

**PACKAGE  
MANAGER /  
DEPENDENCY  
INSTALLER**

# WHY DO WE NEED IT?

- Python is a popular language among data scientists due to its simplicity and the vast array of libraries available.
- However, managing these libraries can be a challenge!
- Two of the most popular tools for managing Python packages are “pip” and “conda”.

# WHY ANACONDA AND NOT PIP?

- Use pip if you are working with pure Python projects and need access to the vast array of libraries.
- Use Anaconda if you are working with projects that use multiple languages, need different versions of Python, or require complex binary dependencies.
- If you're a beginner in data science, use Anaconda;
- If you're more experienced with the command line and cannot find packages for your project (that can be outside the data science domain), then go for Python's pip
- So, for **Ease Of Use**, we use "Anaconda". Also pip comes already installed with anaconda ^

# INSTALLING ANACONDA

# WEBSITE:

<https://www.anaconda.com/products/individual>

## Anaconda Installers



Windows



Mac



Linux

### Python 3.11

[↓ 64-Bit Graphical Installer \(898.6 MB\)](#)

### Python 3.11

[↓ 64-Bit Graphical Installer \(610.5 MB\)](#)

[↓ 64-Bit Command Line Installer \(612.1 MB\)](#)

[↓ 64-Bit \(M1\) Graphical Installer \(643.9 MB\)](#)

[↓ 64-Bit \(M1\) Command Line Installer \(645.6 MB \)](#)

### Python 3.11

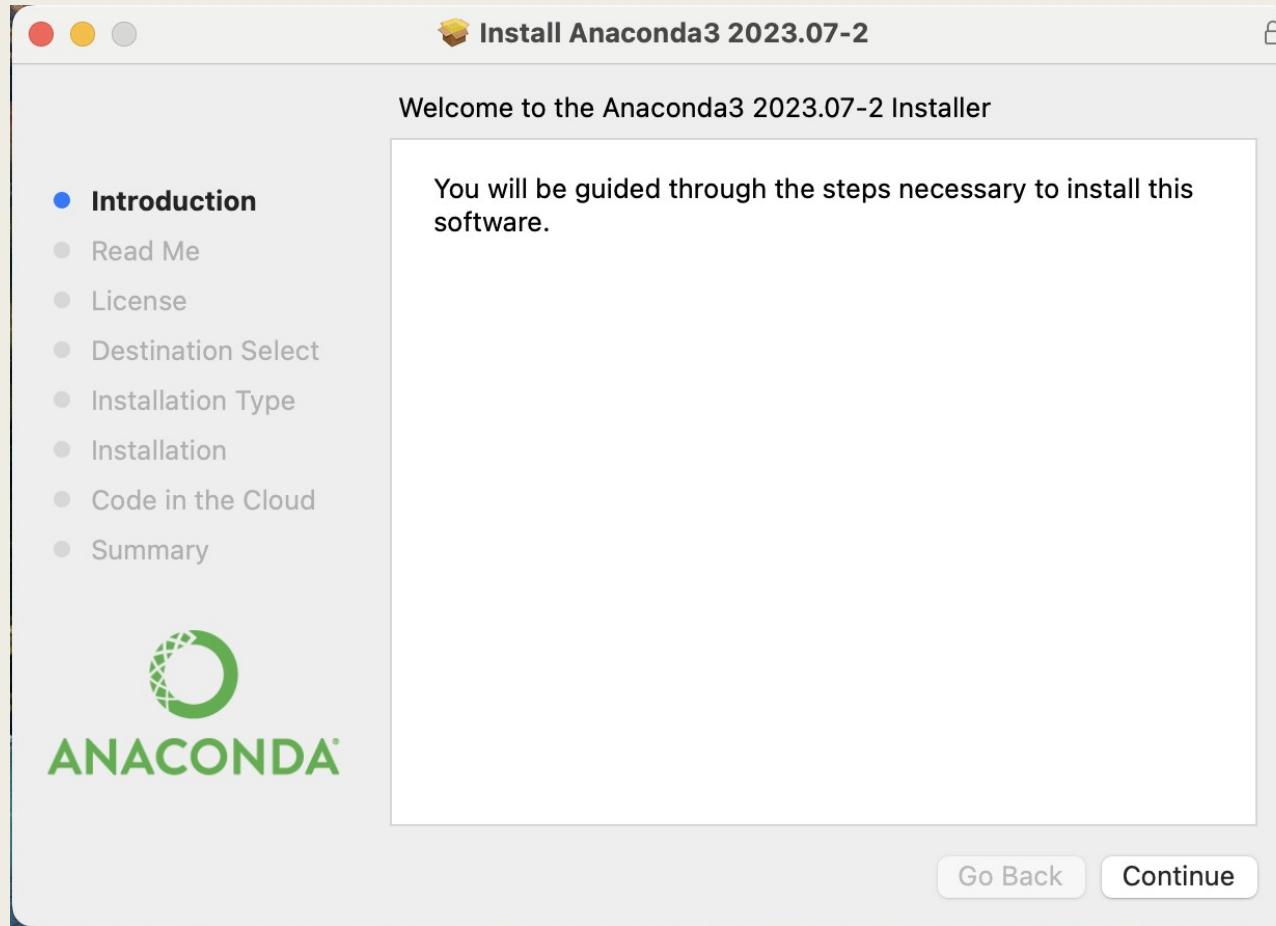
[↓ 64-Bit \(x86\) Installer \(1015.6 MB\)](#)

[↓ 64-Bit \(Power8 and Power9\) Installer \(473.8 MB\)](#)

