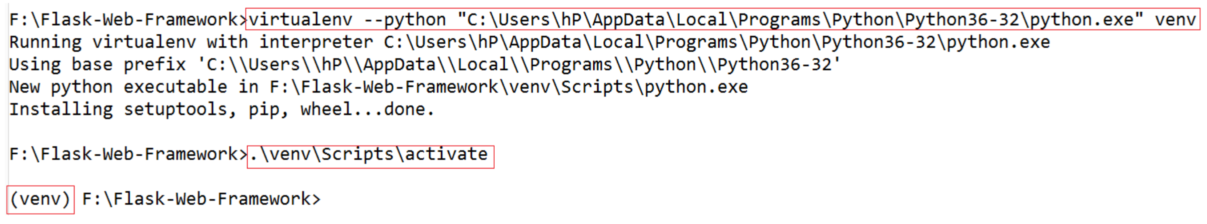
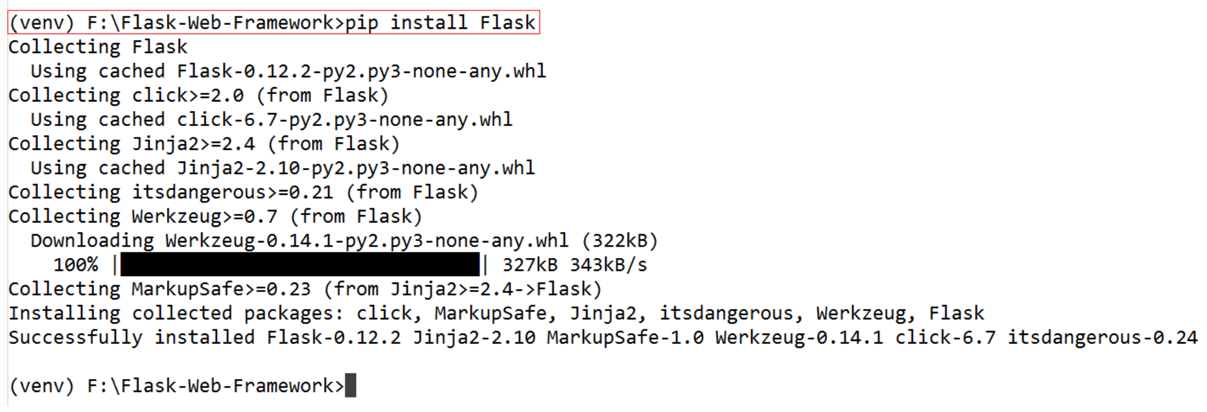
1. Create a directory to have all the web applications with respect to Flask at one place
2. Create a virtual environment and activate the same



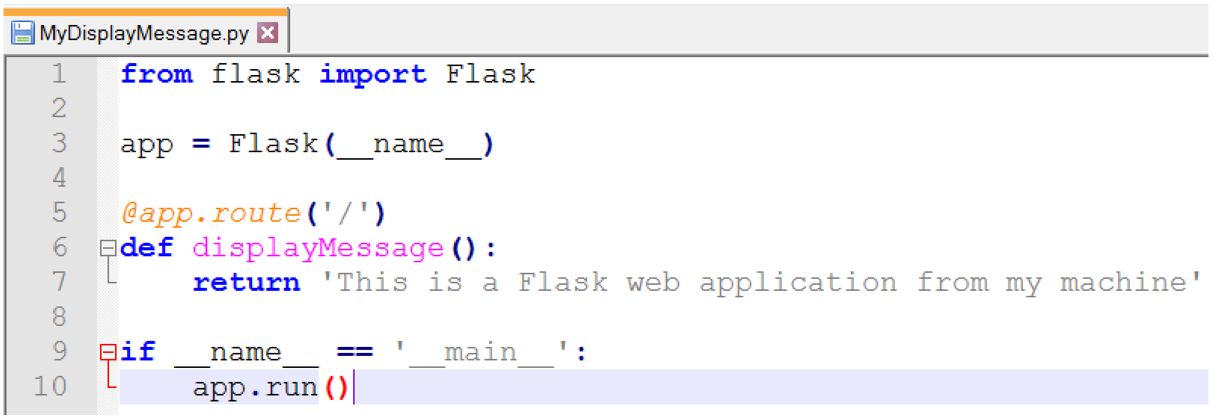
```
F:\Flask-Web-Framework>virtualenv --python "C:\Users\hP\AppData\Local\Programs\Python\Python36-32\python.exe" venv
Running virtualenv with interpreter C:\Users\hP\AppData\Local\Programs\Python\Python36-32\python.exe
Using base prefix 'C:\\Users\\hP\\AppData\\Local\\Programs\\Python\\Python36-32'
New python executable in F:\Flask-Web-Framework\venv\Scripts\python.exe
Installing setuptools, pip, wheel...done.

F:\Flask-Web-Framework>.\venv\Scripts\activate

(venv) F:\Flask-Web-Framework>
```

