venv:

Its an environment

setup.py

Create: setup.py

Google: python setup.py

What will it do?

It will create the machine learning application as a package, and we will be able to use this package further. Anyone else will also be able to use it.

from setuptools import find\_packages, setup

setup(

    name='mlproject',

    version= '0.0.1',

    author='Apoorva',

    author\_email='apoorvadhume@gmail.com',

    packages = find\_packages()

    install\_requires=['pandas', 'numpy', 'seaborn'] #All required packages

or

 install\_requires=get\_requirements('requirement.txt') #Get all the required packages directly from requirements.txt

)

SRC Folder:

Inside the source folder the ML application is to be found as a package.

**\_\_init\_\_.py:**

find\_packages from setup.py will help to determine how many packages are running inside \_\_init\_\_.py

**requirements.txt:**

All the required packages are mentioned here.

**-e .**

It indicates that setup.py is present and setup.py should be triggered.

The below code is mentioned in setup.py to read files from requirements.txt:

  install\_requires=['pandas', 'numpy', 'seaborn'] #All required packages

or

 install\_requires=get\_requirements('requirement.txt') #Get all the required packages directly from requirements.txt

**Basically, both setup.py and requirements.txt are connected to each other.**

**Component Folder:**

It is a part of module

Ex.:

* Data ingestion
* Data Transformation
* Hyperparameter tunning

**exception.py:**

In Python exception handling, exc\_tb.tb\_frame.f\_code.co\_filename is a common way to extract the filename where an exception occurred. This is part of accessing the traceback information associated with an exception.

Here's a breakdown of what each part signifies:

* **exc\_tb:**

This refers to the traceback object. When an exception is caught using except Exception as ex:, you can often get the traceback object by accessing ex.\_\_traceback\_\_ or by using sys.exc\_info(), which returns a tuple (exc\_type, exc\_value, exc\_tb).

* **.tb\_frame:**

This attribute of the traceback object points to the stack frame where the exception occurred. A stack frame contains information about the execution context of a function call.

* **.f\_code:**

This attribute of the frame object refers to the code object being executed in that frame. The code object contains static information about the function or module, including its name and the filename where it's defined.

* **.co\_filename:**

This attribute of the code object directly provides the name of the file in which the code object was defined and where the exception originated.

Purpose:

This sequence of attribute access is crucial for debugging and logging purposes. It allows you to pinpoint the exact file where an error originated, which is essential for understanding and resolving issues within a larger codebase.

Example Usage:

import sys  
  
def divide\_by\_zero():  
 return 1 / 0  
  
try:  
 divide\_by\_zero()  
except Exception as e:  
 exc\_type, exc\_value, exc\_tb = sys.exc\_info()  
 if exc\_tb:  
 file\_name = exc\_tb.tb\_frame.f\_code.co\_filename  
 line\_number = exc\_tb.tb\_lineno  
 print(f"Error occurred in file: {file\_name}, line: {line\_number}")  
 print(f"Error message: {e}")