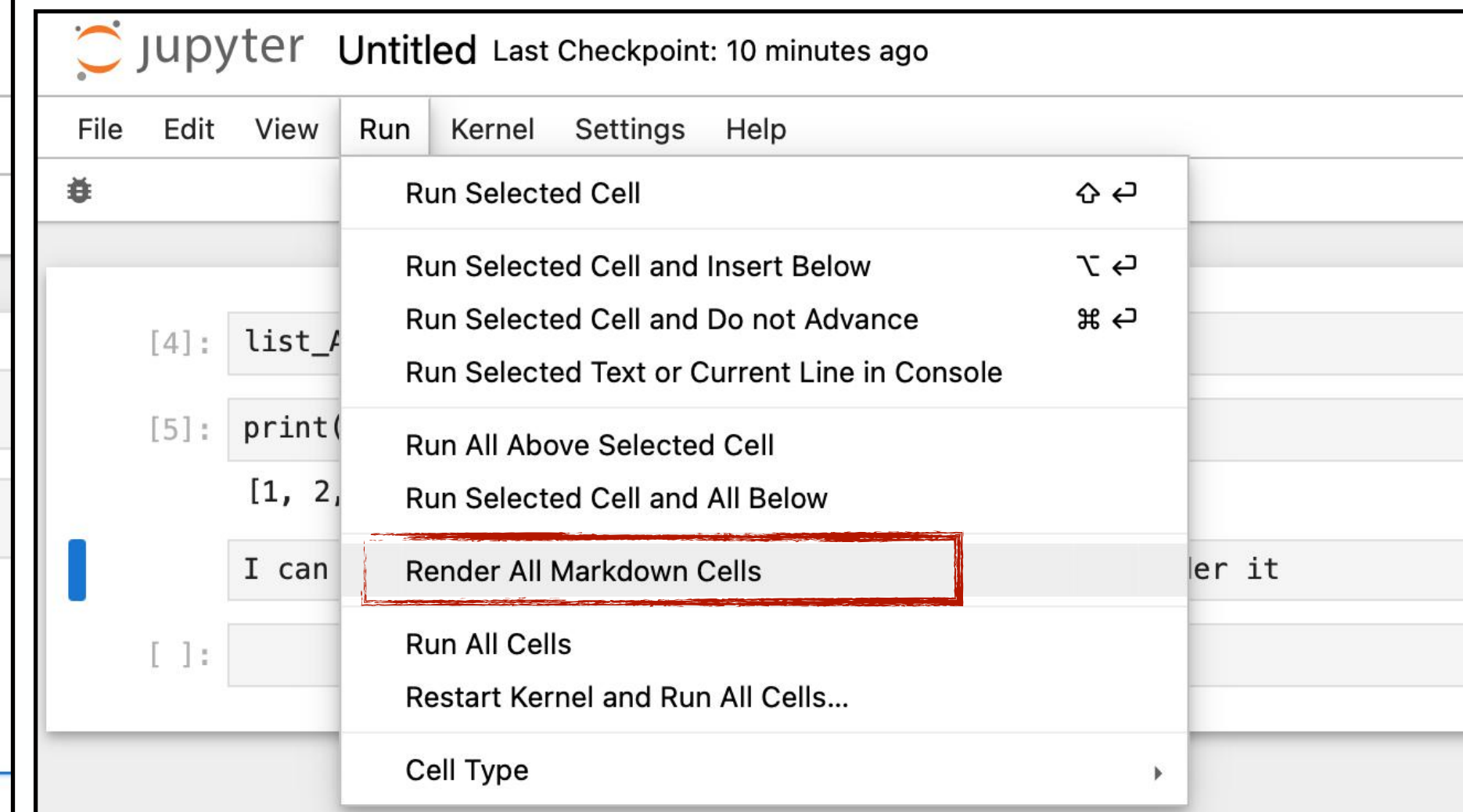
Let's use Anaconda via shell

Launch it:



The screenshot shows a Jupyter Notebook window titled "jupyter Untitled" with a last checkpoint of 10 minutes ago. The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. Below the menu bar is a toolbar with a bug icon. The notebook contains three cells:

- Cell [4]: `list_A=[1,2,3]`
- Cell [5]: `print(list_A)` with output `[1, 2, 3]`
- Cell []: A markdown cell containing the text "I can also insert text in a markdown cell and then render it". The words "insert text" are highlighted with a red box.



The screenshot shows the same Jupyter Notebook window as the previous one, but with the "Run" menu open. The menu items are:

- Run Selected Cell (⌘ ↵)
- Run Selected Cell and Insert Below (⇧ ↵)
- Run Selected Cell and Do not Advance (⌘ ↵)
- Run Selected Text or Current Line in Console
- Run All Above Selected Cell
- Run Selected Cell and All Below
- Render All Markdown Cells** (highlighted with a red box)
- Run All Cells
- Restart Kernel and Run All Cells...
- Cell Type (dropdown arrow)

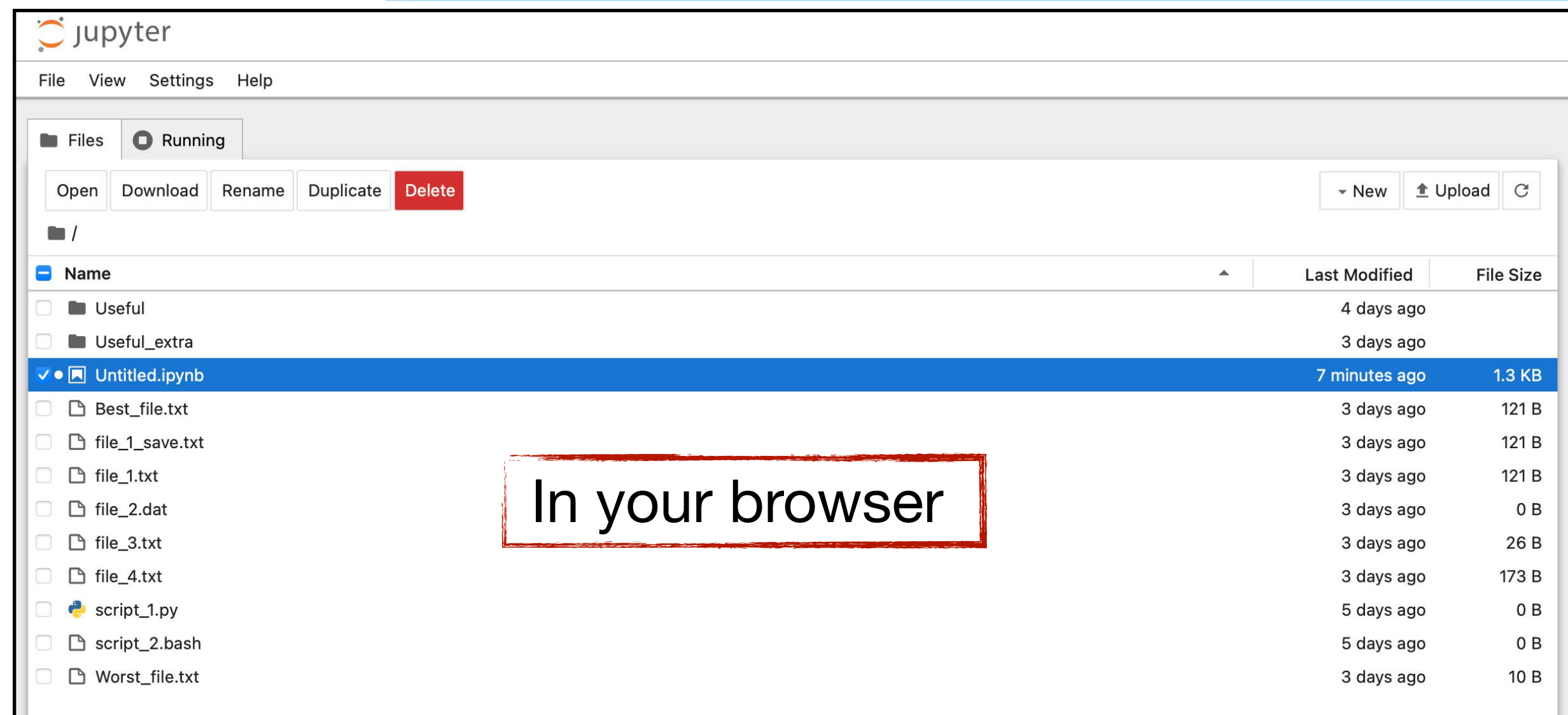
Setting up the working environment with Anaconda

Let's use Anaconda via shell

Launch it:

```
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$ jupyter notebook
[I 2023-09-22 15:16:08.718 ServerApp] Package notebook took 0.0000s to import
[I 2023-09-22 15:16:10.127 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip c
onfirmation).
[C 2023-09-22 15:16:10.140 ServerApp]

To access the server, open this file in a browser:
  file:///Users/milenavalentini/Library/Jupyter/runtime/jpserver-70912-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/tree?token=84fd8e0bc4e833913be7f0e14d7bbc6a8650cf79f8d4ae03
  http://127.0.0.1:8888/tree?token=84fd8e0bc4e833913be7f0e14d7bbc6a8650cf79f8d4ae03
[I 2023-09-22 15:28:31.982 ServerApp] Saving file at /Untitled.ipynb
```



Setting up the working environment with Anaconda

Let's use Anaconda via shell

Install libraries
within an active environment:

```
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$ conda install matplotlib
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /Users/milenavalentini/opt/anaconda3/envs/TRMD_2023

added / updated specs:
- matplotlib

The following packages will be downloaded:
```

package	build		
kiwisolver-1.4.5	py38h15a1a5b_1	59 KB	conda-forge
matplotlib-3.2.2	1	6 KB	conda-forge
Total:		65 KB	

```
The following NEW packages will be INSTALLED:

cyclor          conda-forge/noarch::cyclor-0.11.0-pyhd8ed1ab_0
freetype        conda-forge/osx-64::freetype-2.12.1-h60636b9_2
kiwisolver      conda-forge/osx-64::kiwisolver-1.4.5-py38h15a1a5b_1
matplotlib      conda-forge/osx-64::matplotlib-3.2.2-1
matplotlib-base conda-forge/osx-64::matplotlib-base-3.2.2-py38h1300a51_1
pyparsing       conda-forge/noarch::pyparsing-3.1.1-pyhd8ed1ab_0

Proceed ([y]/n)? yes

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(TRMD_2023) MacBook-Pro-2:TRM_Dati milenavalentini$
```

