

# Lesson Plan:

## flask -class-1



# Topics to be covered:

- Overview of What is a web framework?
- Overview of Flask
- Difference between Flask and Django
- Pros and Cons of Flask
- Steps to Create a Virtual Environment in Python
- Flask Installation
- Create a Basic Flask Application
- RESTful APIs

## What is a web framework?

A Web framework is a collection of packages or modules which allow developers to write Web applications (see WebApplications) or services without having to handle such low-level details as protocols, sockets, or process/thread management.

A Web framework or Web Application Framework helps to build Web applications. It provides tools and libraries to simplify common web development operations. It includes web services, APIs, and other resources.

A Web Framework can help you in many ways. As a Developer every time you build a new Web Application you have to focus very much on the logic and business part but if you have to write the basic framework that you use repeatedly then it can be a distraction for your main development and that's why researchers write those basic frameworks and libraries at a place so that every time you can reuse the code and templates.

The Usage of Frameworks:-

- No need to write code from scratch which can save you time.
- Inbuilt Security Features
- As it follows a certain pattern so using this framework can make testing and debugging easier
- You can write clean code
- These Frameworks include a lot of inbuilt tools which help in better development.

Python has many web frameworks. These frameworks fall into categories of **micro web frameworks** and **macro web frameworks**.

- **Django**, TurboGears, Web2Py, and Pyramid are some of the macro web frameworks of Python.
- **Flask**, CherryPy, and Bottle are some of the micro web frameworks.

## Overview of Flask:

Flask is a web framework written in Python. It is considered to be a microweb framework as it does not require tools or libraries and it depends on us what libraries we want to use.

Flask was created by Armin Ronacher as an April Fool's joke in 2010. Despite that, it gains huge popularity for being an alternative to Django projects with their monolithic structure and dependencies. Flask is based on the **Werkzeug**, **WSGI toolkit**, and the **Jinja2 template engine**. It is suitable for smaller and less complicated web applications.

- **WSGI:** WSGI (Web Server Gateway Interface) is a specification that describes the communication between web servers and Python web applications or frameworks.
- **Werkzeug:** Werkzeug is a WSGI toolkit that implements request, response objects, and utility functions. This is what enables a web frame to be built on it.
- **Jinja2:** Jinja2 is a Popular Python template engine used to create HTML, XML, or other markup formats that are returned to the user through an HTTP response.



## Difference between Flask and Django:

Flask follows Model-Views-Controller (MVC) structure, unlike Django which follows Model-View-Template (MVT) structure.

Flask is a microframework — a framework that doesn't come with tools or libraries. It's missing conveniences that a full framework has. For example, Flask doesn't come with data abstraction layers, meaning that if you have the need to store data long-term and you want to save the data in a database such as MySQL, SQLite, or PostgreSQL, you have to write a connector yourself and write pure SQL code to read or write to the database.

Django is an open-source macro web framework that uses Model-Template-View (MTV) pattern. An MTV pattern gives you a design pattern and a template for best practices that is convenient and easy to modify. Django gives ORM(Object Relational Mapper – It is a programming technique for converting data between type systems using object-oriented programming languages.) and it's up to you what kind of databases to choose (MySQL, PostgreSQL, Oracle, etc.). Django follows MTV pattern which makes it easier for beginners to transmit data to the server and navigate stored data than Flask.

## Pros and Cons of Flask:

### Advantages of Flask

1. **Easy to understand** – Even a novice programmer can understand how to work with the framework. Flask has a simple structure and intuitive syntax. Unlike other frameworks, Flask allows programmers to have complete control over the development processes.
2. **Flexible** – Only some parts of the Flask framework are inaccessible for modifications because they are already very much simple and optimized. Programmers can independently edit most of the framework tools to suit their needs. With Flask, the programmer gets a templating engine that allows the same user interface to be used across multiple pages.
3. **Good testing tools** – Flask has integrated testing and debugging tools. Developers have full-fledged unit tests, a built-in development server, a debugger, and a request handler at their disposal.
4. Flask make database integration and routing URL very easy.

