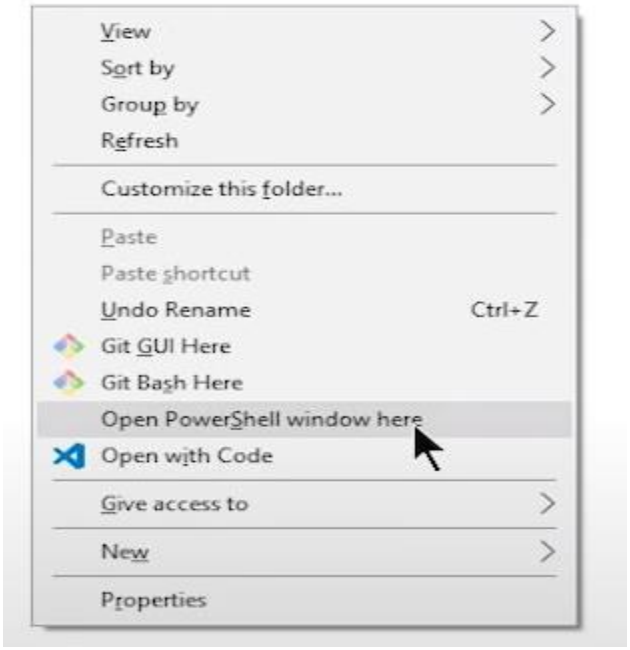
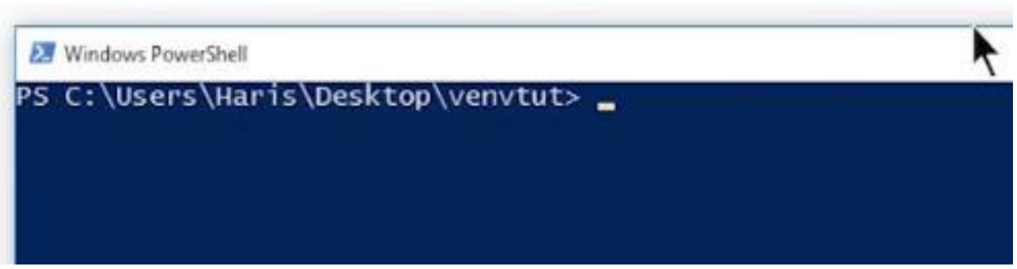
Create python virtual environment using window power shell

Virtual Environment & Requirements.txt

So, guys in this tutorial our introduction will be a little different than usual as we are not going be to using Pycharm. Instead today we are going to work with PowerShell. It may be a little unusual but don't worry, as I am here to guide you through every step.
So, let's get started. First, we will create a new folder. After opening the folder we will now open our PowerShell window using shift + mouse right-click.



Our PowerShell window will appear like this:



```
PS C:\Users\Haris\Desktop\venvtut> _
```

Note that the path it is showing is of our current folder. That is actually the benefit of using shift + mouse right-click.
Now let us move on to some theoretical concept.
A virtual environment is a tool or an aid provided to us by Python to keep the dependencies that we have utilized earlier in a few projects, constant. In simple terms, after some duration, Python keeps on launching and upgrading its versions. The new versions yet being better can have certain disadvantages for few users,

because, in every new update, new functions are added to the modules, and previous ones may be upgraded. So, there is a chance that a function that used to work earlier will not work as it used to.
To save ourselves from such situations, Python has allowed us to use of a virtual environment.

