

# TECNICHE DI RAPPRESENTAZIONE E MODELLIZZAZIONE DEI DATI

– Part 1 –

(2 CFU out of 6 total CFU)

**Link moodle:** <https://moodle2.units.it/course/view.php?id=14486>

Teams code: d2cmkh8

# Intro

## Timeslots:


**Wednesdays:** 14:15 → 15:30 break 15:45 → 17:00


**Fridays:** 9:15 → 10:30 break 10:45 → 12:00

Settembre						
Lu	Ma	Me	Gi	Ve	Sa	Do
						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						

Ottobre						
Lu	Ma	Me	Gi	Ve	Sa	Do
	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			

Office:

 Astronomical Observatory of Trieste  
via G.B. Tiepolo 11, 34143 - Trieste  
Office: 2100 (first floor)  
Phone: 040 3199283

 Please, always send an email to [milena.valentini@units.it](mailto:milena.valentini@units.it) to schedule a meeting

# Intro

---

## 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:** Introduction to operative systems, main Linux commands, working environments, Anaconda

**Lecture 2:** Jupyter Notebook and Git

**Lecture 3:** Essentials of bash

**Lecture 4:** Python - introduction, main concepts, errors, variables, scripts

**Lecture 5:** Python - operators, conditions, functions

**Lecture 6:** Python - strings, lists, tuples, dictionaries

**Lecture 7:** Python - data structures, how to read/write from/a file, input/output, input from command line

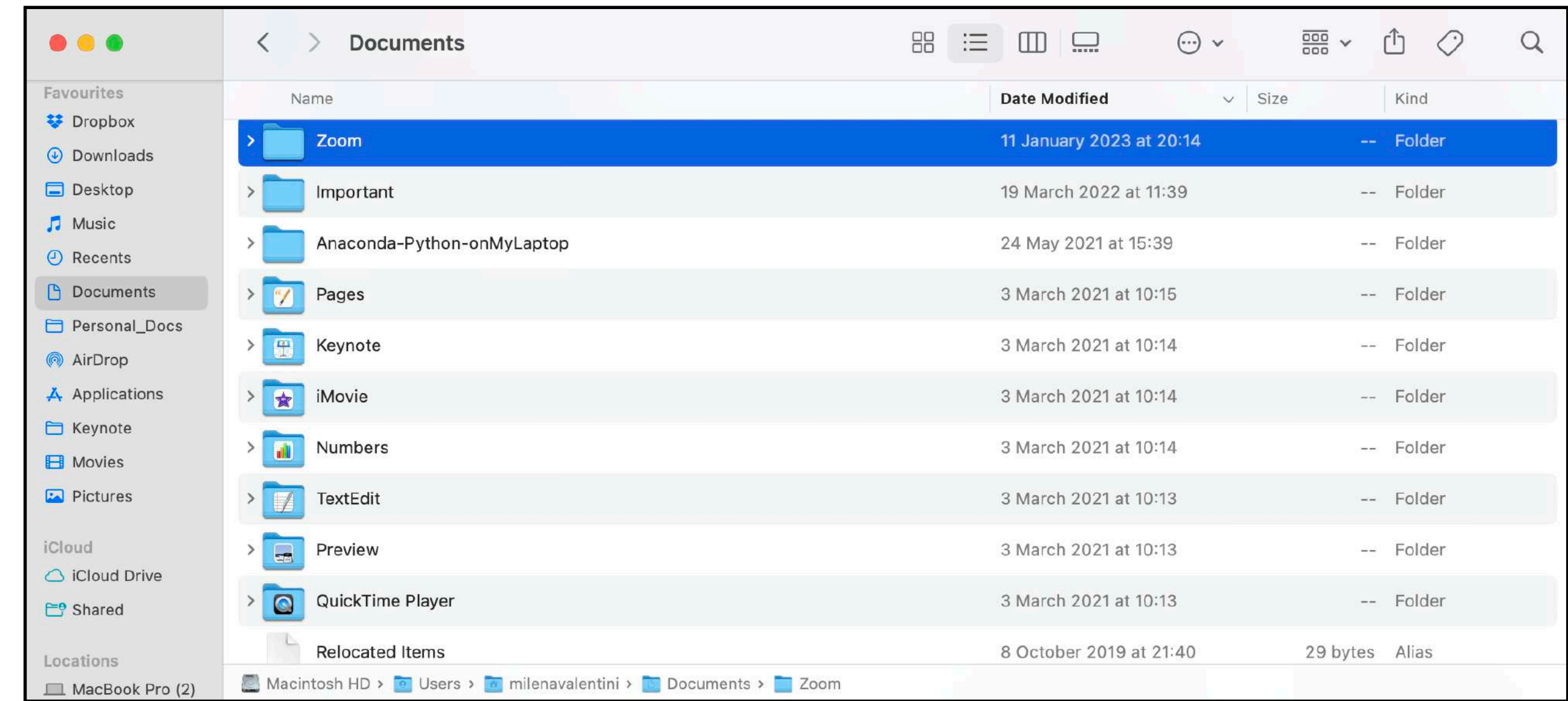
**Lecture 8:** Python - module import, libraries, arrays, objects, try/except; Python exercises



# Useful working tools

## File browser or manager:

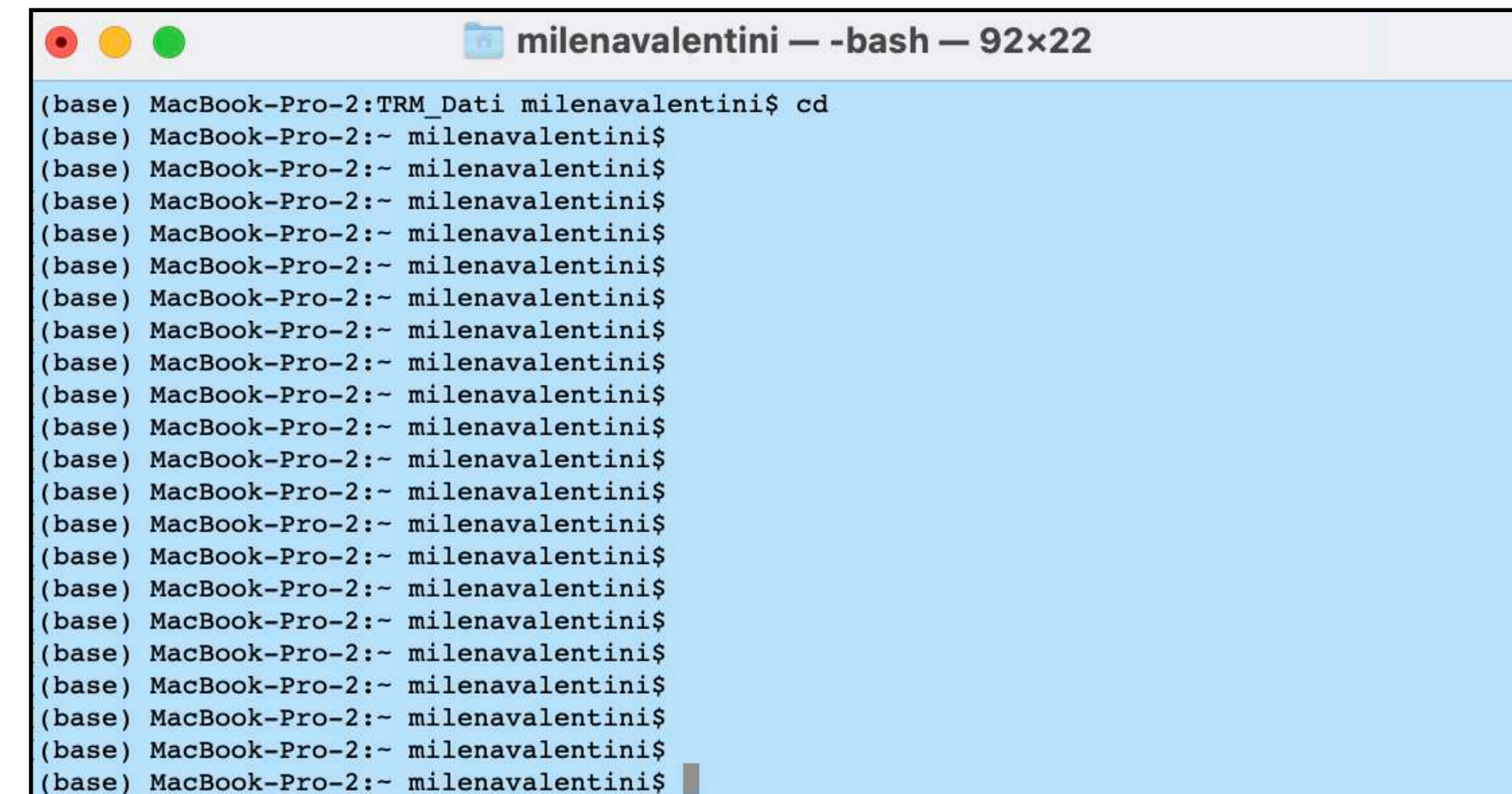
program of an Operative System (OS) which provides you with a user interface to manage folders and files



## Shell/terminal/console/command prompt:

interface to interact with the computer via command line

without relying on graphical user interfaces

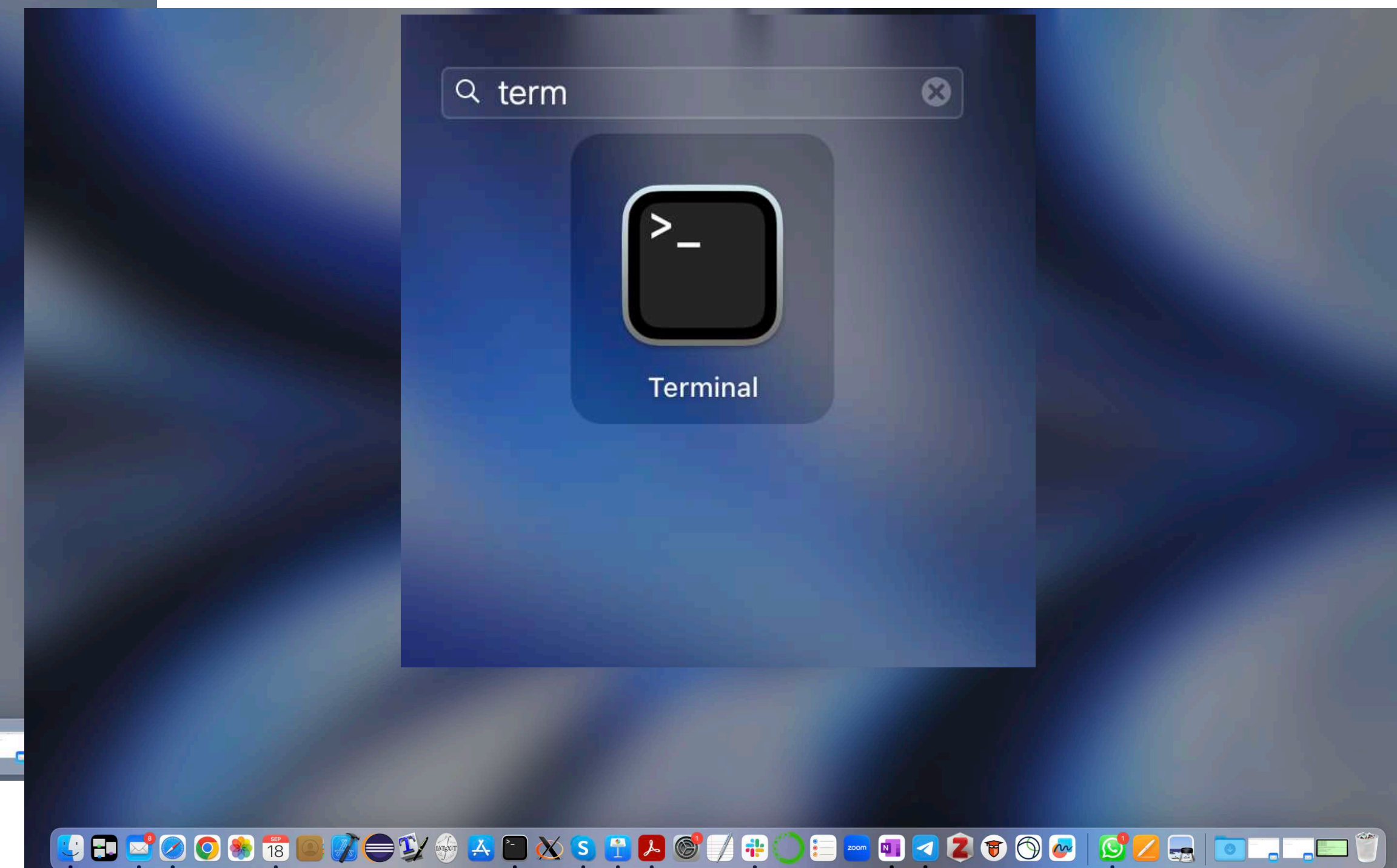
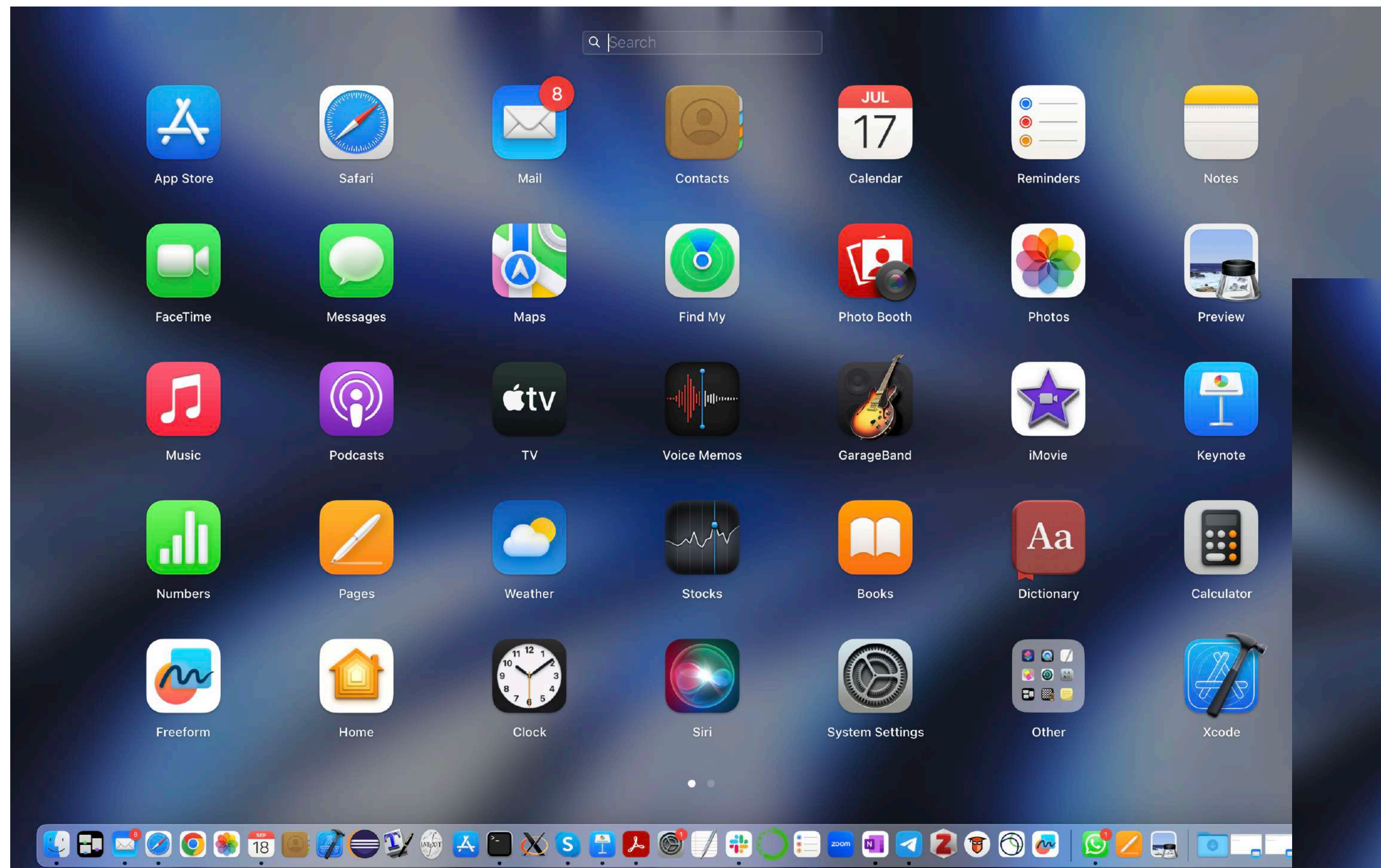