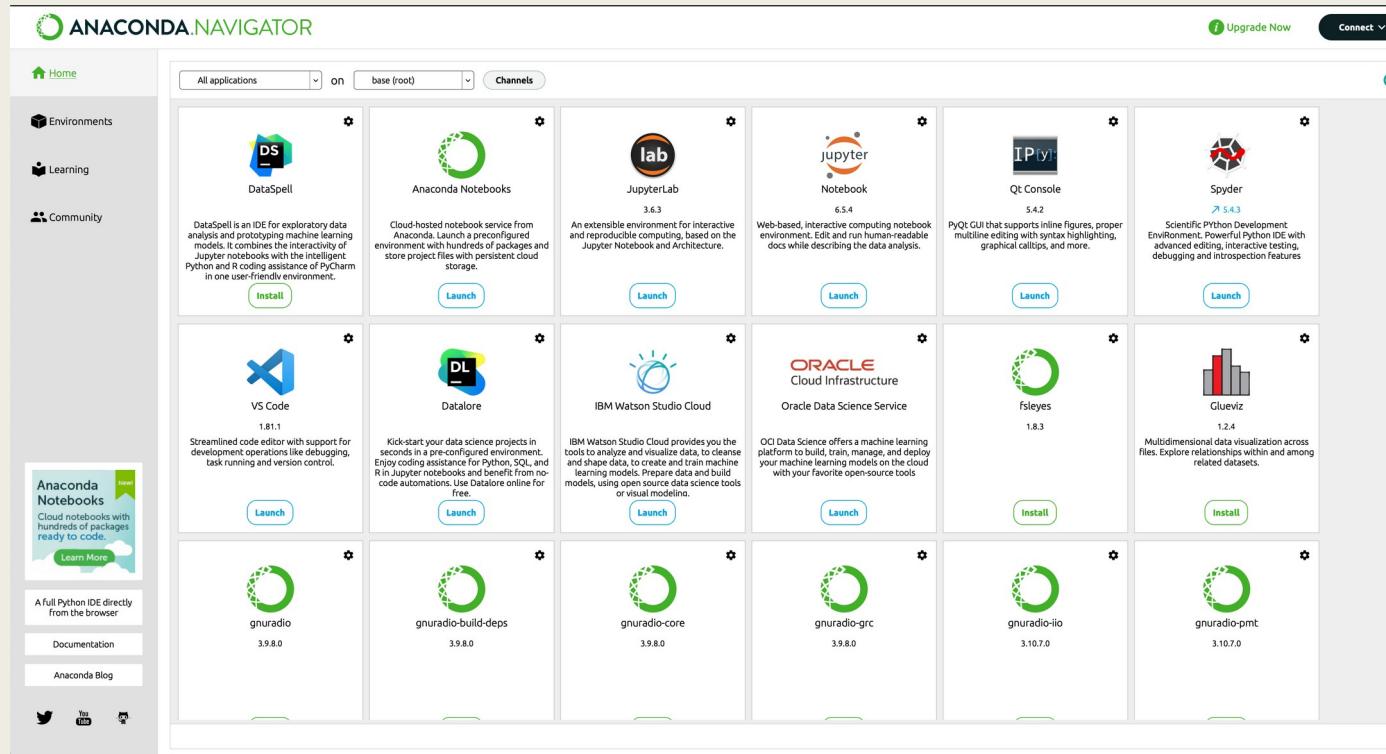
[↓ 64-Bit \(AWS Graviton2 / ARM64\) Installer \(727.4 MB\)](#)

[↓ 64-bit \(Linux on IBM Z & LinuxONE\) Installer \(340.8 MB\)](#)

- Scroll down to near the bottom of the webpage until you see this.
- Click on the appropriate **64-bit graphical installer link** for your computer