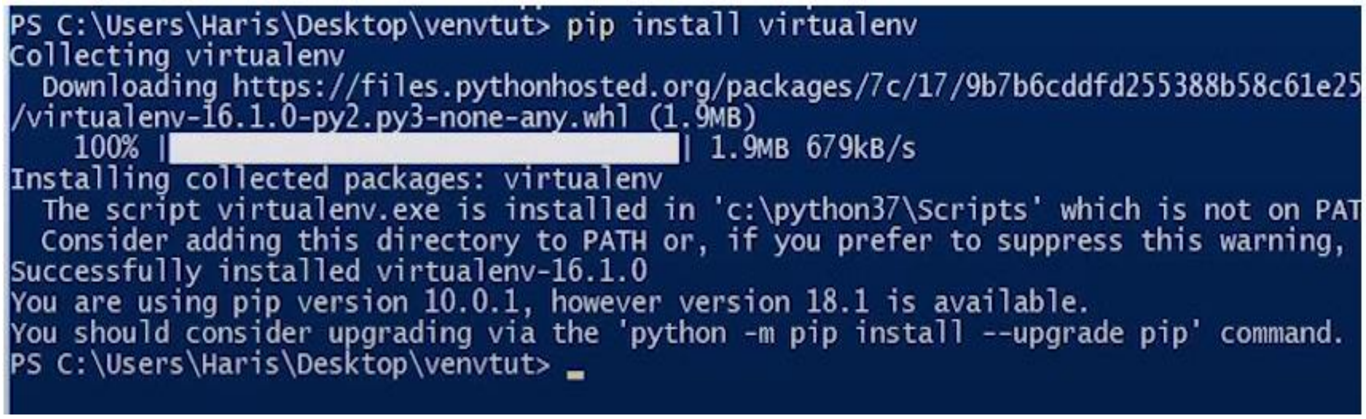
What virtual environment does?

Virtual Environment saves the current state of our compiler along with the state of their modules and libraries. So in this way even if Python has made certain changes in its module, our virtual environment can still work as before even after years. We can also install different packages and **"dataframes"** in our virtual environment.
To be more clear, the virtual environment works exactly the same way as the Python we have installed on our windows/mac/Linux currently because a virtual environment is just a clone of the original product.

Using Virtual Environments

To get started, install the virtualenv tool with pip:

```
$ pip install virtualenv
```

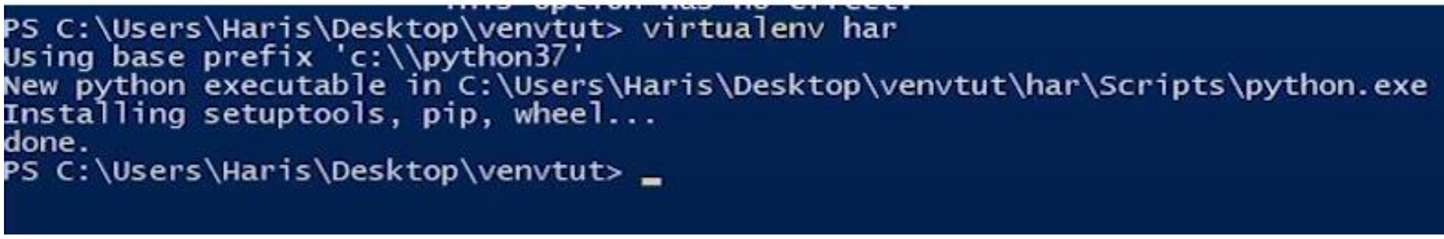


```
PS C:\Users\Haris\Desktop\venvtut> pip install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/7c/17/9b7b6cddfd255388b58c61e25
/virtualenv-16.1.0-py2.py3-none-any.whl (1.9MB)
    100% |████████████████████████████████| 1.9MB 679kB/s
Installing collected packages: virtualenv
  The script virtualenv.exe is installed in 'c:\python37\Scripts' which is not on PAT
  Consider adding this directory to PATH or, if you prefer to suppress this warning,
Successfully installed virtualenv-16.1.0
You are using pip version 10.0.1, however version 18.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\Users\Haris\Desktop\venvtut> _
```

virtualenv is a tool. It is used to create isolated Python environments.
To assign a name to your virtual environment, use command

