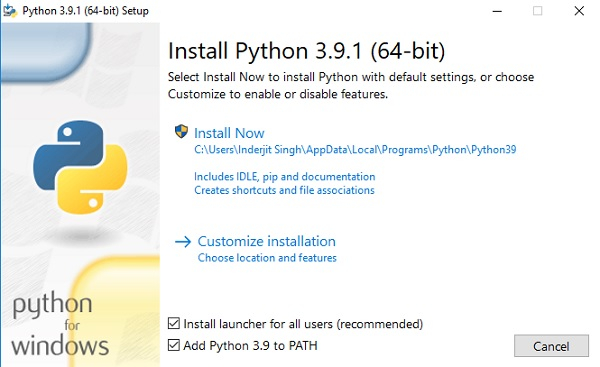
**Data Visualization By Chart**

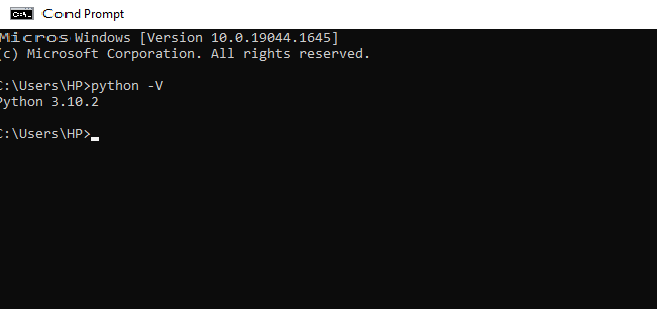
1. **Install Requirements:**

**Python**

* Download python from here: <https://www.python.org/ftp/python/3.10.4/python-3.10.4-amd64.exe>
* You can verify your **python** installation by executing the following command in cmd.

python –V

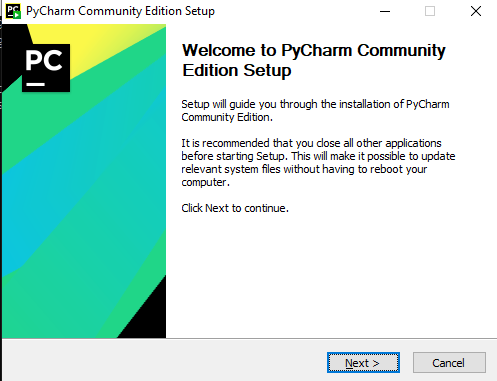




**Pycharm**

* **PyCharm** is the best IDE I've ever used. With **PyCharm**, you can access the command line, connect to a database, create a virtual environment, and manage your version control system all in one place.
* Download it from the following link:

<https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows&code=PCC>

****

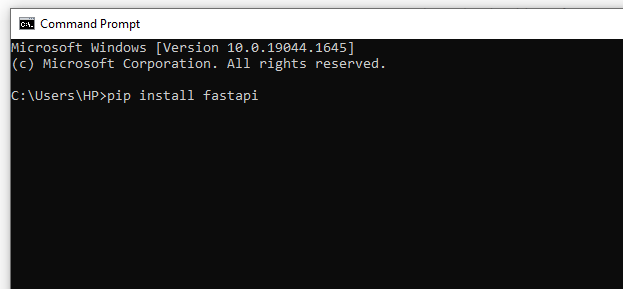
**FastAPI**

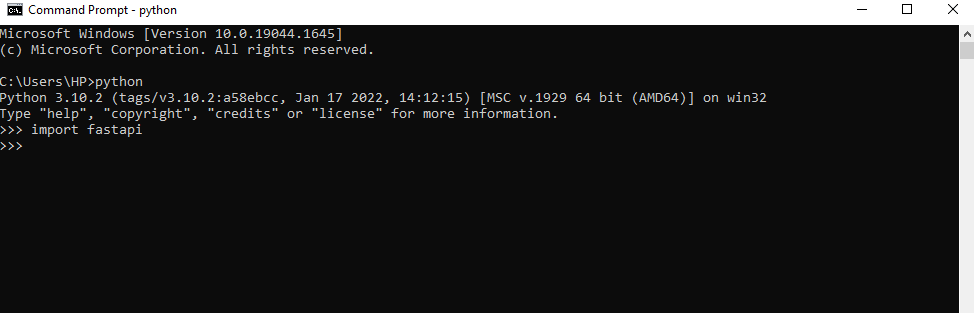
* FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type.
* **Fast**: Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic). [One of the fastest Python frameworks available](https://pypi.org/project/fastapi/#performance).
* Install it using the command on cmd:

pip install fastapi

* You can verify your **fastapi** installation by executing the following command in cmd.