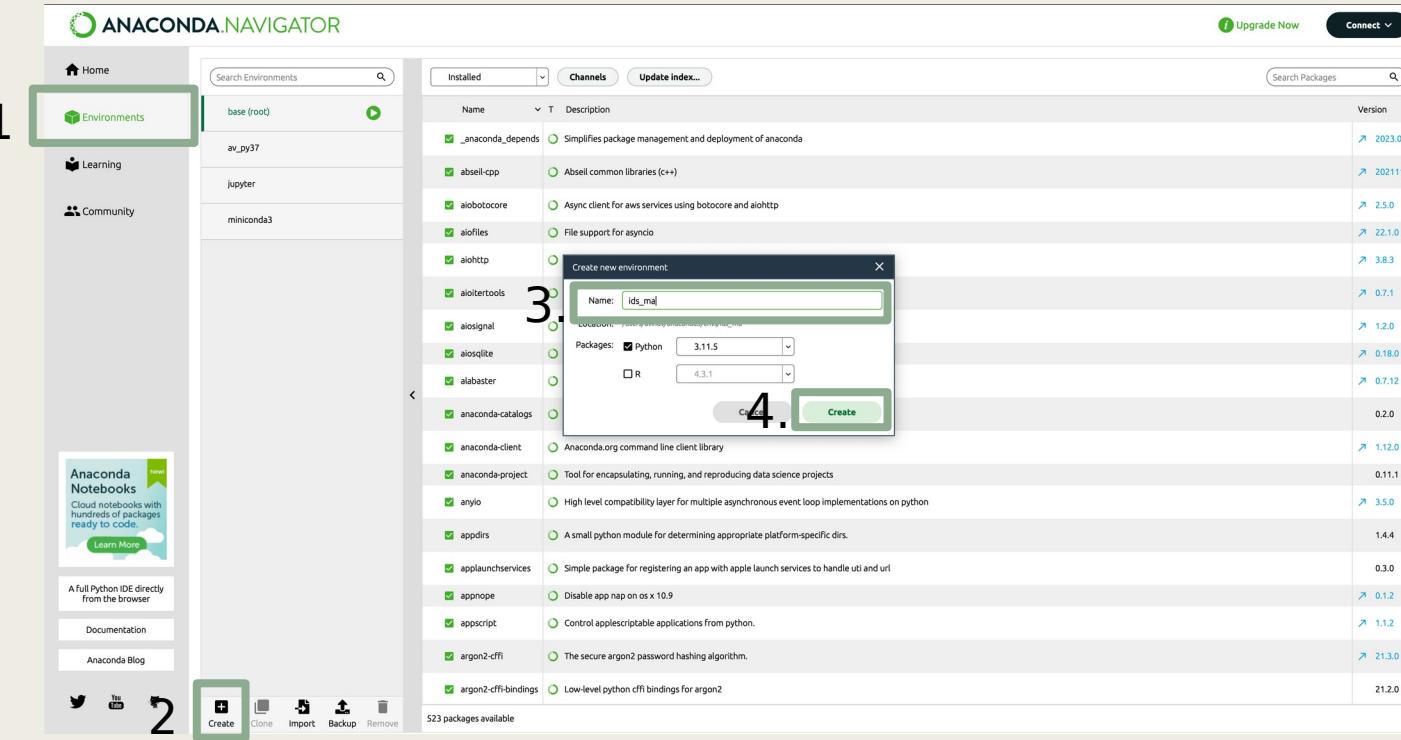


## ■ Follow download instructions



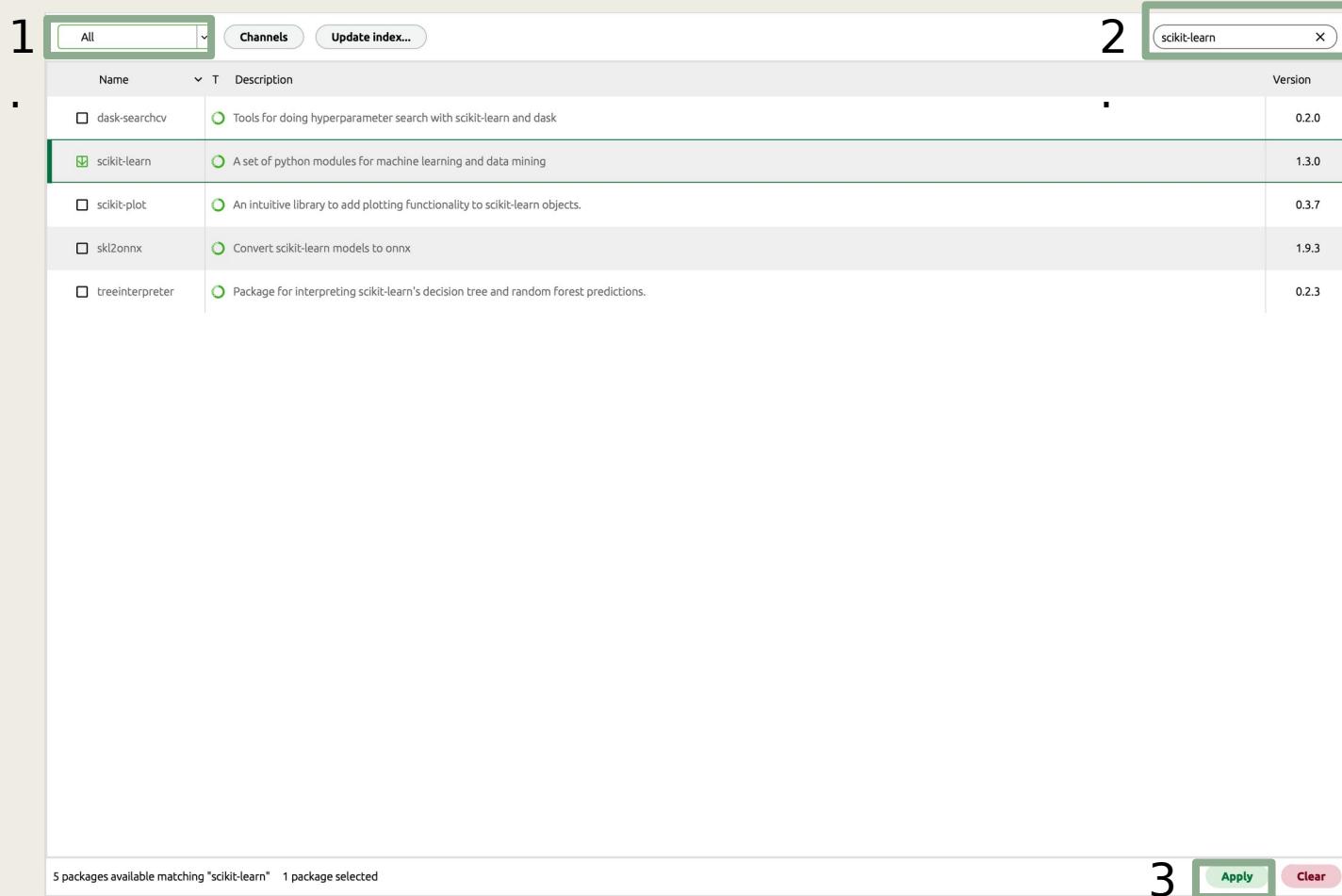
- Locate "Anaconda Navigator" in your applications folder
- When you open it, this is what it should look like

# SETTING UP YOUR ENVIRONMENT



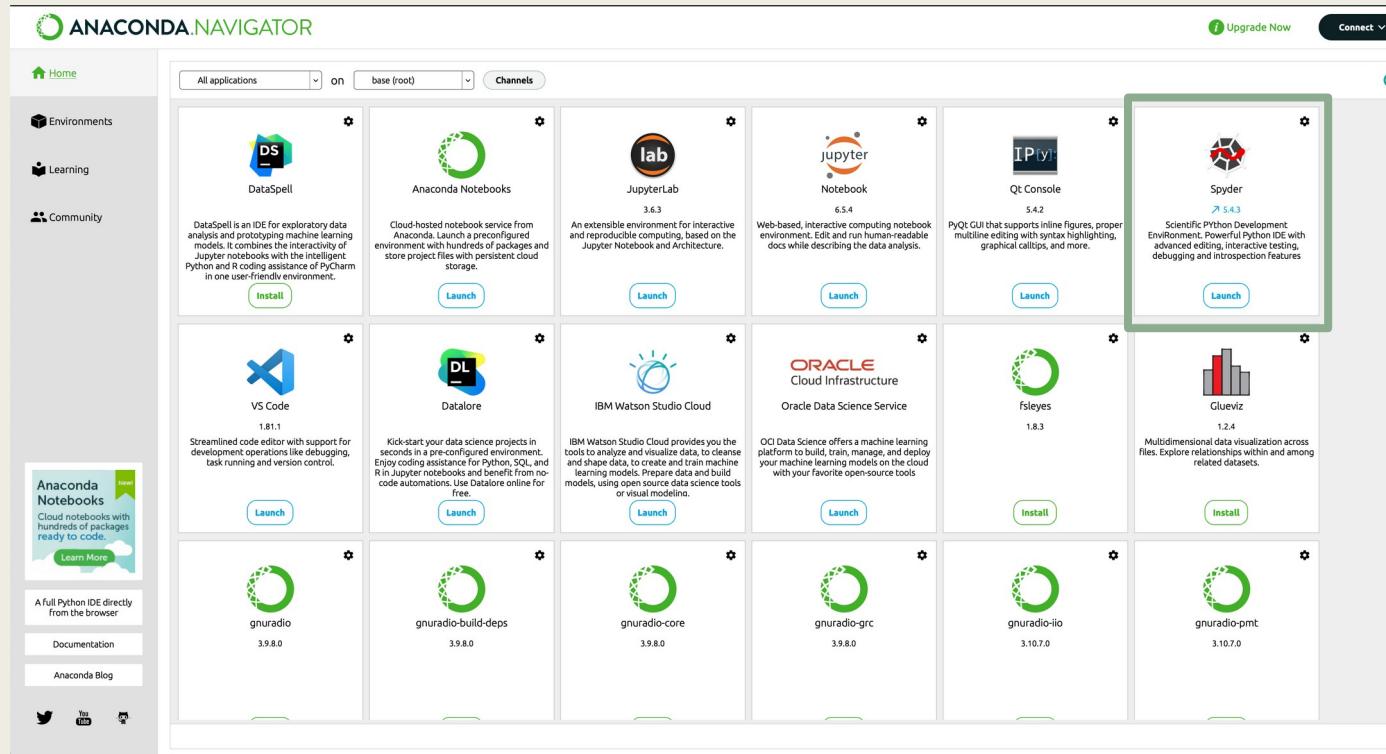
1. Go to “Environments”
2. Create a new Environment
3. Add a name for your new environment
4. Click Create and it’s done!