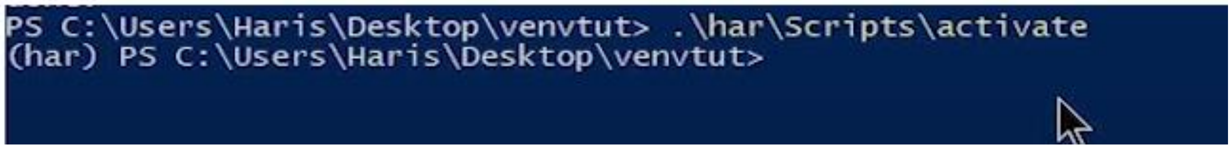
```
$ virtualenv virtualenv_name
```

After running this command, a directory named folder_name will be created. This directory will contain all the necessary executables to use the packages that the Python project would need. Python packages will be installed in this directory.


```
PS C:\Users\Haris\Desktop\venvtut> virtualenv har
Using base prefix 'c:\\python37'
New python executable in C:\Users\Haris\Desktop\venvtut\har\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
PS C:\Users\Haris\Desktop\venvtut> _
```

Now, after creating a virtual environment, you need to activate it. When you open the directory, it contains folders like include, lib, script, and tcl. When you open the script folder, you'll find a file activate.bat. You can activate the virtual environment by simply clicking on this file or using the command prompt and writing the following command.

```
$ .\virtualenv_name\Scripts\activate
```

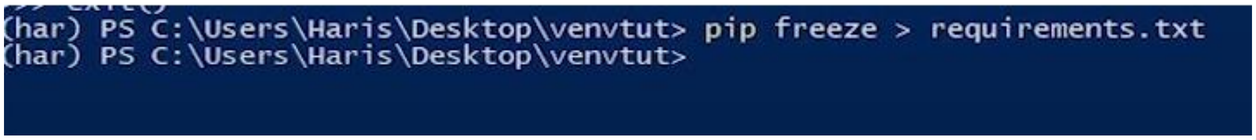

```
PS C:\Users\Haris\Desktop\venvtut> .\har\Scripts\activate
(har) PS C:\Users\Haris\Desktop\venvtut>
```

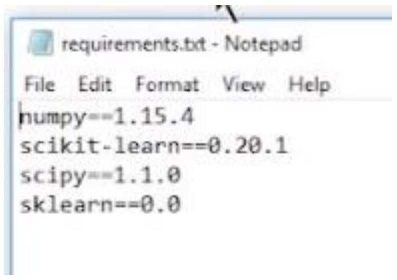
Once the virtual environment is activated, the name of your virtual environment will appear on the terminal's left side.
When you are done working in the virtual environment for the moment, you can deactivate it:

```
(virtualenv_name)$ deactivate
```

Now moving onto **requirement.txt**. When we run a certain command pip **freeze** > **requirement.txt,** a file will be generated in the directory where our virtual environment is based. The file will contain all the details related to the external packages that we have installed along with their versions. By having the requirement.txt file, we can create our virtual machine again easily by downloading all the same libraries, having the same versions by a simple command.

```
freeze > requirement.txt
```

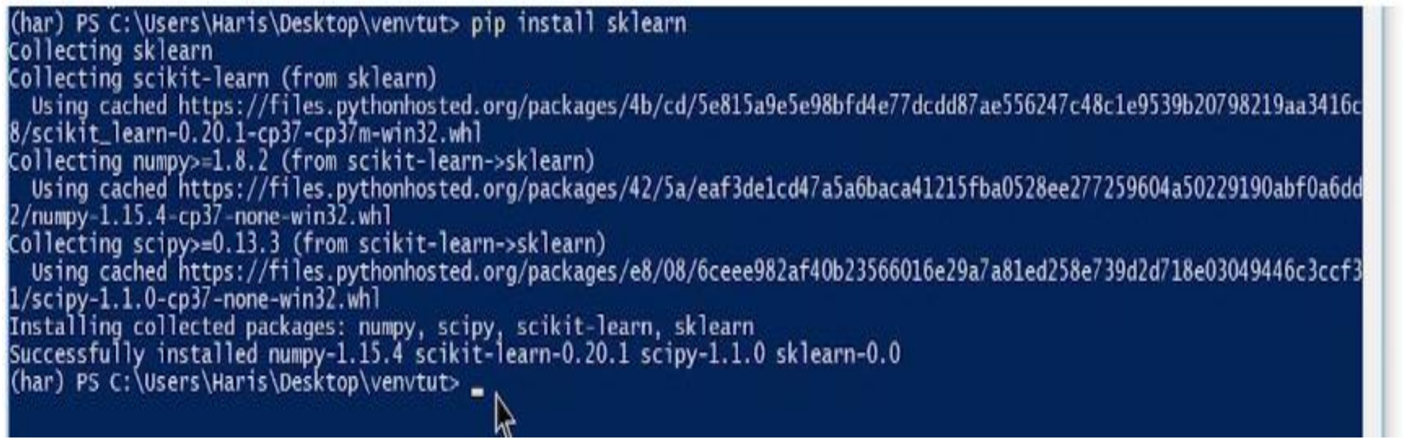

```
(har) PS C:\Users\Haris\Desktop\venvtut> pip freeze > requirements.txt
(har) PS C:\Users\Haris\Desktop\venvtut>
```