import fastapi



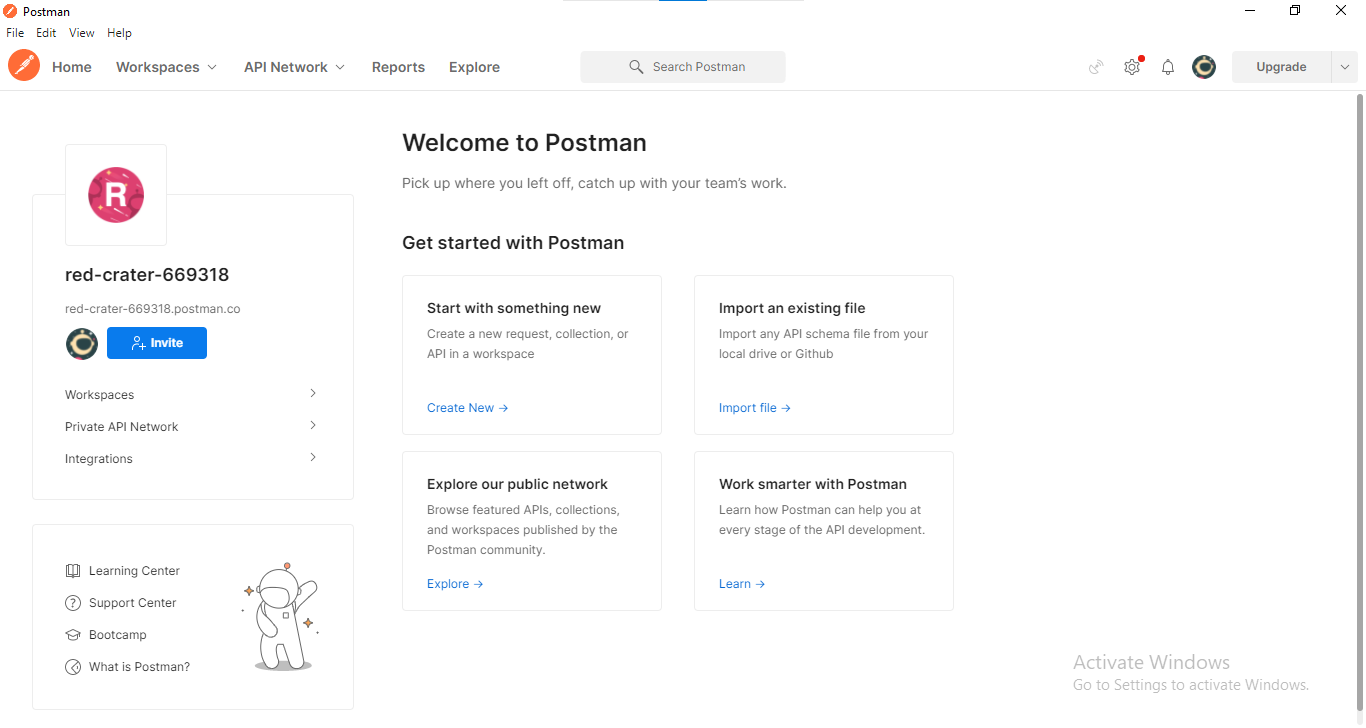
****

**Postman**

* Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.
* Download it from the following link:

<https://dl.pstmn.io/download/latest/win64>

* After downloading, it can be simply opened, just search postman on yours system and open it.



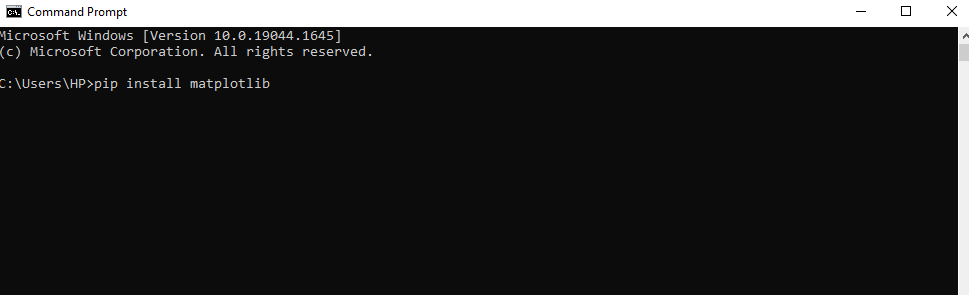
**Matplotlib**

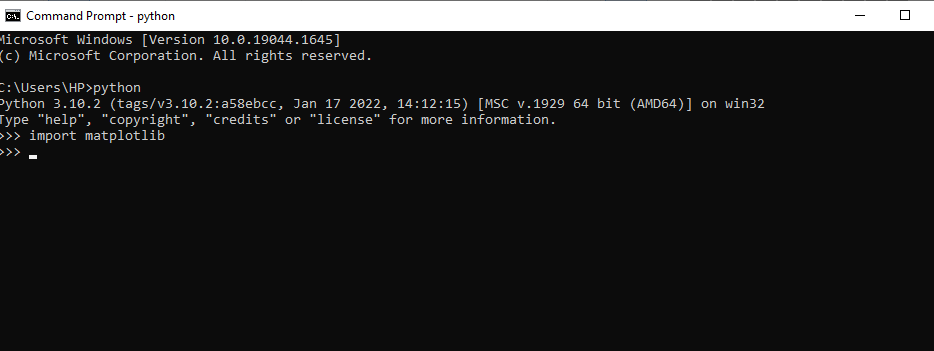
* Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.
* Install it using the command on cmd:

pip install matplotlib

* You can verify your **matplotlib** installation by executing the following command in cmd.

import matplotlib





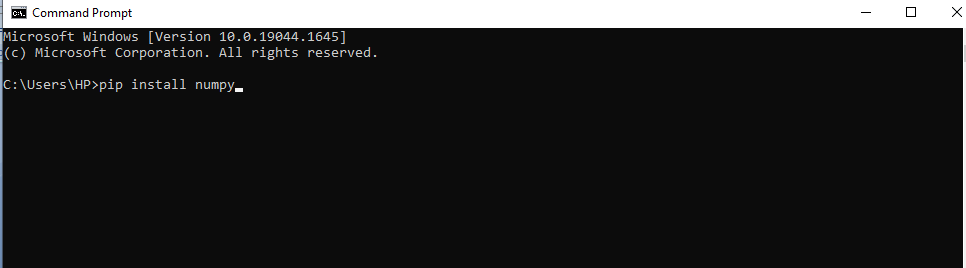
**Numpy**

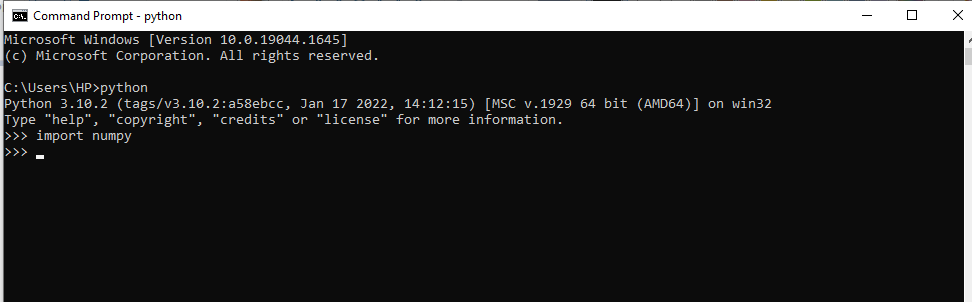
* NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.
* Install it using the command on cmd:

pip install matplotlib

* You can verify your **numpy** installation by executing the following command in cmd.