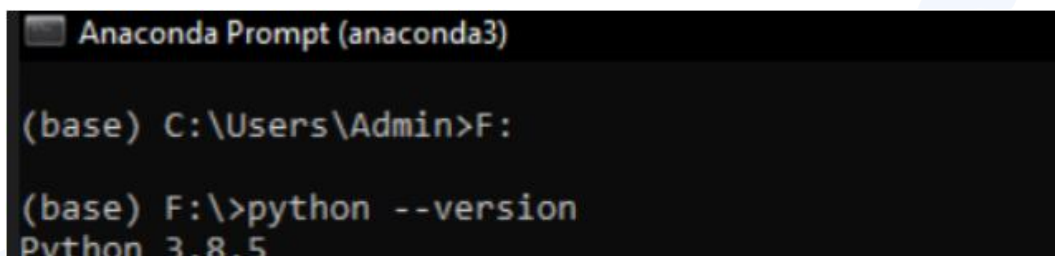
## Disadvantages of Flask

- 1. Doesn't support asynchrony** – Flask processes all requests in one thread and for that, each request blocks the thread for the duration of its execution and then passes the queue to the next request. If we compare it is like only one cash counter at a big bank, with a large number of customers(requests) so, the cash register will not cope, and a large queue will be created.
- 2. Lack of opportunities** – When it comes to developing large web applications, Flask lacks capabilities.

## Steps to Create a Virtual Environment in Python:

To create a Python virtual environment and use it, we use the module `venv`. Normally it installs the latest version of Python but you can also define a specific version.

To check your Python version on Windows you can open Anaconda prompt or command prompt and for Linux, you can open a terminal and write the command – `"python --version"`



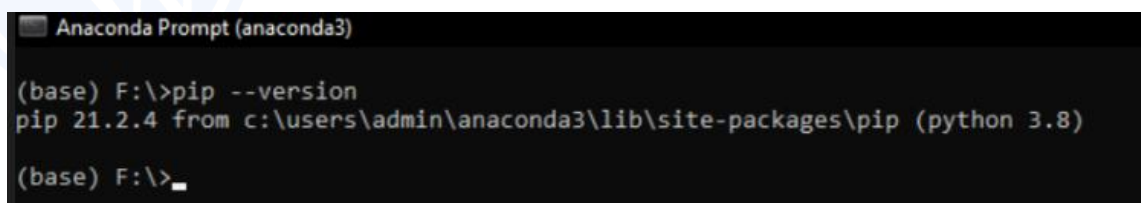
```
Anaconda Prompt (anaconda3)

(base) C:\Users\Admin>F:

(base) F:\>python --version
Python 3.8.5
```

Now we can also check if pip is installed or not

command = **"pip --version"**



```
Anaconda Prompt (anaconda3)

(base) F:\>pip --version
pip 21.2.4 from c:\users\admin\anaconda3\lib\site-packages\pip (python 3.8)

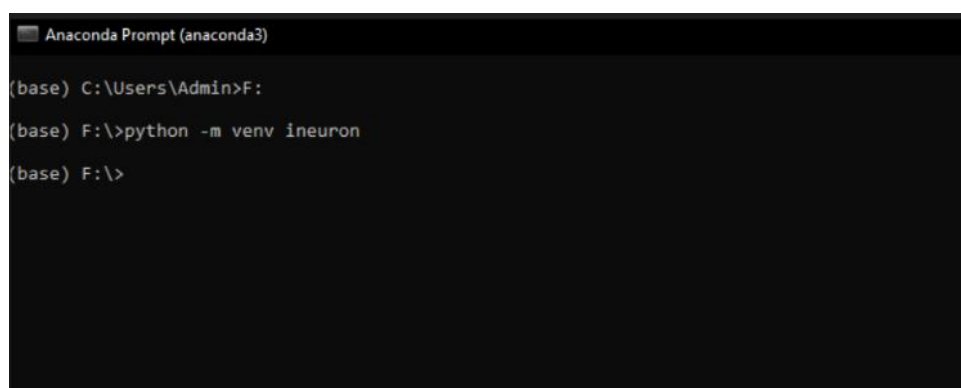
(base) F:\>_
```

Now that we have checked everything we can start creating our virtual environment in Python.