The requirement.txt folder will contain data like this: 

```
numpy==1.15.4
scikit-learn==0.20.1
scipy==1.1.0
sklearn==0.0
```

We can install all the packages one by one by a command:

```
pip install package_name == version
```



```
(har) PS C:\Users\Haris\Desktop\venvtut> pip install sklearn
Collecting sklearn
Collecting scikit-learn (from sklearn)
  Using cached https://files.pythonhosted.org/packages/4b/cd/5e815a9e5e98bfd4e77dcdd87ae556247c48c1e9539b20798219aa3416c
8/scikit_learn-0.20.1-cp37-cp37m-win32.whl
Collecting numpy>=1.8.2 (from scikit-learn->sklearn)
  Using cached https://files.pythonhosted.org/packages/42/5a/eaf3de1cd47a5a6baca41215fba0528ee277259604a50229190abf0a6dd
2/numpy-1.15.4-cp37-none-win32.whl
Collecting scipy>=0.13.3 (from scikit-learn->sklearn)
  Using cached https://files.pythonhosted.org/packages/e8/08/6ceee982af40b23566016e29a7a81ed258e739d2d718e03049446c3ccf3
1/scipy-1.1.0-cp37-none-win32.whl
Installing collected packages: numpy, scipy, scikit-learn, sklearn
Successfully installed numpy-1.15.4 scikit-learn-0.20.1 scipy-1.1.0 sklearn-0.0
(har) PS C:\Users\Haris\Desktop\venvtut> _
```

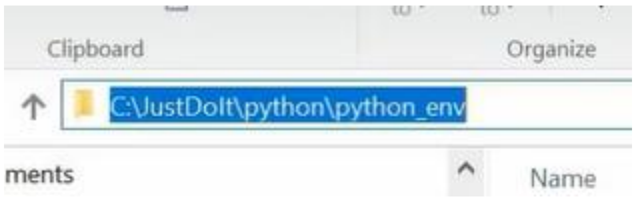
But in case we have a large number of libraries installed, this will take a massive amount of time as we have to install each one by one so we have another method by which we can install all the packages at once by using the requirement.txt file. The syntax would be:

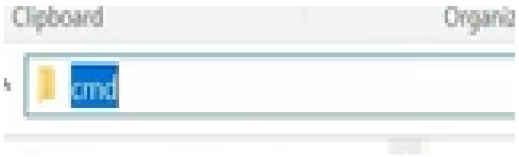
```
pip install -r .\requirements.txt
```

Code file as described in the video

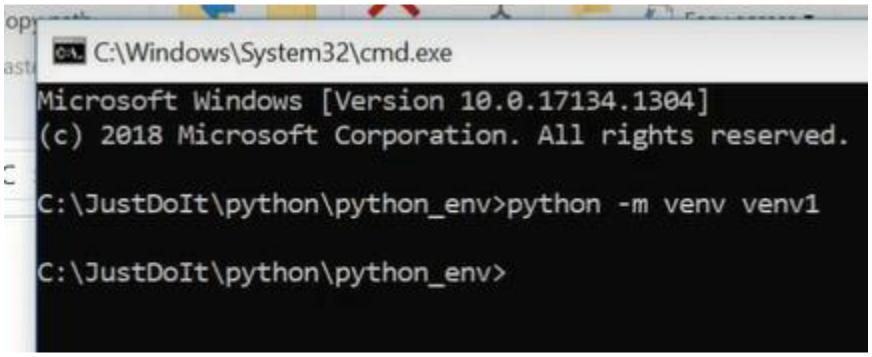
```
numpy==1.15.4
scikit-learn==0.20.1
scipy==1.1.0
sklearn==0.0
```