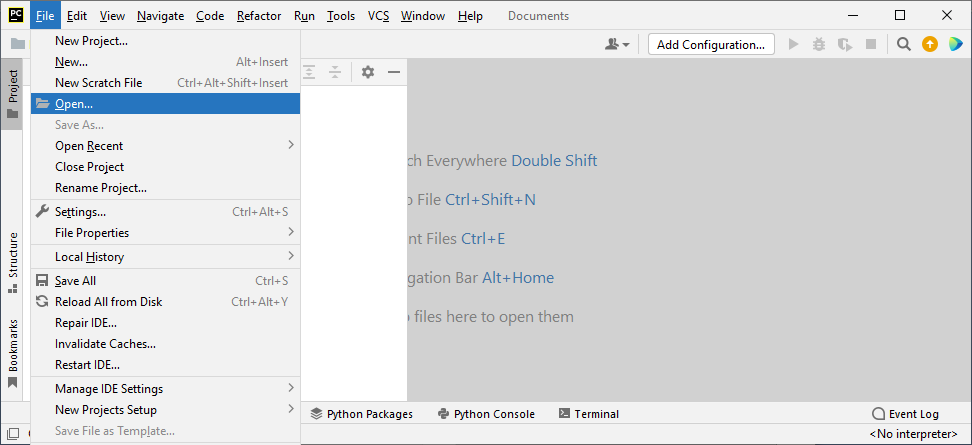
import numpy

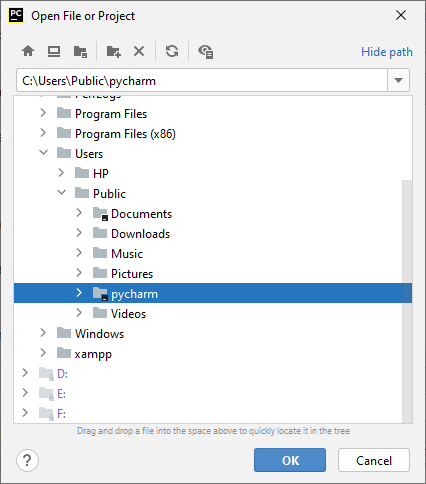
****

****

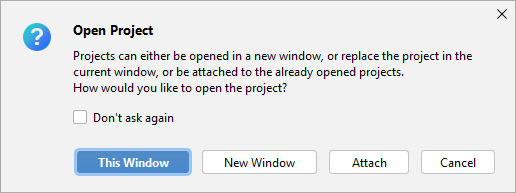
1. **Create Project**

* Open Pycharm.
* Select the Option and select Path where you want to create your project.



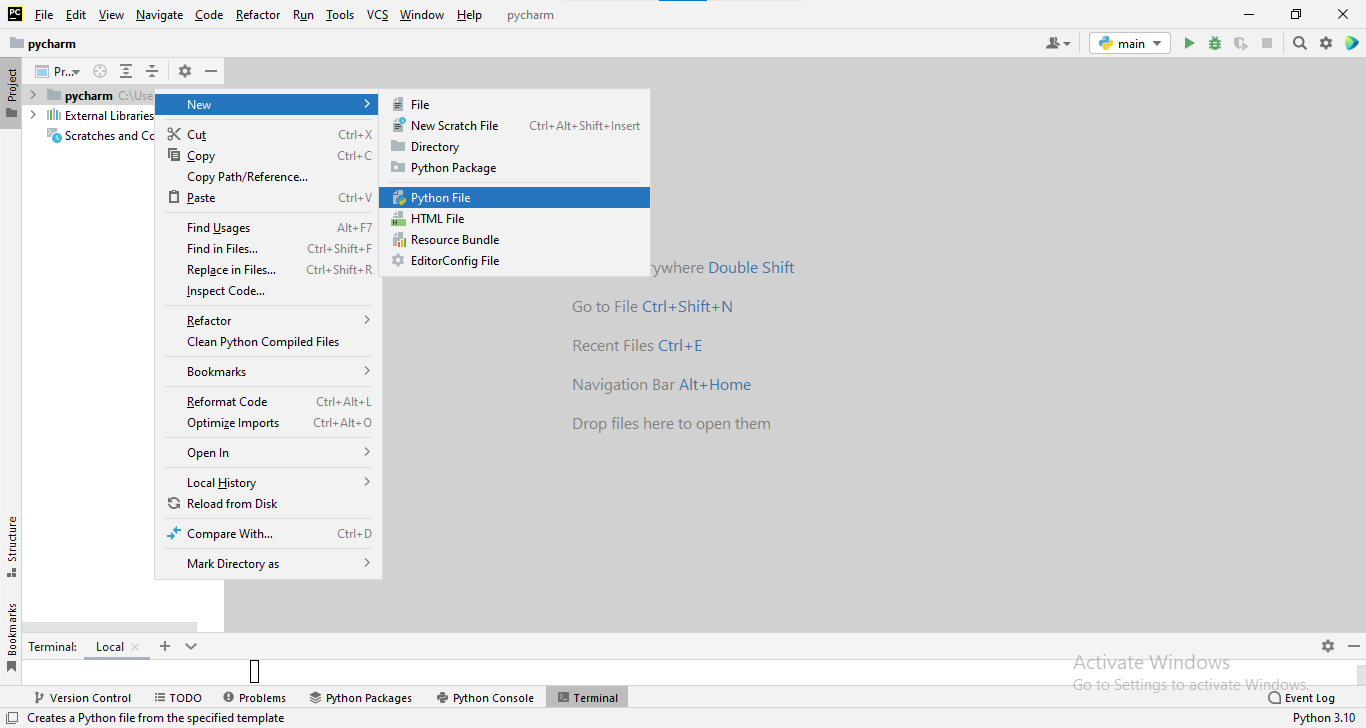


* Select option “This Window”.