3. After creating the virtual environment, install the Flask as shown below:



```
(venv) F:\Flask-Web-Framework>pip install Flask
Collecting Flask
  Using cached Flask-0.12.2-py2.py3-none-any.whl
Collecting click>=2.0 (from Flask)
  Using cached click-6.7-py2.py3-none-any.whl
Collecting Jinja2>=2.4 (from Flask)
  Using cached Jinja2-2.10-py2.py3-none-any.whl
Collecting itsdangerous>=0.21 (from Flask)
Collecting Werkzeug>=0.7 (from Flask)
  Downloading Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
    100% |████████████████████████████████| 327kB 343kB/s
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->Flask)
Installing collected packages: click, MarkupSafe, Jinja2, itsdangerous, Werkzeug, Flask
Successfully installed Flask-0.12.2 Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24

(venv) F:\Flask-Web-Framework>
```

4. Create a sample python program to which returns a message to be displayed. I have created MyDisplayMessage.py as shown below:

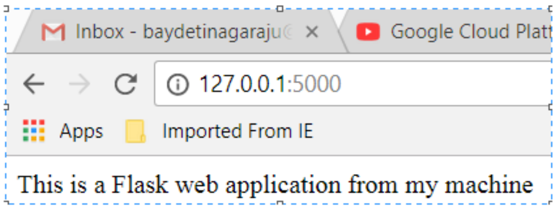


```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def displayMessage():
    return 'This is a Flask web application from my machine'

if __name__ == '__main__':
    app.run()
```

5. Execute the MyDisplayMessage.py

   By default, the web application runs on localhost and port 5000

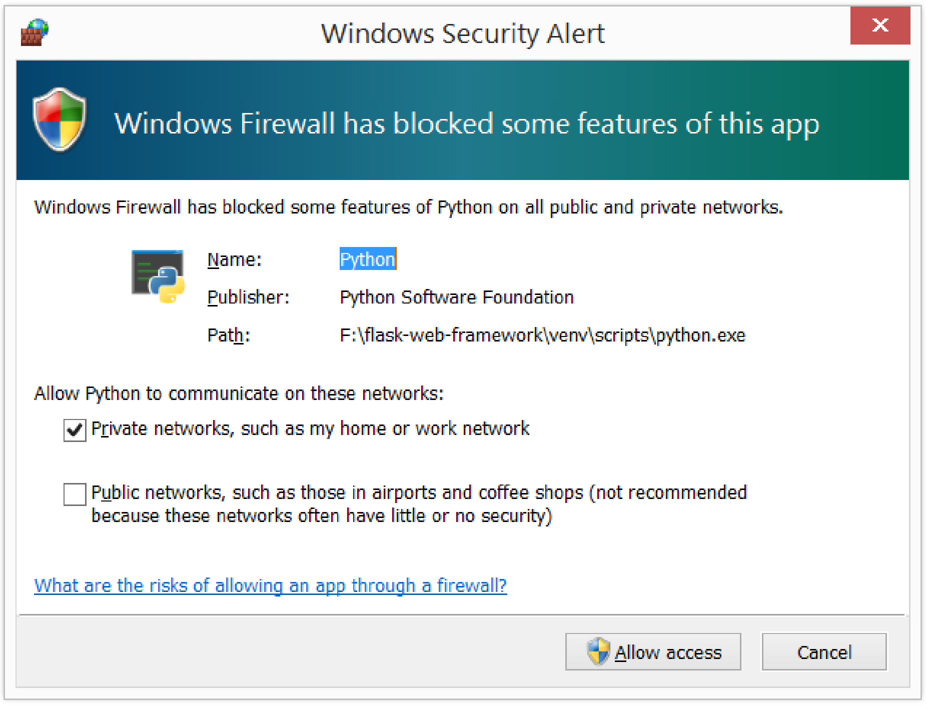
```
(venv) F:\Flask-Web-Framework>py MyDisplayMessage.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [03/Feb/2018 08:06:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Feb/2018 08:06:37] "GET /favicon.ico HTTP/1.1" 404 -
```



This is a Flask web application from my machine

6. The application can be run on a specific host and port number by specifying them in the app.run() method as shown below. If there is any firewall security alert, you can allow the access for the smooth running of the above said web application

```
(venv) F:\Flask-Web-Framework>py MyDisplayMessage.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
```


Windows Security Alert

Windows Firewall has blocked some features of this app

Windows Firewall has blocked some features of Python on all public and private networks.