# The command line tool

OS: Mac

Look for the Terminal among the applications

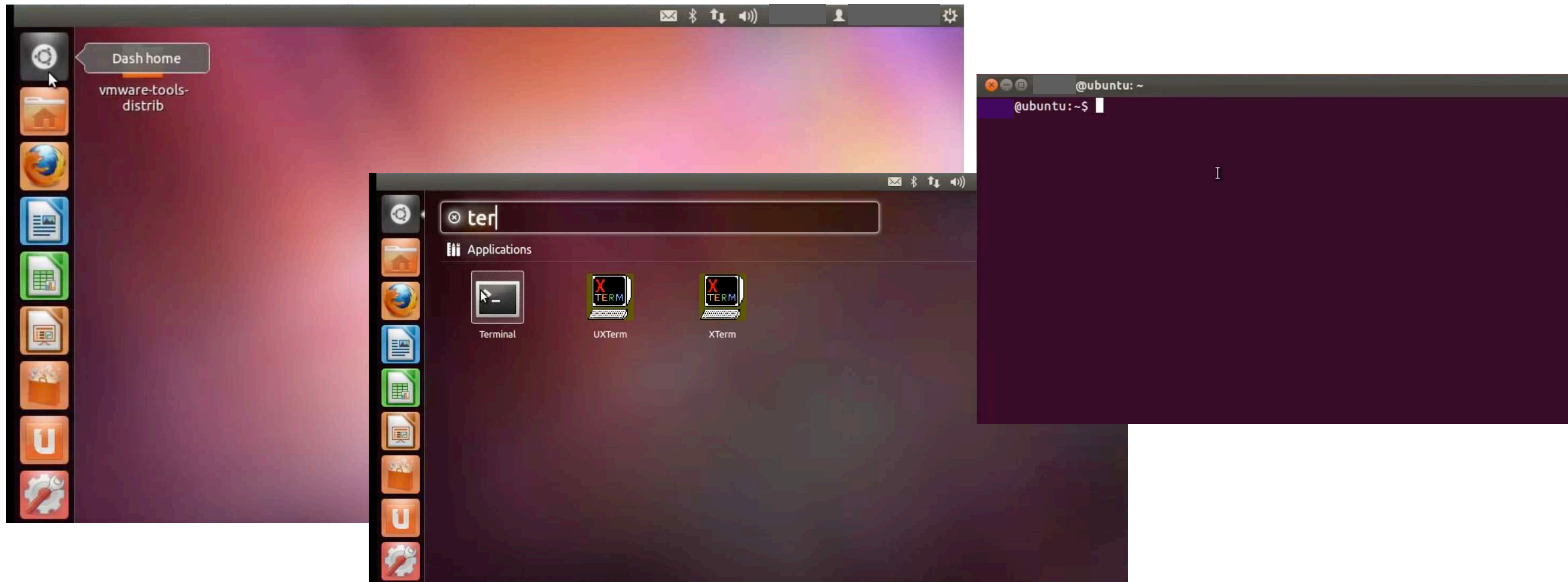




# The command line tool

OS: Linux/Unix

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

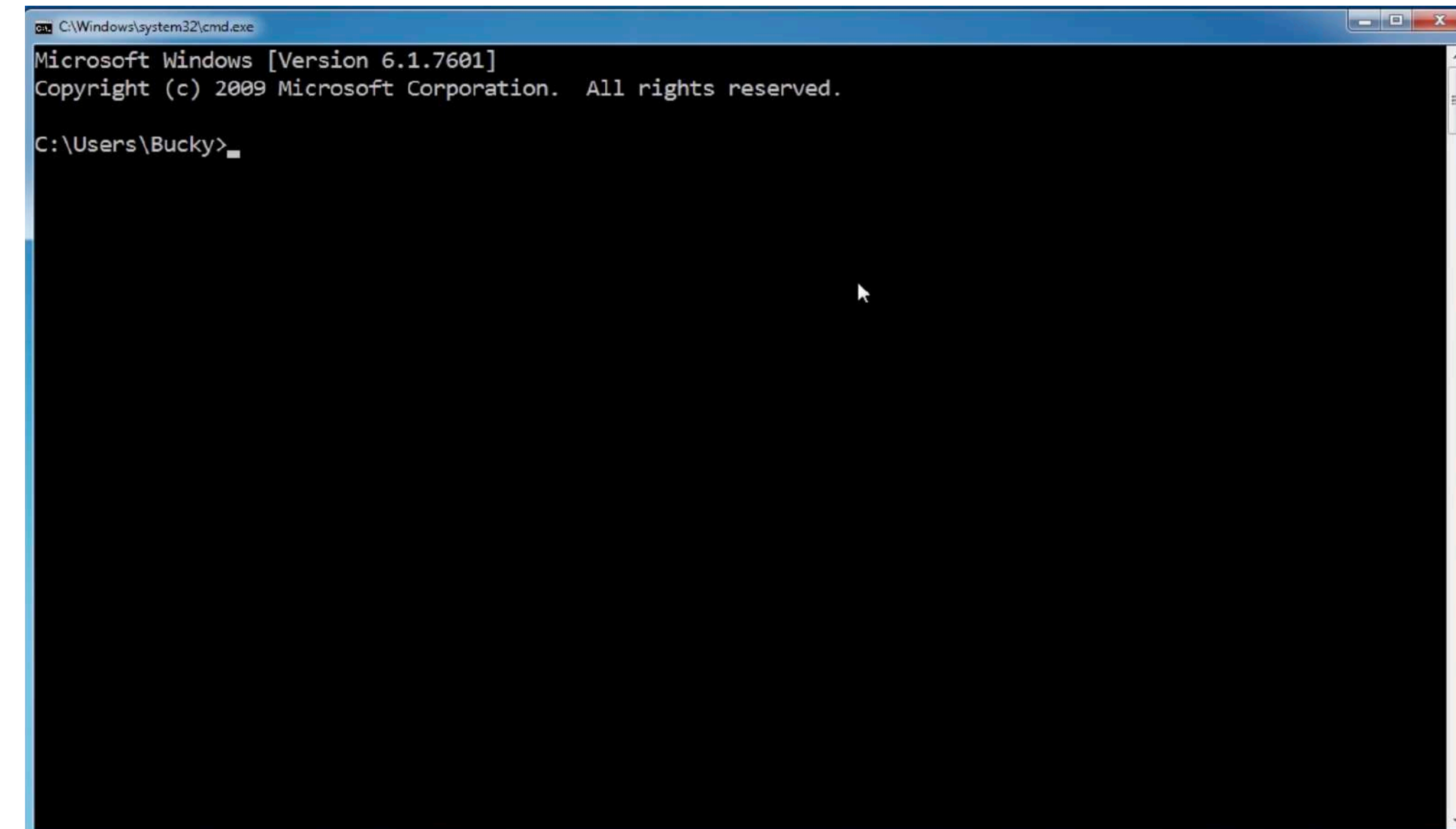
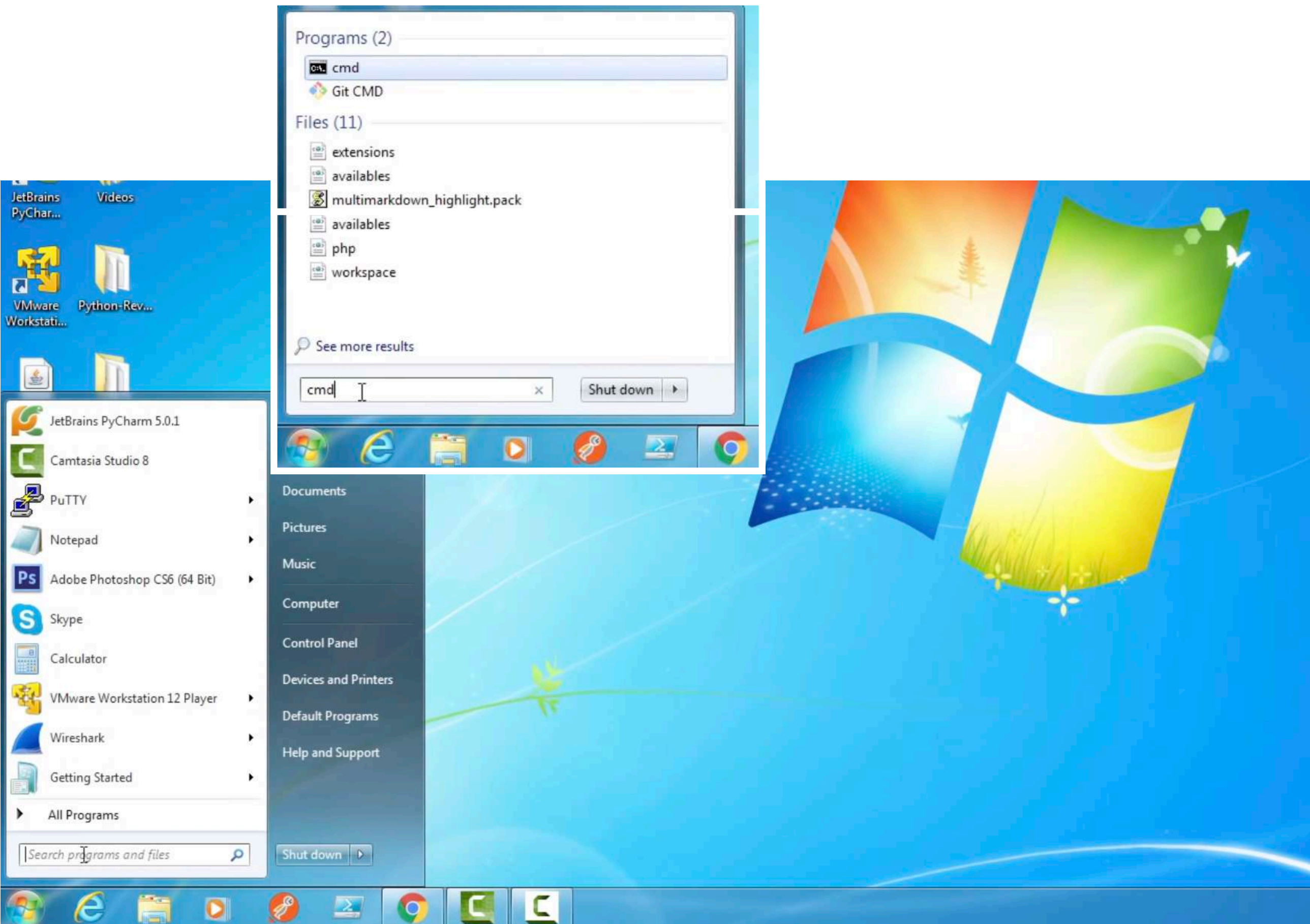
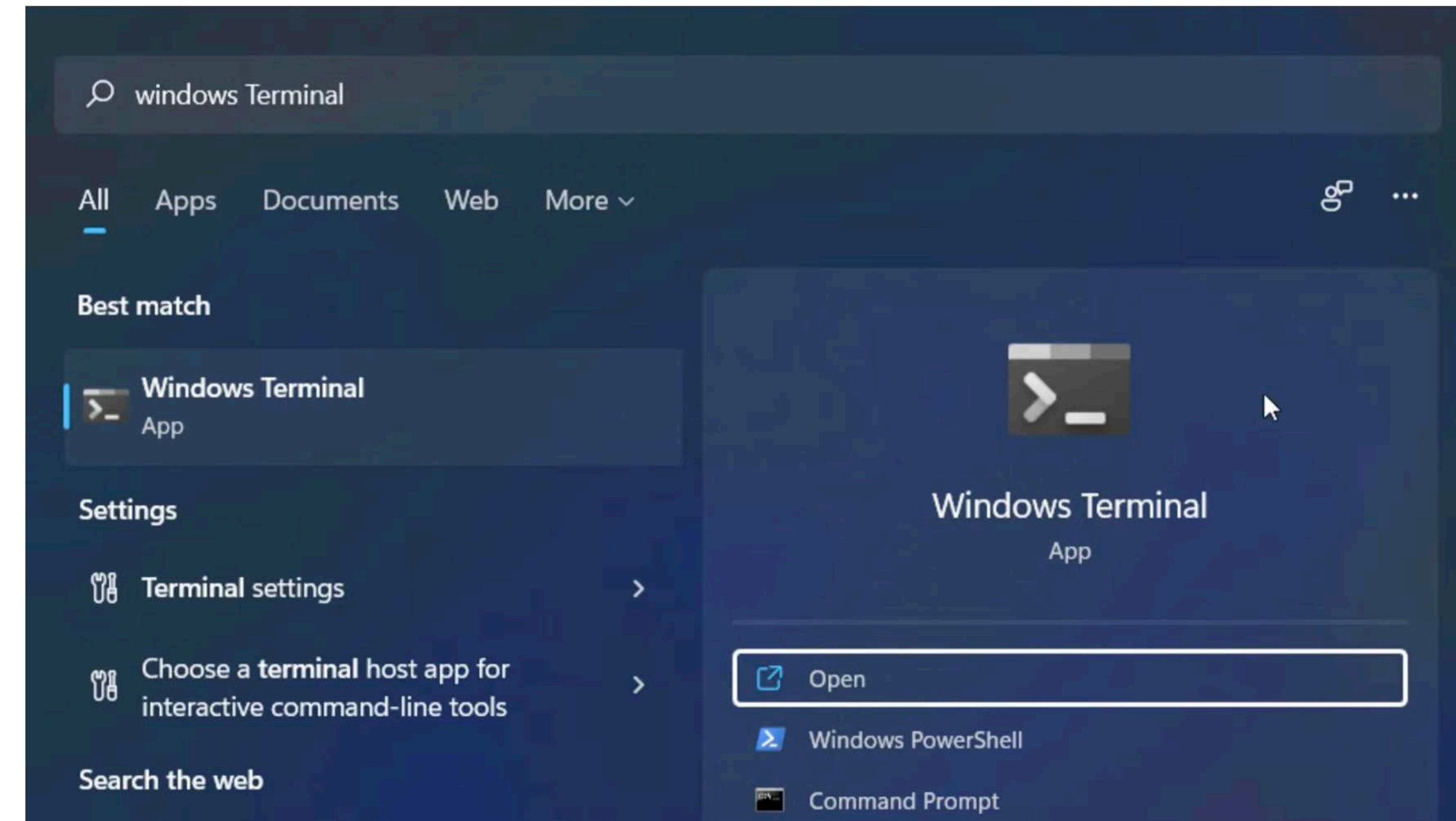




# The command line tool

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$
```



# 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$
```

All commands have a return code (0 if the command line execution has been completed successfully)  
To access the content of the return code: echo \$?

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo $?
0
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



# Command line

Command, general form:

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

All UNIX commands  
have a help documentation  
**man** [command]

```
LS(1)                                General Commands Manual                                LS(1)

NAME
  ls - list directory contents

SYNOPSIS
  ls [-@ABCFGHILOPRSTUWabcdefghijklmnopqrstuvwxyz1%,] [--color=when] [-D format] [file ...]

DESCRIPTION
  For each operand that names a file of a type other than directory, ls displays its name as well as any requested, associated information. For each operand that names a file of type directory, ls displays the names of files contained within that directory, as well as any requested, associated information.

  If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical order.

  The following options are available:

  -@      Display extended attribute keys and sizes in long (-l) output.

  -A      Include directory entries whose names begin with a dot ('.') except for . and ..
          Automatically set for the super-user unless -I is specified.

  -B      Force printing of non-printable characters (as defined by ctype(3) and current
          locale settings) in file names as \xxx, where xxx is the numeric value of the
  :
```



# 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$
```

example:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -ltr
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 script_2.bash
```

**-l** (The lowercase letter "ell".) List files in the long format, as described in the [The Long Format](#) subsection below.