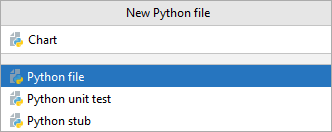
****

1. **Create Chart file**

* Click Right on Pycharm and Create a new python file as given below:

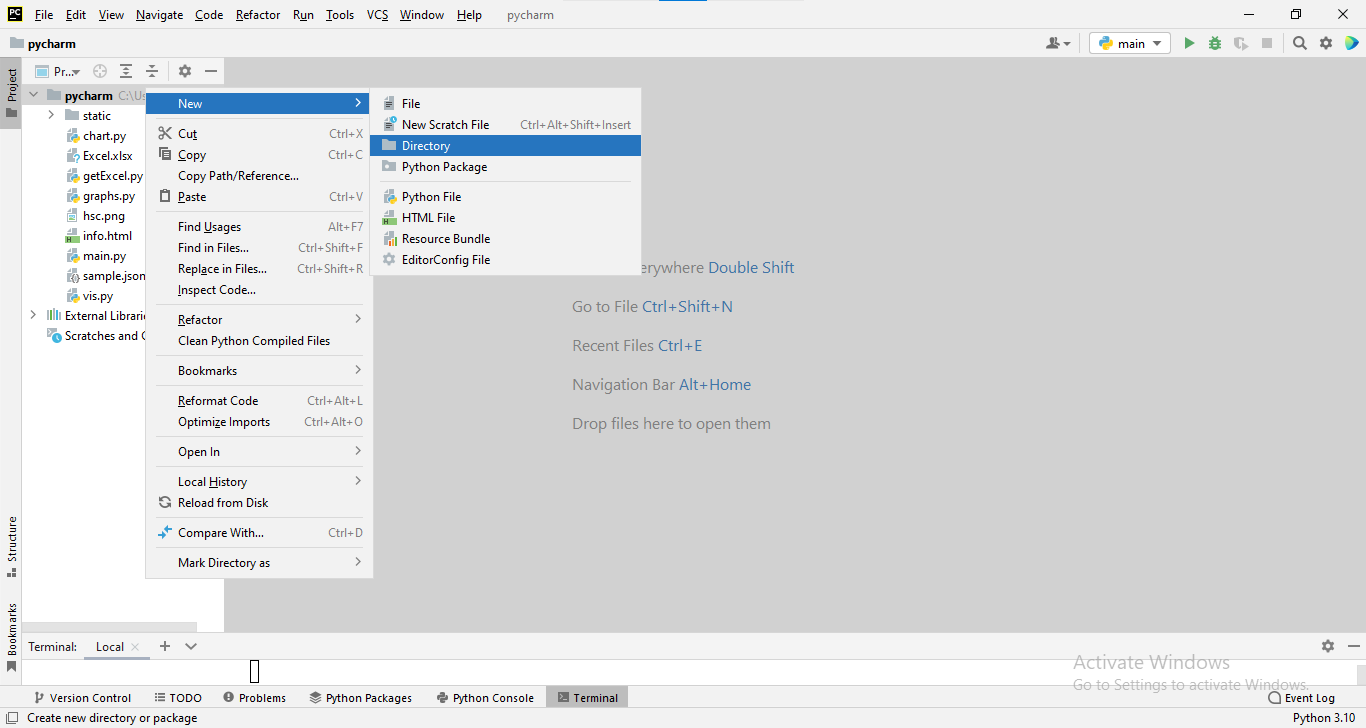
****

* Select Python File and Name it “Chart” and press Enter.



* Create “static” folder to store static files

Click right on Pycham->New->Directory->static.

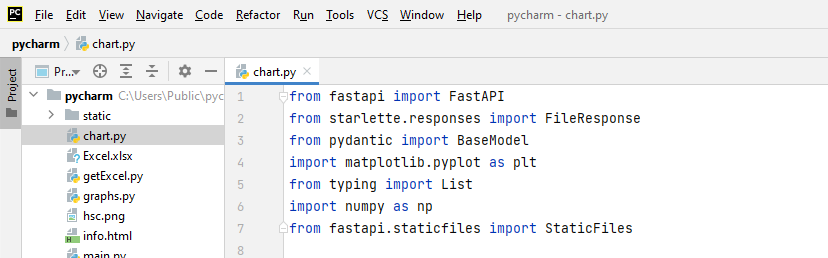


* Write “static” and press OK and then you will get the subdirectory static.

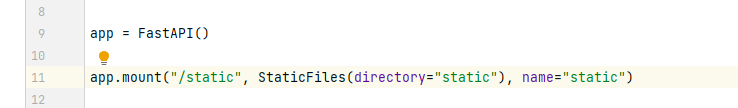


1. **Building API’s**

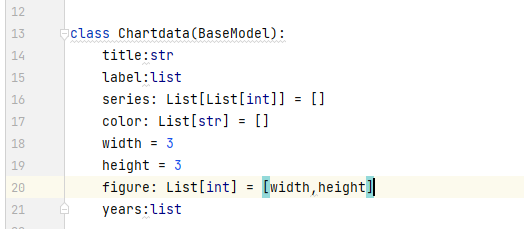
* Now open Chart.py and import all the required libraries as given below.



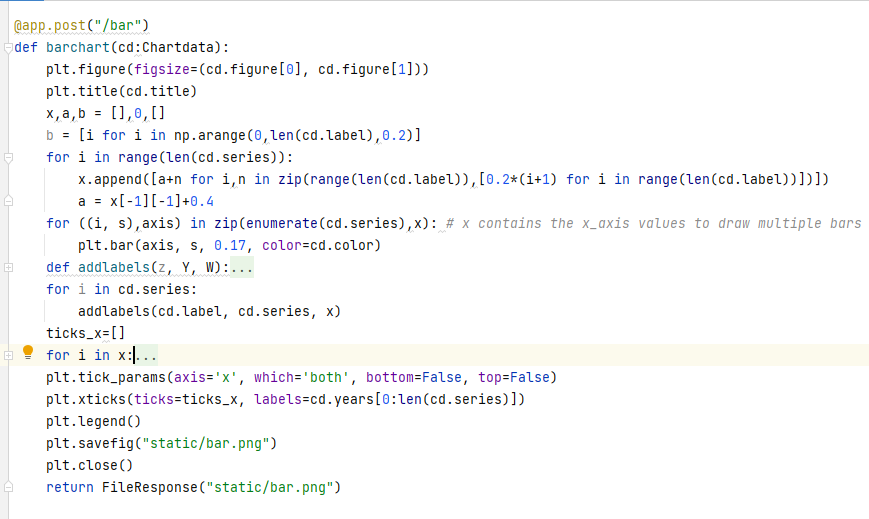
* Now we create the object of FASTAPI, this app object will be used for building API’s with different methods like get and post.
* We have also registered the static folder through app object.