Name: Python
Publisher: Python Software Foundation
Path: F:\flask-web-framework\venv\scripts\python.exe

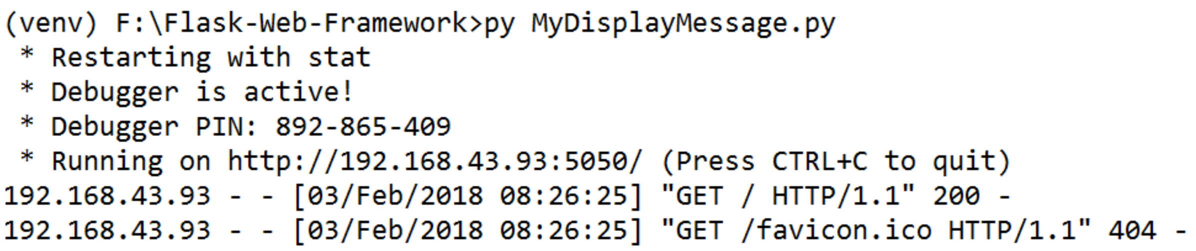
Allow Python to communicate on these networks:

[✓] Private networks, such as my home or work network

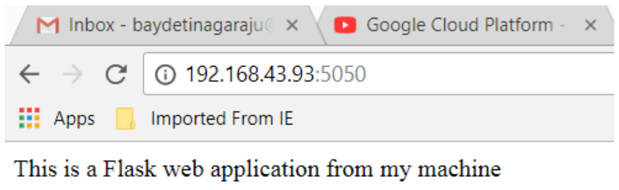
[ ] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

The application will be hosted on specific IP Address and the port number as shown below:

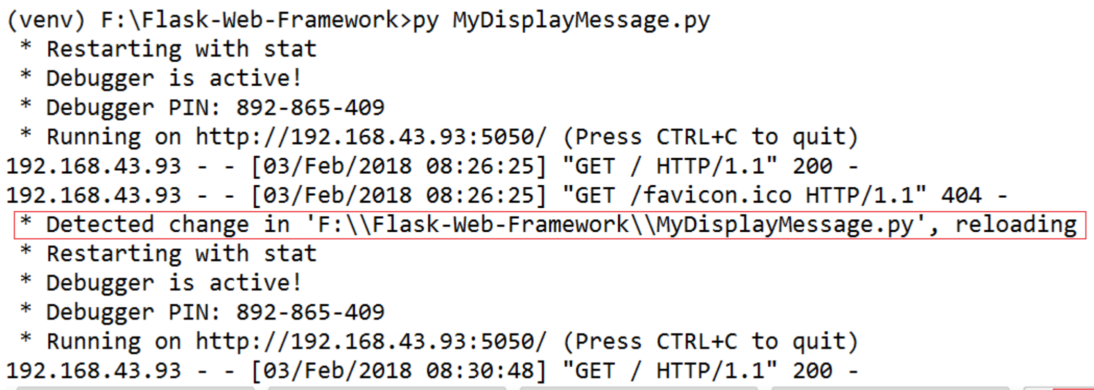


```
(venv) F:\Flask-Web-Framework>py MyDisplayMessage.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
192.168.43.93 - - [03/Feb/2018 08:26:25] "GET / HTTP/1.1" 200 -
192.168.43.93 - - [03/Feb/2018 08:26:25] "GET /favicon.ico HTTP/1.1" 404 -
```

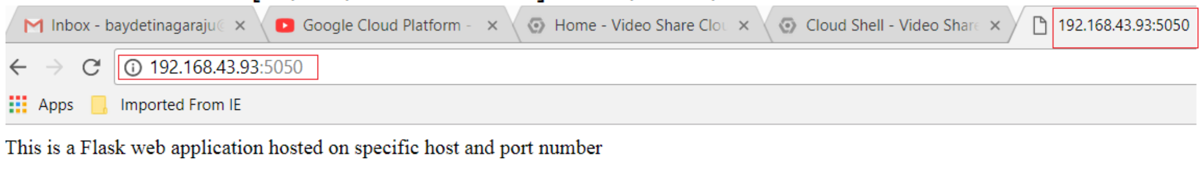


This is a Flask web application from my machine

As the debug option is set to TRUE, any changes in the code will automatically get reflected in your web application. For testing purpose, the message which is being displayed is changed and we can observe that the reloading happens automatically while the application is still under execution phase.

