Flask Vs. Django

Flask	Django
Flask is a light-weight framework popularly categorized as a micro framework. Flask comes with some standard functionalities and allows developers to add any number of libraries or plugins for an extension. If you have a simple, innovative use case to be added to an existing application, Flask should be your choice as it offers flexibility. Flask comes with a small set of easy to learn API, and the documentation is excellent.	Django is a web application framework that takes care of many of the standard functionalities to build secure and maintainable websites. As a developer, all you need is to build your business logic. Django is free and open-source and has very active and helpful community support with loads of documentation.
If you want to dig more into coding and learn core concepts, Flask helps you understand how each component from the back-end works to get a simple web application up and running.	Django follows lots of design patterns, and hence you learn a lot of exciting concepts.
Light-weight framework with minimalistic features. Flask is single-threaded and may not perform too well under heavy load.	Full-stack web framework that follows the batteries-included approach. Django is a production-ready framework.
Flask doesn't have any such feature to handle administration tasks	Django comes with a ready-to-use admin framework that can be customized.
Flask's template engine Jinja2 is based on Django's template engine.	It comes with a built-in template engine that saves a lot of development time.
Each project can be a single application, however, multiple models and views can be added to the single application.	It allows users to divide a single project into multiple small applications which makes them easy to develop and maintain.
Admin features are not as prominent as in Django.	The Django-admin tool is a built-in bootstrapping tool with which developers can build web applications without any external input.
Companies that use Flask are Netflix, Lyft, Reddit, Zillow, MailGui	The companies that use Django are Instagram, Pinterest, Udemy, Coursera, Zapier
Flask has built-in security against the number of common threats such as CSRF, XSS, and SQL injection.	Django is more secure in comparison with other web frameworks. It consists of a much smaller codebase, so there is less possibility of getting attacked by an unauthorized person. To make it more secure, it is needed to evaluate and monitor third-party libraries and extensions.

Do you want to know which framework is better to use for web development, Flask vs Django? Many Python-based web frameworks enable developers to build scalable applications quickly. From simple to complex websites, these frameworks can do it all. Out of the many popular choices, Django vs Flask are the most talked about – and mostly because both are similar in some ways and different in many other ways!

Try out Flask program:

Command: pip install flask

hello.flask.py

```
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def hello():
   name = request.args.get("name", "World")
   return f'Hello, {escape(name)}!'
```

Then start the Flask server from the command line:

```
$ env FLASK_APP=hello_flask.py flask run
* Serving Flask app "hello_flask.py"
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0. 1:5000/ (Press CTRL+C to quit)
```

Click on the above link, and it will print Hello World on a webpage.

Try out Django program:

pip install django command

```
from django.conf import settings
from django.core.handlers.wsgi import WSGIHandler
from django.core.management import execute_from_command_line
from django.http import HttpResponse
from django.urls import path

settings.configure(
    ROOT_URLCONF=__name__,
    DEBUG=True,
)

def hello_world(request):
    return HttpResponse("Hello, Django!")

urlpatterns = [
    path('', hello_world)
]

application = WSGIHandler()

if __name__ == "__main__":
    execute_from_command_line()
```

Type following command in your terminal:

```
python hello_django.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 17, 2019 - 13:48:54
Django version 3.0, using settings None
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

When you click on the above link, it will display the Hello World on a webpage.