# INSTALLING NEW PACKAGES



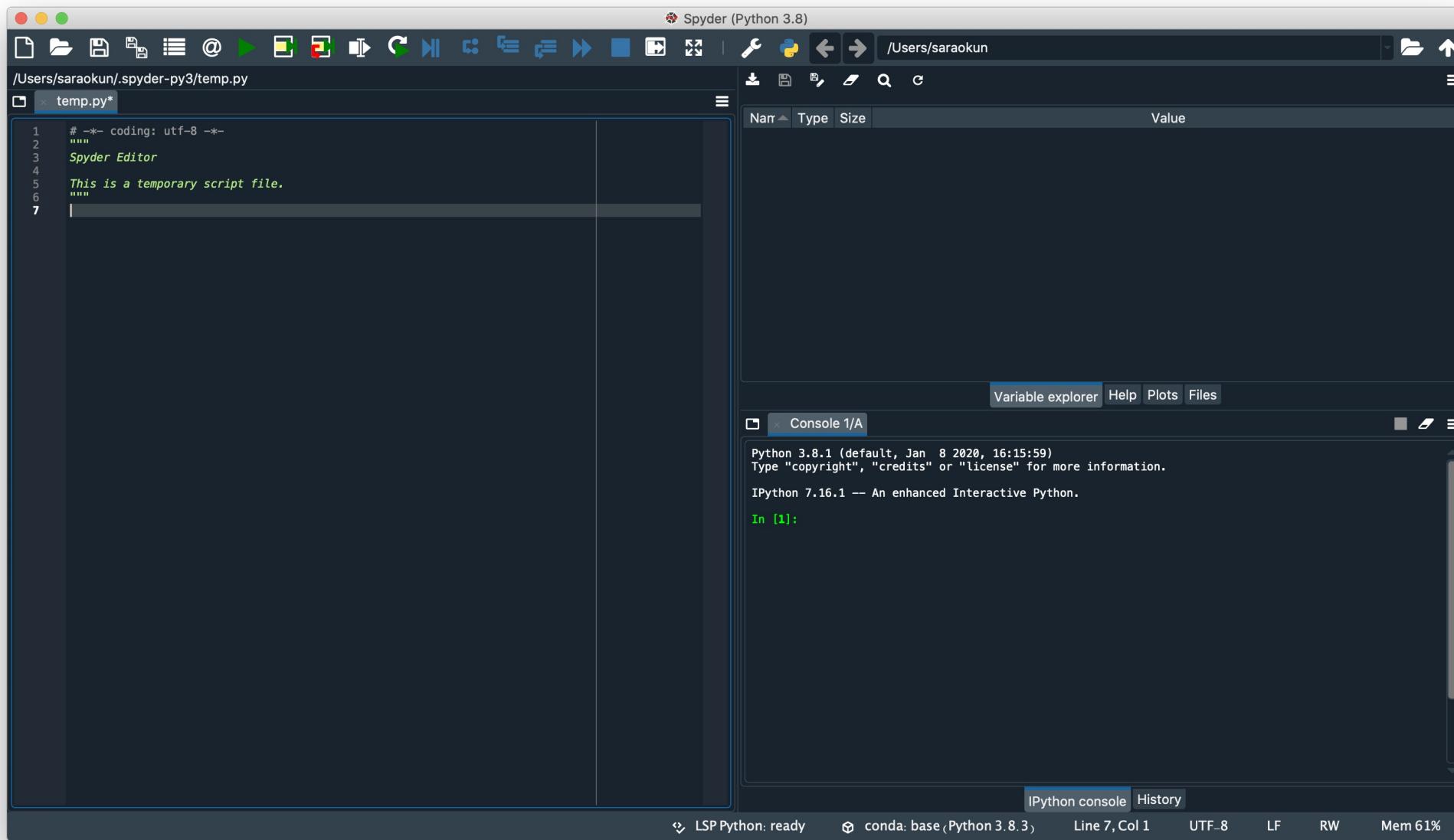
1. Set Package Discovery to all
2. Search for a new Environment, example scikit-learn
3. Click on apply
4. And viola! it installs the new package

# USING IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)



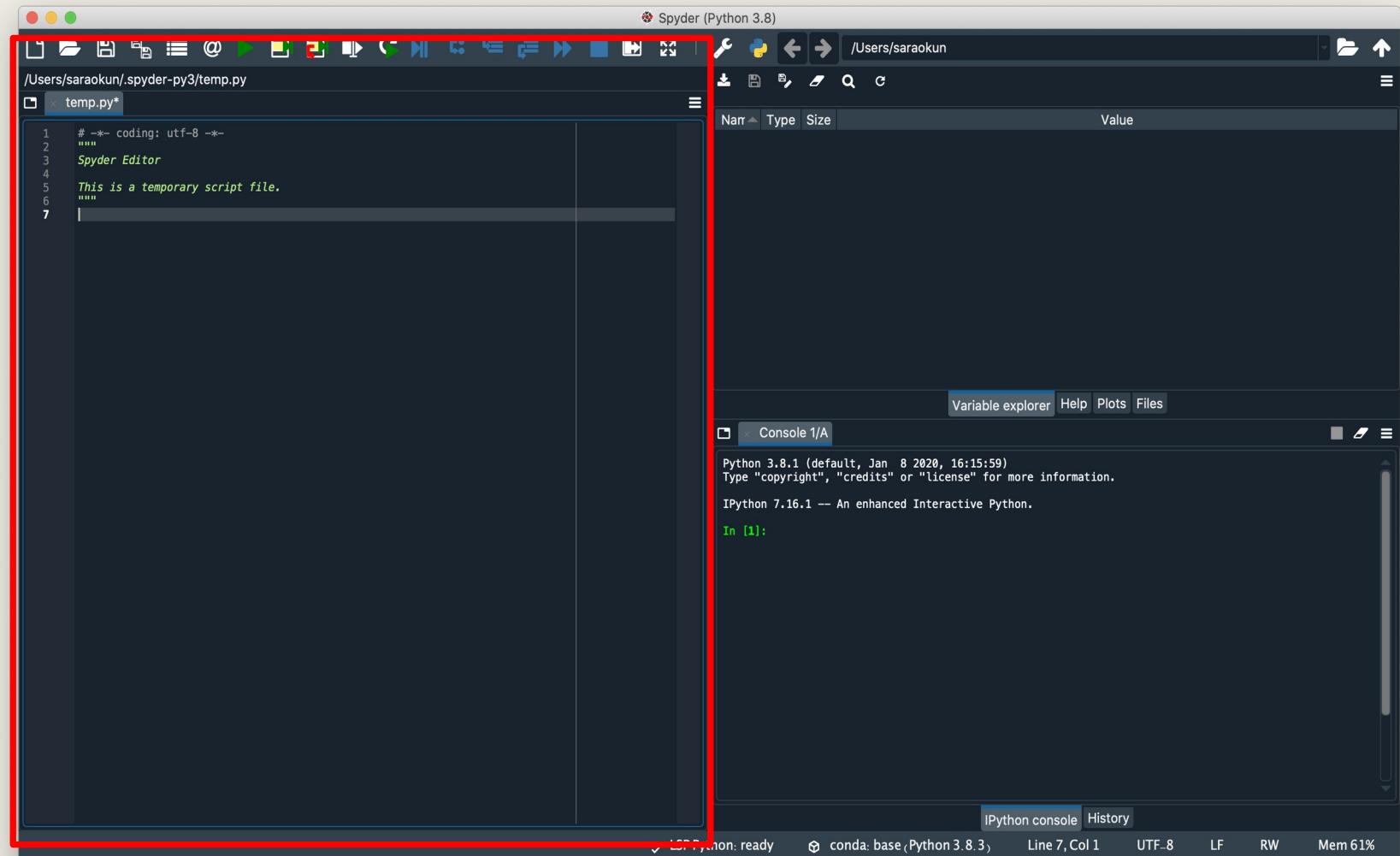
- We will be using Spyder
- If Not installed, Install Spyder(You can find it if you scroll down)
- Click “launch” to open Spyder