How to create Virtual Environment In Jupyter Notebook using cmd





Create Virtual 5nvironment

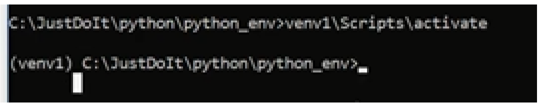


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\JustDoIt\python\python_env>python -m venv venv1

C:\JustDoIt\python\python_env>
```

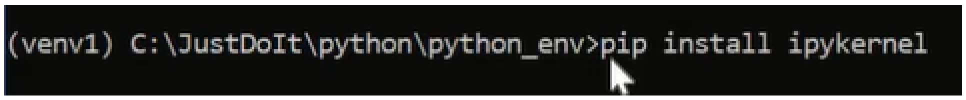
Activate virtual environment



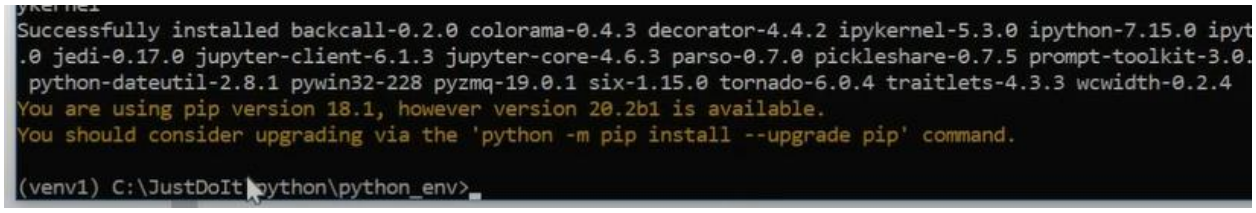
```
C:\JustDoIt\python\python_env>venv1\Scripts\activate

(venv1) C:\JustDoIt\python\python_env>_
```

Install kernel



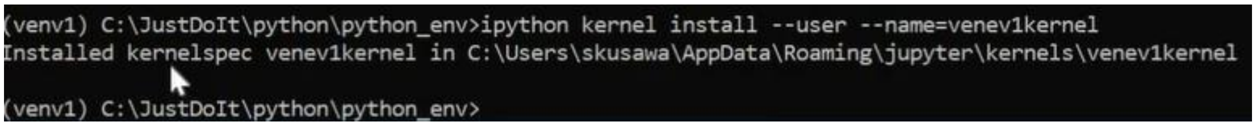
```
(venv1) C:\JustDoIt\python\python_env>pip install ipykernel
```