* When you need to send data from a client (let's say, a browser) to your API, you send it as a **request body**.
* A **request** body is data sent by the client to your API. A **response** body is the data your API sends to the client.
* Your API almost always has to send a **response** body. But clients don't necessarily need to send **request** bodies all the time.
* To declare a **request** body, you use [Pydantic](https://pydantic-docs.helpmanual.io/" \t "_blank) models(**BaseModel**) with all their power and benefits.
* To declare variables with data type we made a data model(class) “Chartdata” and this data model as a class inherits from **BaseModel**.

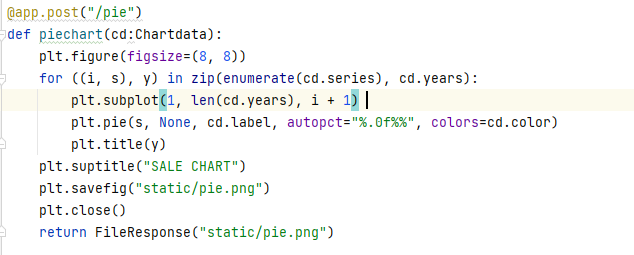


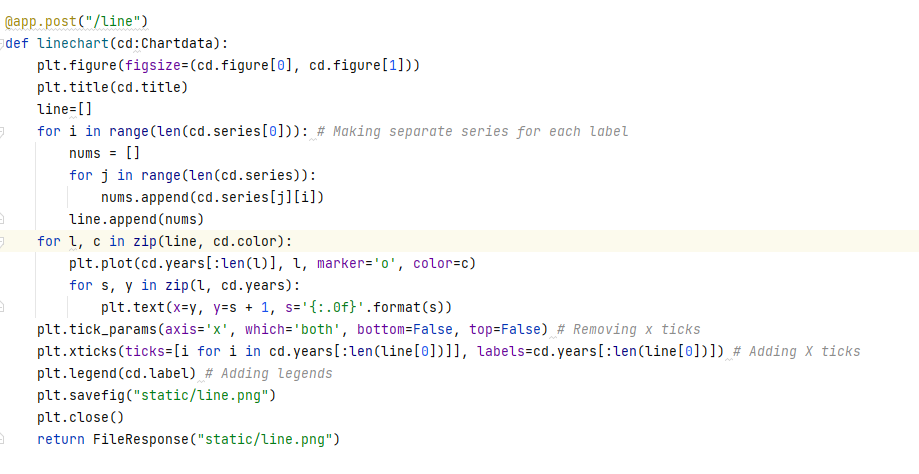
* With just that Python type declaration, **FastAPI** will:
  + Read the body of the request as JSON.
  + Convert the corresponding types (if needed).
  + Validate the data.
    - If the data is invalid, it will return a nice and clear error, indicating exactly where and what was the incorrect data.
  + Give you the received data in the parameter Chartdata.
* To send data, you should use one of: POST (the more common), PUT, DELETE or PATCH.We Used POST.
* Here we made a function barchart with parameter. This parameter is nothing but a class Chartdata with its object . This object has access of all attributes declared in Chartdata class.



* This above post request has it’s url “/bar”and it is binded with barchart function, whenever we call this url, this function will always run and return the response.

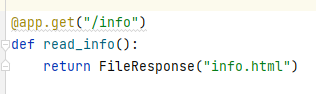
* Same as barchart, we have made piechart and linechart with the help of matplotlib library.



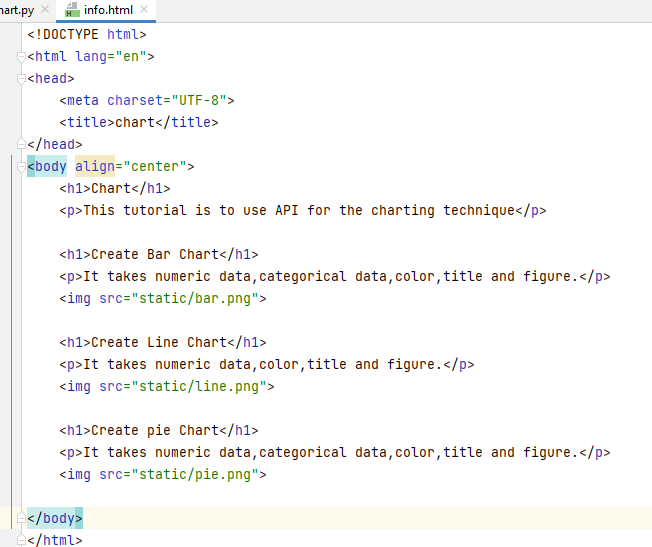


* There is one more api was made with get request “/info”.

This api is binded with read\_info function which returns html file as response.



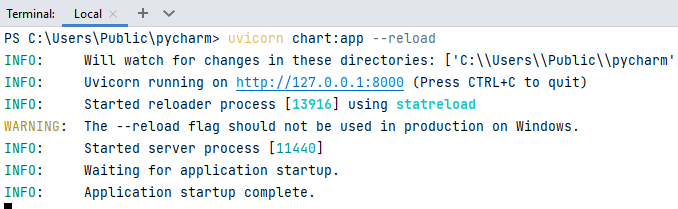
* Html file “info.html” which shows all the charts on the web page.



* Now we have completed our code Implementation part. FastApi includes ASGI server for production like Uvicorn or Hypercorn.
* We can start the server by typing the below command on terminal of Pycharm:

uvicorn Chart:app --reload

this reload option reloads the Chart.py file automatically when any change is made.