### 1. Create

To create a virtual environment first you have to decide a directory where you want to create it and went to that place. Now you can run the `venv` module as a script with the directory path:

Command – **"python -m venv env\_name"**



```
Anaconda Prompt (anaconda3)

(base) C:\Users\Admin>F:

(base) F:\>python -m venv ineuron

(base) F:\>
```

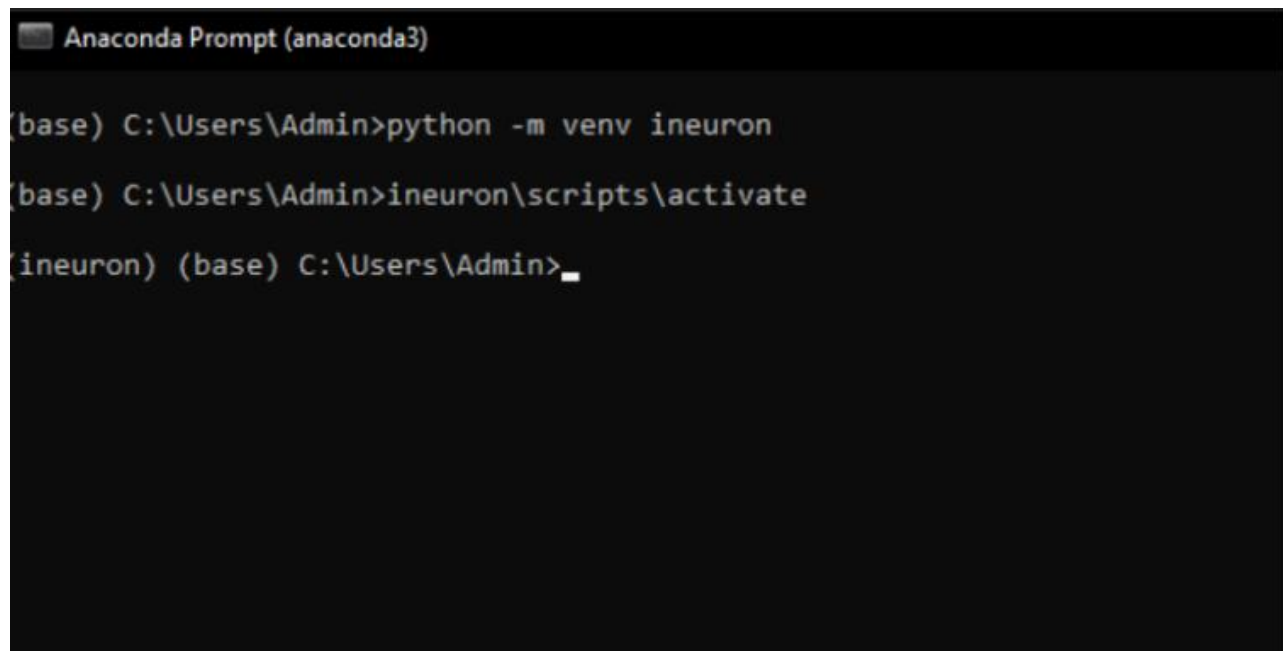
Here python -m flag stands for the module where some modules have main entry points and in order to run this it has to be used.

## 2. Activate

After creating before we start installing our packages in our virtual environment we need to activate the environment. Every time you work on the project you need to activate the relevant virtual environment.