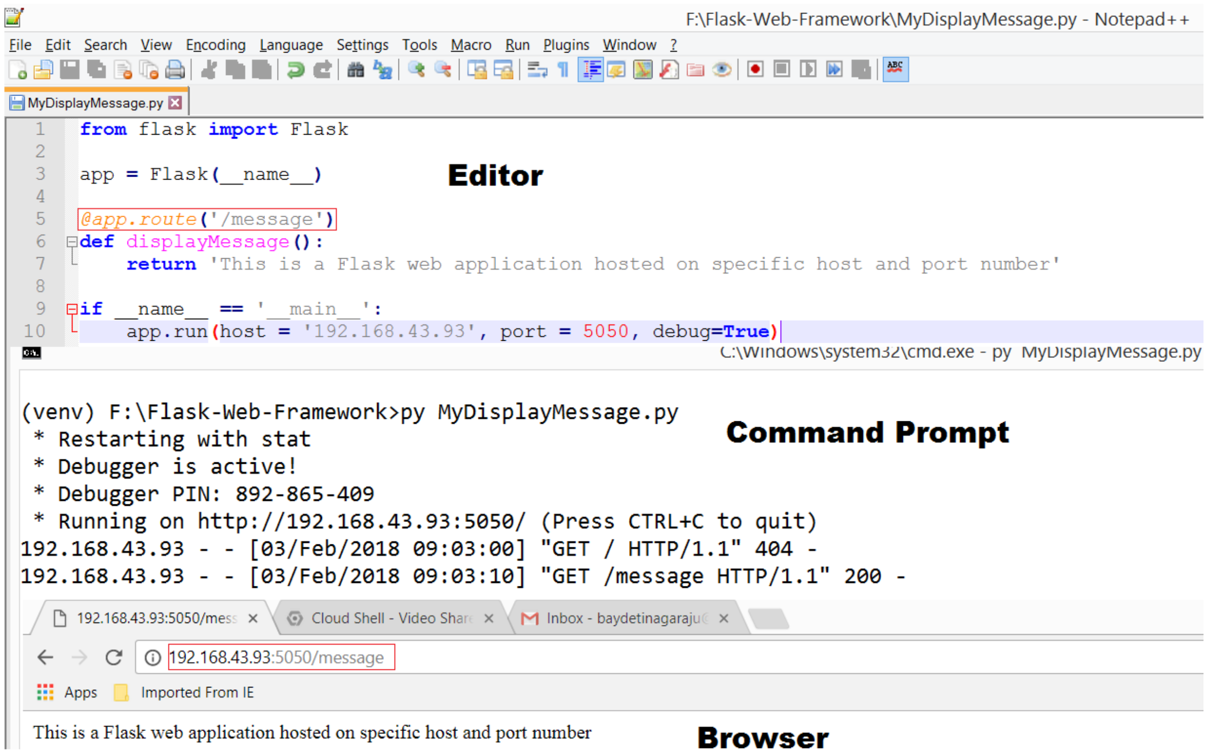


```
(venv) F:\Flask-Web-Framework>py MyDisplayMessage.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
192.168.43.93 - - [03/Feb/2018 08:26:25] "GET / HTTP/1.1" 200 -
192.168.43.93 - - [03/Feb/2018 08:26:25] "GET /favicon.ico HTTP/1.1" 404 -
 * Detected change in 'F:\\Flask-Web-Framework\\MyDisplayMessage.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
192.168.43.93 - - [03/Feb/2018 08:30:48] "GET / HTTP/1.1" 200 -
```