# This is what Spyder looks like!



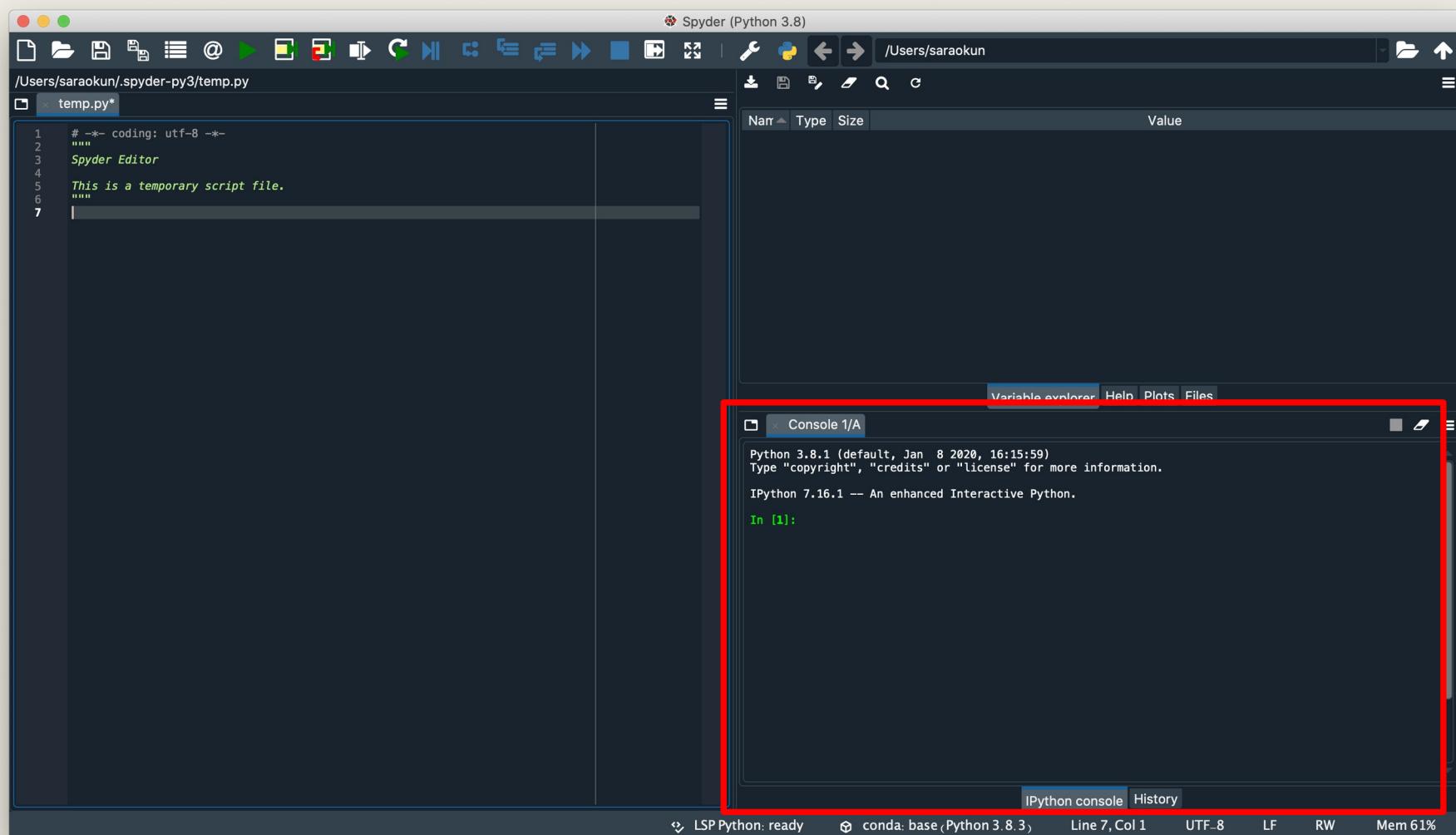
# Editor

- This is where you can write python scripts and save your work as a file.