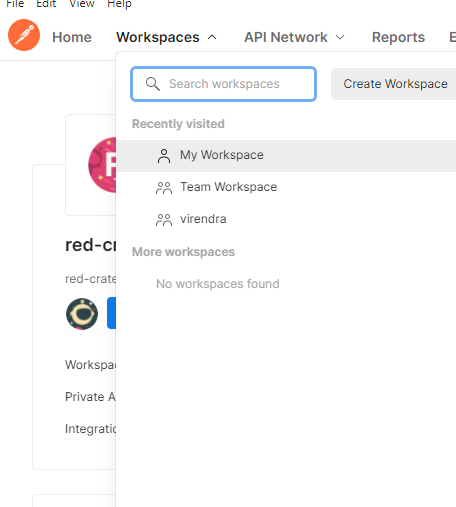


* Untill Server is on, we can use this API’s on production. Later we will use this API’s through Postman.
* Server can be stopped by pressing **Ctl+C** on the terminal.

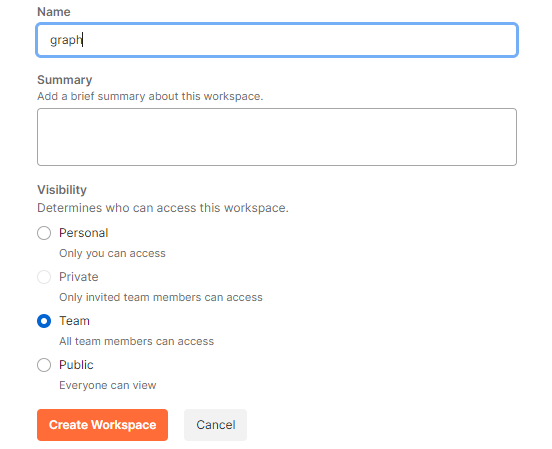
1. **Testing API’s**

* So far we have built all API’s in Chart.py, now we are ready to test all these API’s on postman.
* Firstly, open the postman, click on Workspaces and click on create workspace.

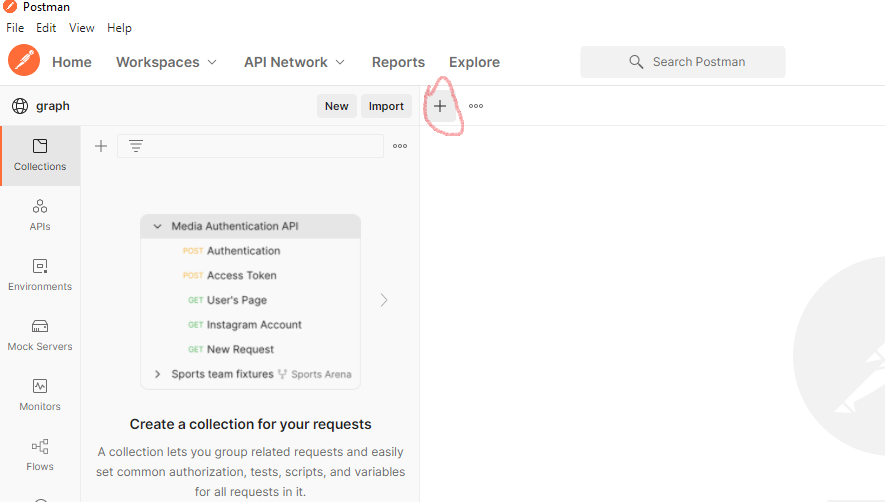
Workspaces->Create Workspace



* Now write the name of workspace(graph), summary and choose visibility option(Public preferred) and click on Create Workspace.



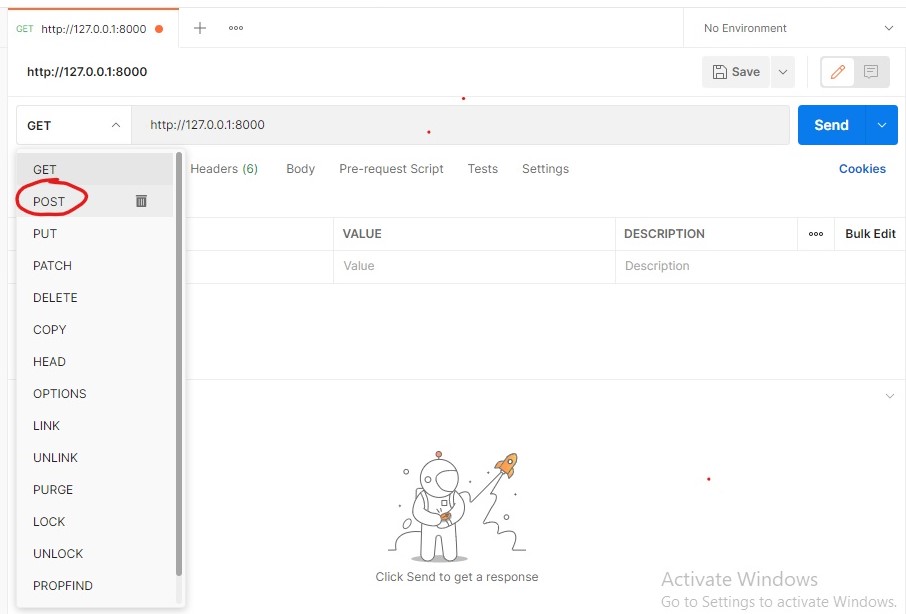
* Click on ‘+’ icon.