**-t** Sort by descending time modified (most recently modified first). If two files have the same modification timestamp, sort their names in ascending lexicographical order. The **-r** option reverses both of these sort orders.

**-r** Reverse the order of the sort.



# Command line

The **echo** command displays on the screen (i.e. standard output) the strings provided as arguments

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo The sky is blue  
The sky is blue  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

command

line of text

**echo** can also output the value of a specific variable, if this is preceded by the \$ character:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ a=3  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo The number is $a  
The number is 3  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ _
```

variable



# Command line

---

A few useful commands:

The **passwd** command allows you to change the user's password

The **who** command prints information about users currently logged in

**whoami** outputs the username associated to the actual user ID

The **export** command exports environment variables

If called with no arguments, **export** prints all the variables in the shell environment

The command **unset** frees the variables

# Command line

---

The **export** command exports environment variables

Environment variables are variables which set and define the behaviour of the environment. They are typically accessed through the shell.

To view all exported variables on your shell: `milenaValentini$ export -p`

They are set when you open a new shell session. If you change any of the variable values at anytime, the export command allows you to update the current shell session about the change.

Some among the commonly used environment variables:

`$USER`, `$PATH`, `$PWD`, `$LANG`, `$UID`, `$SHELL`, `$HOME`

The `$PATH` variable contains search path for commands and executables.

The **pwd** command returns the actual path/current working directory



# Command line

give the command → locate the command → execute the command

To locate the command → search path  
↓  
list of directories  
to locate commands

Modify the search path → \$PATH variable

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo $PATH  
/opt/homebrew/bin:/opt/homebrew/sbin:/Users/milenavalentini/opt/anaconda3/bin:/Users/milenavalentini/opt/anaconda3/condabin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/TeX/texbin
```

The system checks these directories from left to right when running a program.

How to add a directory to the \$PATH environment variable:

```
milenavalentini$ export PATH=path_to_add:$PATH
```

# Command line

---

The **ls** command is used to inspect properties of files and directories.

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(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$
```

**ls** takes the names of one or more filenames or directories as arguments.

The file and directory names are optional: if not provided, UNIX processes the current directory. By default, the list of files within a directory is sorted by filename (the sort order can be modified using relevant flags).

Note that **ls** does not process files starting with `.` (hidden files, mainly used to store user preferences) unless you use the `-a` flag (same for those starting with `..`)

To use the **ls** command on a directory and its files, read permissions on those directory and files are needed



# Command line

To manipulate files and manage directories and their content, the user needs permissions

Main rights of a user:

<b>r</b>	read
<b>w</b>	write
<b>x</b>	execute

Representation:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxr-xr-x  2 milenavalentini  staff   64 Sep 17 15:21 Useful
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

first digit: d identifies a folder, - a file

2nd, 3rd, 4th digits: readable, writable, executable by the owner (u, i.e. user)

5th, 6th, 7th digits: readable, writable, executable by the group (g, i.e. group)

8th, 9th, 10th digits: readable, writable, executable by everybody (a or o, i.e. all/others)



# Command line

---

To manipulate files and manage directories and their content, the user needs permissions

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxr-xr-x  2 milenavalentini  staff   64 Sep 17 15:21 Useful
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ _
```



# Command line

To manipulate files and manage directories and their content, the user needs permissions

How to change permissions  
by exploiting the chmod command:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt  
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod u+x file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt  
-rwxr--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod -r file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt  
--wx-----  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod +r file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt  
-rwxr--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod g+wx file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt  
-rwxrwxr--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



# Command line

To manipulate files and manage directories and their content, the user needs permissions

How to change permissions by exploiting the chmod command:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
--wx----- 1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod +r file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rwxr--r-- 1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod g+wx file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rwxrwxr-- 1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

How to deal with directories' permissions:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxr-xr-x  2 milenavalentini  staff  64 Sep 17 15:21 Useful
-rwxrwxr--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod -R u=rwx,go=rw Useful
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxrw-rw-  2 milenavalentini  staff  64 Sep 17 15:21 Useful
```



# Command line

---

The **touch** command creates a file

```
milenaValentini$ touch file_1.txt
```



# Command line

The **mkdir** command creates a directory

```
MKDIR(1)                                General Commands Manual                                MKDIR(1)

NAME
  mkdir - make directories

SYNOPSIS
  mkdir [-pv] [-m mode] directory_name ...

DESCRIPTION
  The mkdir utility creates the directories named as operands, in the order specified, using mode "rwxrwxrwx" (0777) as modified by the current umask(2).

  The options are as follows:

  -m mode      Set the file permission bits of the final created directory to the specified mode. The mode argument can be in any of the formats specified to the chmod(1) command. If a symbolic mode is specified, the operation characters '+' and '-' are interpreted relative to an initial mode of "a=rwx".

  -p             Create intermediate directories as required. If this option is not specified, the full path prefix of each operand must already exist. On the other hand, with this option specified, no error will be reported if a directory given as an operand already exists. Intermediate directories are created with permission bits of "rwxrwxrwx" (0777) as modified by the current umask, plus write and search permission for the owner.

  -v            Be verbose when creating directories, listing them as they are created.

  The user must have write permission in the parent directory.

EXIT STATUS
  The mkdir utility exits 0 on success, and >0 if an error occurs.
```



# Command line

The **mkdir** command creates a directory — examples:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l  
total 0  
drwxrw-rw-  2 milenavalentini  staff  64 Sep 17 15:21 Useful  
-rwxrwxr--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt  
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat  
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py  
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ mkdir Useful_extra  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l  
total 0  
drwxrw-rw-  2 milenavalentini  staff  64 Sep 17 15:21 Useful  
drwxr-xr-x  2 milenavalentini  staff  64 Sep 17 17:17 Useful_extra  
-rwxrwxr--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt  
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat  
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py  
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



# Command line

The **mkdir** command creates a directory — examples:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l  
total 0  
drwxrw-rw-  2 milenavalentini  staff   64 Sep 17 15:21 Useful  
-rwxrwxr--  1 milenavalentini  staff    0 Sep 17 12:58 file_1.txt  
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 file_2.dat  
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_1.py  
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_2.bash  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ mkdir Useful_extra  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l  
total 0  
drwxrw-rw-  2 milenavalentini  staff   64 Sep 17 15:21 Useful  
drwxr-xr-x  2 milenavalentini  staff   64 Sep 17 17:17 Useful_extra  
-rwxrwxr--  1 milenavalentini  staff    0 Sep 17 12:58 file_1.txt  
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 file_2.dat  
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_1.py  
-rw-r--r--  1 milenavalentini  staff    0 Sep 17 12:58 script_2.bash  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls Useful  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ mkdir -p ./Useful/OtherResources/Useful_extra  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls Useful  
OtherResources  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls Useful/OtherResources/  
Useful_extra  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



# Command line

---

The **touch** command creates a file

```
milenaValentini$ touch file_1.txt
```

The **cp** command copies one file from a directory to another  
or a given file into another file

```
(base) MacBook-Pro-2:TRM_Dati milenaValentini$  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cp file_1.txt Useful/OtherResources/  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cp file_1.txt file_3.txt  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$
```



# Command line

The **touch** command creates a file

```
milenaValentini$ touch file_1.txt
```

The **cp** command copies one file from a directory to another  
or a given file into another file

```
(base) MacBook-Pro-2:TRM_Dati milenaValentini$  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cp file_1.txt Useful/OtherResources/  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cp file_1.txt file_3.txt  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$
```

cp overwrites (e.g. if file\_3.txt is not empty, it will then be the same as file\_1.txt)

```
(base) MacBook-Pro-2:TRM_Dati milenaValentini$  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cp -i file_1.txt file_3.txt  
overwrite file_3.txt? (y/n [n]) no  
not overwritten  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$
```



# Command line

The **touch** command creates a file

```
milenaValentini$ touch file_1.txt
```

The **cp** command copies one file from a directory to another or a given file into another file

```
(base) MacBook-Pro-2:TRM_Dati milenaValentini$  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cp file_1.txt Useful/OtherResources/  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cp file_1.txt file_3.txt  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$
```

The **rsync** command behaves similarly to the **cp** command: it deals with file transfer and allows you to only transfer the differences between two sets of files using a checksum-search algorithm. It's especially useful when network/ssh connections are involved.

The **cksum** command displays a check value, the total number of octets in the file, and the filename itself.

```
(base) MacBook-Pro-2:TRM_Dati milenaValentini$ cksum file_1.txt  
3000425221 121 file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenaValentini$
```



# How to manipulate files

**rm** removes files and directories

**rmdir** removes an empty directory

**rm -rf** forces to recursively delete a non empty directory

```
RM(1)                                General Commands Manual                                RM(1)

NAME
  rm, unlink - remove directory entries

SYNOPSIS
  rm [-f | -i] [-dIRrvWx] file ...
  unlink [--] file

DESCRIPTION
  The rm utility attempts to remove the non-directory type files specified on the command line.
  If the permissions of the file do not permit writing, and the standard input device is a
  terminal, the user is prompted (on the standard error output) for confirmation.

  The options are as follows:

  -d      Attempt to remove directories as well as other types of files.

  -f      Attempt to remove the files without prompting for confirmation, regardless of the
          file's permissions.  If the file does not exist, do not display a diagnostic message
          or modify the exit status to reflect an error.  The -f option overrides any previous
          -i options.

  -i      Request confirmation before attempting to remove each file, regardless of the file's
          permissions, or whether or not the standard input device is a terminal.  The -i
          option overrides any previous -f options.

  -I      Request confirmation once if more than three files are being removed or if a
          directory is being recursively removed.  This is a far less intrusive option than -i
          yet provides almost the same level of protection against mistakes.

  -R      Attempt to remove the file hierarchy rooted in each file argument.  The -R option
          implies the -d option.  If the -i option is specified, the user is prompted for
          confirmation before each directory's contents are processed (as well as before the
          attempt is made to remove the directory).  If the user does not respond
          affirmatively, the file hierarchy rooted in that directory is skipped.

  -r      Equivalent to -R.
```



# How to manipulate files

## The **mv** command moves files

```
MV(1)                                     General Commands Manual                                     MV(1)

NAME
  mv - move files

SYNOPSIS
  mv [-f | -i | -n] [-hv] source target
  mv [-f | -i | -n] [-v] source ... directory

DESCRIPTION
  In its first form, the mv utility renames the file named by the source operand to the destination path named by the target operand. This form is assumed when the last operand does not name an already existing directory.

  In its second form, mv moves each file named by a source operand to a destination file in the existing directory named by the directory operand. The destination path for each operand is the pathname produced by the concatenation of the last operand, a slash, and the final pathname component of the named file.
```

```
MacBook-Pro-2:TRM_Dati milenavalentini$
MacBook-Pro-2:TRM_Dati milenavalentini$ mv file_1.txt file_4.txt
MacBook-Pro-2:TRM_Dati milenavalentini$
```

equivalent to: `cp file_1.txt file_4.txt; rm file_1.txt`

```
MacBook-Pro-2:TRM_Dati milenavalentini$
MacBook-Pro-2:TRM_Dati milenavalentini$ mv file_3.txt Useful/
MacBook-Pro-2:TRM_Dati milenavalentini$
```

1st form of the manual

```
MacBook-Pro-2:TRM_Dati milenavalentini$
MacBook-Pro-2:TRM_Dati milenavalentini$ mv file_2.dat file_4.txt Useful/
MacBook-Pro-2:TRM_Dati milenavalentini$
```

2nd form of the manual



# How to manipulate files

The **cat** command is used to display a text file or to concatenate multiple files into a single file.

By default, the **cat** command prints outputs to the standard output.

To simply inspect the file content, you can also use to command **more**

If you want to know how many lines a file is made of, you can use to command **wc -l** (it counts lines or words)

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_3.txt
# letters
a
b
c
d
e
f
g
h
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ more file_3.txt
# letters
a
b
c
d
e
f
g
h
file_3.txt (END)
```

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ wc -l file_1.txt
41 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ wc -l file_3.txt
9 file_3.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

# How to manipulate files

The **cat** command is used to display a text file or to concatenate multiple files into a single file.

By default, the **cat** command prints outputs to the standard output.

To see the first part of a file, or its first n lines, use the **head** command:

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_3.txt
# letters
a
b
c
d
e
f
g
h
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ head file_1.txt
# numbers
1
2
3
4
5
6
7
8
9
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ head -5 file_1.txt
# numbers
1
2
3
4
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



# How to manipulate files

The **cat** command is used to display a text file or to concatenate multiple files into a single file.

By default, the **cat** command prints outputs to the standard output.

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_3.txt
# letters
a
b
c
d
e
f
g
h
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

To see the first part of a file, or its first n lines, use the **head** command:

To access the bottom part of a file, or its last n lines, use the **tail** command:

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ tail file_1.txt
31
32
33
34
35
36
37
38
39
40
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ tail -3 file_1.txt
38
39
40
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

# How to manipulate files

The **cat** command is used to display a text file or to concatenate multiple files into a single file.

By default, the **cat** command prints outputs to the standard output.

The **cat** command takes in one or more filenames as its arguments.

The files are concatenated in the order they appear in the argument list.

As for almost every command, the **cat** command generates the output to standard output, which can be redirected to a file (using the UNIX direction operator `>`; use `>>` to append)

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_3.txt
# letters
a
b
c
d
e
f
g
h
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_1.txt file_3.txt > file_4.txt
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ wc -l file_4.txt
  50 file_4.txt
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ tail -15 file_4.txt
35
36
37
38
39
40
# letters
a
b
c
d
e
f
g
h
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



# How to manipulate files

The **cat** command is used to display a text file or to concatenate multiple files into a single file.

By default, the **cat** command prints outputs to the standard output.

The **cat** command takes in one or more filenames as its arguments.

The files are concatenated in the order they appear in the argument list.

As for almost every command, the **cat** command generates the output to standard output, which can be redirected to a file (using the UNIX direction **operator >**; use **>>** to append)

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_3.txt
# letters
a
b
c
d
e
f
g
h
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_1.txt file_3.txt > file_4.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ wc -l file_4.txt
  50 file_4.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ tail -15 file_4.txt
35
36
37
38
39
40
# letters
a
b
c
d
e
f
g
h
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

# How to manipulate files

The **cat** command is used to display a text file or to concatenate multiple files into a single file.

By default, the **cat** command prints outputs to the standard output.

The **cat** command takes in one or more filenames as its arguments.

The files are concatenated in the order they appear in the argument list.

As for almost every command, the **cat** command generates the output to standard output, which can be redirected to a file (using the UNIX direction operator **>** ; use **>>** to append)

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_3.txt
# letters
a
b
c
d
e
f
g
h
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cat file_3.txt >> file_4.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ wc -l file_4.txt
  59 file_4.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ tail -25 file_4.txt
34
35
36
37
38
39
40
# letters
a
b
c
d
e
f
g
h
# letters
a
b
c
d
e
f
g
h
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```



# How to manipulate files

The **ln** command provides a given file with an alternative name.

It links a file name to another one.

It is possible to link a file to another in the same directory or even to the same name in another directory.

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

When linking a filename to another filename, only two arguments can be specified: the source filename and the target filename.