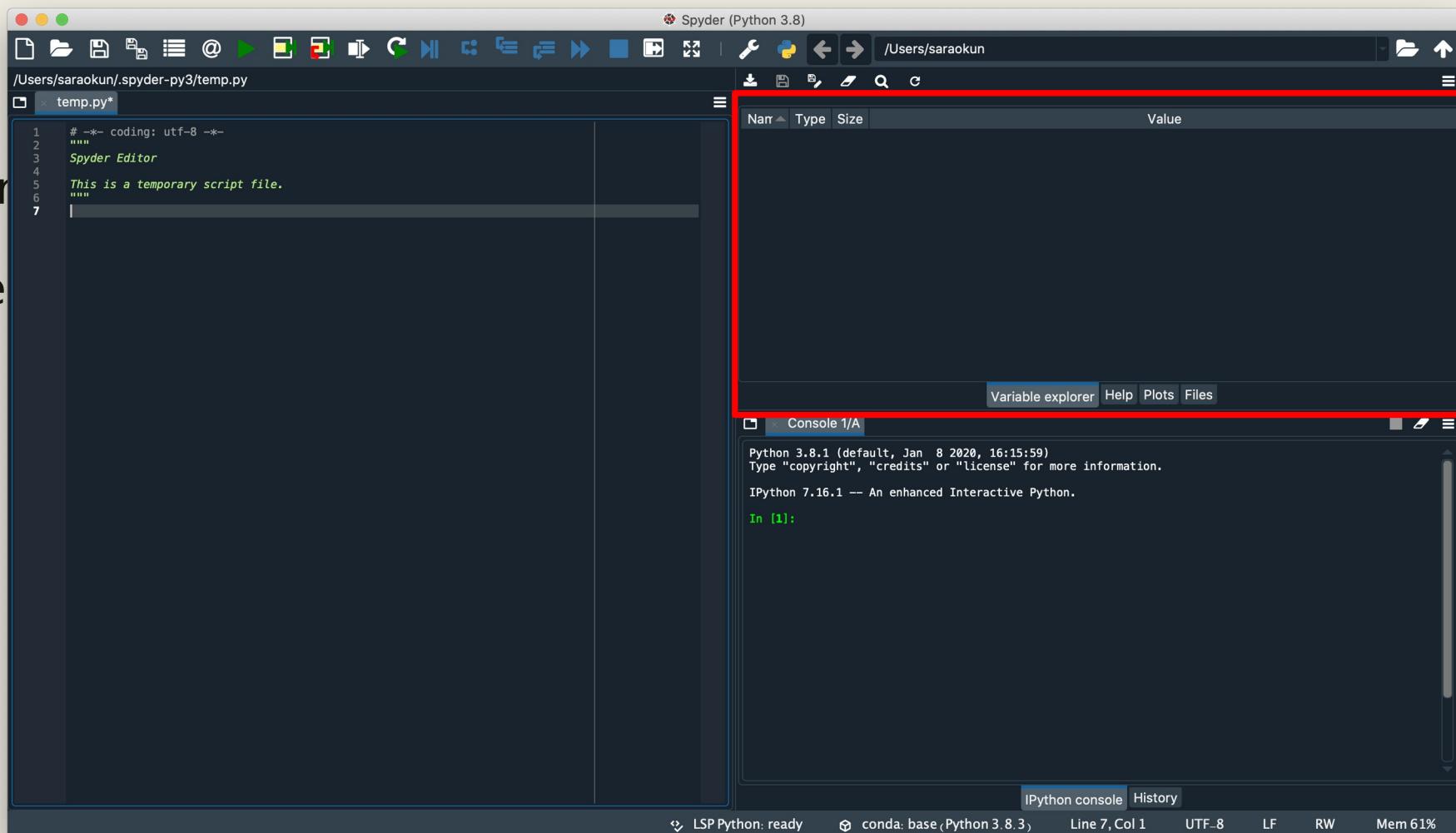
# Console

- This is where you will see your code output



# Variable explorer

- The variable explorer tab will show you all the objects you have created



# Simple example

- Here is what my environment looks like when I run my code
- I am writing a script here so that I can later save all of my code

The screenshot shows the Spyder Python IDE interface. A red box highlights the code editor window on the left, which displays a Python script named `temp.py`. The script contains the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6
7 #here is an example of a real simple script to show you what the editor,
8 #console and variable explorer look like when you're writing code
9
10 #simple math
11 print(2+2)
```

To the right of the code editor is the Variable explorer, showing a table with one row:

Name	Type	Size	Value

Below the code editor is the IPython console window, titled "Console 3/A". It shows the following output:

```
Python 3.8.1 (default, Jan  8 2020, 16:15:59)
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runcell(0, '/Users/saraokun/.spyder-py3/temp.py')
4

In [2]:
```

The status bar at the bottom of the IDE provides system information: LSP Python: ready, conda: base (Python 3.8.3), Line 10, Col 13, UTF-8, LF, RW, Mem 61%.

# Simple example

- Here is the output of my code

The screenshot shows the Spyder Python IDE interface. On the left is the code editor with a file named 'temp.py' containing the following code:

```
1  # -*- coding: utf-8 -*-
2  """
3      Spyder Editor
4
5      This is a temporary script file.
6
7      #here is an example of a real simple script to show you what the editor,
8      #console and variable explorer look like when you're writing code
9
10 #simple math
11 print(2+2)
```

In the center is the Variable explorer, which is currently empty. On the right is the IPython console, which displays the following output:

```
Python 3.8.1 (default, Jan  8 2020, 16:15:59)
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runcell(0, '/Users/sarokun/.spyder-py3/temp.py')
4
```

The output from the IPython console is highlighted with a red rectangle.

# Variables!

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

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7 #here is an example of a reall simple script to show you what the editor,
8 #console and variable explorer look like when you're writing code
9
10 #simple math
11 print(2+2)
12
13 #now i want to make a variable and store this information
14 x = 2+2
15 print(x)
```

A red box highlights the assignment statement `x = 2+2` and the subsequent `print(x)` call. The variable explorer window on the right shows a table with one entry:

Name	Type	Size	Value
x	int	1	4

The IPython console at the bottom shows the execution of the script:

```
Python 3.8.1 (default, Jan  8 2020, 16:15:59)
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

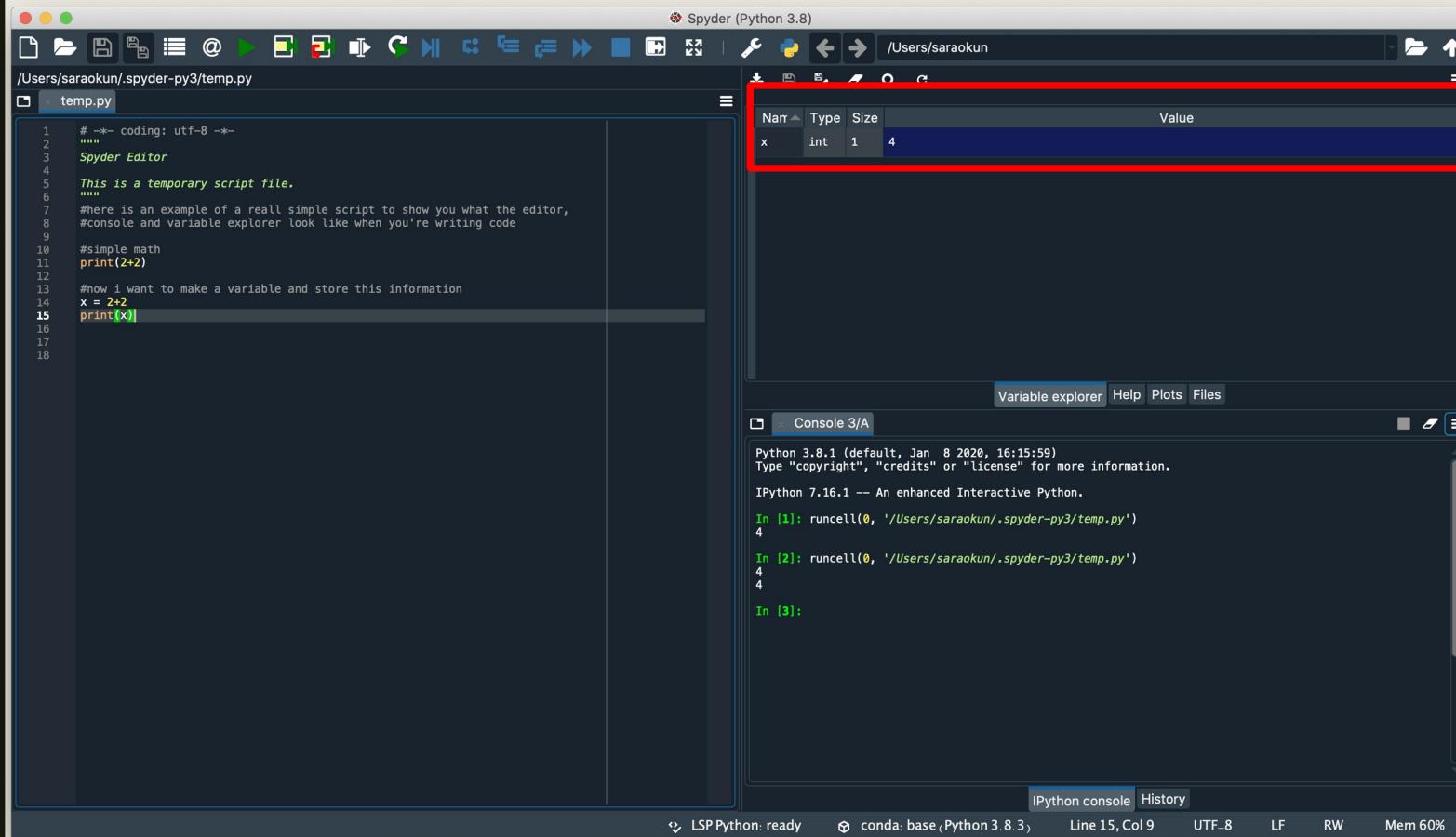
In [1]: runcell(0, '/Users/saraokun/.spyder-py3/temp.py')
4

In [2]: runcell(0, '/Users/saraokun/.spyder-py3/temp.py')
4
4

In [3]:
```

■ I then created a variable in my script

# Variables!



The screenshot shows the Spyder Python IDE interface. On the left is the code editor with a file named `temp.py` containing the following code:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.

here is an example of a reall simple script to show you what the editor,
console and variable explorer look like when you're writing code

#simple math
print(2+2)

#now i want to make a variable and store this information
x = 2+2
print(x)

```

In the center is the Variable explorer window, which lists variables with their type and size. A red box highlights the row for variable `x`, which has a type of `int`, a size of `1`, and a value of `4`.

At the bottom is the IPython console, showing the following session:

```
Python 3.8.1 (default, Jan  8 2020, 16:15:59)
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runcell(0, '/Users/saraokun/.spyder-py3/temp.py')
4

In [2]: runcell(0, '/Users/saraokun/.spyder-py3/temp.py')
4
4

In [3]:
```

The status bar at the bottom indicates `LSP Python: ready`, `conda: base, Python 3.8.3`, `Line 15, Col 9`, `UTF-8`, `LF`, `RW`, and `Mem 60%`.

- I then created a variable in my script
- After running the script, you can now see a variable and its value in the variable explorer

# Variables!

The screenshot shows the Spyder Python 3.8 IDE interface. The code editor on the left contains a script named `temp.py` with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6
7 here is an example of a reall simple script to show you what the editor,
8 #console and variable explorer look like when you're writing code
9
10 #simple math
11 print(2+2)
12
13 #now i want to make a variable and store this information
14 x = 2+2
15 print(x)
```

The variable explorer in the top right shows a single variable `x` of type `int` with a value of `4`.

The IPython console at the bottom has the following history:

```
In [1]: runcell(0, '/Users/saraokun/.spyder-py3/temp.py')
4
In [2]: runcell(0, '/Users/saraokun/.spyder-py3/temp.py')
4
4
In [3]:
```