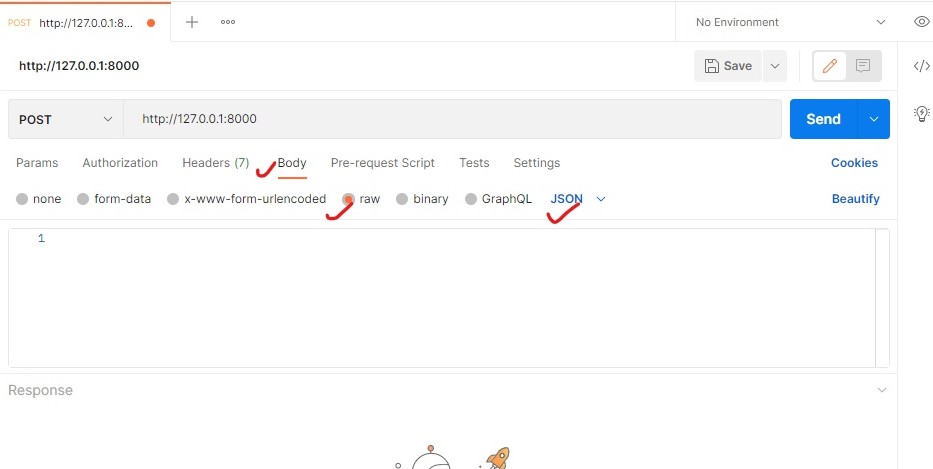
* Now copy the URL from the Pycharm terminal:

http://127.0.0.1:8000

* And Paste it with Post request as given below:



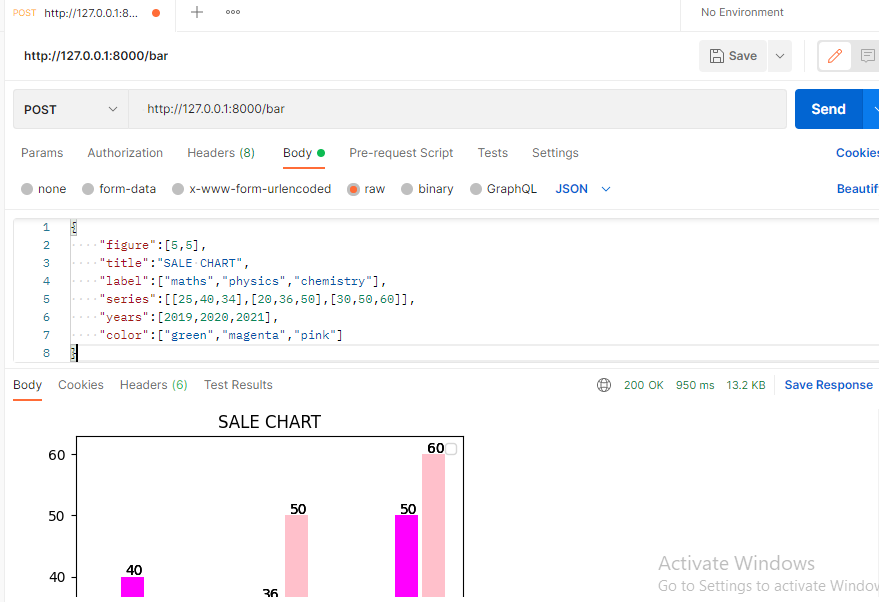
* Select Body-> raw-> JSON



* Here we are going to give all the data in JSON format. So we are giving all the attribute values of Chartdata class which we made in Chart.py file.
* Firstly, we will give the chart API’s with post request as given below:

<http://127.0.0.1:8000/bar>

* After giving all the values in JSON format , click on Send button and then you will get the response in Body as given below.



* Same as above url, we will send request with other API’s and get the respective charts with same data we used for bar chart.