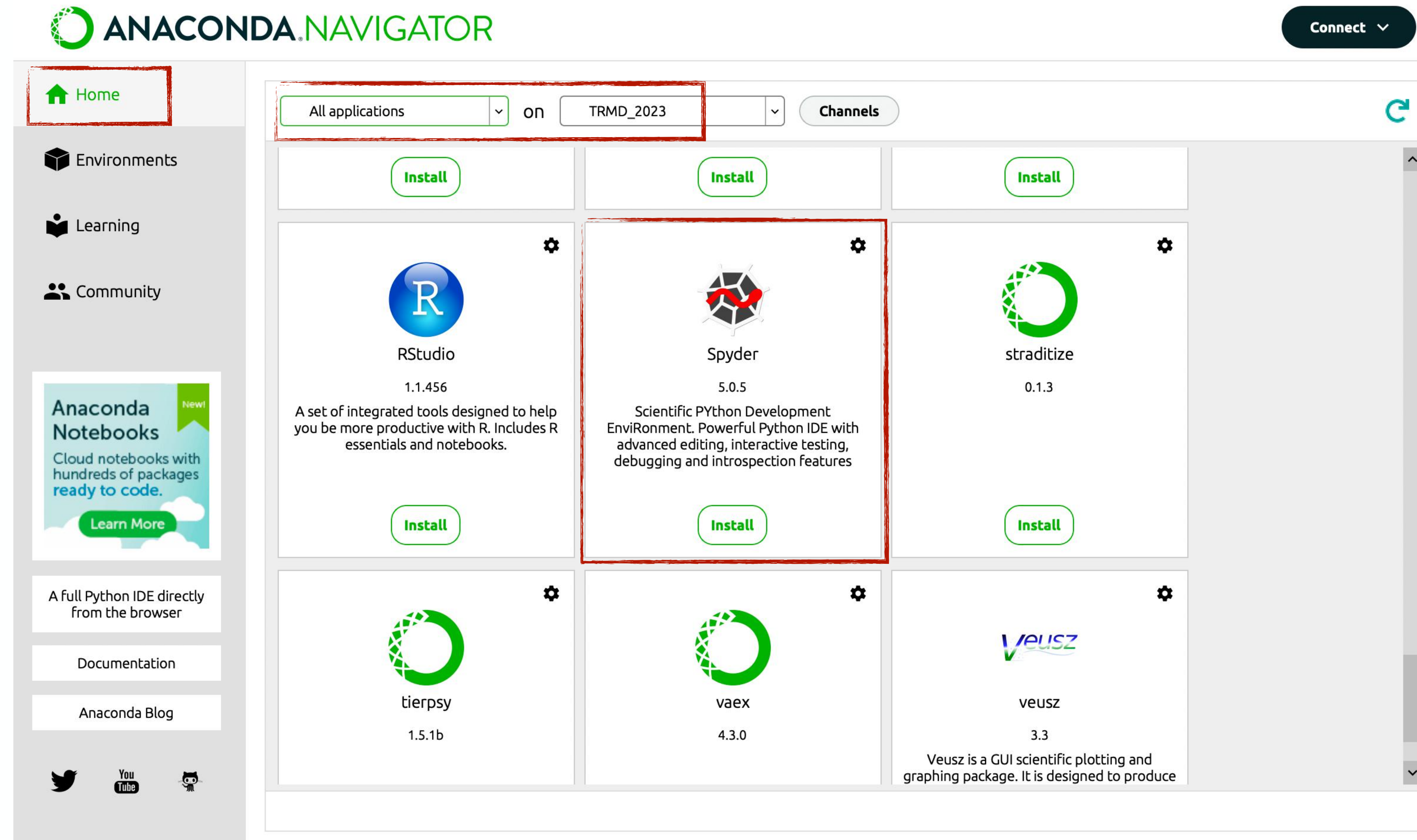
matplotlib
*scientific library
for publication quality
figures in Python*

Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

A very useful application which can be used as a code editor:



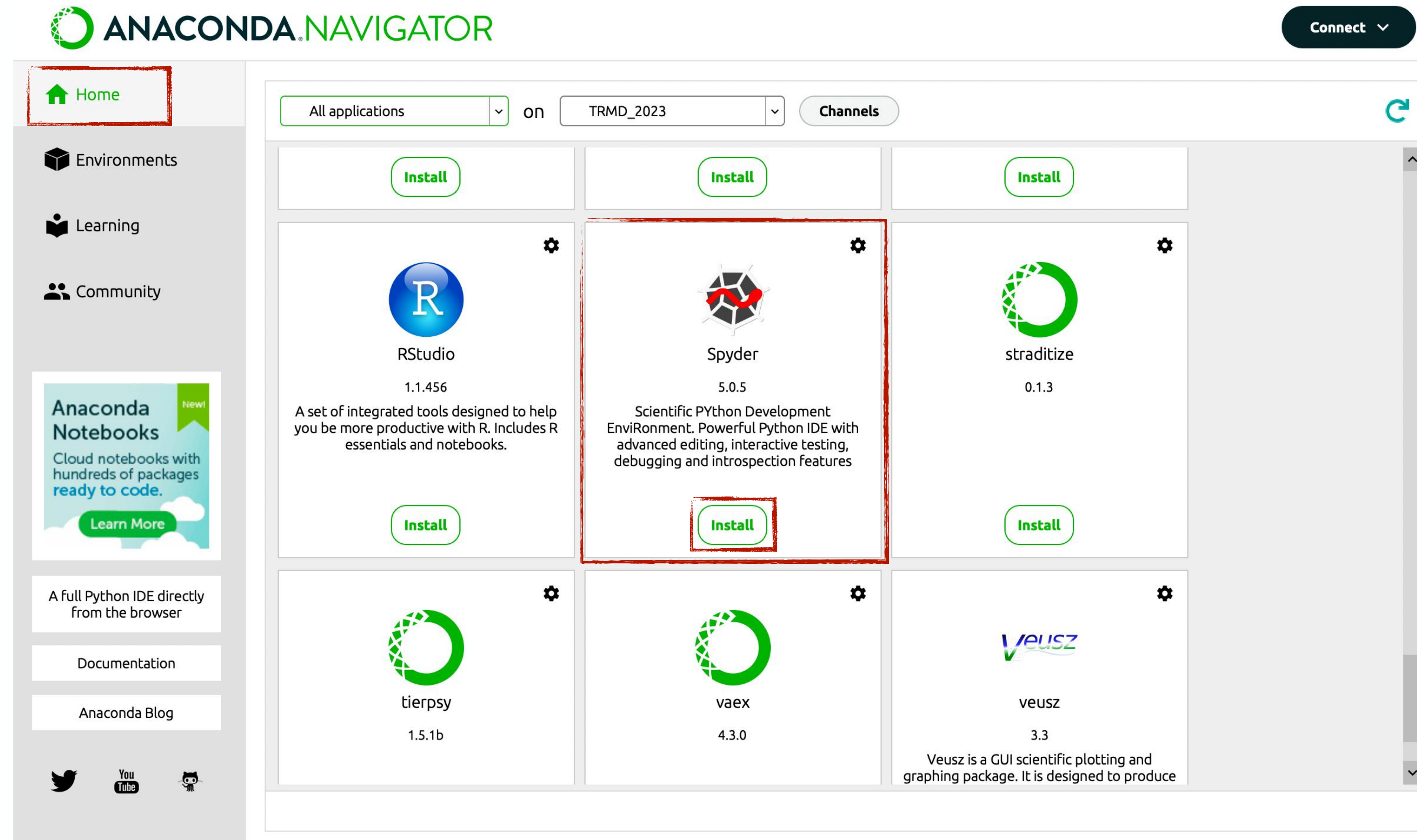
Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

A very useful application which can be used as a code editor:

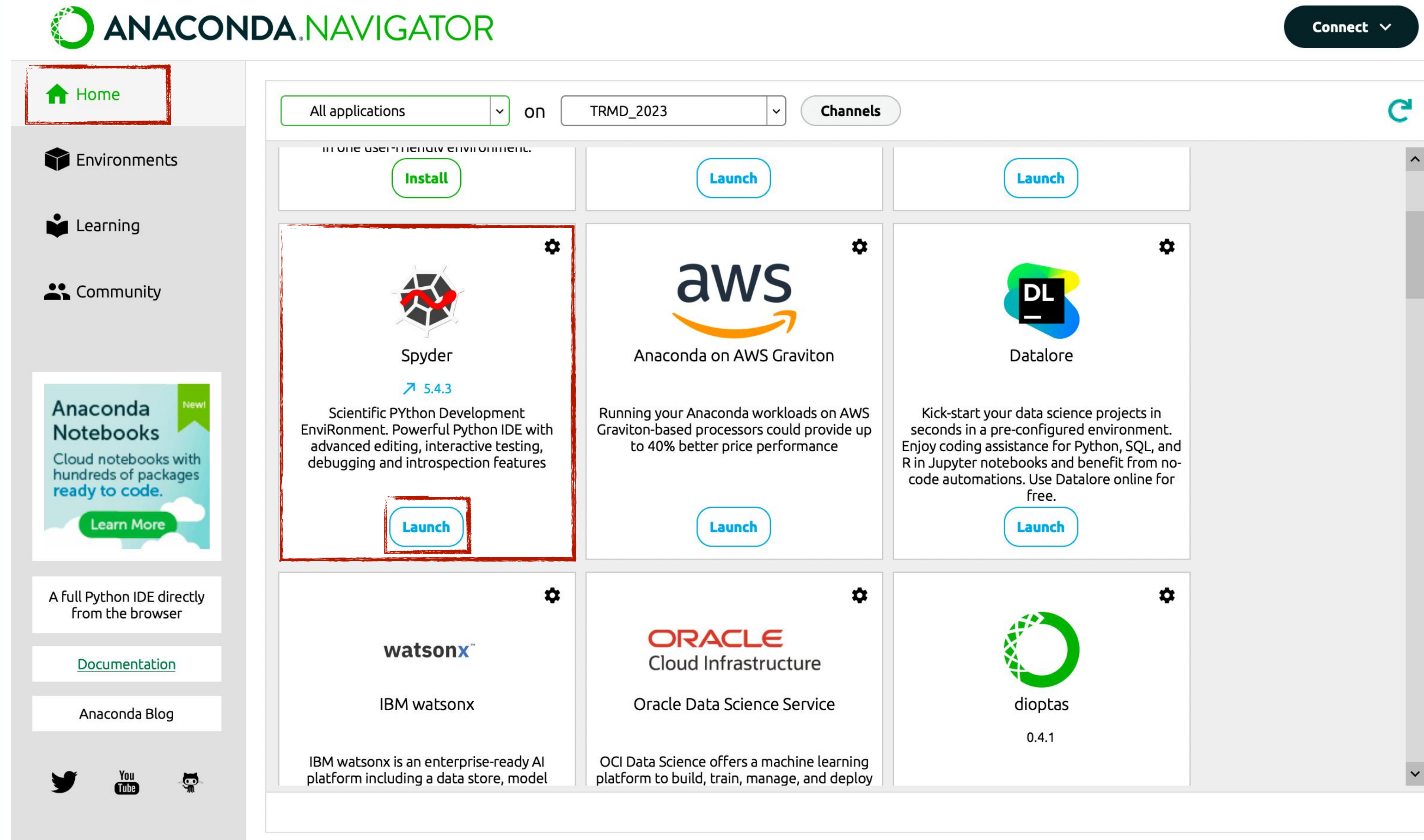
Select the applications to be installed in the environment among available ones



Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:



A very useful application which can be used as a code editor:

Select the applications to be installed in the environment

The application has just been installed and can be launched

Spyder

The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.8)". The top toolbar contains various icons for file operations and execution. The main editor displays a Python script named "temp.py" with the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4 This is a temporary script file.
5 """
6
7 import numpy as np
8 import matplotlib.pyplot as plt
9 import healpy as hp
10 import math
11
12
13 NSIDE = 64
14 print(
15     "Approximate resolution at NSIDE {} is {:.2} deg".format(
16         NSIDE, hp.nside2resol(NSIDE, arcmin=True) / 60
17     )
18 )
19
20 NPIX = hp.nside2npix(NSIDE)
21 print(NPIX)
22
23 my_map_I = hp.read_map("test.h.fits")
24
25 hp.mollview(
26     my_map_I,
27     title="Fiducial model sky map",
28     unit="cm$^{-2}$",
29     # norm="hist",
30     norm="log",
31     min=math.modf(np.amin(my_map_I))[1],
32     max=math.modf(np.amax(my_map_I))[1]
33     # min=605,
34     # max=11525
35 )
36 hp.graticule()
37
38 plt.savefig("My_healpix_map.png", dpi=300)
39
40
```

The right-hand side of the interface features a "Console" tab with a "Usage" help message:

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

[New to Spyder? Read our tutorial](#)

Below the help message are tabs for "Help", "Variable Explorer", "Plots", and "Files". The "Console" tab is active, showing the following output:

```
Python 3.8.17 | packaged by conda-forge | (default, Jun 16 2023, 07:11:34)
Type "copyright", "credits" or "license" for more information.

IPython 8.4.0 -- An enhanced Interactive Python.