This is a Flask web application hosted on specific host and port number

7. It is possible to access the desired web page directly without traversing from the home page. Binding an URL to a function can be accomplished in two ways:
   a. route() decorator in Flask
   b. add_url_rule() function of an application object



```
from flask import Flask

app = Flask(__name__)

@app.route('/message')
def displayMessage():
    return 'This is a Flask web application hosted on specific host and port number'

if __name__ == '__main__':
    app.run(host = '192.168.43.93', port = 5050, debug=True)
```

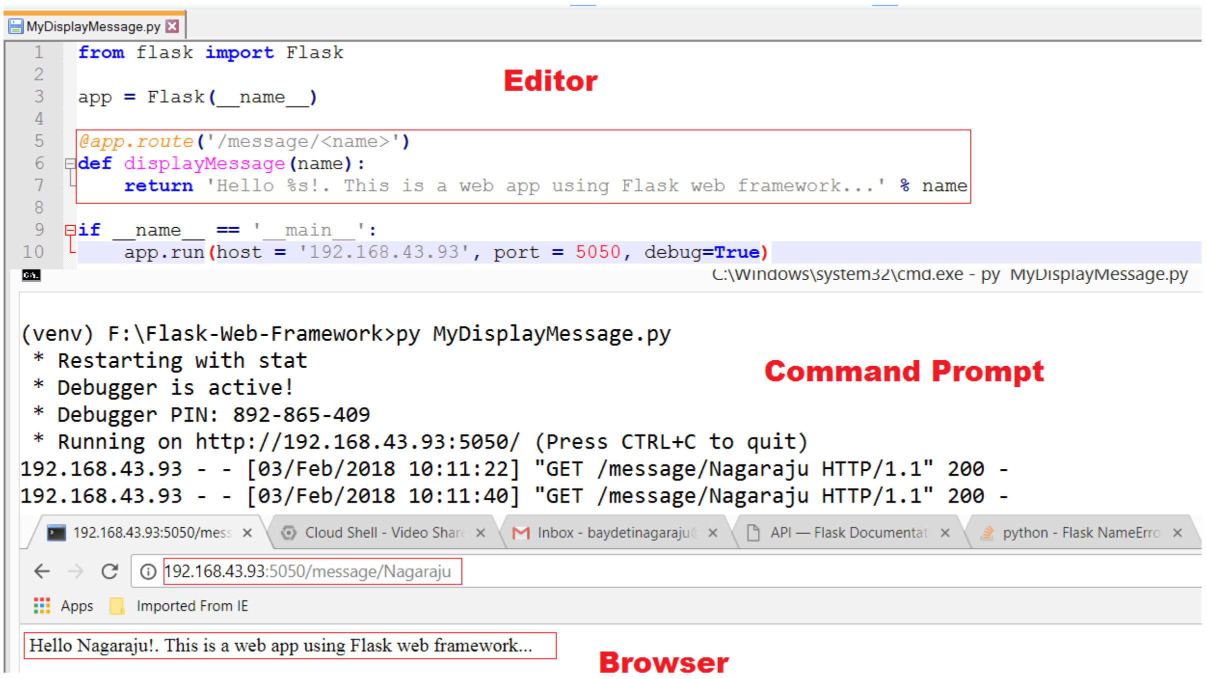
```
(venv) F:\Flask-Web-Framework>py MyDisplayMessage.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
192.168.43.93 - - [03/Feb/2018 09:03:00] "GET / HTTP/1.1" 404 -
192.168.43.93 - - [03/Feb/2018 09:03:10] "GET /message HTTP/1.1" 200 -
```

192.168.43.93:5050/message

This is a Flask web application hosted on specific host and port number

F:\Flask-Web-Framework\MyDisplayMessage.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
MyDisplayMessage.py
Editor
C:\Windows\system32\cmd.exe - py MyDisplayMessage.py
Command Prompt
192.168.43.93:5050/mess
Cloud Shell - Video Share
Inbox - baydetinagaraju
Apps
Imported From IE
Browser

8. It is also possible to bind the URL dynamically by passing a variable from the URL



```
from flask import Flask

app = Flask(__name__)

@app.route('/message/<name>')
def displayMessage(name):
    return 'Hello %s!. This is a web app using Flask web framework...' % name

if __name__ == '__main__':
    app.run(host = '192.168.43.93', port = 5050, debug=True)
```