When linking a filename to a directory, you can specify multiple filenames to be linked to the same directory.

```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls  
Best_file.txt Useful_extra file_1.txt file_2.dat file_4.txt script_2.bash  
Useful Worst_file.txt file_1_save.txt file_3.txt script_1.py  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls Useful_extra/  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ln file_1.txt Useful_extra/  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls Useful_extra/  
file_1.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ln file_4.txt Useful_extra/  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls Useful_extra/  
file_1.txt file_4.txt  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

# How to manipulate files

---

The **ln** command provides a given file with an alternative name.

Link two files in the current directory:

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

The flags that can be used with the **ln** command are as follows:

- s to create a soft link to another file or directory.

In a soft link, the linked file contains the name of the original file.

When an operation on the linked filename is done, the name of the original file in the link is used to reference the original file.

- f to ensure that the destination filename is replaced by the linked filename if the file already exists.




# How to manipulate files

---

The **ln** command provides a given file with an alternative name.


Link two files in the current directory:

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

 It is linked to file\_1.txt (hard link)  
If one of the files is removed, the other  
will not undergo changes.

To create a symbolic link of the first argument in the current directory:

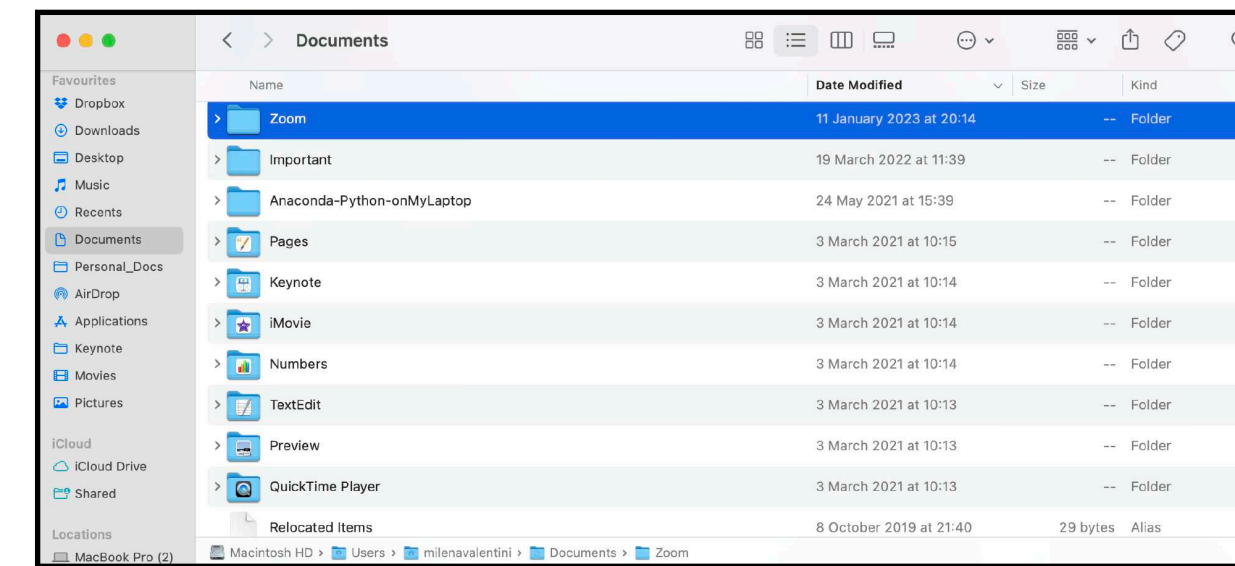
```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$  
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ln -s file_3.txt Worst_file.txt
```

 This linked file  
only contains the name of file\_3.txt  
If you remove file\_3.txt, you will be left with an orphan Worst\_file.txt,  
which points to nowhere.

# 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

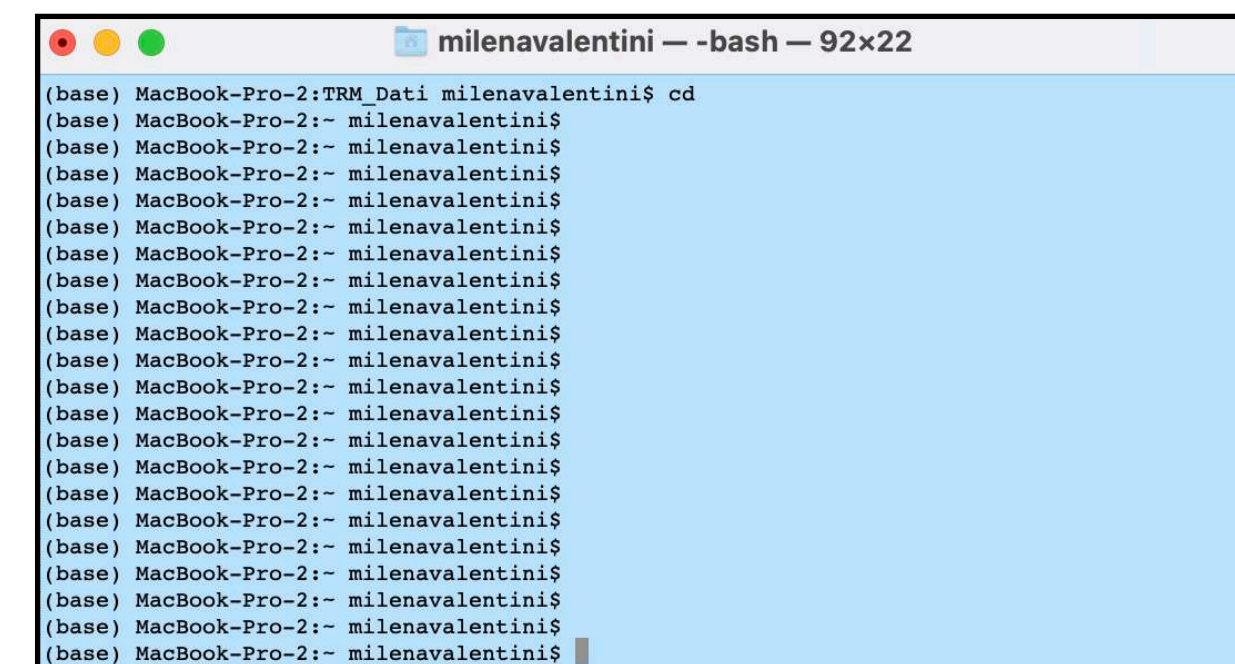


## Shell/terminal/console/command line prompt:

interface to interact with the computer

via command line

without relying on graphical unit 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:

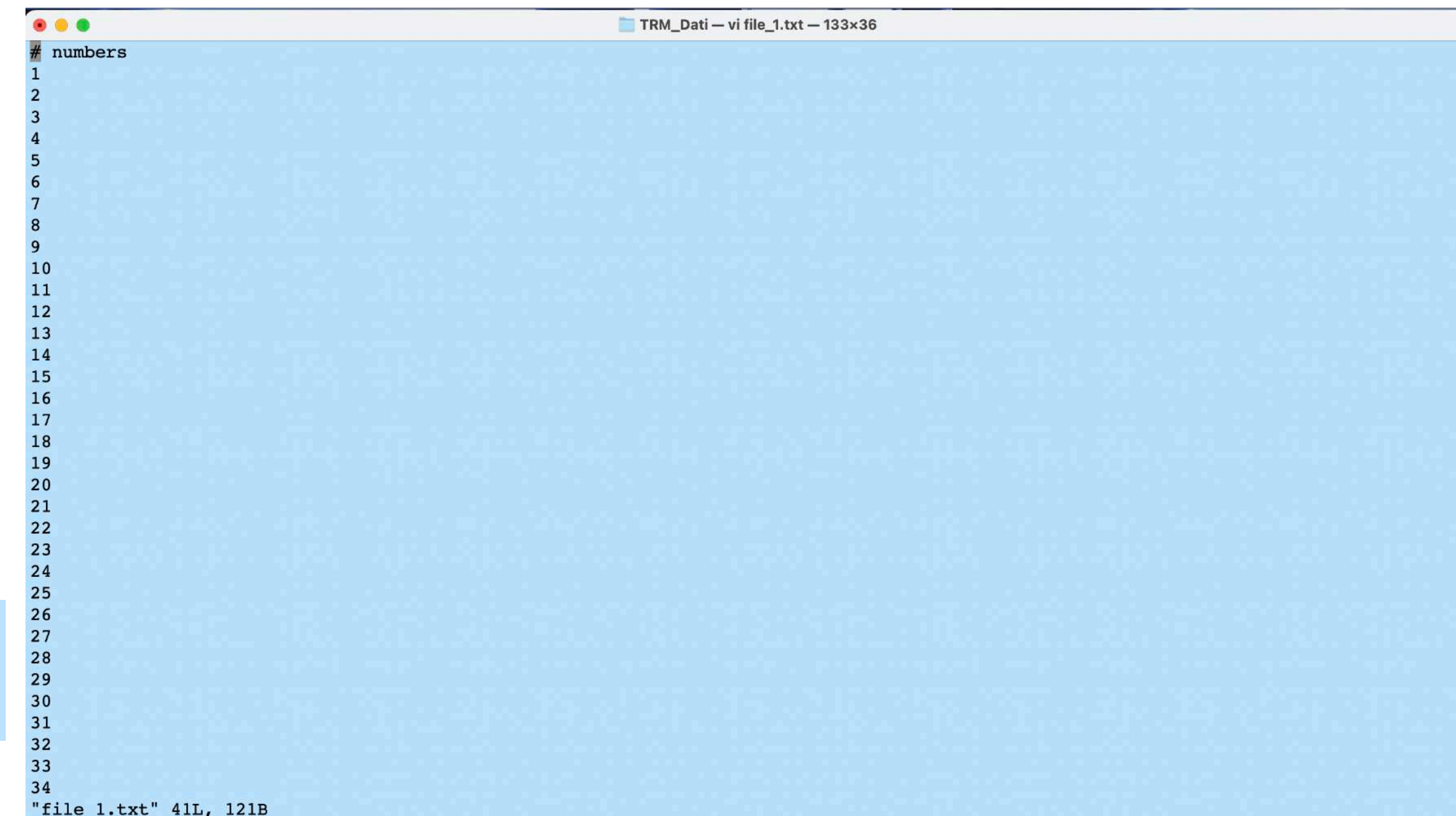
1. — *Command mode*: commands produce actions to be taken on the file, and
2. — *Insert mode*: entered text is written into the file.

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](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$
```





# The text editor

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

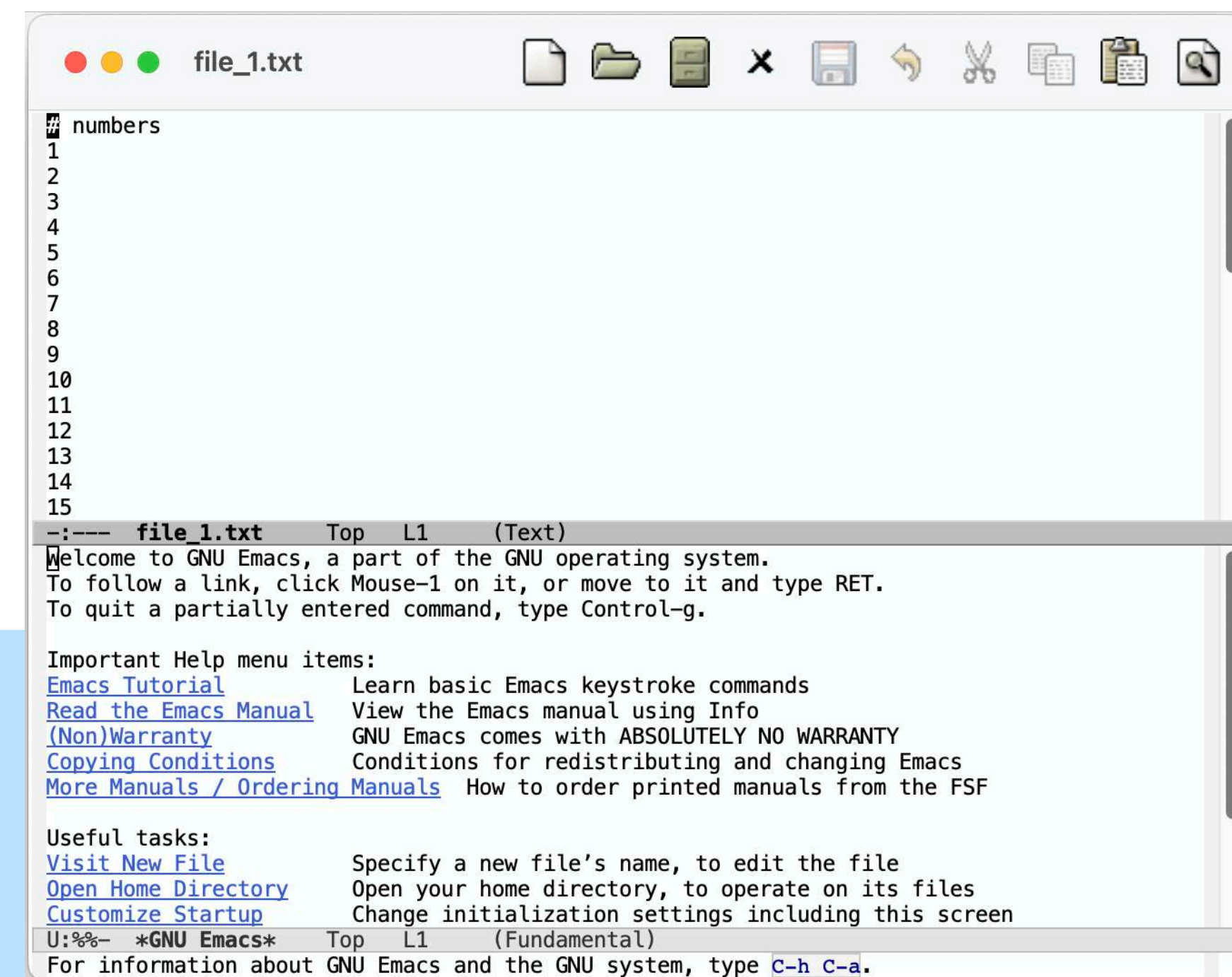
You can follow the instructions to download and install it here:

<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](https://www.gnu.org/software/emacs/manual/html_node/emacs/index.html)