In [1]:
```

The bottom status bar displays: conda: TRMD_2023 (Python 3.8.17) | Completions: conda(TRMD_2023) | LSP: Python | Line 40, Col 1 | UTF-8 | LF | RW | Mem 94%

Spyder

The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script named `temp.py` with the following code:

```
1  # -*- coding: utf-8 -*-
2  """
3  Spyder Editor
4  This is a temporary script file.
5  """
6
7  import numpy as np
8  import matplotlib.pyplot as plt
9  import healpy as hp
10 import math
11
12
13 NSIDE = 64
14 print(
15     "Approximate resolution at NSIDE {} is {:.2} deg".format(
16         NSIDE, hp.nside2resol(NSIDE, arcmin=True) / 60
17     )
18 )
19
20 NPIX = hp.nside2npix(NSIDE)
21 print(NPIX)
22
23 my_map_I = hp.read_map("test.h.fits")
24
25 hp.mollview(
26     my_map_I,
27     title="Fiducial model sky map",
28     unit="cm$^{-2}$",
29     # norm="hist",
30     norm="log",
31     min=math.modf(np.amin(my_map_I))[1],
32     max=math.modf(np.amax(my_map_I))[1]
33     # min=605,
34     # max=11525
35 )
36 hp.graticule()
37
38 plt.savefig("My_healpix_map.png", dpi=300)
39
40
```

The right sidebar contains a 'Usage' help panel with the following text:

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

[New to Spyder? Read our tutorial](#)

Below the help panel are tabs for 'Help', 'Variable Explorer', 'Plots', and 'Files'. The console window shows the IPython prompt:

```
Python 3.8.17 | packaged by conda-forge | (default, Jun 16 2023, 07:11:34)
Type "copyright", "credits" or "license" for more information.

IPython 8.4.0 -- An enhanced Interactive Python.

In [1]:
```

The status bar at the bottom indicates: conda: TRMD_2023 (Python 3.8.17) | Completions: conda(TRMD_2023) | LSP: Python | Line 40, Col 1 | UTF-8 | LF | RW | Mem 94%

Spyder

The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.8)". The top toolbar contains various icons for file operations and execution. The main editor displays a Python script named "temp.py" with the following code:

```
1  # -*- coding: utf-8 -*-
2  """
3  Spyder Editor
4  This is a temporary script file.
5  """
6
7  import numpy as np
8  import matplotlib.pyplot as plt
9  import healpy as hp
10 import math
11
12
13 NSIDE = 64
14 print(
15     "Approximate resolution at NSIDE {} is {:.2} deg".format(
16         NSIDE, hp.nside2resol(NSIDE, arcmin=True) / 60
17     )
18 )
19
20 NPIX = hp.nside2npix(NSIDE)
21 print(NPIX)
22
23 my_map_I = hp.read_map("test.h.fits")
24
25 hp.mollview(
26     my_map_I,
27     title="Fiducial model sky map",
28     unit="cm$^{-2}$",
29     # norm="hist",
30     norm="log",
31     min=math.modf(np.amin(my_map_I))[1],
32     max=math.modf(np.amax(my_map_I))[1]
33     # min=605,
34     # max=11525
35 )
36 hp.graticule()
37
38 plt.savefig("My_healpix_map.png", dpi=300)
39
40
```

The right-hand side of the interface is divided into two panels. The top panel, titled "Console", shows a "Usage" message:

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

[New to Spyder? Read our tutorial](#)

The bottom panel, titled "Console 1/A", shows the IPython console output:

```
Python 3.8.17 | packaged by conda-forge | (default, Jun 16 2023, 07:11:34)
Type "copyright", "credits" or "license" for more information.

IPython 8.4.0 -- An enhanced Interactive Python.

In [1]:
```

The bottom status bar displays: conda: TRMD_2023 (Python 3.8.17) | Completions: conda(TRMD_2023) | LSP: Python | Line 40, Col 1 | UTF-8 | LF | RW | Mem 94%

Spyder

The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.8)". The top toolbar contains various icons, with the "Run" icon (a green play button) highlighted by a red box. Below the toolbar, the file path is shown as "/Users/milenaValentini/.spyder-py3/temp.py". The code editor displays the following Python code:

```
1  # -*- coding: utf-8 -*-
2  """
3  Spyder Editor
4  This is a temporary script file.
5  """
6
7  import numpy as np
8  import matplotlib.pyplot as plt
9  import healpy as hp
10 import math
11
12
13 NSIDE = 64
14 print(
15     "Approximate resolution at NSIDE {} is {:.2} deg".format(
16         NSIDE, hp.nside2resol(NSIDE, arcmin=True) / 60
17     )
18 )
19
20 NPIX = hp.nside2npix(NSIDE)
21 print(NPIX)
22
23 my_map_I = hp.read_map("test.h.fits")
24
25 hp.mollview(
26     my_map_I,
27     title="Fiducial model sky map",
28     unit="cm$^{-2}$",
29     # norm="hist",
30     norm="log",
31     min=math.modf(np.amin(my_map_I))[1],
32     max=math.modf(np.amax(my_map_I))[1]
33     # min=605,
34     # max=11525
35 )
36 hp.graticule()
37
38 plt.savefig("My_healpix_map.png", dpi=300)
39
40
```

The console window shows the following output:

```
Python 3.8.17 | packaged by conda-forge | (default, Jun 16 2023, 07:11:34)
Type "copyright", "credits" or "license" for more information.

IPython 8.4.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/milenaValentini/.spyder-py3/temp.py')
Traceback (most recent call last):

  File ~/opt/anaconda3/envs/TRMD_2023/lib/python3.8/site-packages/spyder_kernels/py3compat.py:356 in
  compat_exec
    exec(code, globals, locals)

  File ~/spyder-py3/temp.py:9 in <module>
    import healpy as hp

ModuleNotFoundError: No module named 'healpy'

In [2]:
```

The status bar at the bottom indicates: conda: TRMD_2023 (Python 3.8.17) | Completions: conda(TRMD_2023) | LSP: Python | Line 40, Col 1 | UTF-8 | LF | RW | Mem 94%

Spyder

The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.8)" and shows a code editor on the left and a console on the right. The code editor contains a Python script named "temp.py" with the following content:

```
1  #-*- coding: utf-8 -*-
2  """
3  Spyder Editor
4  This is a temporary script file.
5  """
6
7  import numpy as np
8  import matplotlib.pyplot as plt
9  import healpy as hp
10 import math
11
12
13 NSIDE = 64
14 print(
15     "Approximate resolution at NSIDE {} is {:.2} deg".format(
16         NSIDE, hp.nside2resol(NSIDE, arcmin=True) / 60
17     )
18 )
19
20 NPIX = hp.nside2npix(NSIDE)
21 print(NPIX)
22
23 my_map_I = hp.read_map("test.h.fits")
24
25 hp.mollview(
26     my_map_I,
27     title="Fiducial model sky map",
28     unit="cm$^{-2}$",
29     # norm="hist",
30     norm="log",
31     min=math.modf(np.amin(my_map_I))[1],
32     max=math.modf(np.amax(my_map_I))[1]
33     # min=605,
34     # max=11525
35 )
36 hp.graticule()
37
38 plt.savefig("My_healpix_map.png", dpi=300)
39
40
```

The console on the right shows the output of the script execution, including a traceback for a `ModuleNotFoundError` that occurred at line 9 of the script:

```
Python 3.8.17 | packaged by conda-forge | (default, Jun 16 2023, 07:11:34)
Type "copyright", "credits" or "license" for more information.

IPython 8.4.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/milenaValentini/.spyder-py3/temp.py')
Traceback (most recent call last):

  File ~/opt/anaconda3/envs/TRMD_2023/lib/python3.8/site-packages/spyder_kernels/py3compat.py:356 in
  compat_exec
    exec(code, globals, locals)

  File ~/spyder-py3/temp.py:9 in <module>
    import healpy as hp

ModuleNotFoundError: No module named 'healpy'
```

The terminal at the bottom of the console shows the command used to install the missing module:

```
In [2]: MacBook-Pro-2:TRM_Dati milenaValentini$ conda install healpy
```

The status bar at the bottom of the IDE indicates the current environment is `conda: TRMD_2023 (Python 3.8.17)` and shows other details like `Completions: conda(TRMD_2023)`, `LSP: Python`, `Line 40, Col 1`, `UTF-8`, `LF`, `RW`, and `Mem 94%`.

Spyder

The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.8)". The top toolbar contains various icons for file operations and execution. The left pane shows the file explorer with a single file named "temp.py". The central pane is the code editor, displaying the following Python code:

```
1  # -*- coding: utf-8 -*-
2  """
3  Spyder Editor
4  This is a temporary script file.
5  """
6
7  import numpy as np
8  import matplotlib.pyplot as plt
9  import healpy as hp
10 import math
11
12
13 NSIDE = 64
14 print(
15     "Approximate resolution at NSIDE {} is {:.2} deg".format(
16         NSIDE, hp.nside2resol(NSIDE, arcmin=True) / 60
17     )
18 )
19
20 NPIX = hp.nside2npix(NSIDE)
21 print(NPIX)
22
23 my_map_I = hp.read_map("test.h.fits")
24
25 hp.mollview(
26     my_map_I,
27     title="Fiducial model sky map",
28     unit="cm-2",
29     # norm="hist",
30     norm="log",
31     min=math.modf(np.amin(my_map_I))[1],
32     max=math.modf(np.amax(my_map_I))[1]
33     # min=605,
34     # max=11525
35 )
36 hp.graticule()
37
38 plt.savefig("My_healpix_map.png", dpi=300)
39
40
```

The right pane displays a plot titled "Fiducial model sky map". The plot shows a Mollweide projection of a sky map with a color scale ranging from 613 to 11687 cm⁻². The plot is zoomed in to 72%. Below the plot are tabs for "Help", "Variable Explorer", "Plots", and "Files".

The bottom pane is the IPython Console, showing the following output:

```
Python 3.8.17 | packaged by conda-forge | (default, Jun 16 2023, 07:11:34)
Type "copyright", "credits" or "license" for more information.

IPython 8.4.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/milenaValentini/.spyder-py3/temp.py')
Traceback (most recent call last):

  File ~/opt/anaconda3/envs/TRMD_2023/lib/python3.8/site-packages/spyder_kernels/py3compat.py:356 in
  compat_exec
    exec(code, globals, locals)

  File ~/.spyder-py3/temp.py:9 in <module>
    import healpy as hp

ModuleNotFoundError: No module named 'healpy'

In [2]: runfile('/Users/milenaValentini/.spyder-py3/temp.py')
Approximate resolution at NSIDE 64 is 0.92 deg
49152

