Flask

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.

Flask

Flask is a web application framework written in Python. It is developed by **Armin Ronacher**, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine.

[Flask](http://flask.pocoo.org/) is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are:

* [Werkzeug](http://werkzeug.pocoo.org/) a WSGI utility library
* [jinja2](http://jinja.pocoo.org/) which is its template engine

Template Engine

If your website contains only few pages, changing its style will take you some time but is doable. However, if you have a lot of pages (for example the list of items you sell in your store), this task become overwhelming.

Using templates you are able to set a basic layout for your pages and mention which element will change. This way you can define your header once and keep it consistent over all the pages of your website, and if you need to change your header, you will only have to update it in one place.

Using a template engine will save you a lot of time when creating your application but also when updating and maintaining it.

Jinja2

Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Some basic syntax of jinja are:

* {% ... %} for [Statements](http://jinja.pocoo.org/docs/2.10/templates/#list-of-control-structures)
* {{ ... }} for [Expressions](http://jinja.pocoo.org/docs/2.10/templates/#expressions) to print to the template output
* {# ... #} for [Comments](http://jinja.pocoo.org/docs/2.10/templates/#comments) not included in the template output

WSGI

Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

Werkzeug

It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

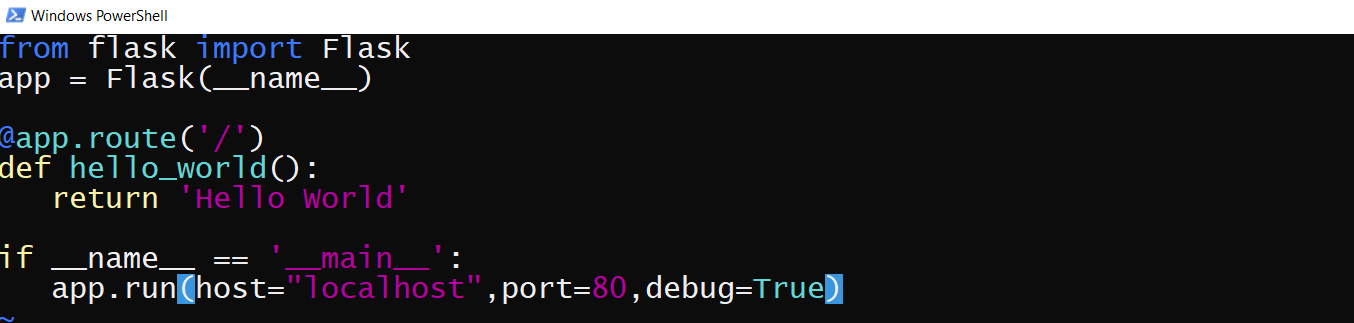
Flask Application

* The module used for flask is “flask”

First install the “flask” module using the following command:

Pip install flask

* Make a folder in which you are going to make your flask application then create two following folders inside your main project:
  + Static
  + Templates
* create a file name app.py and write the following code into:



* The above code shows "Hello, World!" on localhost port 80 in a web browser when run with the python app.py command
* Importing flask module in the project is mandatory. An object of Flask class is our **WSGI** application.
* Flask constructor takes the name of **current module (\_\_name\_\_)** as argument.
* The **route()** function of the Flask class is a decorator, which tells the application which URL should call the associated function.

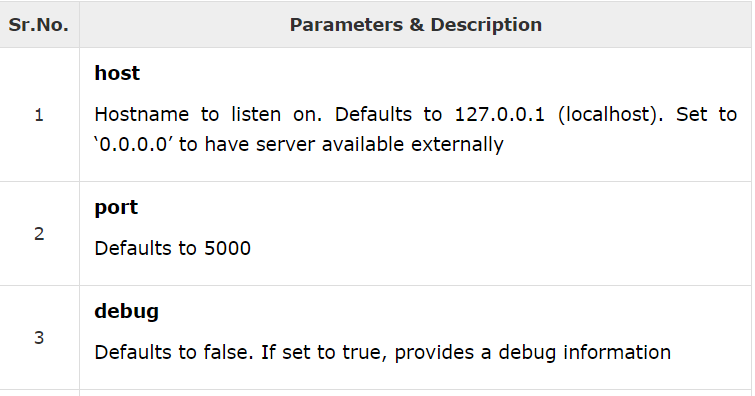
App.route(rule,options)

* + The **rule** parameter represents URL binding with the function.
  + The **options** is a list of parameters to be forwarded to the underlying Rule object.

In the above example, **‘/’** URL is bound with **hello\_world()** function. Hence, when the home page of web server is opened in browser, the output of this function will be rendered.

* the **run()** method of Flask class runs the application on the local development server.

App.run(host,port,debug)



Debug mode

* A **Flask** application is started by calling the **run()** method. However, while the application is under development, it should be restarted manually for each change in the code. To avoid this inconvenience, enable **debug support**. The server will then reload itself if the code changes. It will also provide a useful debugger to track the errors if any, in the application.
* The **Debug** mode is enabled by setting the **debug** property of the **application** object to **True** before running or passing the debug parameter to the **run()** method.