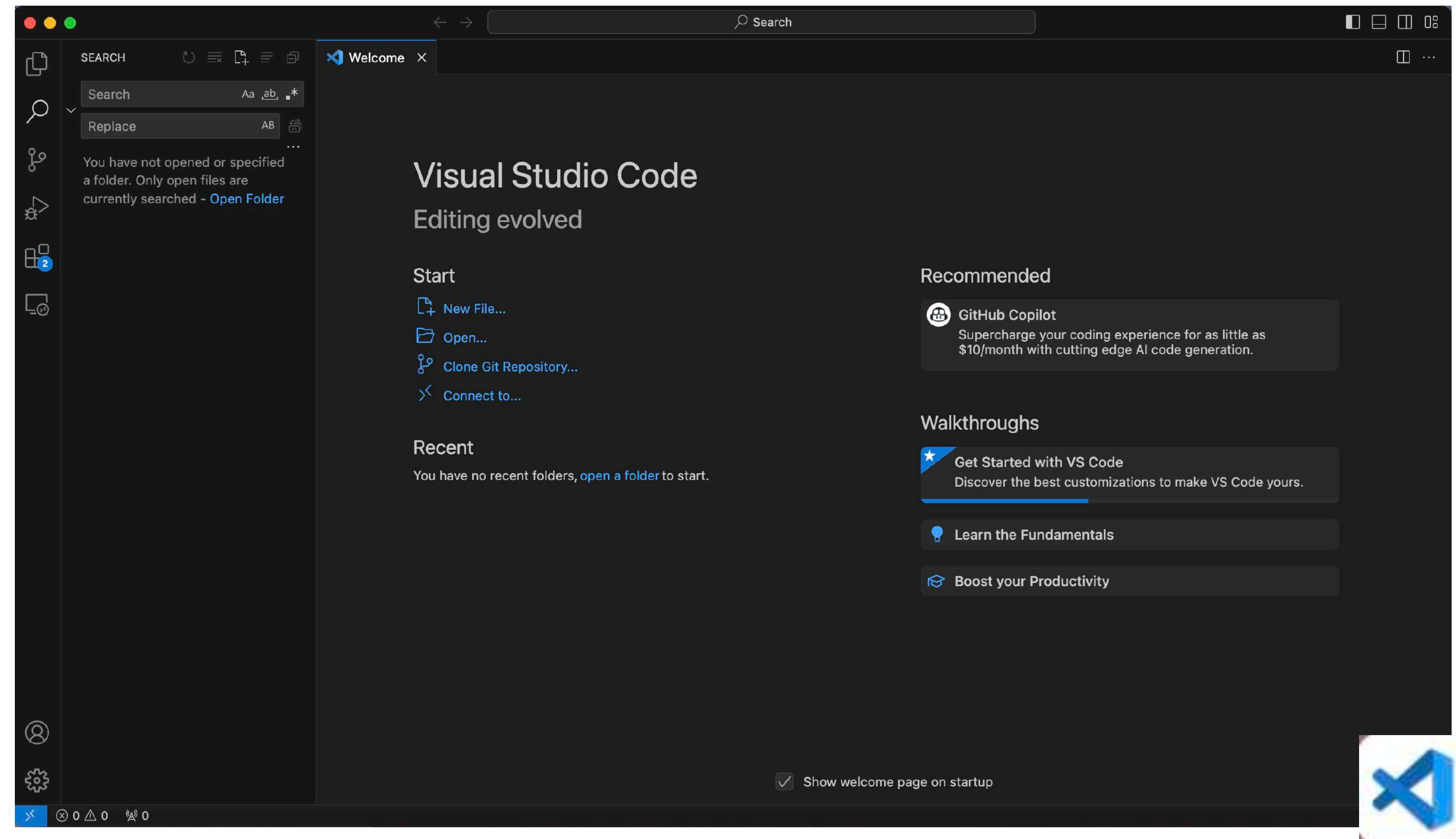


# The text editor

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

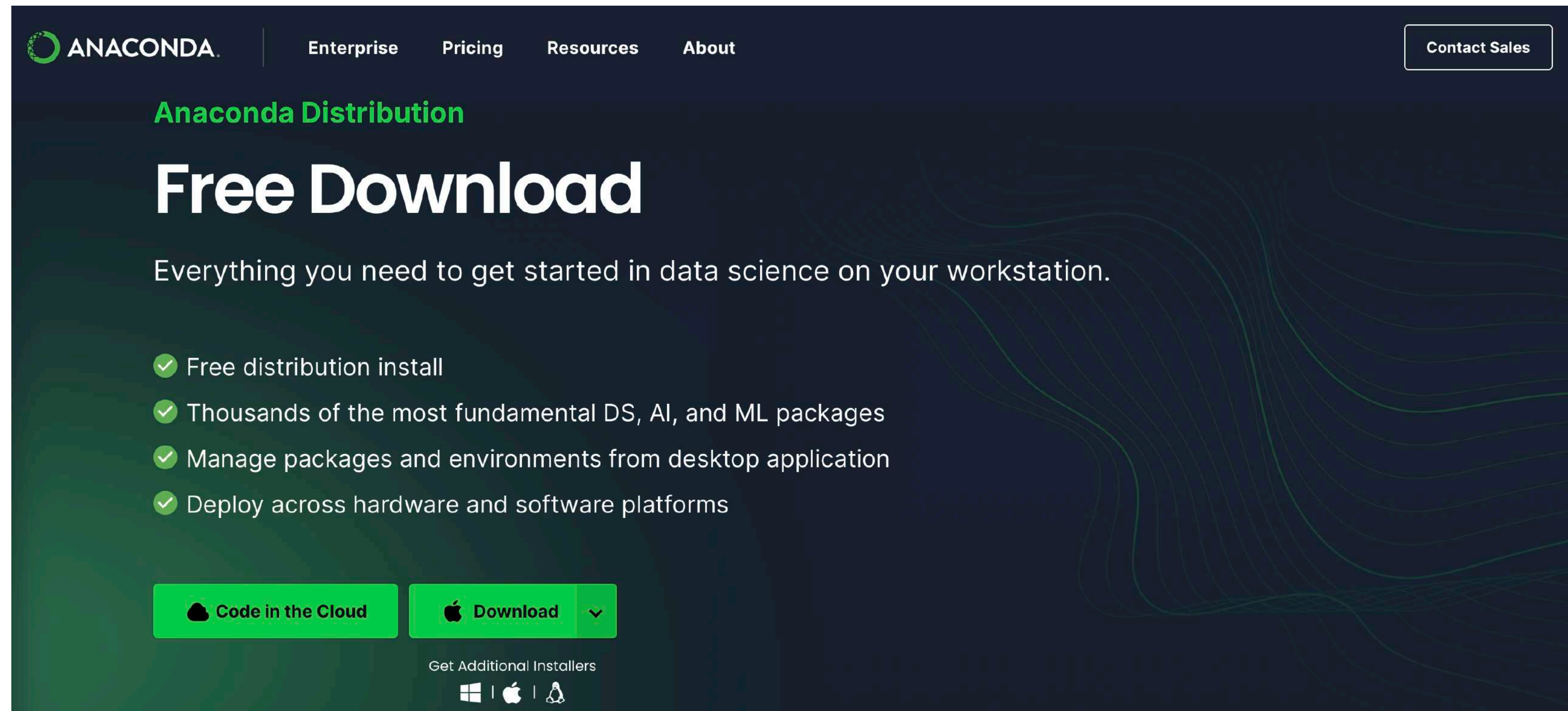
Anaconda is an open-source package and environment management system that runs on Windows, macOS, and Linux.

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



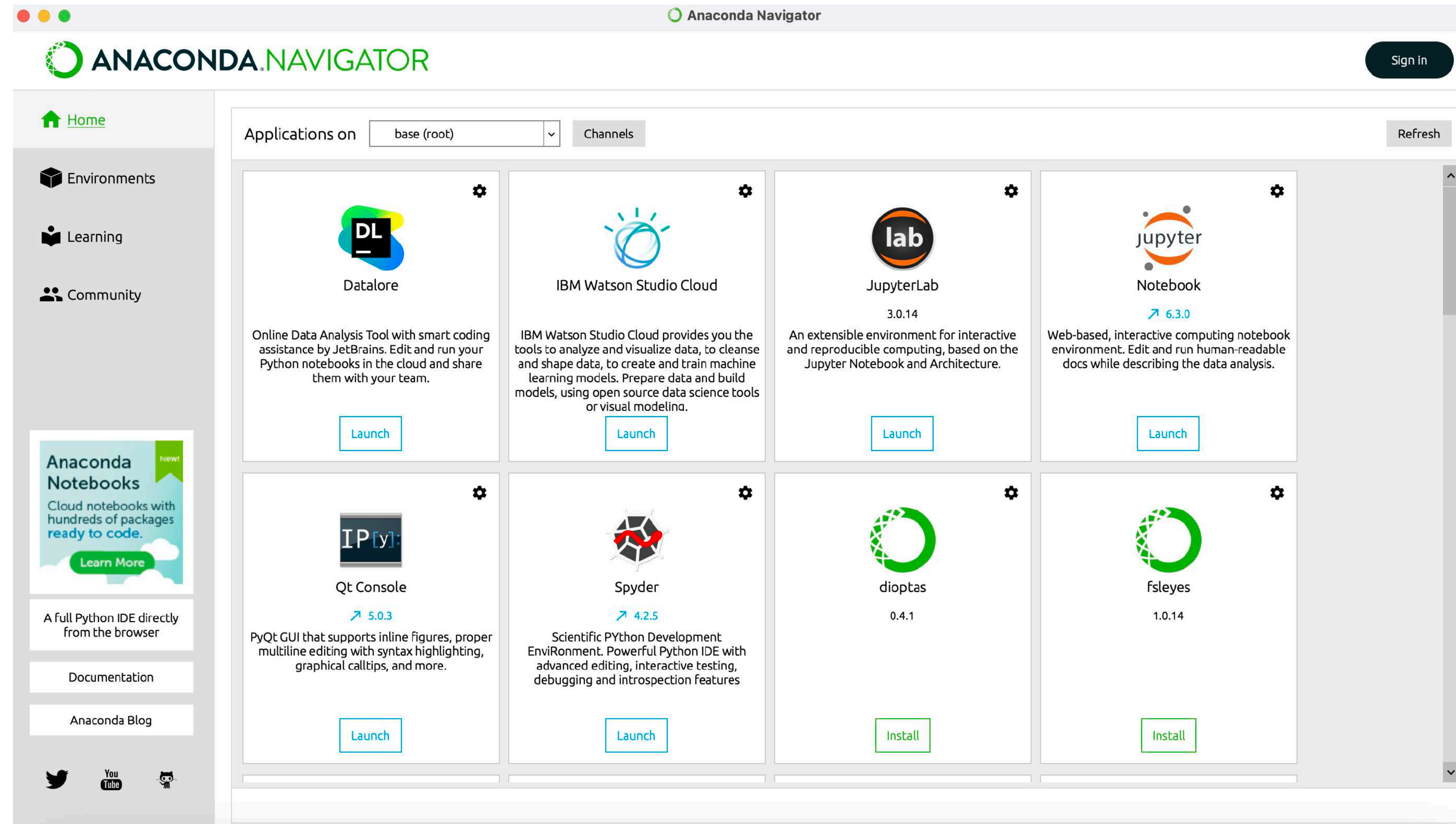
The screenshot shows the Anaconda website's download page. At the top, there is a navigation bar with the Anaconda logo, links for Enterprise, Pricing, Resources, and About, and a Contact Sales button. The main heading is "Anaconda Distribution" in green, followed by "Free Download" in large white text. 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:

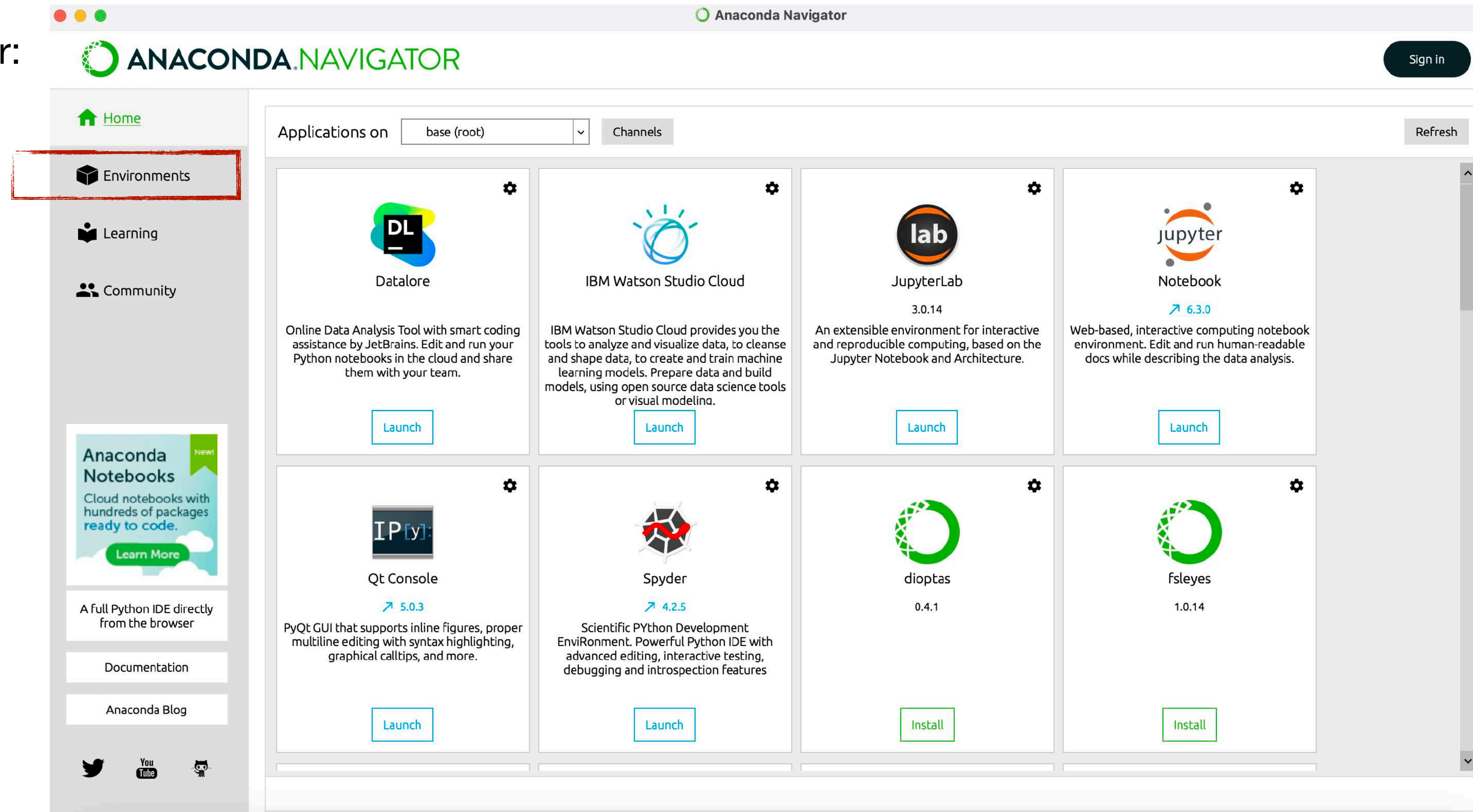


# Setting up the working environment with Anaconda

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 interface is divided into several sections:

- Left Sidebar:** Contains navigation options: Home, Environments (highlighted), Learning, and Community. Below these are promotional cards for Anaconda Notebooks and a Python IDE, along with links to Documentation and the Anaconda Blog.
- Environments Panel:** A search bar labeled "Search Environments" is at the top. Below it, a list of environments is shown: "base (root)", "Jupyter\_...", "spyder\_...", and "spyder\_...". The "base (root)" environment is highlighted with a red box. Below the list, the text "Already existing environments" is displayed.
- Bottom Action Bar:** Features four buttons: "Create" (highlighted with a red box), "Clone", "Import", and "Remove".
- Right Panel:** Shows a list of installed packages. At the top, there are filters for "Installed" and "Channels", and buttons for "Update index..." and "Search Packages". The package list has columns for Name, Description, and Version. The "Create" button is also highlighted with a red box.

Name	Description	Version
✓ _ipyw_lab_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

356 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:

The screenshot displays the Anaconda Navigator application window. The main interface is titled 'ANACONDA NAVIGATOR' and features a sidebar with navigation options: Home, Environments, Learning, and Community. The 'Environments' tab is active, showing a list of environments: 'base (root)', 'Jupyter\_', 'spyder\_', and 'spyder\_'. A 'Create new environment' dialog box is open in the foreground, allowing the user to configure a new environment. The dialog includes fields for 'Name' (set to 'JupyterNotebook\_test'), 'Location' (set to '/Users/milenaValentini/opt/anaconda3/envs/JupyterNotebook\_test'), and 'Packages' (with 'Python 3.8' selected and 'R' unselected). A 'Create' button is visible in the dialog. In the background, a table of installed packages is visible, including \_ipyw\_jlab\_nb\_ex..., alabaster, appnope, appscript, argh, argon2-cffi, asn1crypto, astroid, and astropy. The 'Create' button in the bottom-left corner of the main interface is highlighted with a red box.