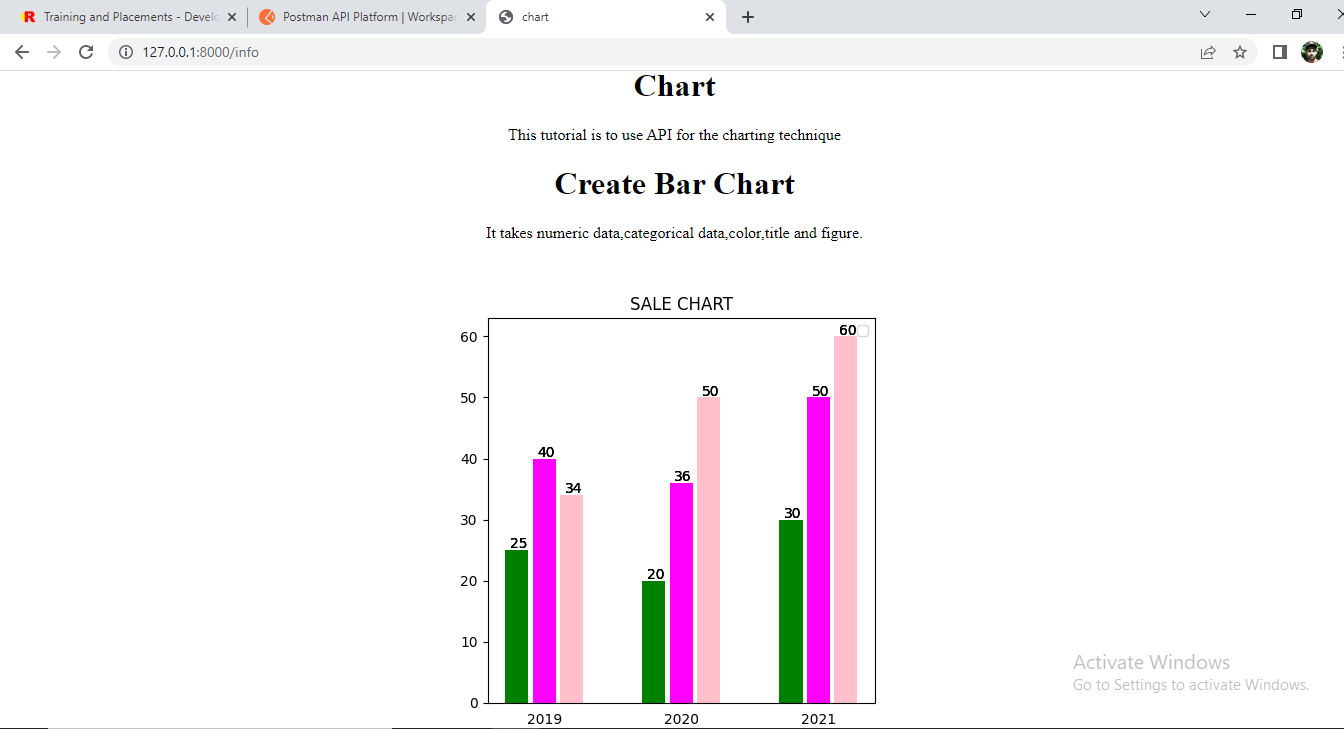
<http://127.0.0.1:8000/pie>

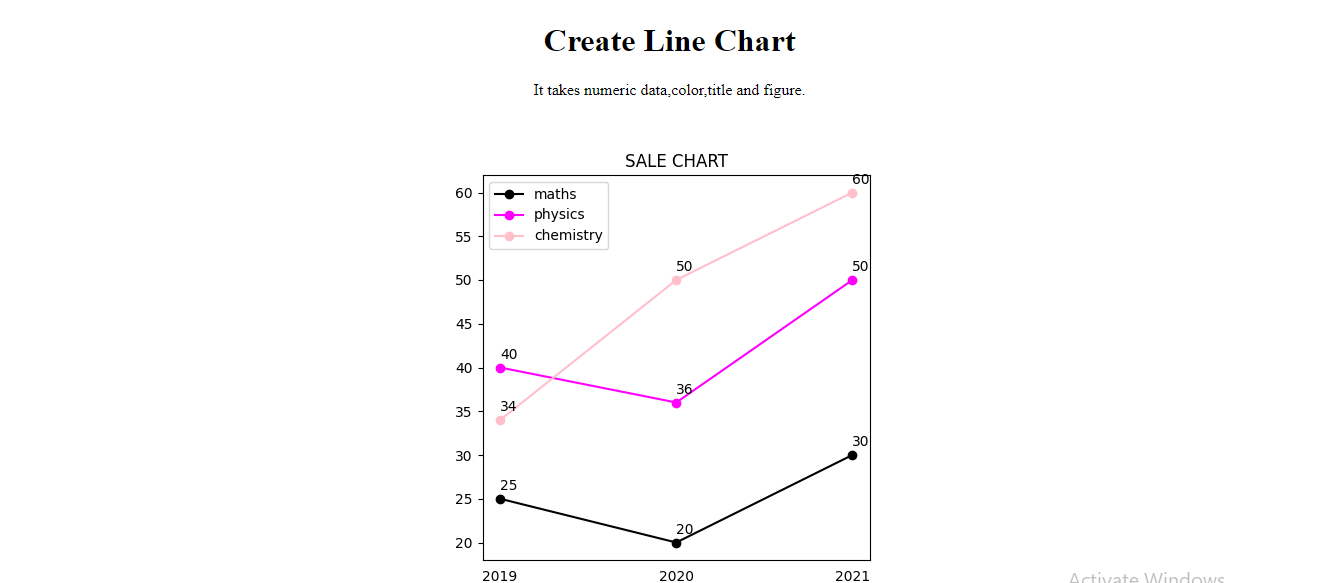
<http://127.0.0.1:8000/line>

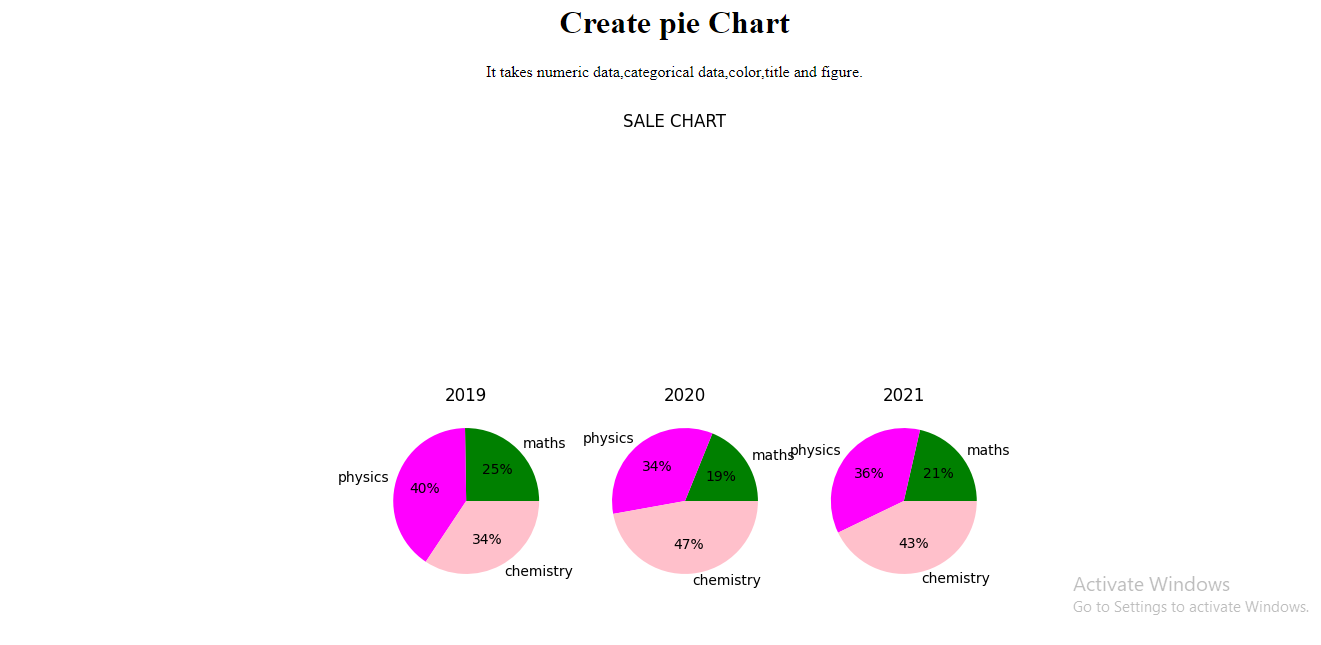
1. **Get Charts**

* To get all the charts over one place, we will use the API that is made with GET request:

<http://127.0.0.1:8000/info>







* NOTE: Always make sure your server is ON whenever you use these API’s.

-------------------------------------X----------------------------------------