Command for Windows:- **"env\_name\scripts\activate"**

Command for Mac or Unix:-**"sample\_venv/bin/activate"**



```
Anaconda Prompt (anaconda3)

(base) C:\Users\Admin>python -m venv ineuron

(base) C:\Users\Admin>ineuron\scripts\activate

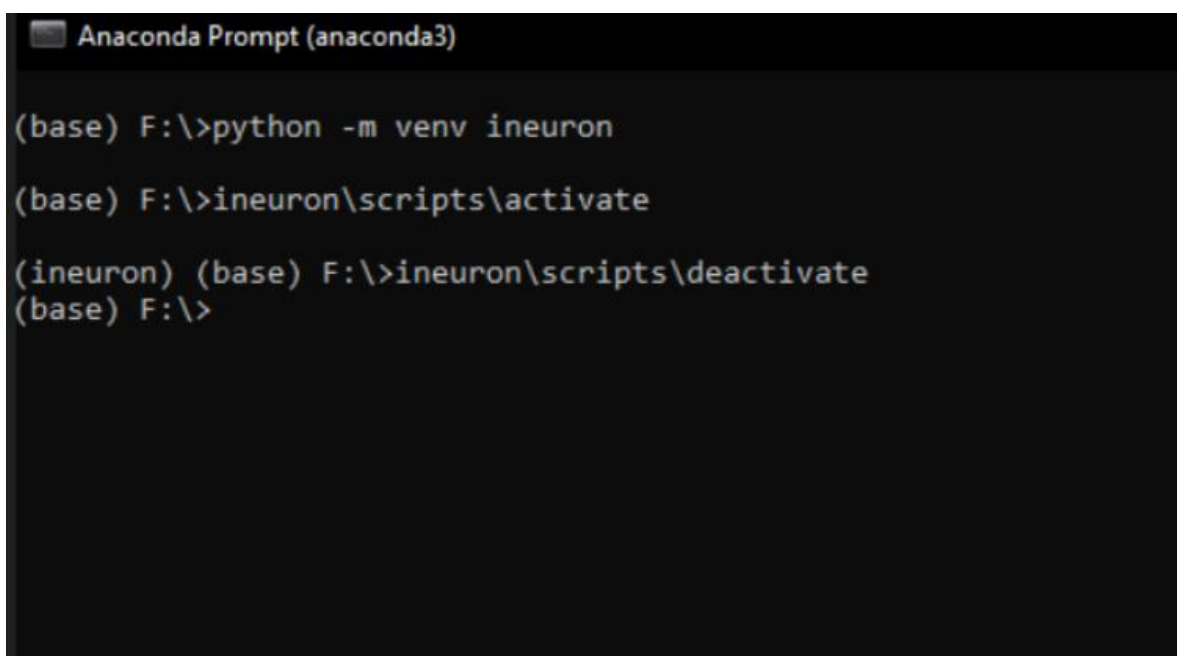
(ineuron) (base) C:\Users\Admin>_
```

## 3. Deactivate

Now that your virtual environment is created and activated you can install any packages of your choice.

You can also deactivate any environment of your choice.

Command :- **"env\_name\scripts\deactivate"**



```
Anaconda Prompt (anaconda3)

(base) F:\>python -m venv ineuron

(base) F:\>ineuron\scripts\activate

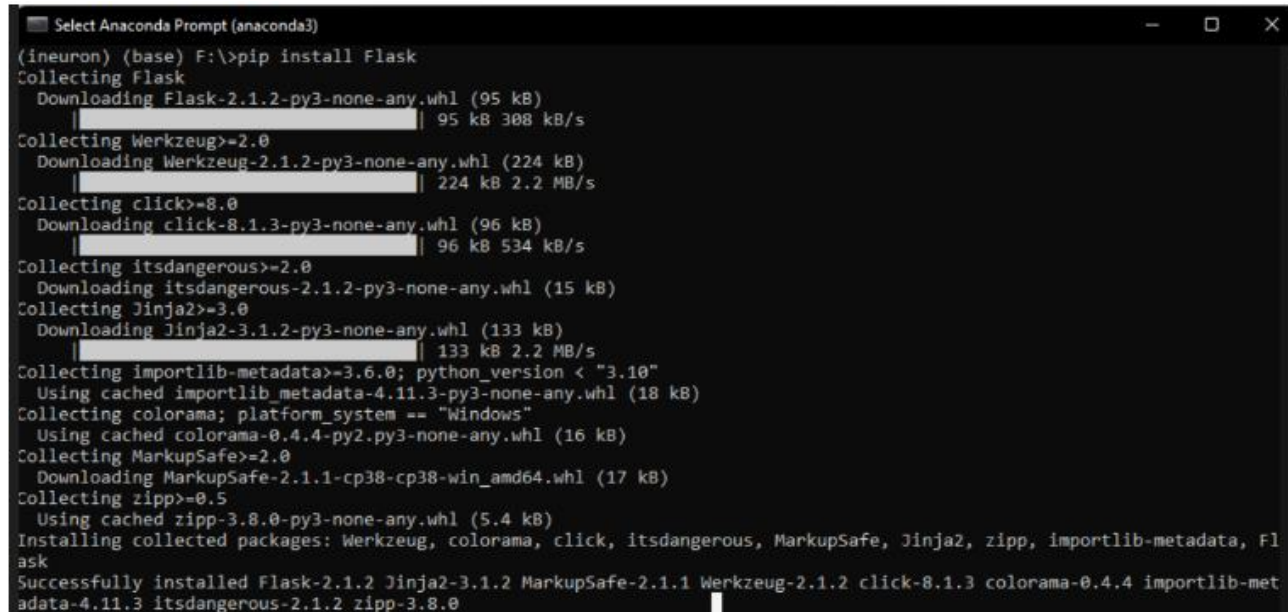
(ineuron) (base) F:\>ineuron\scripts\deactivate

(base) F:\>
```