In [3]:
```

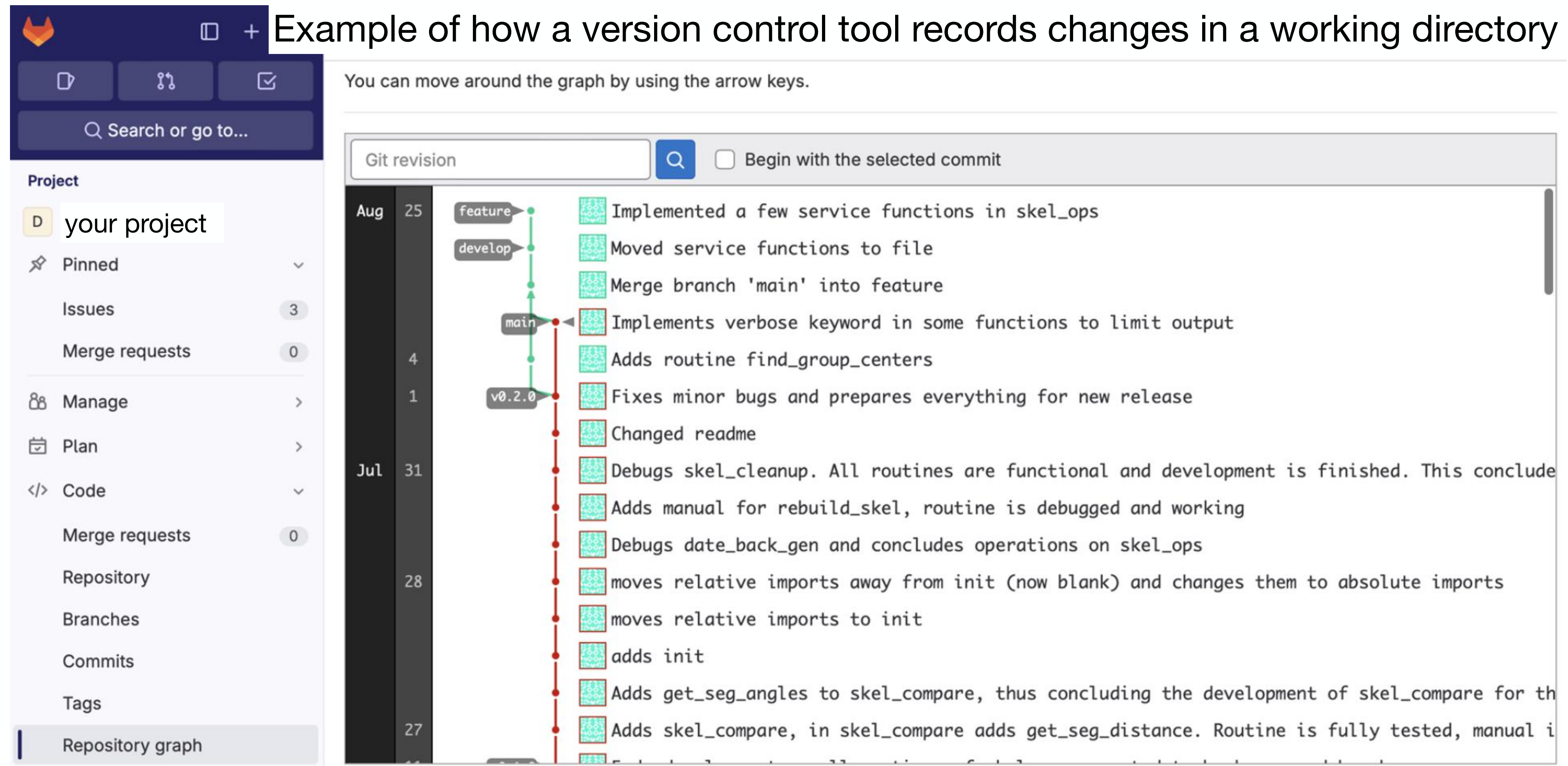
The status bar at the bottom indicates the environment is "conda: TRMD_2023 (Python 3.8.17)", with "Completions: conda(TRMD_2023)", "LSP: Python", "Line 23, Col 25", "UTF-8", "LF", "RW", and "Mem 94%".

Other tools: version control system

Version control system (e.g., GIT): allows you to keep track of changes, useful as a collaborative tool

Repositories are (working) directories hosting your projects.

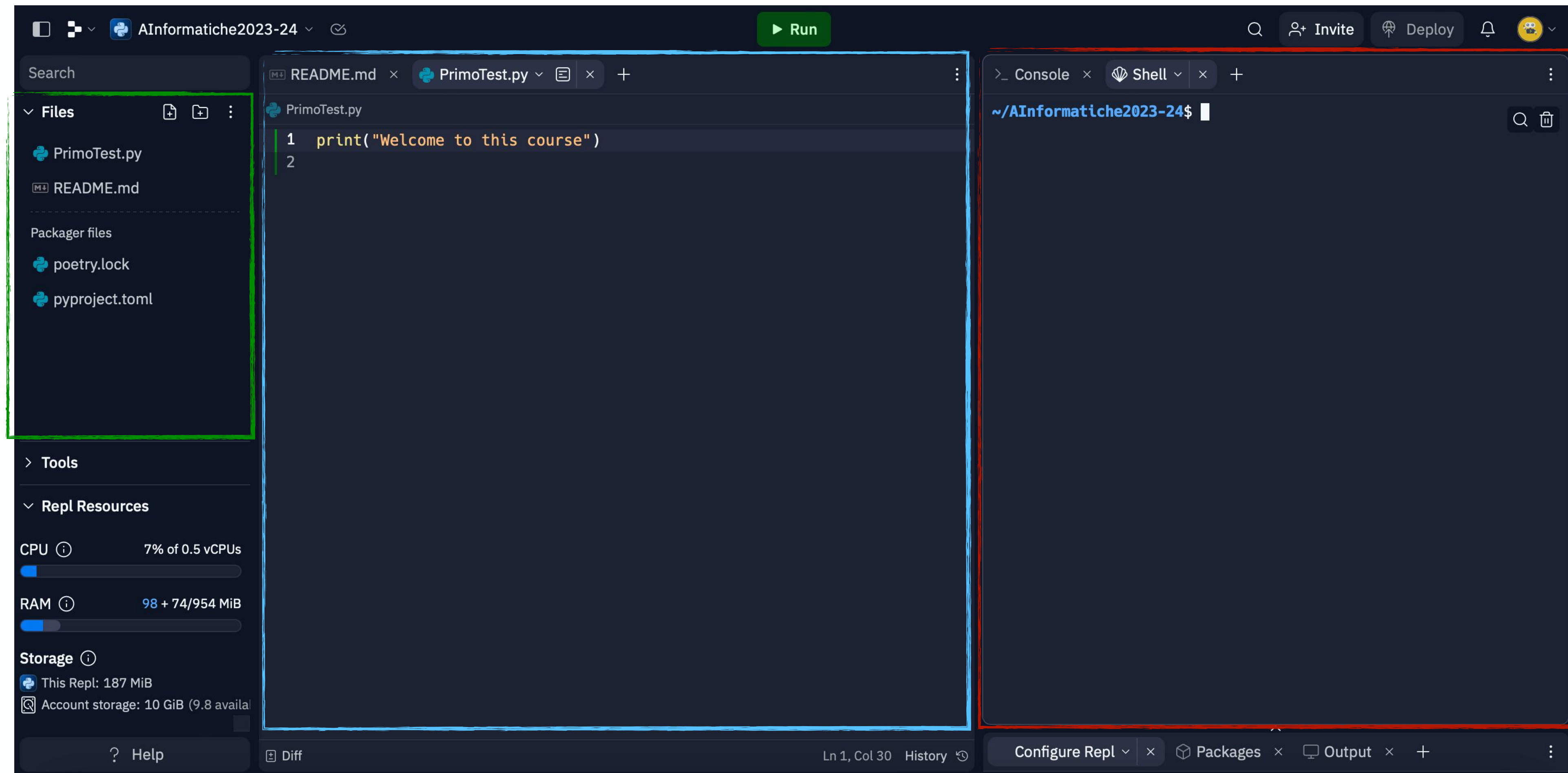
Example of how a version control tool records changes in a working directory



The screenshot displays a Git repository interface. On the left, a sidebar shows the project name 'your project' and various navigation options like 'Pinned', 'Issues', 'Merge requests', 'Manage', 'Plan', 'Code', 'Merge requests', 'Repository', 'Branches', 'Commits', 'Tags', and 'Repository graph'. The main area shows a commit history graph with branches 'feature', 'develop', 'main', and 'v0.2.0'. The graph shows a sequence of commits starting from 'v0.2.0' and moving through 'main' to 'develop' and 'feature'. The commit messages are listed on the right, including 'Implemented a few service functions in skel_ops', 'Moved service functions to file', 'Merge branch 'main' into feature', 'Implements verbose keyword in some functions to limit output', 'Adds routine find_group_centers', 'Fixes minor bugs and prepares everything for new release', 'Changed readme', 'Debugs skel_cleanup. All routines are functional and development is finished. This concludes', 'Adds manual for rebuild_skel, routine is debugged and working', 'Debugs date_back_gen and concludes operations on skel_ops', 'moves relative imports away from init (now blank) and changes them to absolute imports', 'moves relative imports to init', 'adds init', 'Adds get_seg_angles to skel_compare, thus concluding the development of skel_compare for th', and 'Adds skel_compare, in skel_compare adds get_seg_distance. Routine is fully tested, manual i'.

Other tools: Integrated Development Environment

A tool which displays all together a **file manager**, a **code editor**, a version control system, a **console** and other additional functionalities like for e.g. a debugger.



Other tools: Integrated Development Environment

Setup a working environment:

Register on Git (github.com)



Other tools: Integrated Development Environment

Setup a working environment:

Register on Git

Create a public repository on Git

New repository

Type to search

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * MilenaValentini / Repository name * TRM_Dati

✔ TRM_Dati is available.

Great repository names are short and memorable. Need inspiration? How about [silver-waddle](#) ?

Description (optional)

Public repository of the course: Tecniche di Rappresentazione e Modellizzazione Dati

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository

Other tools: Integrated Development Environment

Setup a working environment:

Register on Git

Create a public repository on Git

Register on repl.it (replit.com)



Other tools: Integrated Development Environment

Setup a working environment:

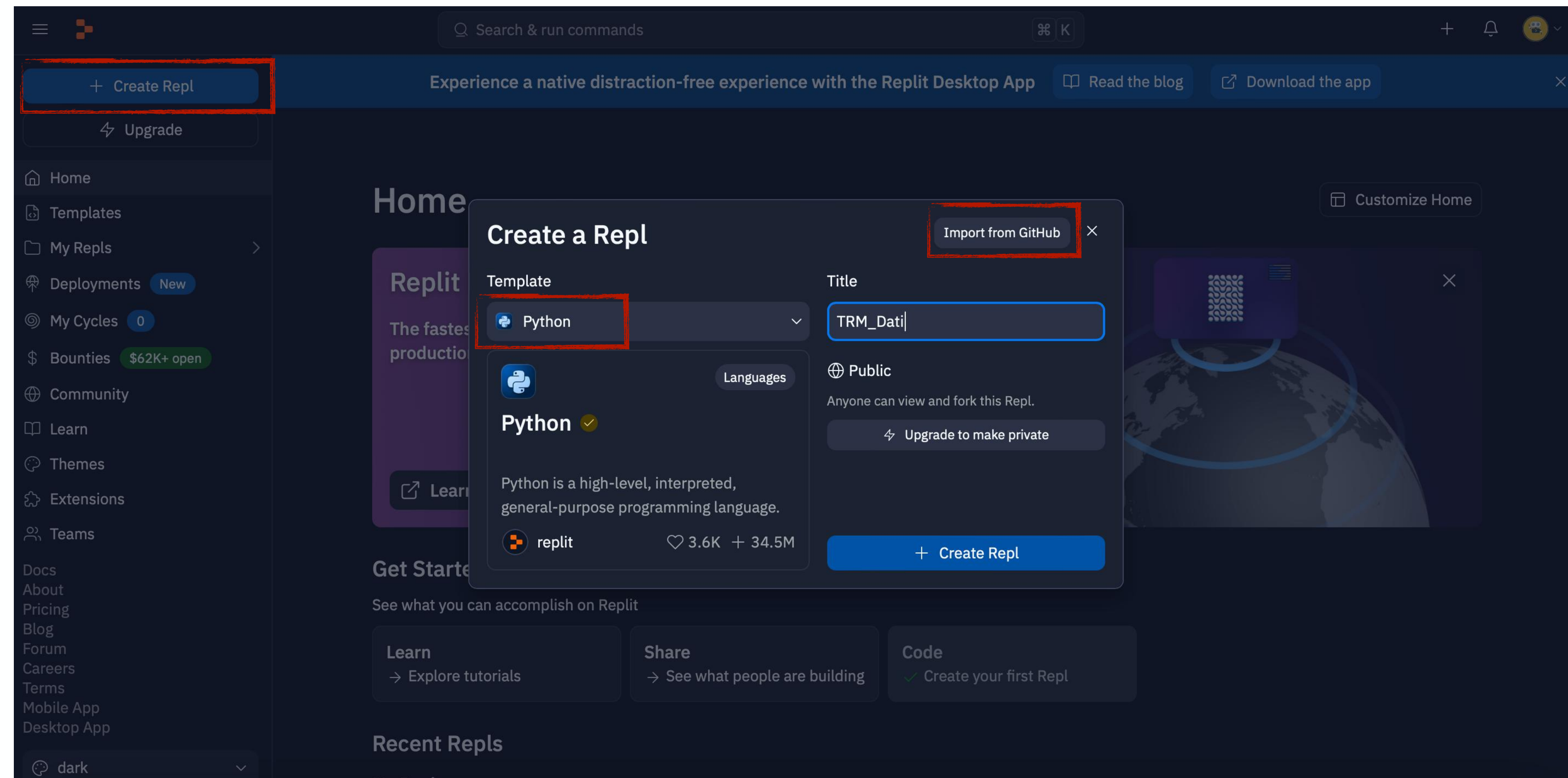
Register on Git

Create a public repository on Git

Register on repl.it (replit.com)

Create a new repl project by importing from Git the repository you have just created there

Set Python as default language



Other tools: Integrated Development Environment

Setup a working environment:

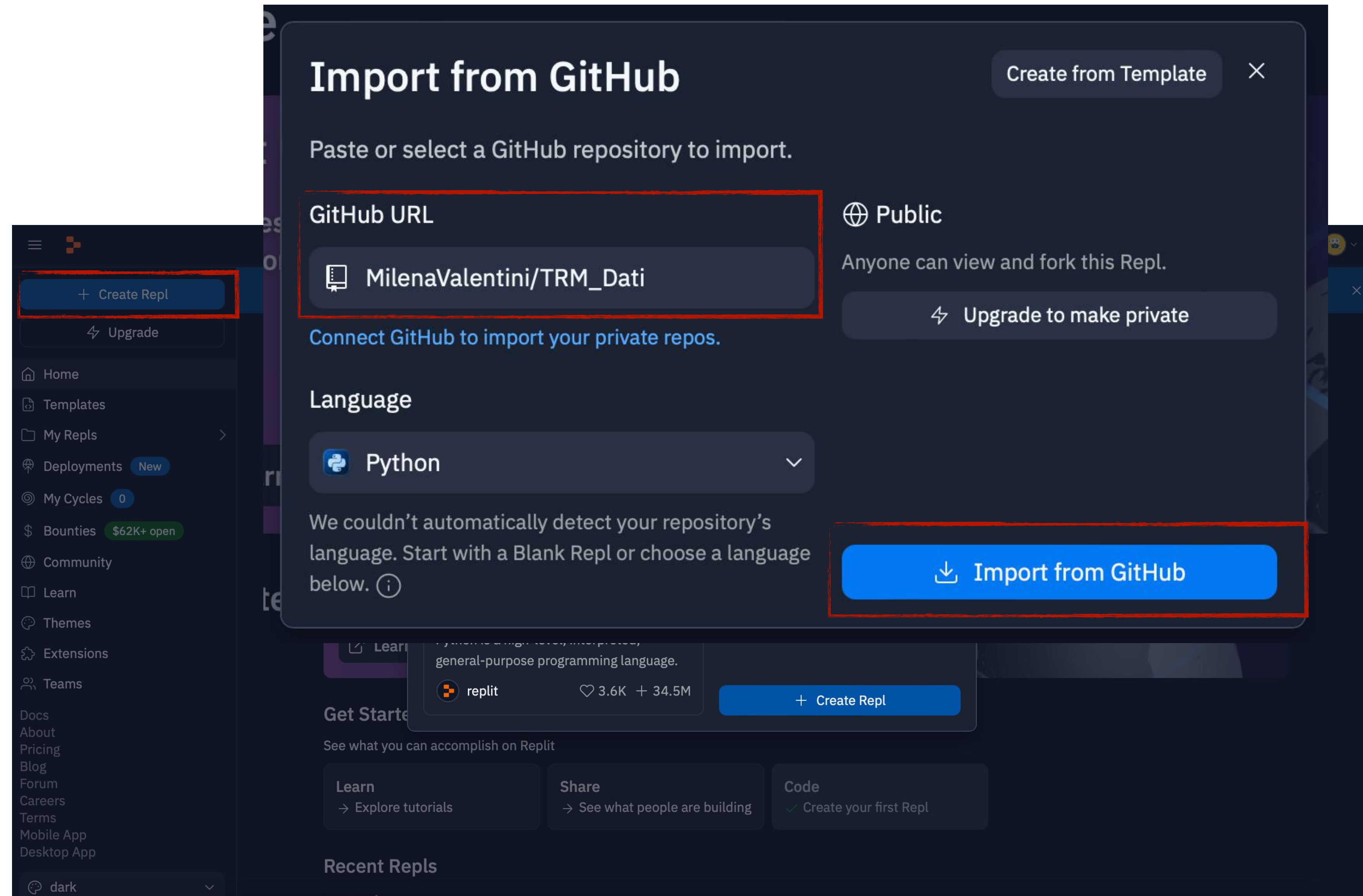
Register on Git

Create a public repository on Git

Register on repl.it (replit.com)

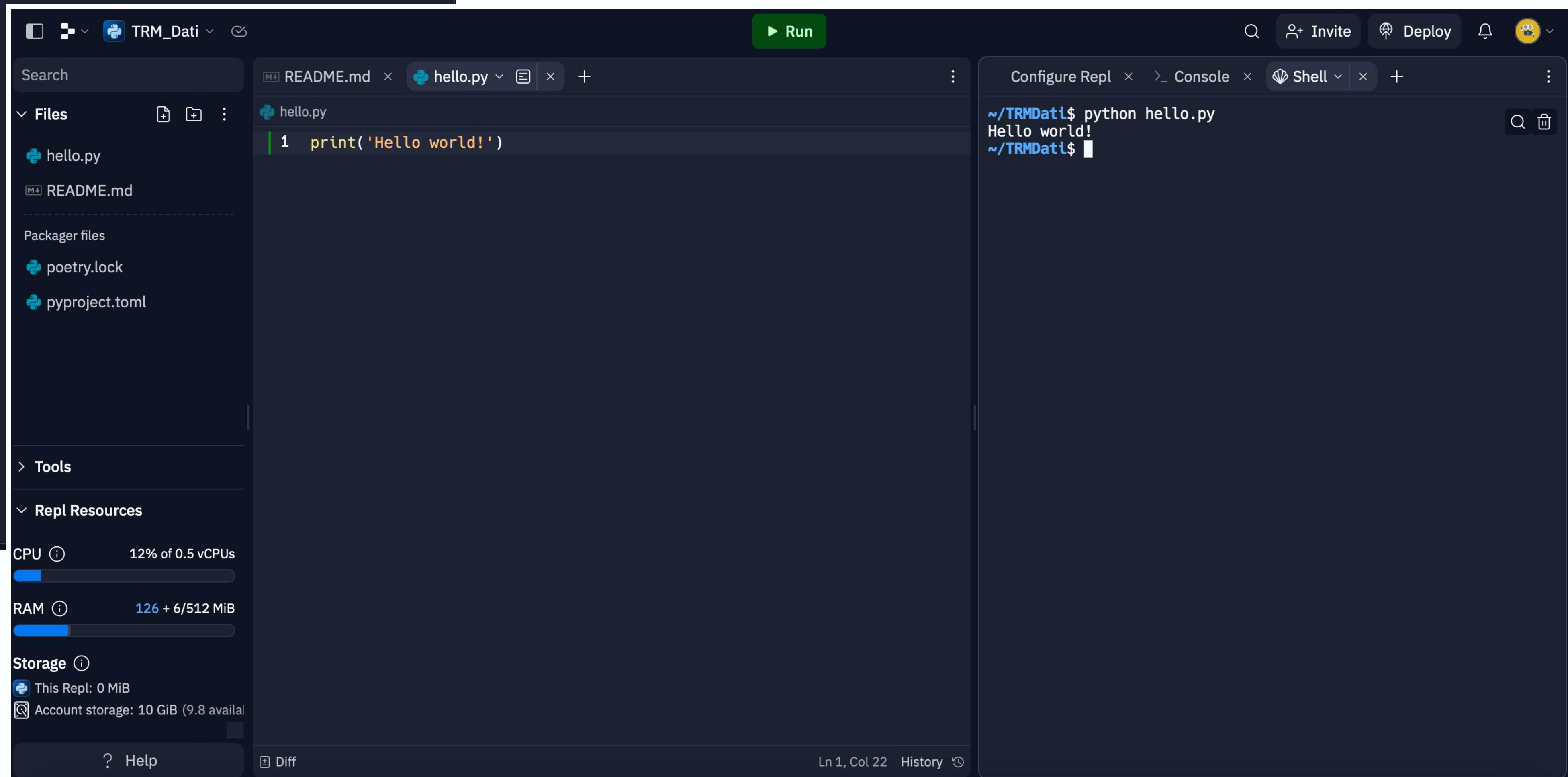
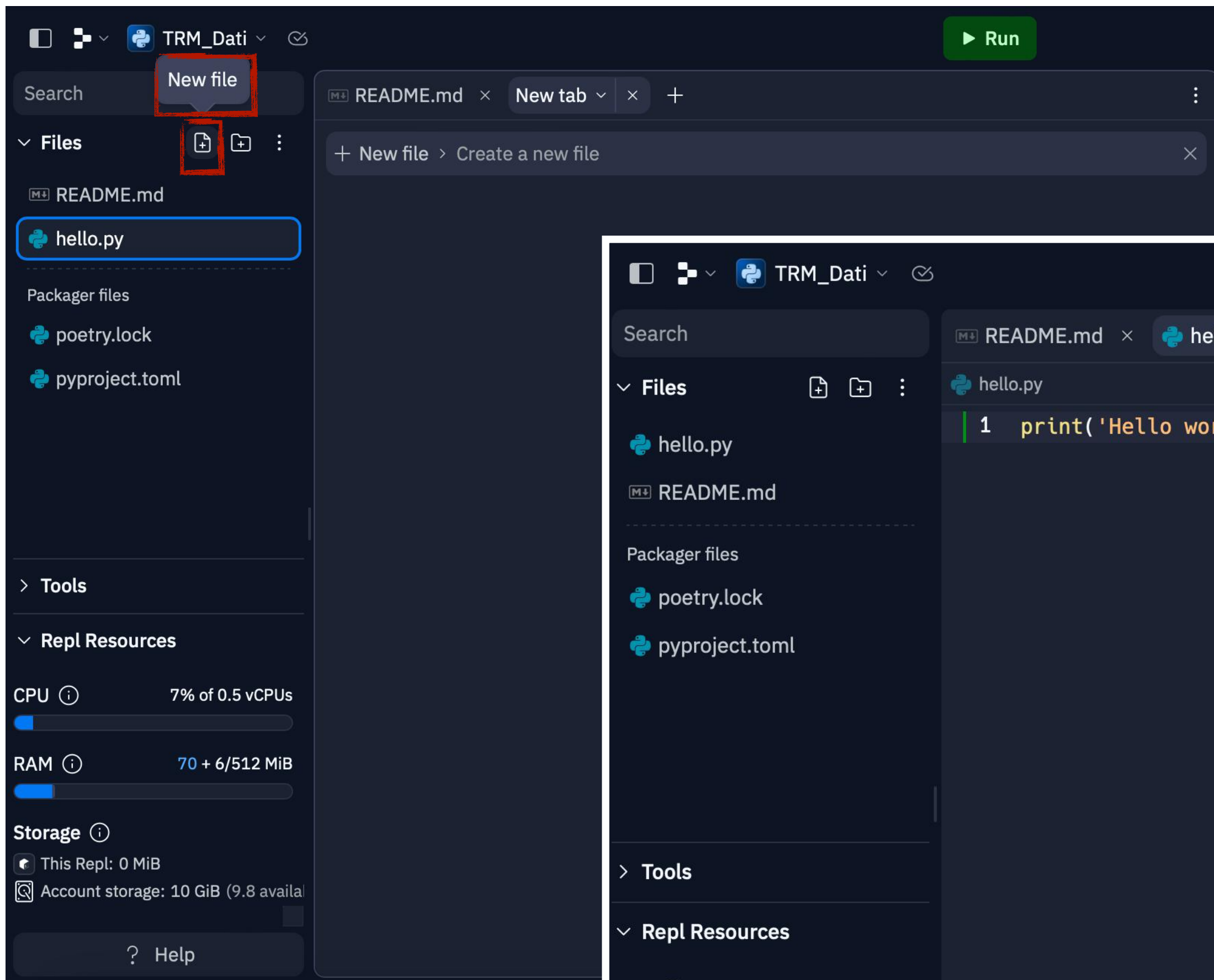
Create a new repl project by importing from Git the repository you have just created there

Set Python as default language



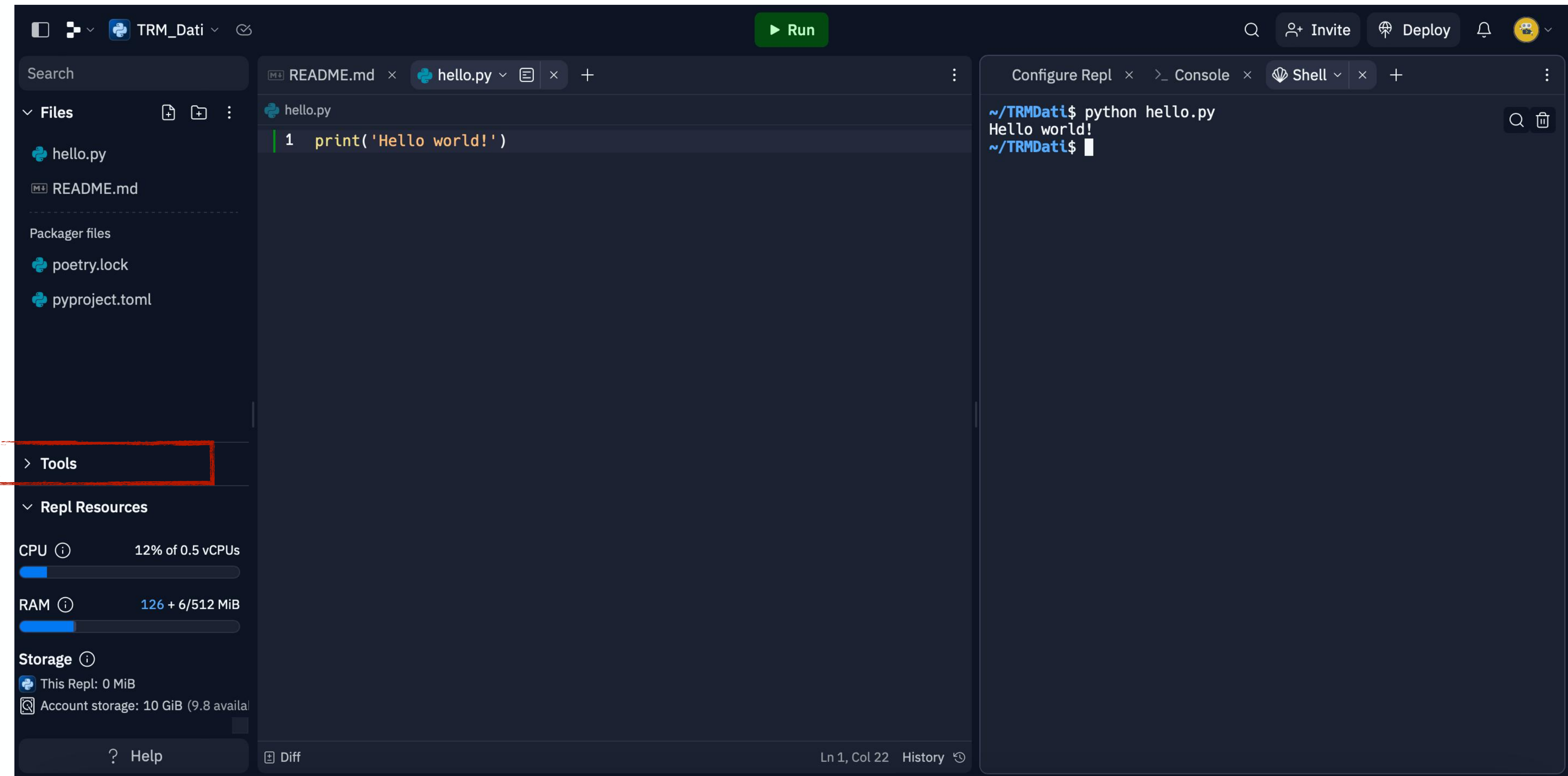
Other tools: Integrated Development Environment

Create a new file to produce the first working script



