**PYTHON FLASK**

**Introduction**

Flask is a framework. Framework is a library or collection of libraries that aims to solve a part of a generic problem instead of a complete specific one. When building web applications, there are some problems that will always need to be solved, such as routing from URLs to resources, inserting dynamic data into HTML, and interacting with an end user.

Flask is a light weight web development micro framework in Python. Flask is a micro framework because it implements only core functionality of web developing. Flask has no native support for accessing databases, validating web forms, authenticating users, or other high-level tasks. These and many other key services most web applications need are available through extensions that integrate with the corepackages. As you know, Django is a full framework of python for web developing.

**Installation**

Before the installation of Flask install python package manager PIP. Before you install PIP, check if PIP is already installed using the following command.

**pip help**

**In ubuntu** use the following commands for the installation of pip

**sudo apt-get update**

**sudo apt-get install python-pip**

**PIP installation in Windows**

Download **get-pip.py** to a folder from <https://bootstrap.pypa.io/>Change the directory to the folder

where the get-pip.py is. Give the following command to install PIP

**python get-pip.py**

Installation of PIP over windows is over.

After the installation of PIP

use the following commanf to install Flask framework.

**pip install flask**

When you execute this command, pip will not only install Flask, but also all of its dependencies.

You can also verify that Flask was correctly installed by starting the Python inter‐ preter and trying

to import it:

**python**

**>>> import flask**

**>>>**

If no errors appear, you are successfully installed python Flask.

**Flask Application Structure**

At first we have to create an application instance. As you know we need a web server for running a web application. The web server passes all requests it receives from clients to this object for handling, using a protocol called Web Server Gateway Interface (WSGI, pronounced “wiz-ghee”).

The application instance is an object of class Flask, usually created as follows:

**from flask import Flask**

**app = Flask(\_\_name\_\_)**

The only required argument to the Flask class constructor is the name of the main module or package of the application. For most cases, \_\_name\_\_ variable is the correct value for this argument.

As I previously mentioned, web browsers send request to the web server and the web server sends these requests to the Flask application instance. As per the received request the application instance executes the appropriate code. For this purpose it keeps a mapping of URLs to Python functions.

The mapping between a URL and a python function is called a route.

The following decorator is used to define a route in Flask application.

**app.route**

The following example shows how a route is declared using this decorator:

**@app.route('/')**

**def index():**

**return '<h1>Hello World!</h1>'**

A complete flask application with with a single route is given below:

**from flask import Flask**

**app = Flask(\_\_name\_\_)**

**@app.route('/')**

**def index():**

**return '<h1>My First Flask Program!</h1>'**

Save this application as sample.py

A development webserver is included with the flask installation. We can use this webserver for testing our appication. To run this webserver use the following command.

**In linux:**

**export FLASK\_APP=sample.py**

**flask run**

**In windows:**

**set FLASK\_APP=sample.py**

**flask run**

After the execution, open your browser and type **http://localhost:5000** in the address bar of the browser. You will get the output.