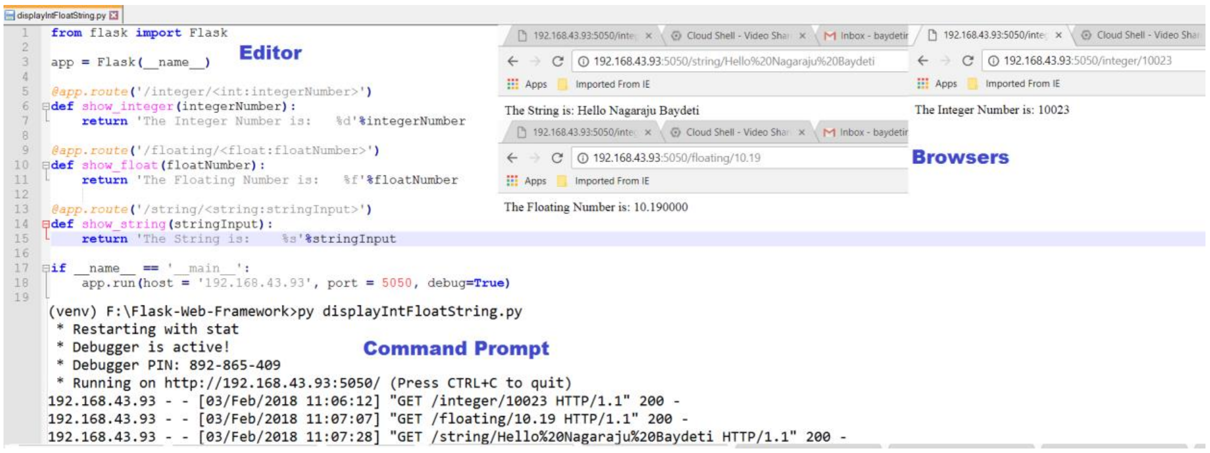
```
(venv) F:\Flask-Web-Framework>py MyDisplayMessage.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
192.168.43.93 - - [03/Feb/2018 10:11:22] "GET /message/Nagaraju HTTP/1.1" 200 -
192.168.43.93 - - [03/Feb/2018 10:11:40] "GET /message/Nagaraju HTTP/1.1" 200 -
```

192.168.43.93:5050/message/Nagaraju

Hello Nagaraju!. This is a web app using Flask web framework...

Not only strings, the variable can be integer, floating point and directory path

9. Depending on the variable, corresponding URL will be dynamically mapped as shown below:



```python
from flask import Flask

app = Flask(__name__)

@app.route('/integer/<int:integerNumber>')
def show_integer(integerNumber):
    return 'The Integer Number is:   %d'%integerNumber

@app.route('/floating/<float:floatNumber>')
def show_float(floatNumber):
    return 'The Floating Number is:   %f'%floatNumber

@app.route('/string/<string:stringInput>')
def show_string(stringInput):
    return 'The String is:    %s'%stringInput

if __name__ == '__main__':
    app.run(host = '192.168.43.93', port = 5050, debug=True)
```

```
192.168.43.93:5050/string/Hello%20Nagaraju%20Baydeti
The String is: Hello Nagaraju Baydeti

192.168.43.93:5050/integer/10023
The Integer Number is: 10023

192.168.43.93:5050/floating/10.19
The Floating Number is: 10.190000
```

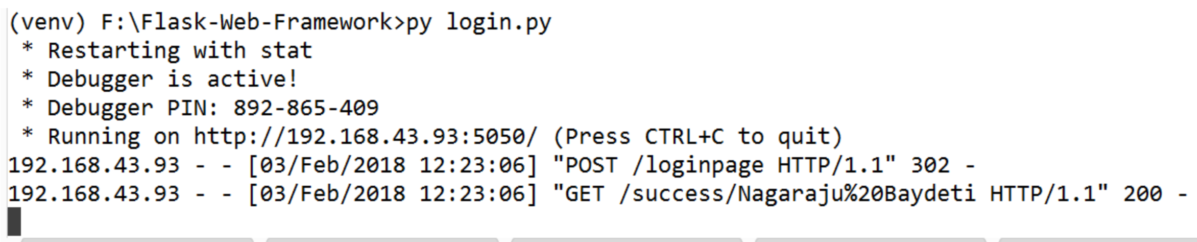
```
(venv) F:\Flask-Web-Framework>py displayIntFloatString.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
192.168.43.93 - - [03/Feb/2018 11:06:12] "GET /integer/10023 HTTP/1.1" 200 -
192.168.43.93 - - [03/Feb/2018 11:07:07] "GET /floating/10.19 HTTP/1.1" 200 -
192.168.43.93 - - [03/Feb/2018 11:07:28] "GET /string/Hello%20Nagaraju%20Baydeti HTTP/1.1" 200 -
```