Other tools: Integrated Development Environment

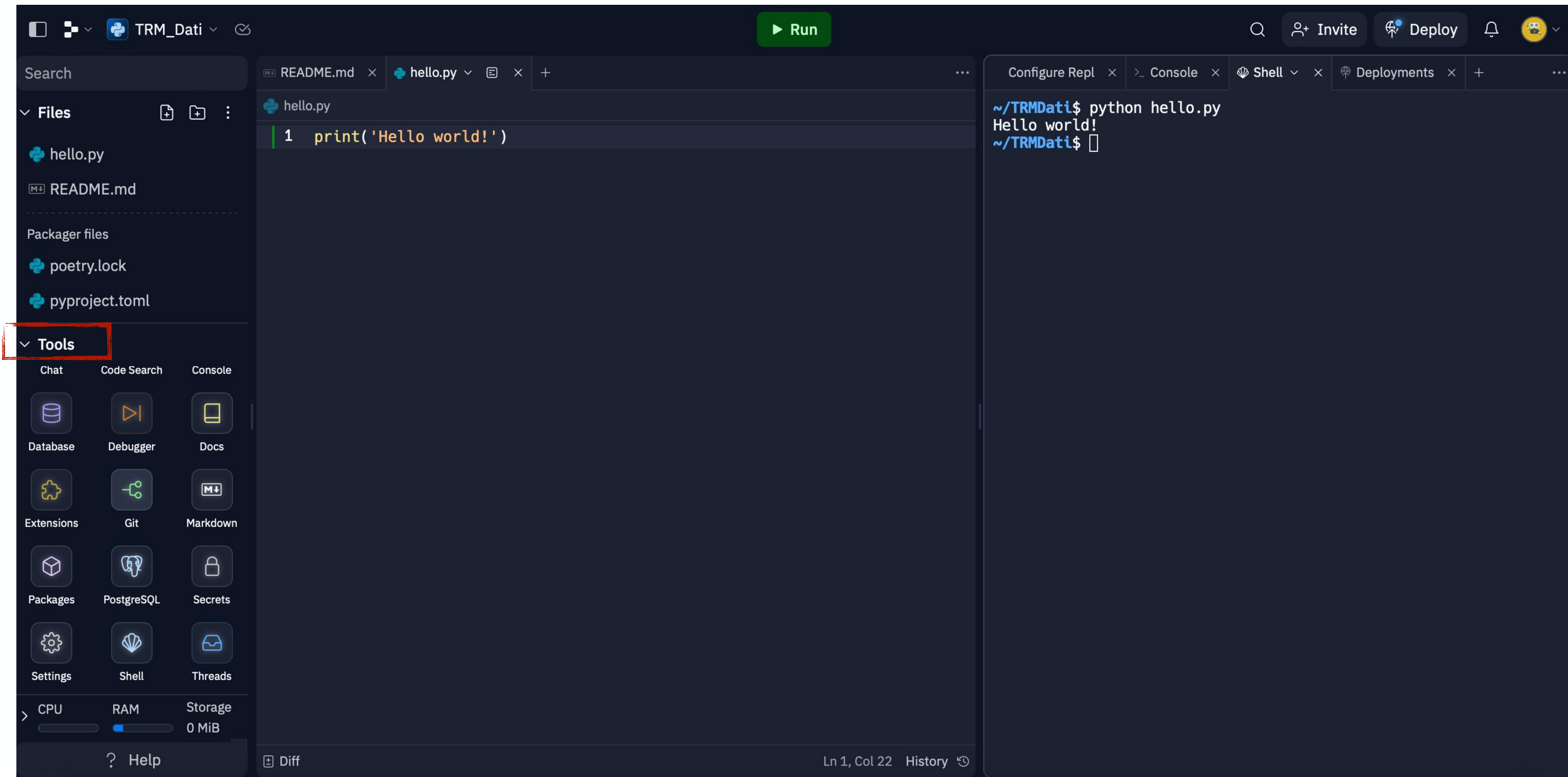
Connecting replit to Git



Other useful material: <https://replit.com/talk/learn/Replit-Git-Tutorial/23331>

Other tools: Integrated Development Environment

Connecting replit to Git




Other tools: Integrated Development Environment

Connecting replit to Git

The screenshot displays the Replit IDE interface. On the left sidebar, the 'Tools' section is expanded, and the 'Git' icon is highlighted with a red box. The main editor area shows a Python file named 'hello.py' with the code `print('Hello world!')`. On the right-hand side, the 'main' branch view is active, and a notification box with a warning icon states: 'Can't push or pull from GitHub. This Repl has a GitHub repository as a remote, but you are not connected to GitHub.' Below this notification is a blue button labeled 'Connect to GitHub'. The bottom right panel shows the 'Commit' section with a list of 5 changed files: `.replit`, `hello.py`, `poetry.lock`, `pyproject.toml`, and `replit.nix`, all marked as 'Added'.

Other tools: Integrated Development Environment



Install & Authorize Replit

Install & Authorize on your personal account Milena Valentini

All repositories
This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

Only select repositories
Select at least one repository.
Also includes public repositories (read-only).

with these permissions:

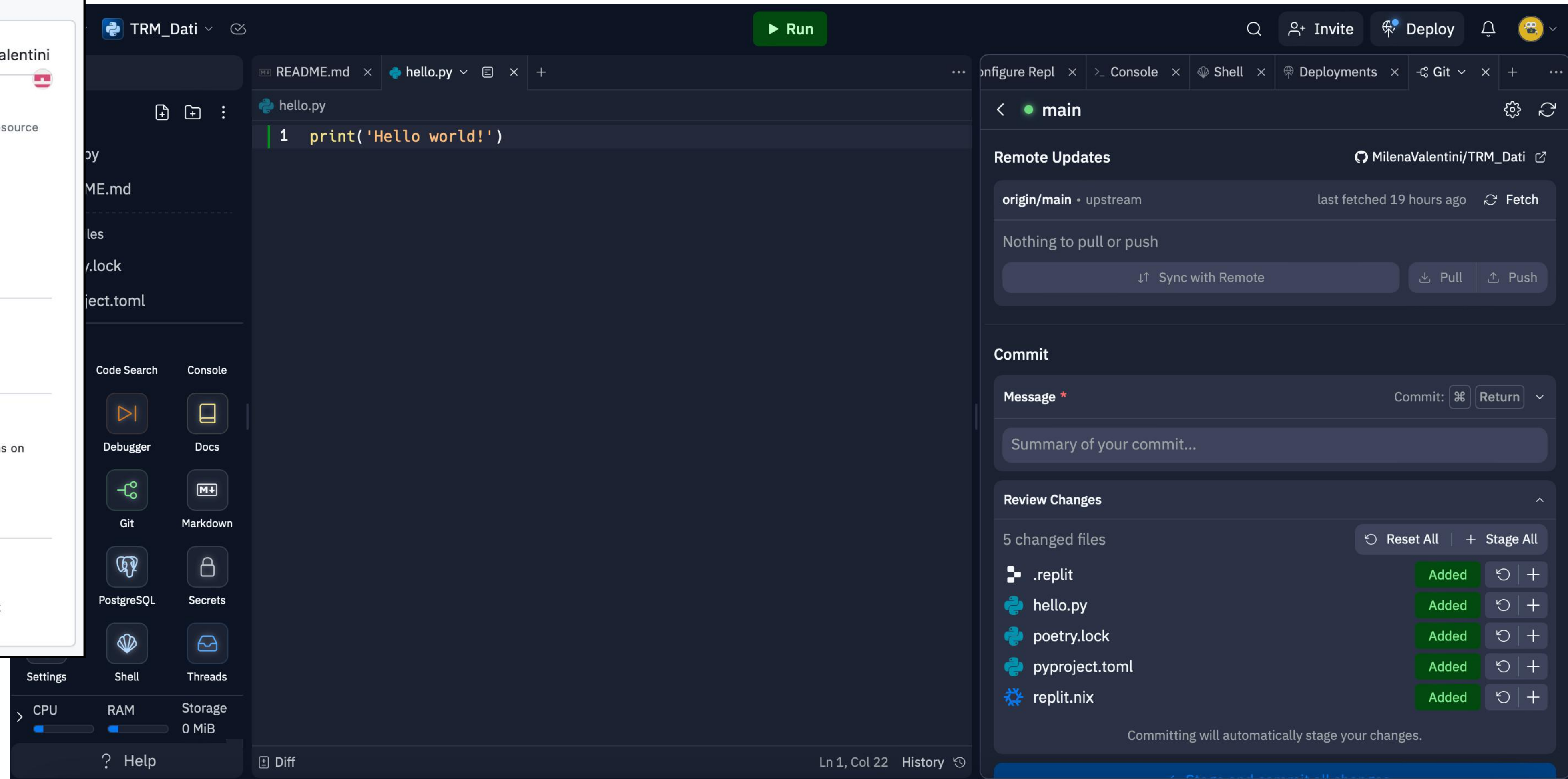
- ✓ Read access to metadata
- ✓ Read and write access to administration and code

User permissions
Installing and authorizing Replit immediately grants these permissions on your account: MilenaValentini.

- ✓ Read access to email addresses

Install & Authorize Cancel

Next: you'll be redirected to <https://replit.com/auth/github/callback>



The screenshot displays the Replit IDE interface. The main editor shows a Python file named `hello.py` with the following code:

```
1 print('Hello world!')
```

The interface includes a sidebar with various tools like Code Search, Console, Debugger, Docs, Git, Markdown, PostgreSQL, Secrets, Shell, and Threads. At the bottom, there are system resource indicators for CPU, RAM, and Storage (0 MiB).

On the right side, the Git interface is visible, showing the current branch as `main` and the remote as `MilenaValentini/TRM_Dati`. The commit dialog is open, with a message field containing "Summary of your commit..." and a list of 5 changed files:

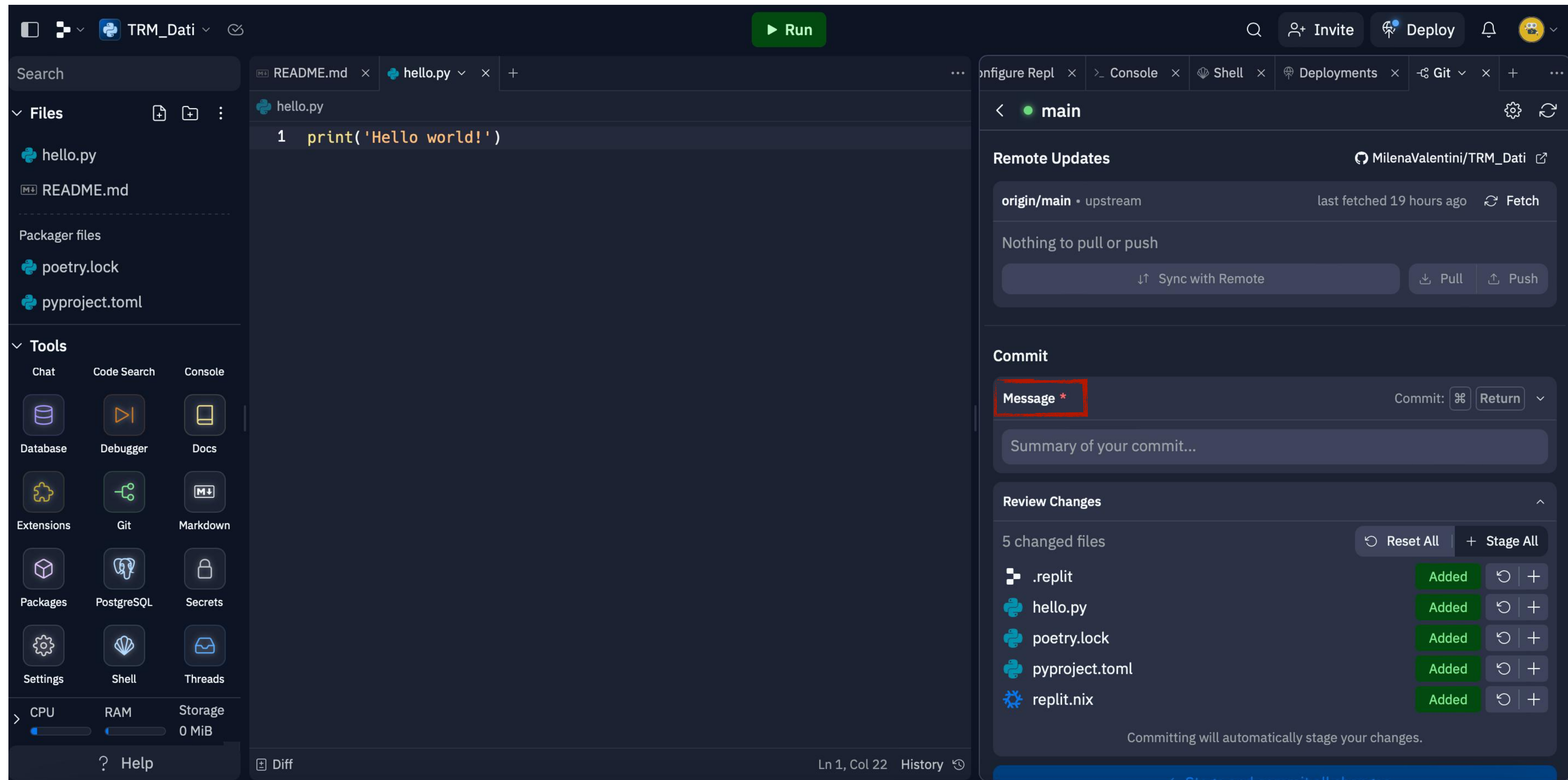
File	Status	Actions
<code>.replit</code>	Added	↺ +
<code>hello.py</code>	Added	↺ +
<code>poetry.lock</code>	Added	↺ +
<code>pyproject.toml</code>	Added	↺ +
<code>replit.nix</code>	Added	↺ +

At the bottom of the commit dialog, it states: "Committing will automatically stage your changes."

Authorize Replit on Git

Other tools: Integrated Development Environment

Add message to describe your commit

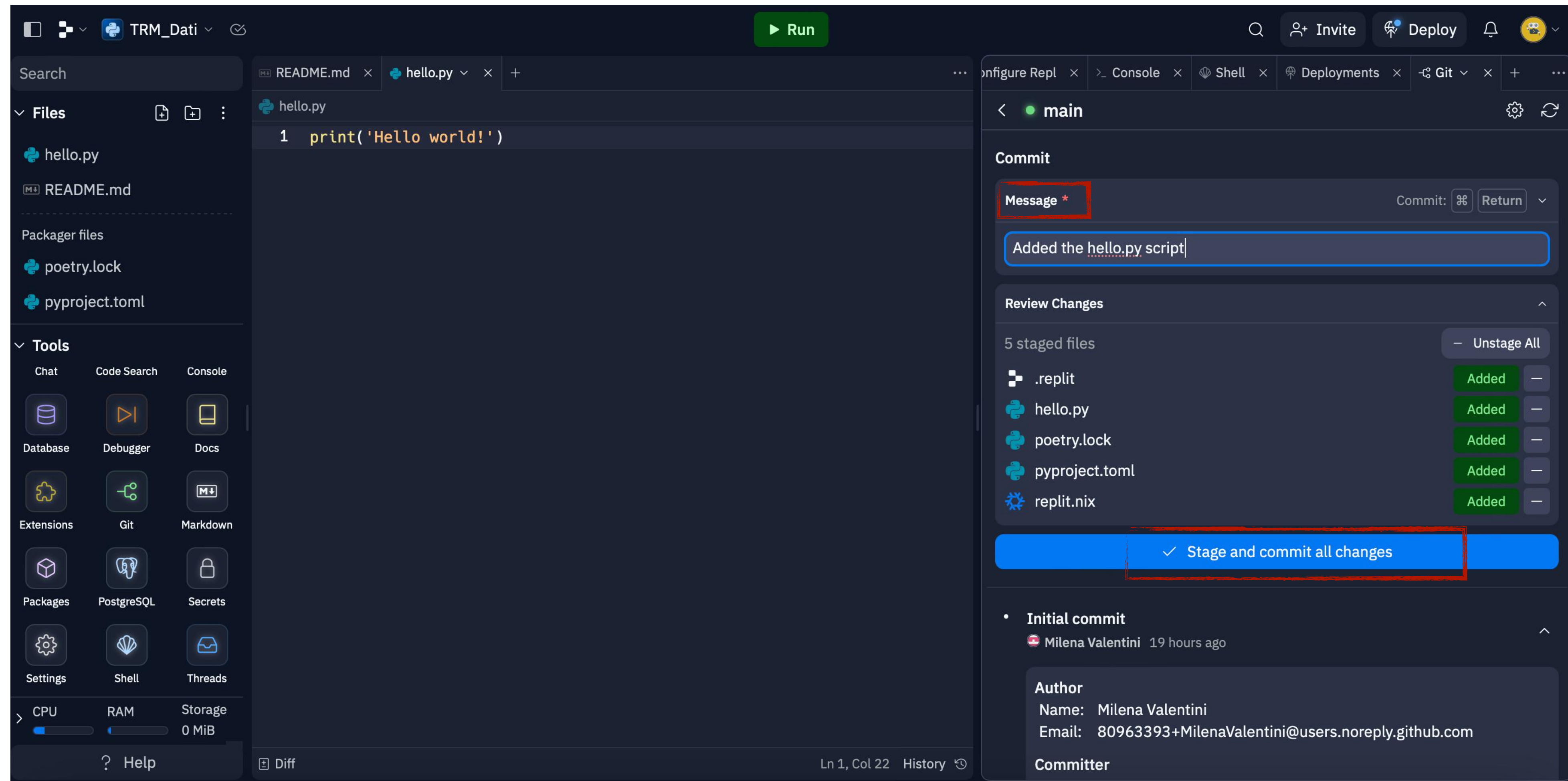


Other tools: Integrated Development Environment

Add message to describe your commit

Stage files to commit (the hello.py script, and additional files internal to replit plus related to libraries)

Commit



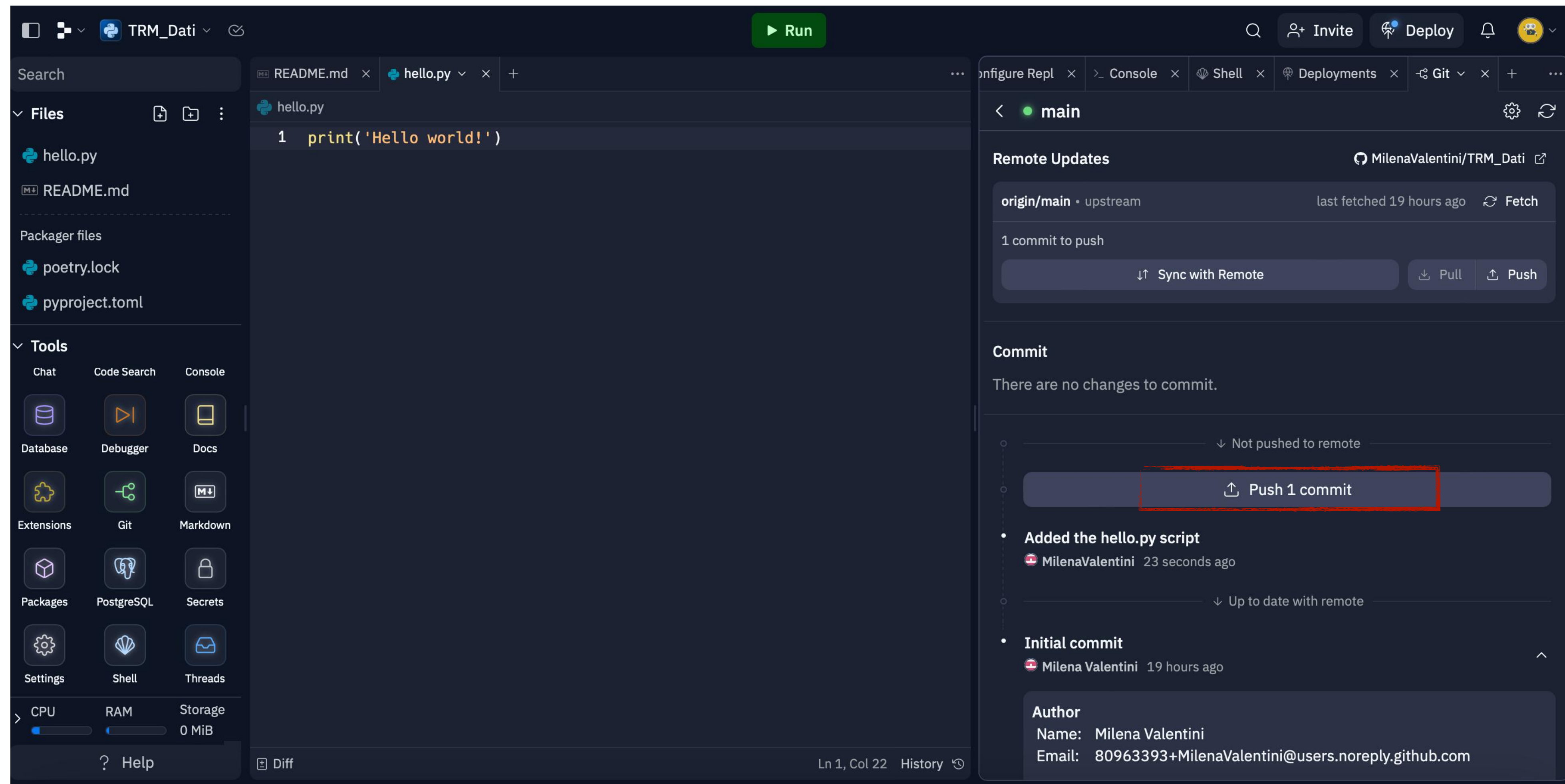
Other tools: Integrated Development Environment

Add message to describe your commit

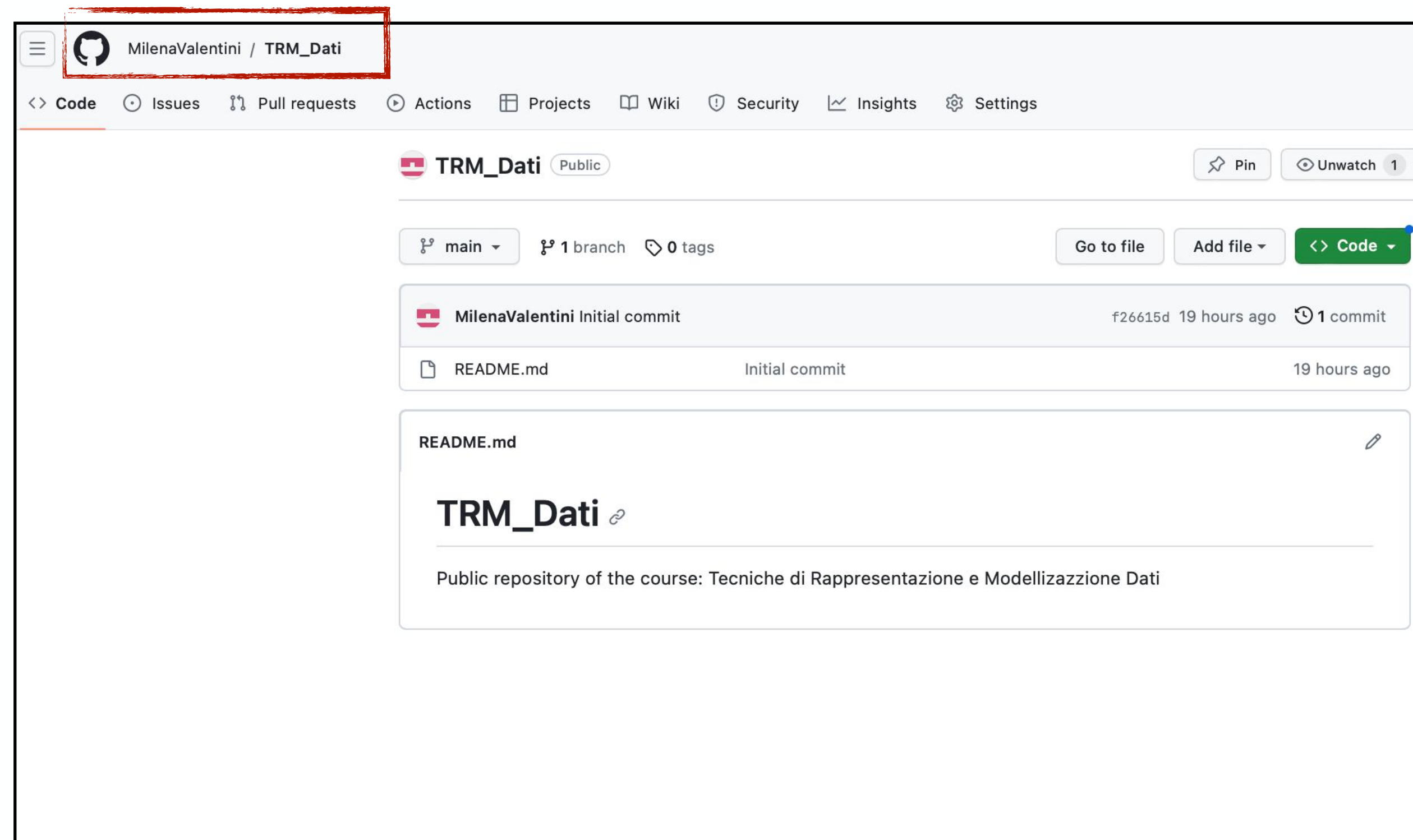
Stage files to commit (the hello.py script, and additional files internal to replit plus related to libraries)

Commit

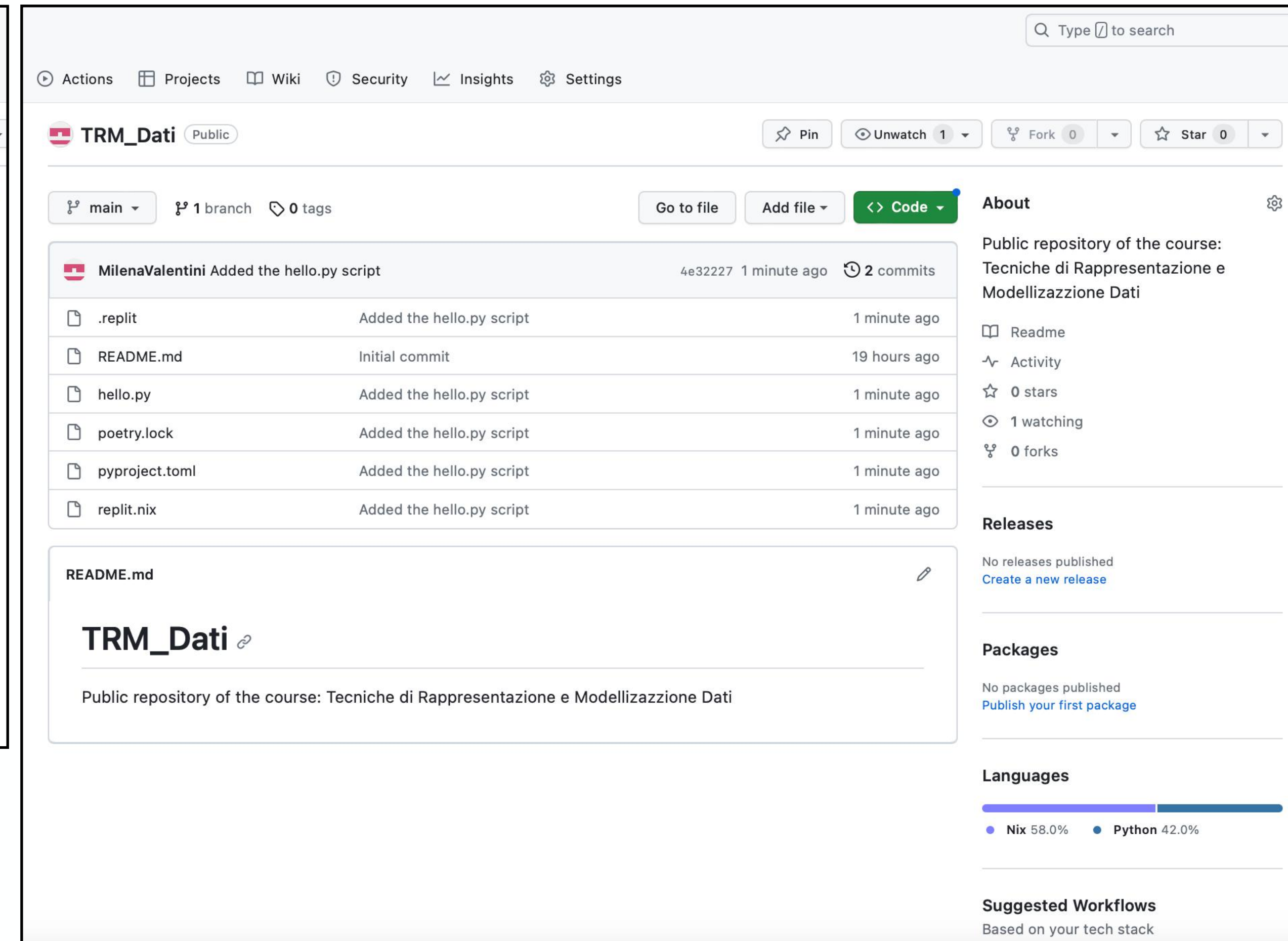
Push to Git



Other tools: Integrated Development Environment



This screenshot shows the GitHub repository page for MilenaValentini / TRM_Dati. The repository is public and has 1 branch (main) and 0 tags. The commit history shows an initial commit by MilenaValentini 19 hours ago, with 1 commit. The README.md file is visible, containing the text "TRM_Dati" and "Public repository of the course: Tecniche di Rappresentazione e Modellizzazione Dati". The user profile "MilenaValentini" is highlighted with a red box.