```
Successfully installed backcall-0.2.0 colorama-0.4.3 decorator-4.4.2 ipykernel-5.3.0 ipython-7.15.0 ipyt
.0 jedi-0.17.0 jupyter-client-6.1.3 jupyter-core-4.6.3 parso-0.7.0 pickleshare-0.7.5 prompt-toolkit-3.0.
 python-dateutil-2.8.1 pywin32-228 pyzmq-19.0.1 six-1.15.0 tornado-6.0.4 traitlets-4.3.3 wcwidth-0.2.4
You are using pip version 18.1, however version 20.2b1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(venv1) C:\JustDoIt\python\python_env>_
```

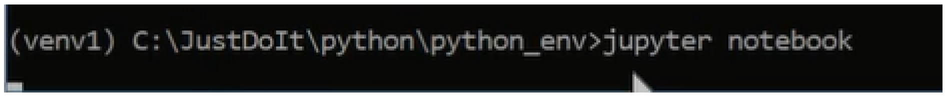
Install new kernel



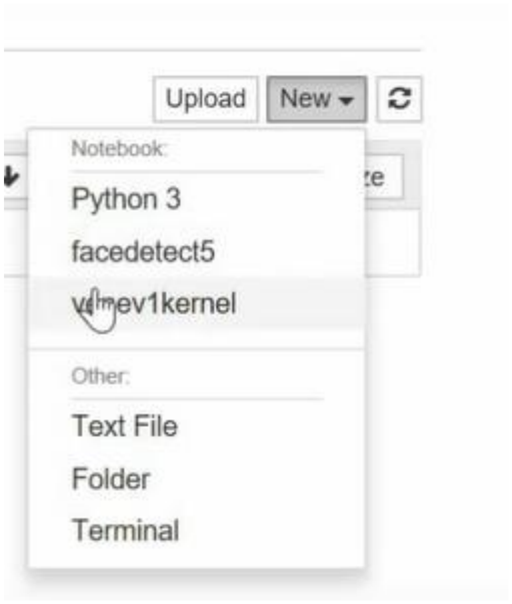
```
(venv1) C:\JustDoIt\python\python_env>ipython kernel install --user --name=venev1kernel
Installed kernelspec venev1kernel in C:\Users\skusawa\AppData\Roaming\jupyter\kernels\venev1kernel

(venv1) C:\JustDoIt\python\python_env>
```

Launch jupyter notebook



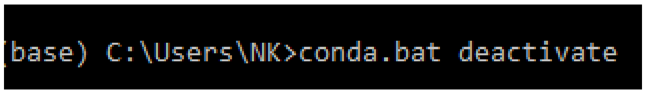
```
(venv1) C:\JustDoIt\python\python_env>jupyter notebook
```



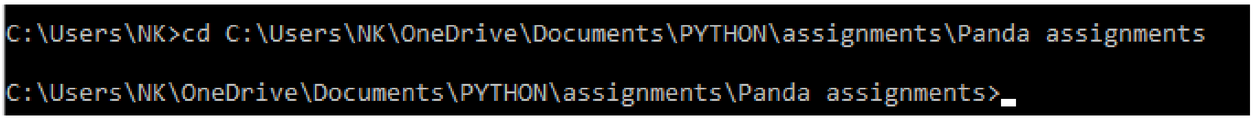
- **List Kernels**
  - **Jupyter kernelspec list**
- **Remove kernel**
  - **Jupyter kernelspec remove <kernel_name>**

How to create Virtual Environment In Jupyter Notebook using Anaconda Prompt

- Deactivate from current environment and change the directory



```
(base) C:\Users\NK>conda.bat deactivate
```



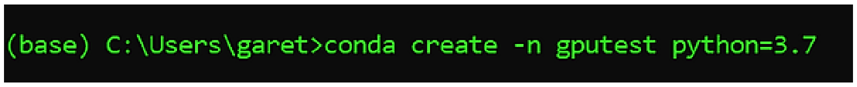
```
C:\Users\NK>cd C:\Users\NK\OneDrive\Documents\PYTHON\assignments\Panda assignments

C:\Users\NK\OneDrive\Documents\PYTHON\assignments\Panda assignments>
```

Creating and Activating A New Environment - Step 3

Creating a new conda environment prevents conflicts between environment libraries. To put it simply, creating a new environment stops you having to constantly install and uninstall dependencies every time you want to switch between projects. Instead, you can just switch environments. That way, your environment has it's own kernel along with it's own libraries / dependencies.

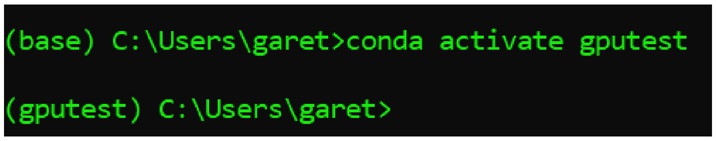
You can create a new environment with the following command. After -n is where we specify the environment name. This is followed by the version of python we want to install.