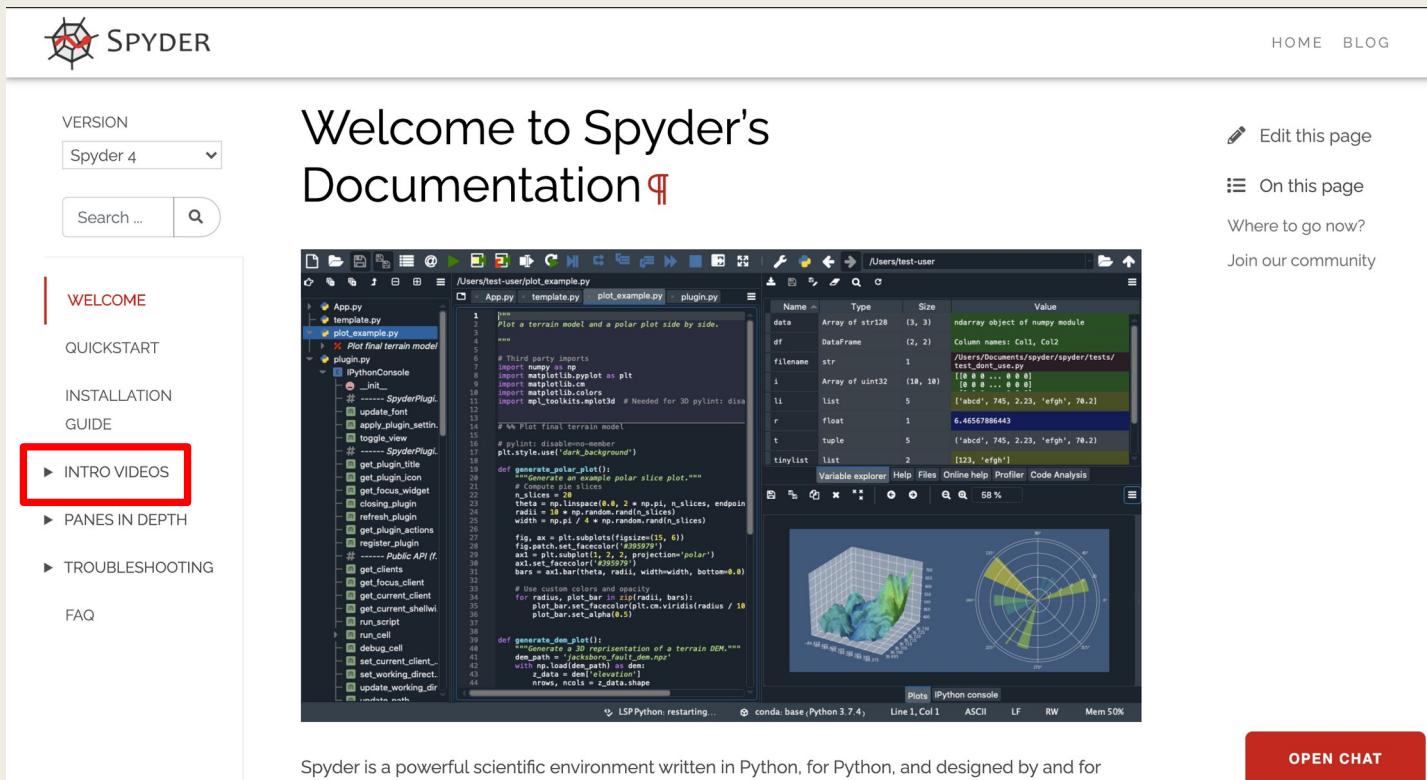
A red box highlights the output of the second run cell command in the IPython console.

- I then created a variable in my script
- After running the script, you can now see a variable and its value in the variable explorer
- Here is the output the second time I ran the code, after creating the variable

# Helpful links

■ <https://docs.spyder-ide.org/current/index.html>

- *There are some helpful intro videos I would recommend going through if you get confused*
- *You can find them under “Intro Videos”*



Spyder is a powerful scientific environment written in Python, for Python, and designed by and for

OPEN CHAT

# REPRODUCIBILIT Y EXAMPLE

# Example of Random Variable!

The screenshot shows the Spyder IDE interface. On the left, the code editor displays a file named 'temp.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 Example to demonstrate Reproducibility
6 """
7 import random
8 print(random.random())
9
```

On the right, the IPython console window shows the following output:

```
Python 3.11.5 (main, Sep 11 2023, 08:31:25) [Clang 14.0.6 ]
Type "copyright", "credits" or "license" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.5340282781919221

In [2]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.12938275045532488

In [3]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.20851441691310413

In [4]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.6844532973874464

In [5]:
```

- Random is used in almost every Data Science Library, Machine Learning on the inside(hidden)
- So every time you run a function(maybe ML/Data Science), you might get a different value
- How to solve this problem then?

# Example of Random Variable!

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a file named 'temp.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 Example to demonstrate Reproducibility
6 """
7 import random
8 random.seed(69420)
9 print(random.random())
10
```

On the right, the IPython Console shows the output of running the script multiple times:

```
In [5]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.681993770479116

In [6]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.681993770479116

In [7]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.681993770479116

In [8]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.681993770479116

In [9]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.681993770479116

In [10]: runfile('/Users/avinav/.spyder-py3/temp.py', wdir='/Users/avinav/.spyder-py3')
0.681993770479116

In [11]:
```

The status bar at the bottom indicates the environment: conda: ids\_ma (Python 3.11.5), Completions: conda(ids\_ma), LSP: Python, Line 10, Col 1, UTF-8, LF, RW, Mem 77%.

- Let's set the “seed”!
- The `seed()` method is used to initialize the random number generator. The random number generator needs a number to start with (a seed value), to be able to generate a random number.
- By default the random number generator uses the current system time.

IS ANYTHING IN  
COMPUTERS  
REALLY  
“RANDOM”?



# FREQUENTLY USED PACKAGES

# NUMPY

## NumPy documentation

Version: 1.26

[Download documentation](#): [Historical versions of documentation](#)

Useful links: [Installation](#) | [Source Repository](#) | [Issue Tracker](#) | [Q&A Support](#) | [Mailing List](#)

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

<https://numpy.org/doc/stable/>

# SCIKIT-LEARN

scikit-learn

*Machine Learning in Python*

Getting Started

Release Highlights for 1.3

GitHub

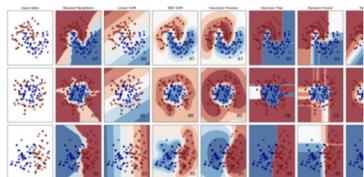
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

## Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** Gradient boosting, nearest neighbors, random forest, logistic regression, and more...



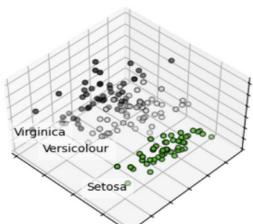
Examples

## Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, feature selection, non-negative matrix factorization, and more...

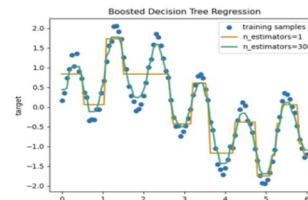


## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** Gradient boosting, nearest neighbors, random forest, ridge, and more...



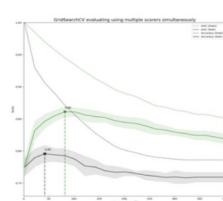
Examples

## Model selection

Comparing, validating and choosing parameters and models.

**Applications:** Improved accuracy via parameter tuning

**Algorithms:** grid search, cross validation, metrics, and more...

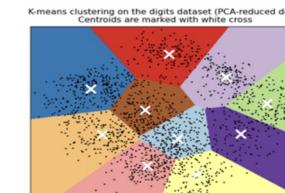


## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, HDBSCAN, hierarchical clustering, and more...



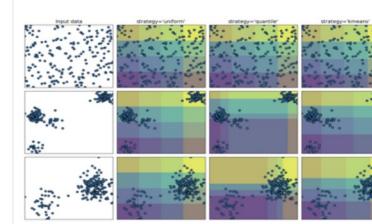
Examples

## Preprocessing

Feature extraction and normalization.

**Applications:** Transforming input data such as text for use with machine learning algorithms.

**Algorithms:** preprocessing, feature extraction, and more...



<https://scikit-learn.org/stable/>