10. Flask – Http Methods (demo for post and get methods using a html page)

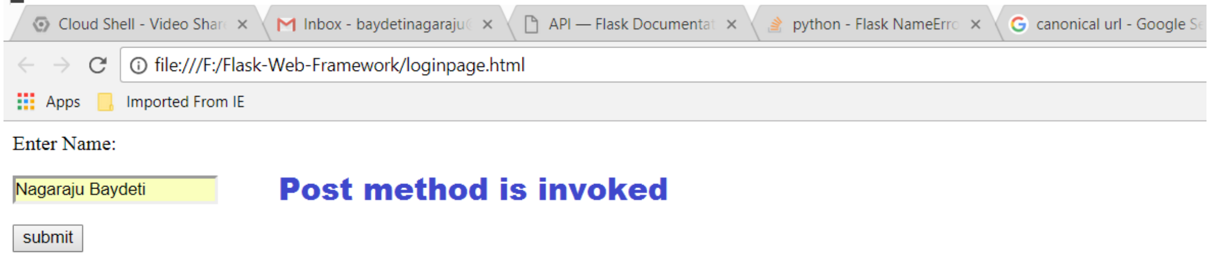
```html
#loginpage.html

<html>
   <body>
      <form action="http://192.168.43.93:5050/loginpage" method="post">
      <p> Enter Name:  </p>
      <p> <input type = "text" name = "nm" /> </p>
      <p> <input type = "submit" value = "submit" /> </p>
      </form>
   </body>
</html>
```

```python
#login.py

from flask import Flask, redirect, url_for, request
app = Flask(__name__)
@app.route('/success/<name>')
def success(name):
   return 'Welcome : %s'%name
@app.route('/loginpage', methods = ['post', 'get'])
def login():
   if request.method == 'POST':
      user = request.form['nm']
      return redirect(url_for('success',name=user))
   else:
      user = request.args.get('nm')
      return redirect(url_for('success',name=user))
if __name__ == '__main__':
   app.run(host = '192.168.43.93', port = 5050, debug=True)
```

Open the loginpage.html and give an input string. The screenshots below explains the flow of execution.



```
(venv) F:\Flask-Web-Framework>py login.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 892-865-409
 * Running on http://192.168.43.93:5050/ (Press CTRL+C to quit)
192.168.43.93 - - [03/Feb/2018 12:23:06] "POST /loginpage HTTP/1.1" 302 -
192.168.43.93 - - [03/Feb/2018 12:23:06] "GET /success/Nagaraju%20Baydeti HTTP/1.1" 200 -
```

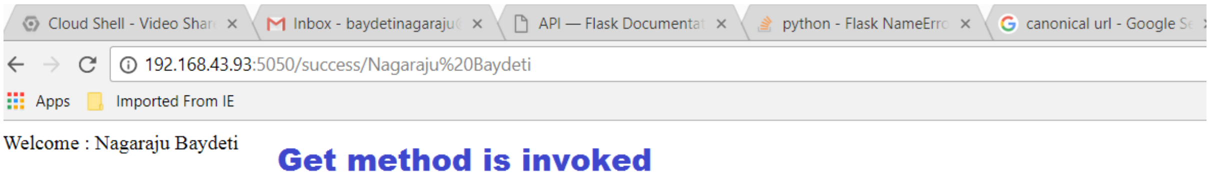


file:///F:/Flask-Web-Framework/loginpage.html

Enter Name:

Nagaraju Baydeti

Post method is invoked



192.168.43.93:5050/success/Nagaraju%20Baydeti

Welcome : Nagaraju Baydeti

Get method is invoked

Flask Web Application – Deployment in Google Cloud Platform

1. Create a virtual environment
2. Create a separate directory for your web application and three files are required: main.py, app.yaml, requirements.txt
3. Open Google Cloud SDK shell in your local machine
4. Execute the command *gcloud auth login*. This command authenticates a google user login for the purpose of using the google could platform. Enter your google credentials for the authentication procedure.
5. Check the execution of the python program in the local machine before deploying to cloud and once done with the local deployment, execute the command *gcloud app deploy* for deploying the app in the cloud and and then *gcloud app browse* for checking the deployed app from the browser.