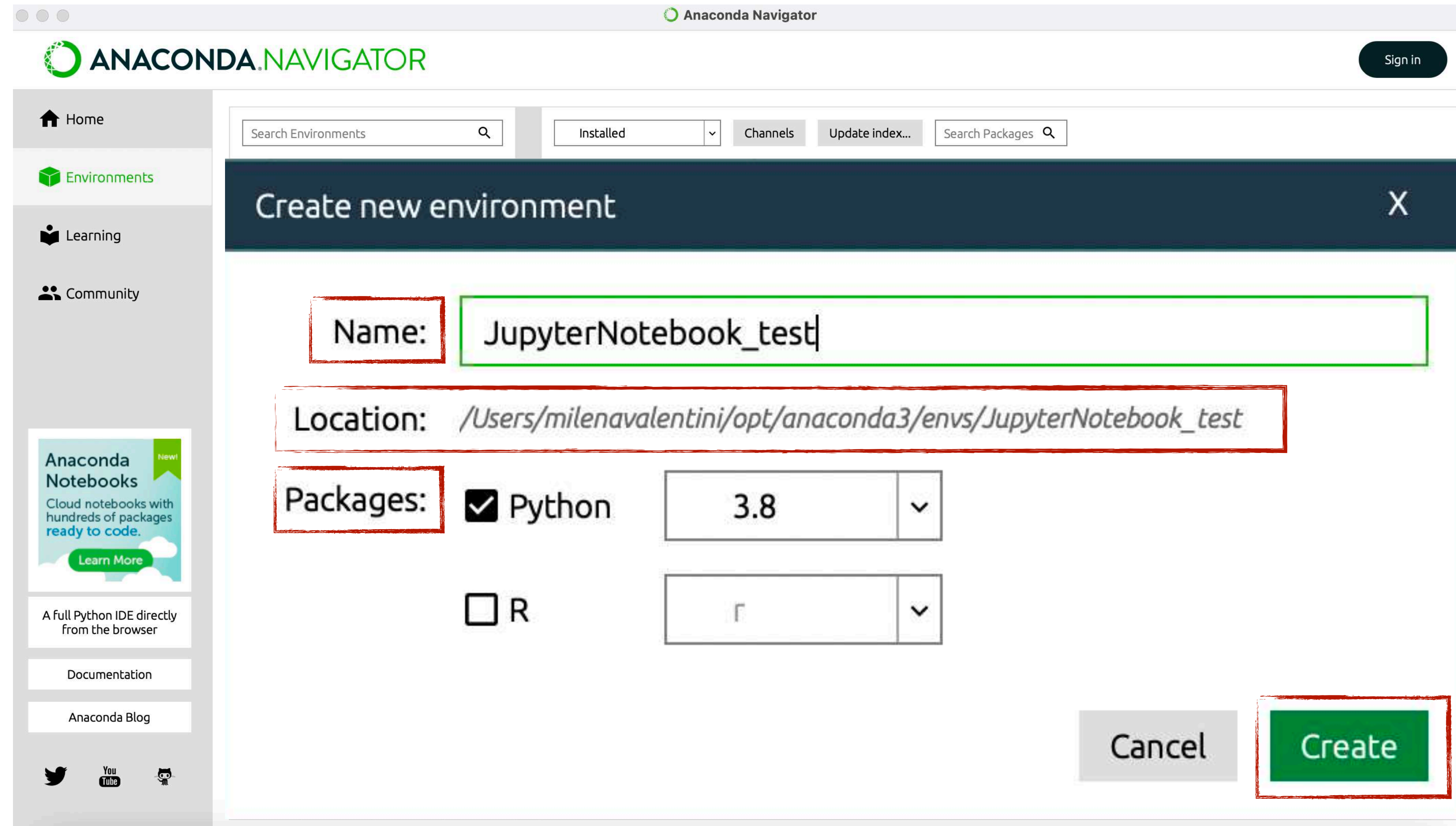
# Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:

Create a new environment:



# Setting up the working environment with Anaconda

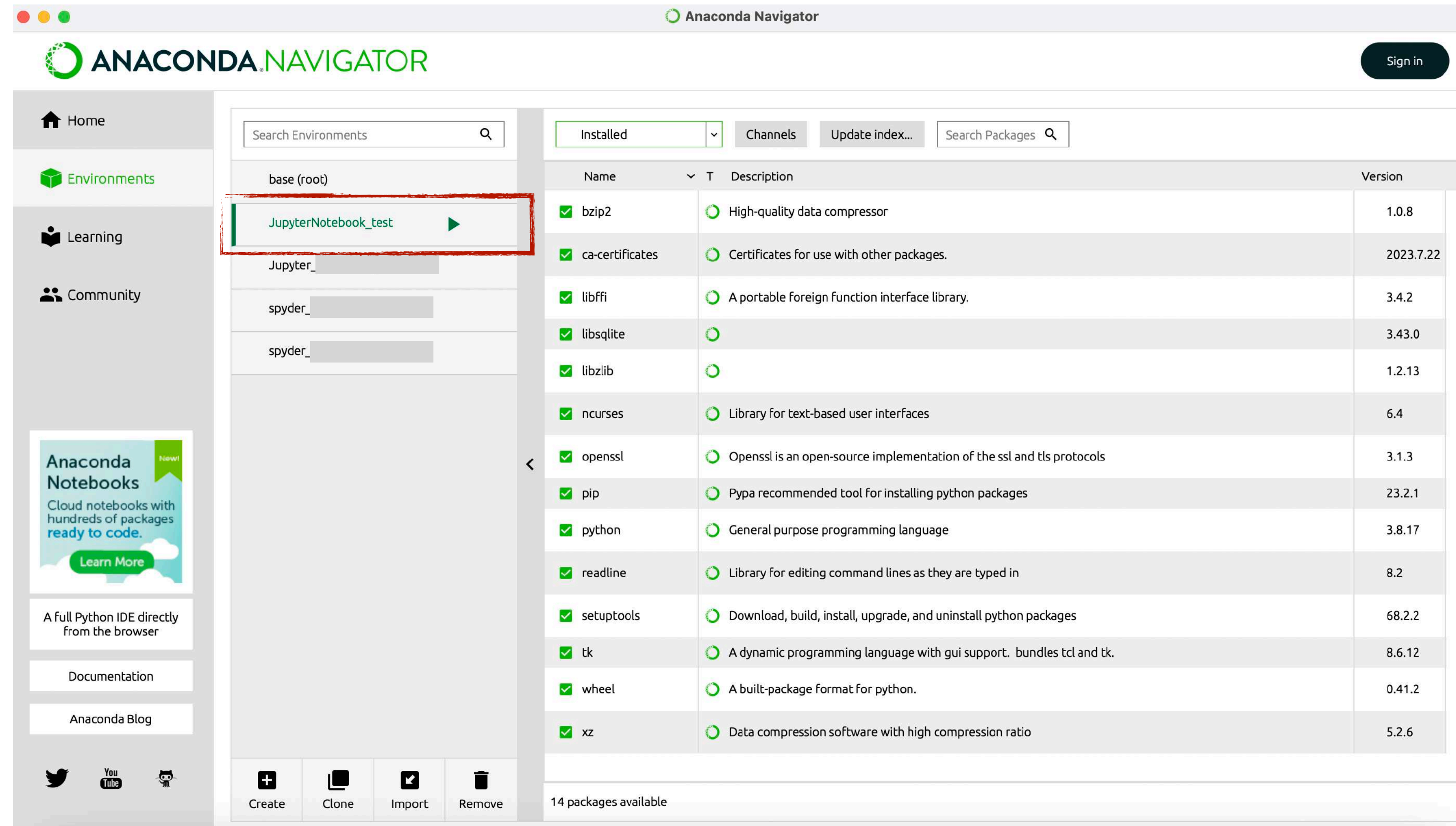
To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

Select Environments:

Create a new environment:

Here is the new environment:



The screenshot displays the Anaconda Navigator application window. The left sidebar contains navigation options: Home, Environments (selected), Learning, and Community. The main area shows a list of environments: 'base (root)', 'JupyterNotebook\_test' (highlighted with a red box), 'Jupyter\_', 'spyder\_', and another 'spyder\_'. Below the environment list are buttons for 'Create', 'Clone', 'Import', and 'Remove'. The right panel shows a table of installed packages with columns for Name, Description, and Version.

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:

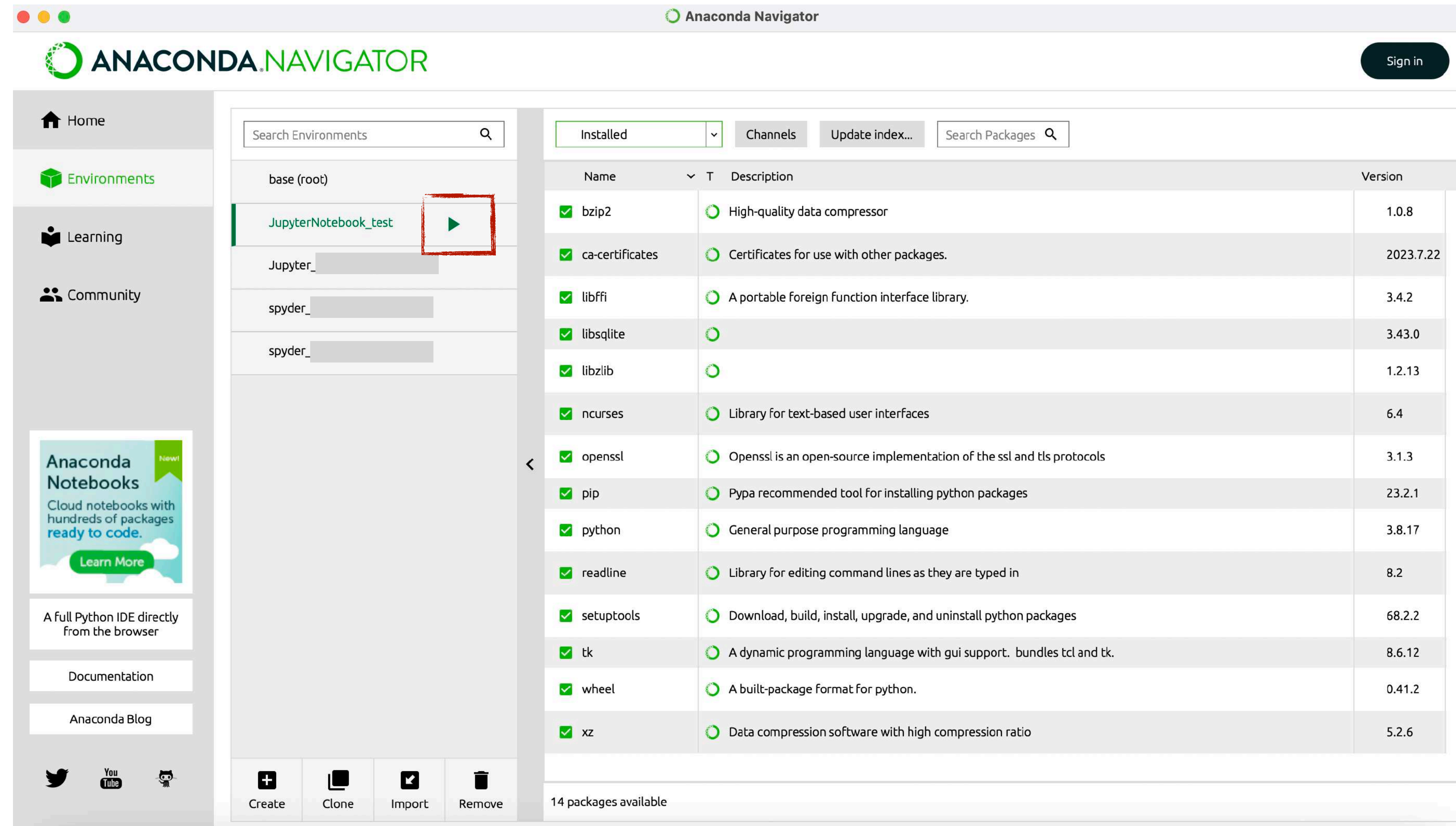
Launch Anaconda-Navigator:

Select Environments:

Create a new environment:

Here is the new environment:

The green arrow tells you that the new environment is active



The screenshot displays the Anaconda Navigator application window. The interface is divided into several sections:

- Left Sidebar:** Contains navigation options: Home, Environments (highlighted in green), Learning, and Community. At the bottom, there are social media icons for Twitter, YouTube, and GitHub.
- Environments Panel:** A list of environments is shown. The 'JupyterNotebook\_test' environment is highlighted with a green bar and a green play button icon (highlighted with a red box), indicating it is the active environment. Other environments listed include 'base (root)', 'Jupyter\_', 'spyder\_', and another 'spyder\_'.
- Right Panel:** Shows a list of installed packages for the active environment. The packages are listed in a table with columns for Name, Description, and Version.

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

At the bottom of the right panel, it indicates '14 packages available'.

# Setting up the working environment with Anaconda

To exploit it via its graphical user interface:

Launch Anaconda-Navigator:

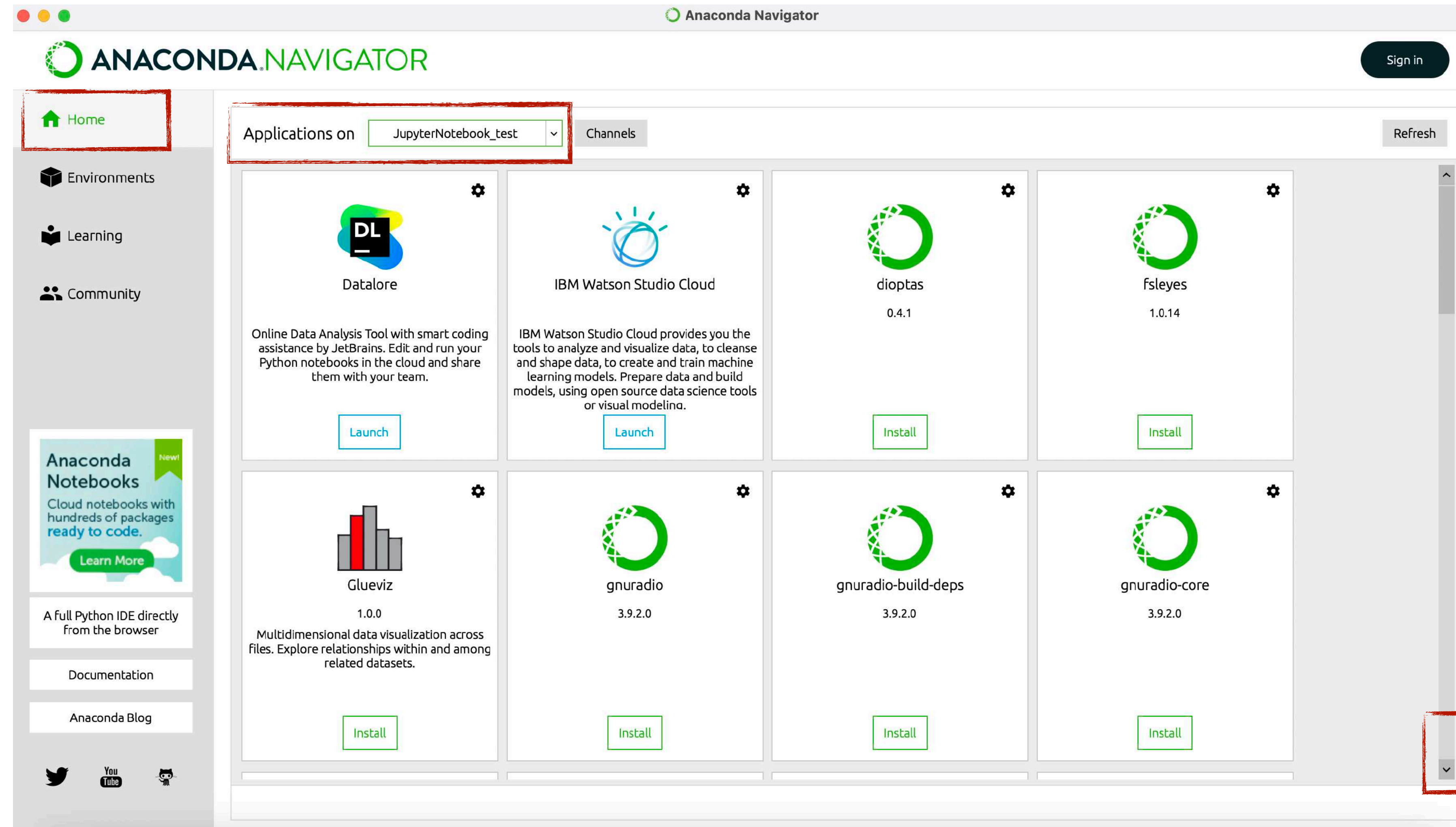
Select Environments:

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:

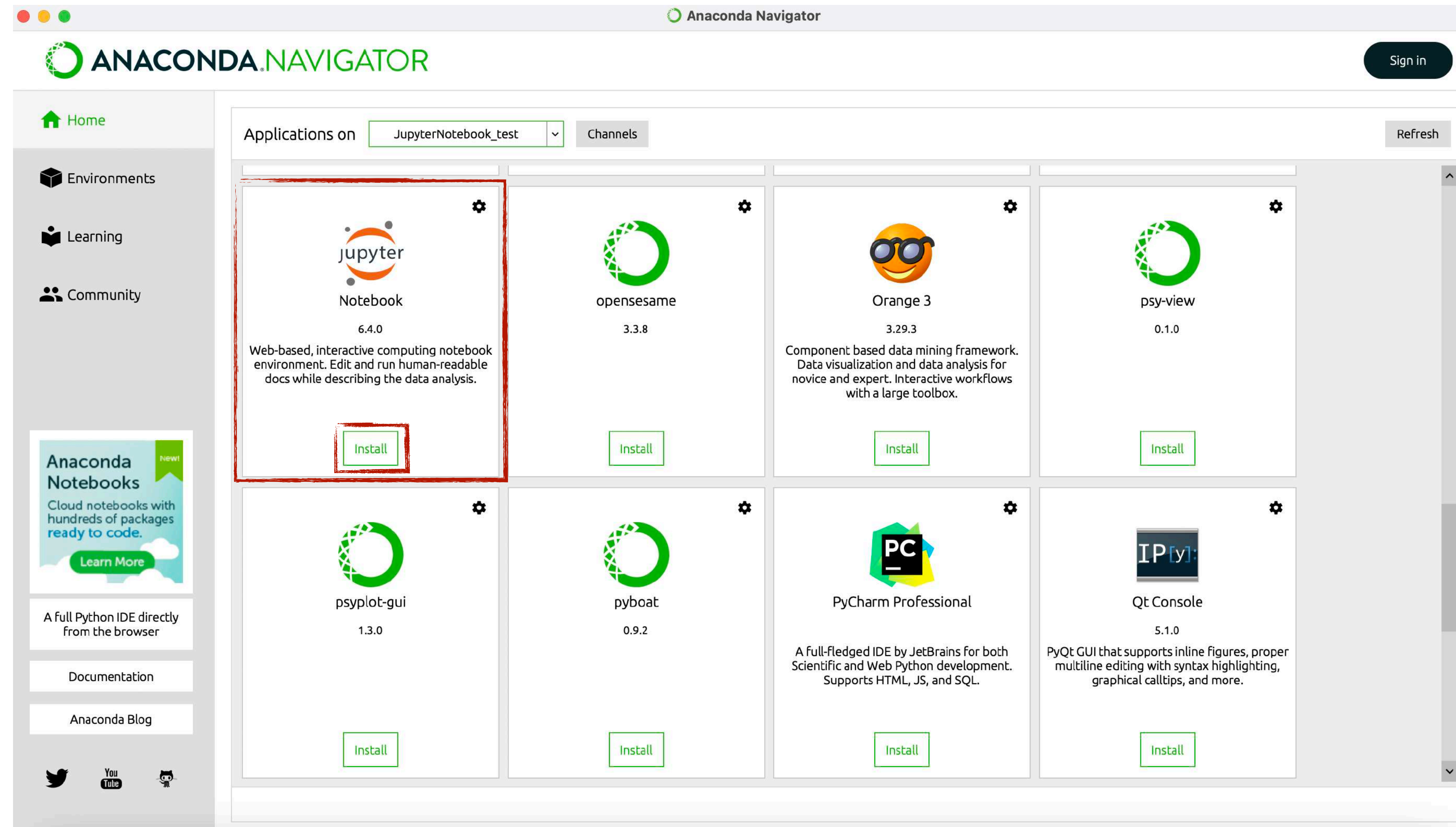
Select Environments:

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:

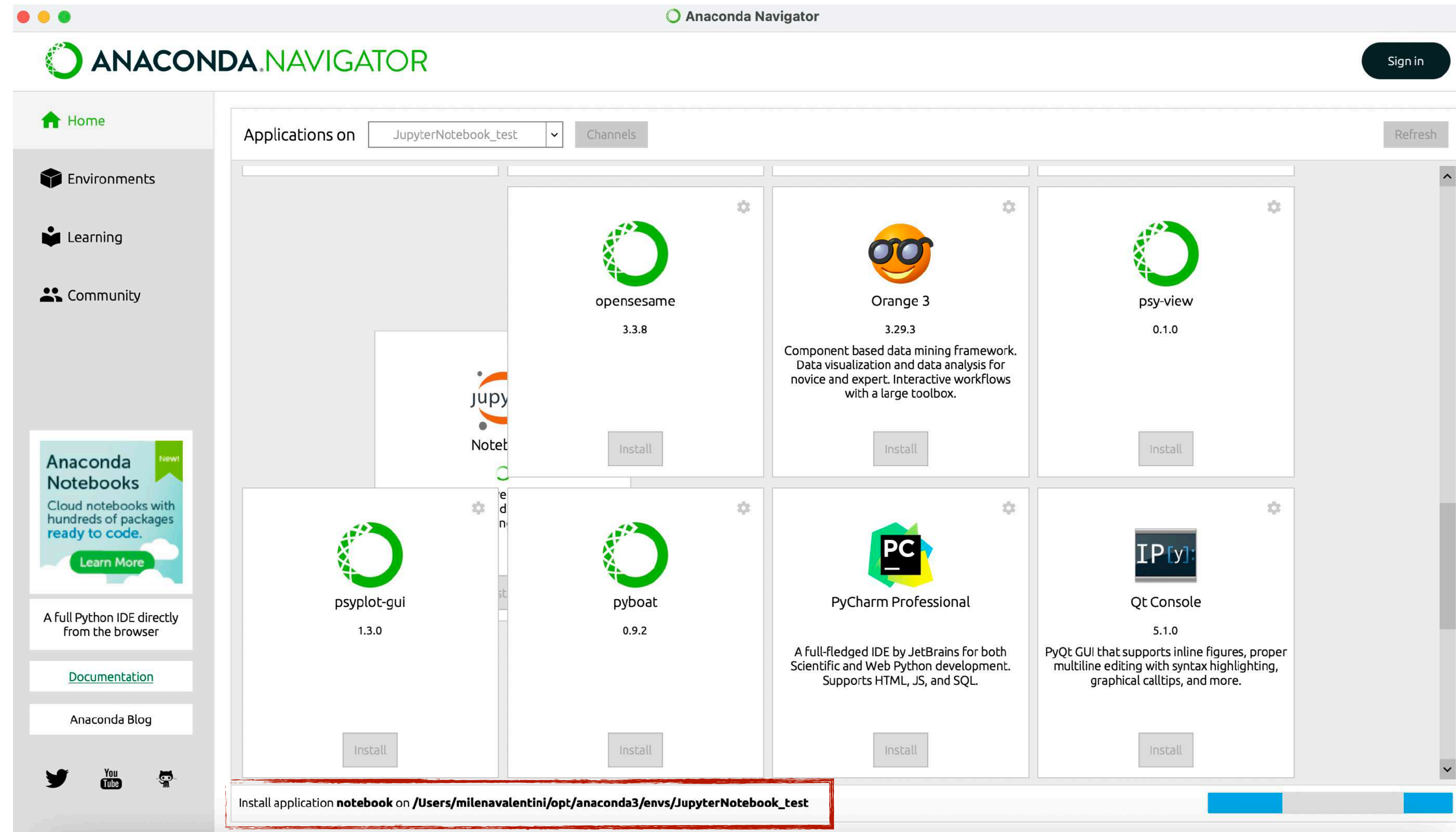
Select Environments:

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:

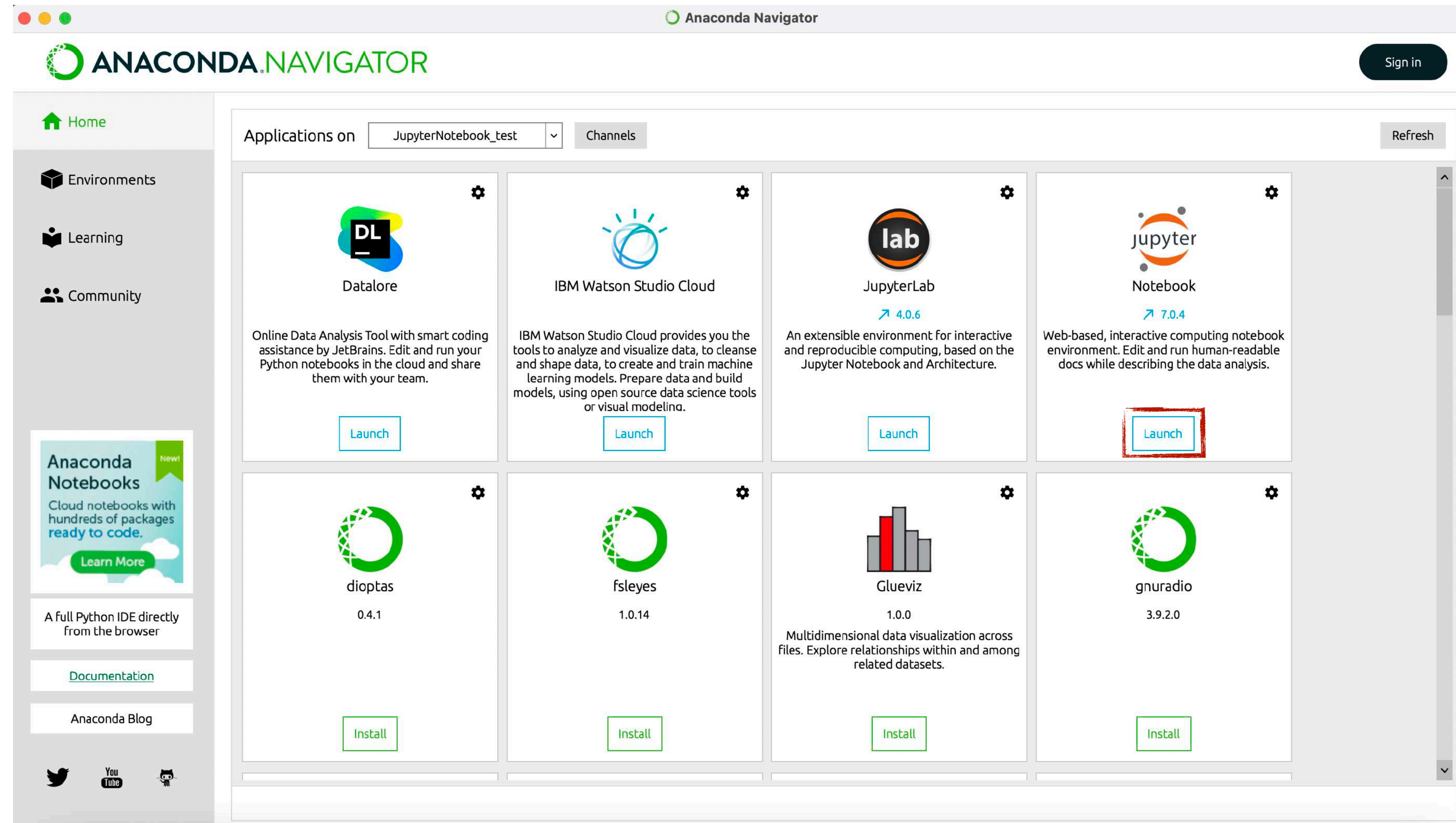
Create a new environment:

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

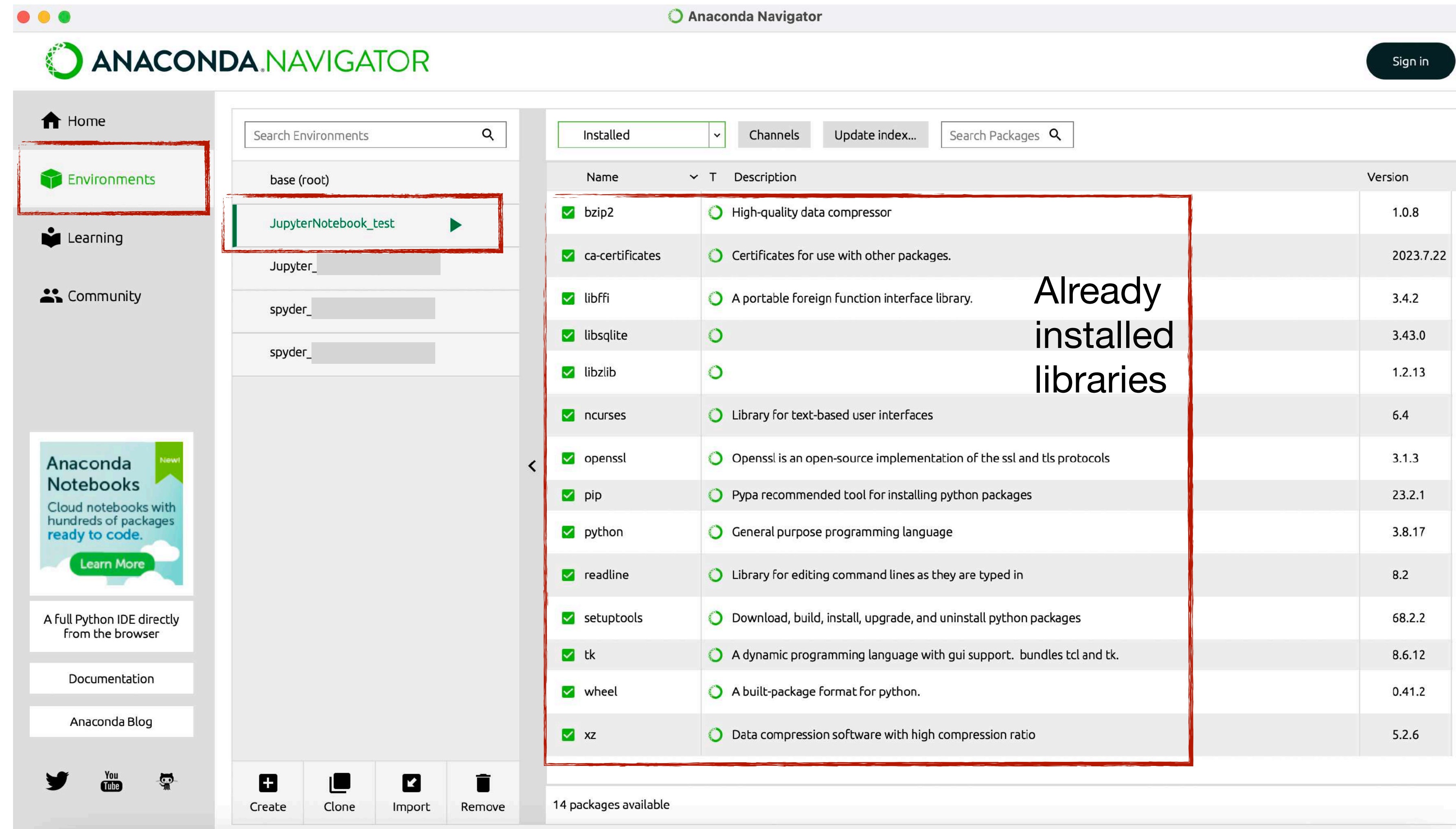
The application has just been installed and can be launched