This screenshot shows the GitHub repository page for MilenaValentini / TRM_Dati after an update. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a new commit by MilenaValentini 1 minute ago, with 2 commits. The commit message is "Added the hello.py script". The README.md file is visible, containing the text "TRM_Dati" and "Public repository of the course: Tecniche di Rappresentazione e Modellizzazione Dati". The right sidebar shows repository statistics: 0 stars, 1 watching, 0 forks, and 0 releases. The languages section shows Nix at 58.0% and Python at 42.0%.

The Git repository has been successfully updated

Other tools: Integrated Development Environment

Make sure to configure the hidden file `.replit` as follows:

language = "bash"
run = "/bin/bash"

Stage, commit and push it to Git

The image displays the Replit IDE interface. The top-left pane shows the file explorer with a context menu open over the `.replit` file. The menu options include 'Upload file', 'Upload folder', 'Open pane', 'Download as zip', 'Hide hidden files', and 'Collapse all'. The top-right pane shows the 'Run' button and the 'Commit' dialog. The commit message is 'Added the hello.py script'. The 'Review Changes' section shows 5 staged files: `.replit`, `hello.py`, `poetry.lock`, `pyproject.toml`, and `replit.nix`. A blue button at the bottom of the commit dialog says 'Stage and commit all changes'. The bottom-right pane shows the commit history, including an 'Initial commit' by Milena Valentini 19 hours ago, with author details: Name: Milena Valentini, Email: 80963393+MilenaValentini@users.noreply.github.com.