Create Environment:

* conda create -p venv python==3.10
* conda activate venv/

Create .gitignore file:

* Files like .venv don’t get uploaded in github

Create folder .github/workflows/main.yaml

* Deployement github actions

Github Setup

* echo "# Credit-Risk-Default-Prediction" >> README.md
* git init
* git add README.md ( git add .)
* git commit -m "first commit"
* git branch -M main
* git remote add origin https://github.com/reevubabbai2003/Credit-Risk-Default-Prediction.git
* git push -u origin main

Setup.py

'''

The setup.py file is an essential part of packaging and

distributing Python projects. It is used by setuptools

(or distutils in older Python versions) to define the configuration

of your project, such as its metadata, dependencies, and more

'''

from setuptools import find\_packages

* find\_packages: scan through every folder and whenever there is \_\_init\_\_.py file consider that folder as a package.

**Why We Create a setup.py File 📦**

Think of a setup.py file as the official instruction manual for your project. It tells packaging tools like pip everything they need to know to handle your code correctly.

Its main purposes are:

1. **To Provide Metadata:** It contains key information about your project, such as its **name**, **version**, **author**, and a brief description. This is what you see when you browse packages on PyPI (the Python Package Index).
2. **To List Dependencies:** The **install\_requires** argument inside setup.py lists all the other Python packages that your project needs to run. When someone installs your project, pip reads this list and automatically installs all of them.
3. **To Find Your Code:** It tells pip which parts of your project are the actual Python packages to be included. The **find\_packages()** function is a common way to do this automatically.
4. **To Make it Installable:** Most importantly, having a setup.py file allows you to bundle your project into standard formats (like a "wheel" file) that can be easily shared and installed by others using a simple pip install command.

Without it, your project is just a collection of scripts that someone would have to manually download and manage.

* **-e** stands for **"editable."**
* **.** is a shortcut for the **current directory** (where your setup.py is).

So, the command means: "Install the project in the current directory in editable mode."

**How It Works: The Shortcut Analogy**

To understand what "editable" means, think of it like creating a shortcut versus making a copy of a file.

* **Normal Install (pip install .):** This is like **making a copy** of your project's files and moving that copy into Python's site-packages directory. If you change your original source code, the installed copy remains unchanged, and you won't see your updates until you reinstall.
* **Editable Install (pip install -e .):** This is like **creating a shortcut**. Instead of copying the files, pip places a link in the site-packages directory that points directly back to your original source code.

**The Connection**

When you run pip install -e ., pip **reads your setup.py file** to understand how to create this "shortcut." It uses the metadata and dependency information from setup.py to link your project into your Python environment.