```
(base) C:\Users\garet>conda create -n gputest python=3.7
```

We then need to activate our environment. The commands differ slightly between Windows and Mac. After entering the command, you should see that (base) has now changed to (your environment name). This means that any commands entered from this point on will be executed within that environment.

Windows:



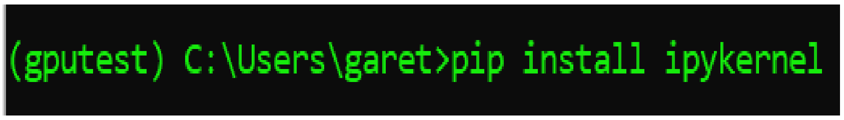
```
(base) C:\Users\garet>conda activate gputest

(gputest) C:\Users\garet>
```

Installing a Python Kernel - Step 4

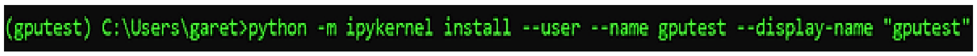
We need to create a Python Kernel inside of our environment so that we can run our code. Anaconda used to create a kernel by default. However, this is no longer the case, so we must do it manually.

First we need install ipykernel:



```
(gputest) C:\Users\garet>pip install ipykernel
```

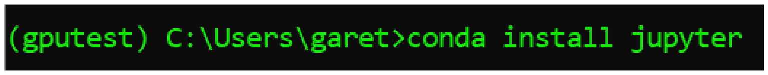
We now need to install a kernel. Again, make sure you have activated your environment, (gputest), in this case. To install a kernel, enter the following command. The variables after '--name' and display-name will need to be set to the name of your environment. If entered correctly, you'll see a response 'Installed kernelspec'.



```
(gputest) C:\Users\garet>python -m ipykernel install --user --name gputest --display-name "gputest"
```

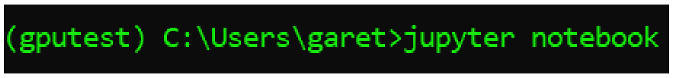
Jupyter Notebook Setup - Step 5

After creating a new environment, we essentially start from scratch with an empty environment. Therefore, to use Jupyter Notebook we will need install it via Anaconda. Again, make sure you have activated your new environment.



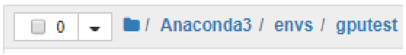
```
(gputest) C:\Users\garet>conda install jupyter
```

After installing Jupyter, we can now launch Jupyter Notebook with the following command:



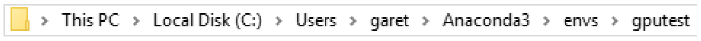
```
(gputest) C:\Users\garet>jupyter notebook
```

After entering the command above, a Jupyter notebook server should launch via your default browser. Using the notebook server file system, navigate to your environment directory. For example:



0 / Anaconda3 / envs / gputest

Which also resembles this directory in your system:



This PC › Local Disk (C:) › Users › garet › Anaconda3 › envs › gputest

To freeze

Pip freeze > requirement.txt

To delete virtual enviroments

Step 1: Find the Conda environment to delete

To find the name of the environment you want to delete, we can get the list of all Conda environments as follov

```
conda env list
```

Let the name of the environment to be delete is `corrupted_env`.

Step 2: Get out of the environment

You cannot delete the conda environment you are within. To get out of the current environme

```
conda deactivate
```

Step 3: Delete the Conda Environment (6 commands)

If the name of the environment to be delete is `corrupted_env`, then use the following command to delete it:

```
conda env remove -n corrupted_env
```

OR

```
conda env remove --name corrupted_env
```

Alternatively, we can use the following command:

```
conda remove --name corrupted_env --all
```

OR

```
conda remove -n corrupted_env --all
```

If you have the path where a conda environment is located, you can directly specify the path instead of name of the conda environment. In this case, the command will be:

```
conda env remove -p /path/to/env
```

OR

```
conda env remove --prefix /path/to/env
```

Delete Directory directly?

It is not advised to delete the directory directly where the conda environment is stored. In some cases, it might be necessary so the steps are:

- Find the path of the conda environment using:

```
conda info --envs
```

- Delete the directory directly:

```
rm -rf /Users/username/.local/share/conda/envs/corrupted_env
```