# Flask Installation:

Now that your environment is created and activated you can install any libraries of your choice using the pip command.

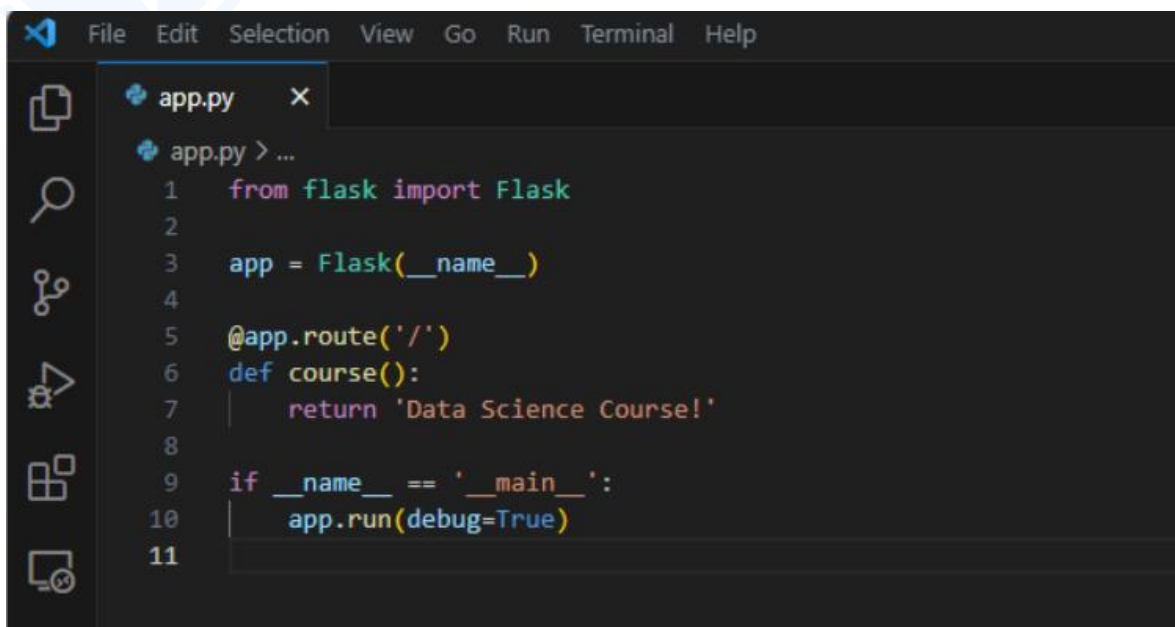
Command:- **"pip install Flask"**



```
Select Anaconda Prompt (anaconda3)
(ineuron) (base) F:\>pip install Flask
Collecting Flask
  Downloading Flask-2.1.2-py3-none-any.whl (95 kB)
    |#####| 95 kB 308 kB/s
Collecting Werkzeug>=2.0
  Downloading Werkzeug-2.1.2-py3-none-any.whl (224 kB)
    |#####| 224 kB 2.2 MB/s
Collecting click>=8.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    |#####| 96 kB 534 kB/s
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2>=3.0
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
    |#####| 133 kB 2.2 MB/s
Collecting importlib-metadata>=3.6.0; python_version < "3.10"
  Using cached importlib_metadata-4.11.3-py3-none-any.whl (18 kB)
Collecting colorama; platform_system == "Windows"
  Using cached colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.1-cp38-cp38-win_amd64.whl (17 kB)
Collecting zipp>=0.5
  Using cached zipp-3.8.0-py3-none-any.whl (5.4 kB)
Installing collected packages: Werkzeug, colorama, click, itsdangerous, MarkupSafe, Jinja2, zipp, importlib-metadata, Flask
Successfully installed Flask-2.1.2 Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.1.2 click-8.1.3 colorama-0.4.4 importlib-metadata-4.11.3 itsdangerous-2.1.2 zipp-3.8.0
```