# 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.

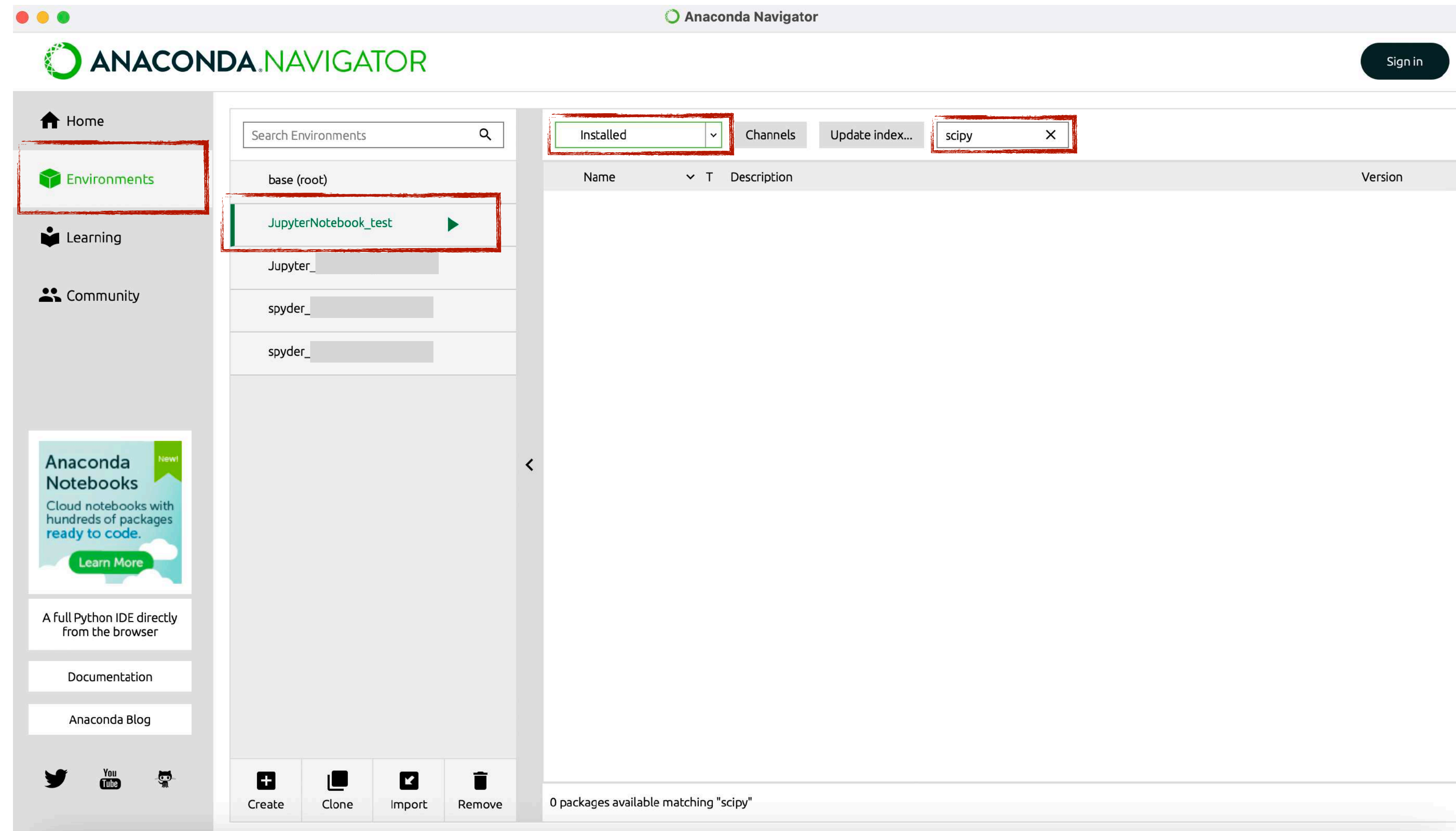
Name	Description	Version
✓ bzip2	High-quality data compressor	1.0.8
✓ ca-certificates	Certificates for use with other packages.	2023.7.22
✓ libffi	A portable foreign function interface library.	3.4.2
✓ libsqlite		3.43.0
✓ libzlib		1.2.13
✓ ncurses	Library for text-based user interfaces	6.4
✓ openssl	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:

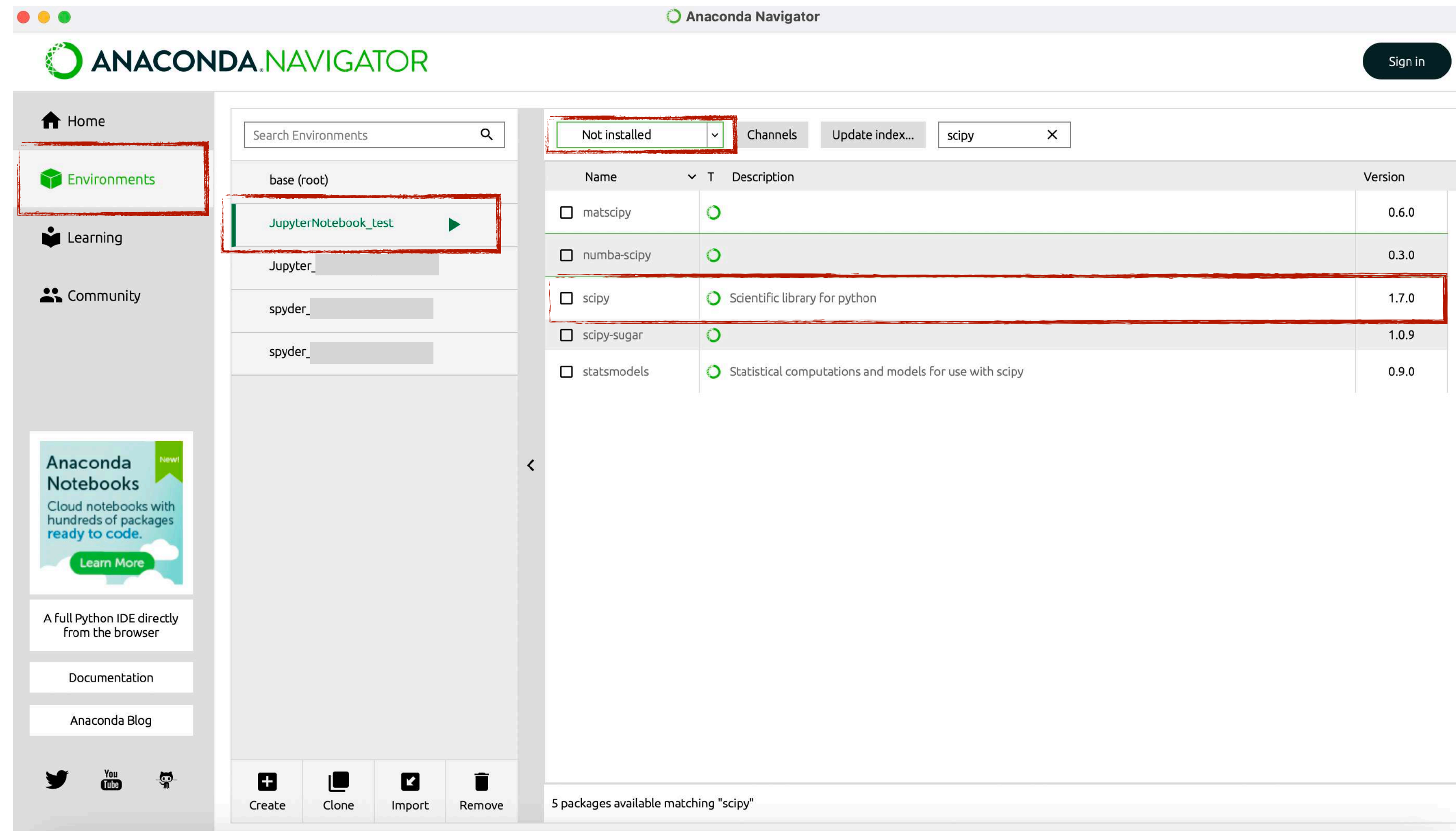
To install libraries  
(instead of applications)  
within a given environment:



# 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:





# 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. The interface is divided into several sections:

- Left Sidebar:** Contains navigation options: Home, Environments (highlighted with a red box), Learning, and Community.
- Environment List:** A list of environments including 'base (root)', 'JupyterNotebook\_test' (highlighted with a red box), and several 'Jupyter\_' and 'spyder\_' environments.
- Environment Details:** A search bar and a dropdown menu set to 'Not installed'. Below this is a table of available packages for the selected environment.

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

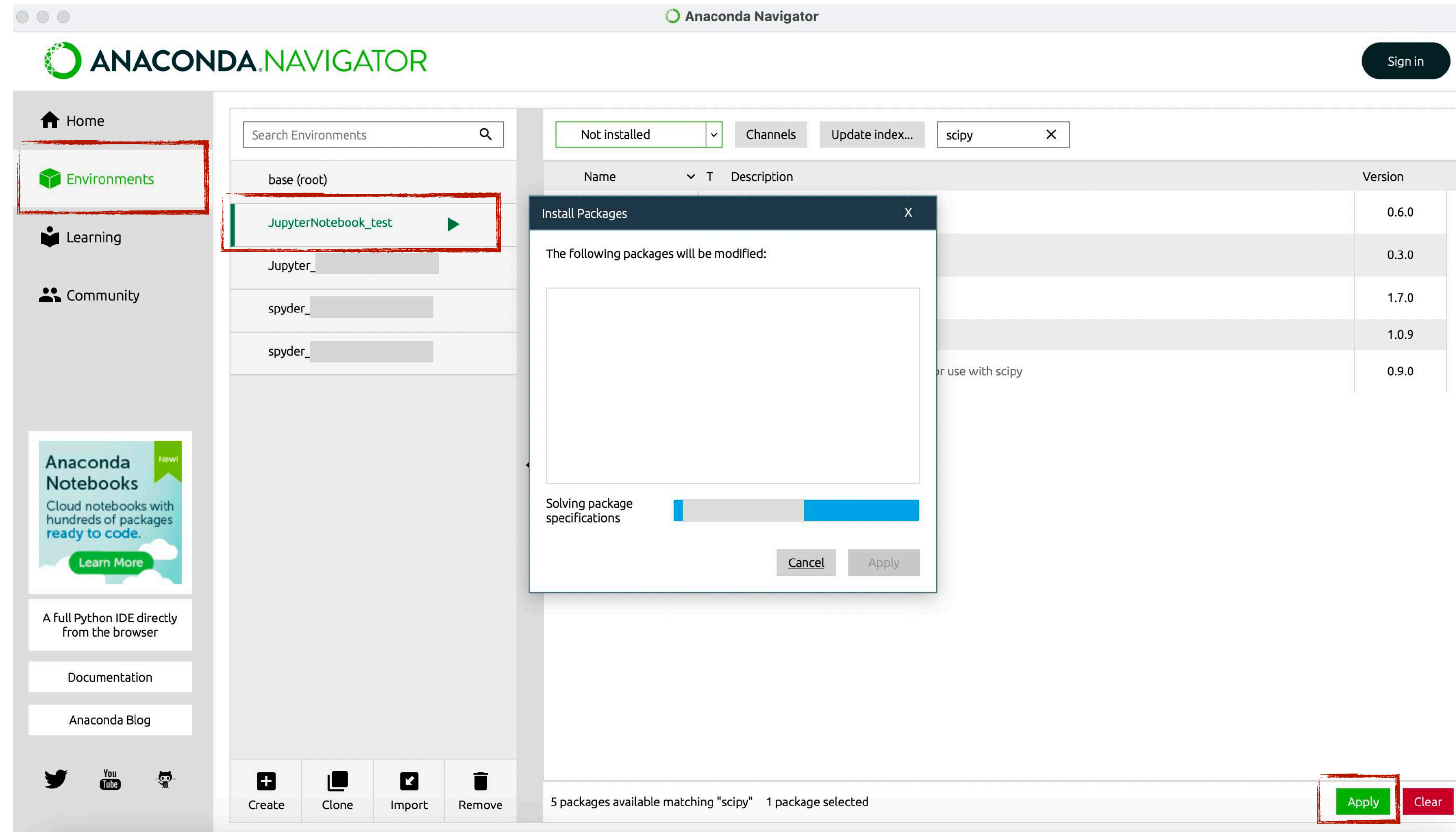
At the bottom of the interface, there are buttons for 'Create', 'Clone', 'Import', and 'Remove'. A status bar at the bottom right indicates '5 packages available matching "scipy" 1 package selected' and features 'Apply' and 'Clear' buttons (both highlighted with red boxes).

# 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





# 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 interface. The 'Environments' tab is selected in the left sidebar. The main panel shows a search for 'scipy' in the 'base (root)' environment. An 'Install Packages' dialog box is open, displaying a list of 10 packages to be installed. The 'Apply' button is highlighted with a red box.

Name	Unlink	Link	Channel	Action
1 *pooch	-	1.7.0	conda-forge	Installed
2 *numpy	-	1.24.4	conda-forge	Installed
3 *llvm-openmp	-	16.0.6	conda-forge	Installed
4 *libopenblas	-	0.3.24	conda-forge	Installed
5 *liblapack	-	3.9.0	conda-forge	Installed
6 *libgfortran5	-	13.2.0	conda-forge	Installed

\* indicates the package is a dependency of a selected package

5 packages available matching "scipy" 1 package selected

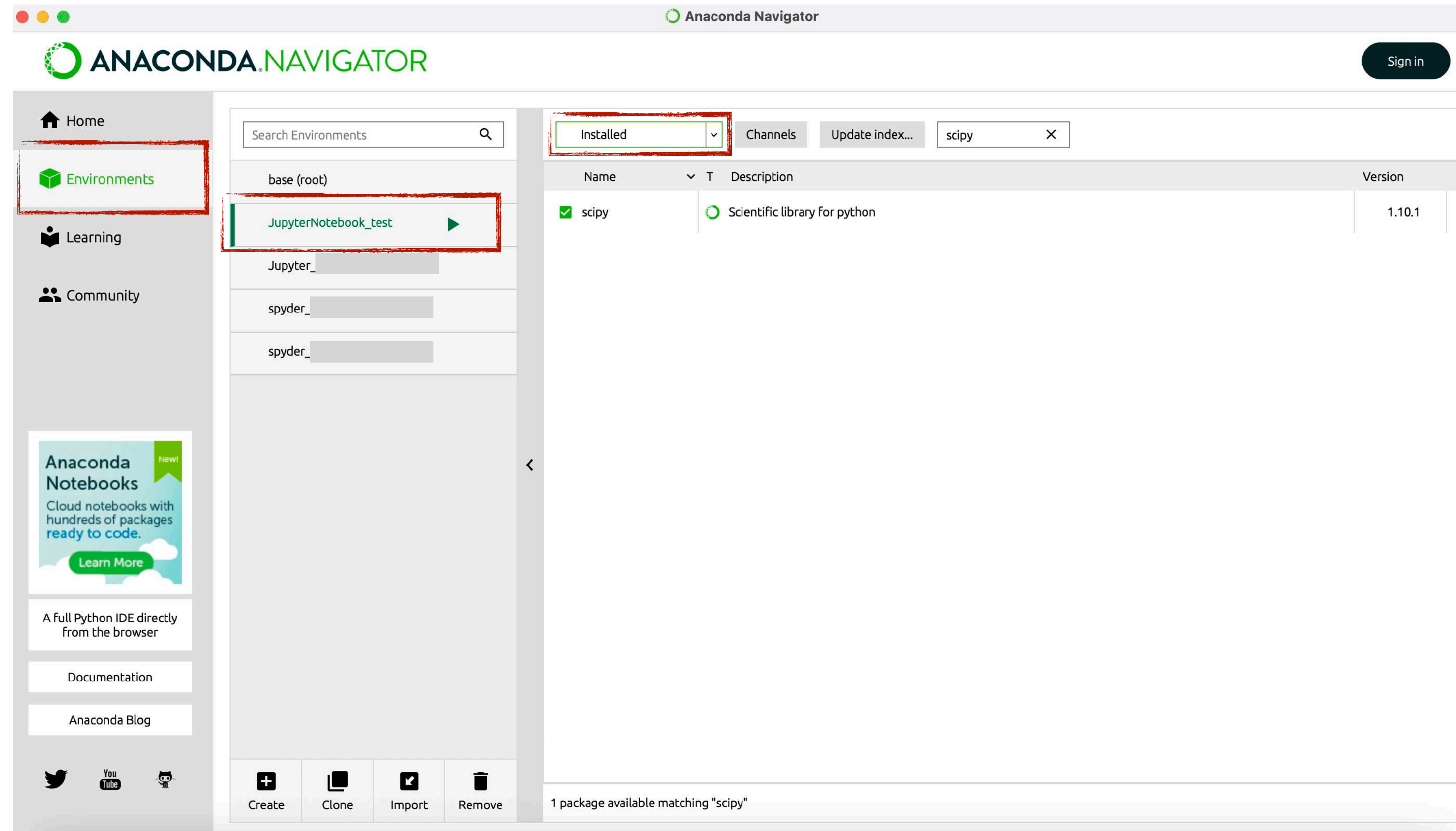
# Setting up the working environment with Anaconda

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 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_

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

Activate it:



# 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`