# Create a Basic Flask Application:

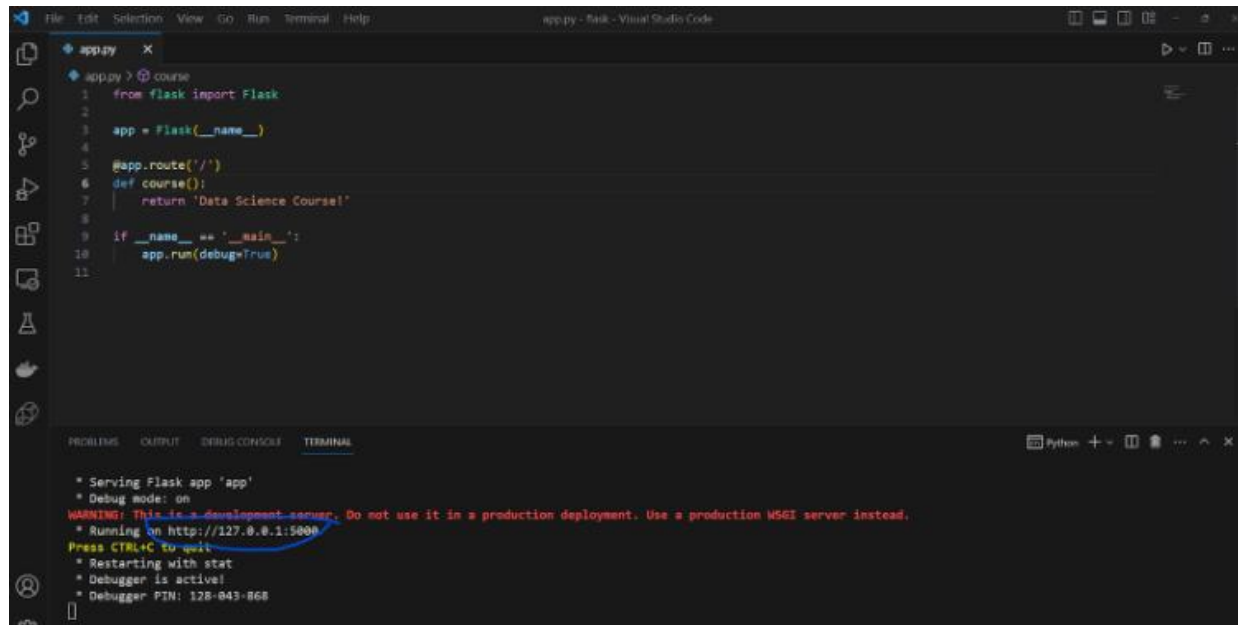
Let's create a minimal Flask application, we need to use the following code to develop the basic Flask application:



```
File Edit Selection View Go Run Terminal Help
app.py x
app.py > ...
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def course():
7      return 'Data Science Course!'
8
9  if __name__ == '__main__':
10     app.run(debug=True)
11
```

After running the app, we need to click on the URL <http://127.0.0.1:5000/> in the terminal.





```

app.py
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def course():
7     return 'Data Science Course!'
8
9 if __name__ == '__main__':
10     app.run(debug=True)
11

```

```

* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 128-043-868

```

Flask app Output:

