How to Run the Application

Contents:

Necessary Installations

1. The easiest way to create/run our application is to do so on a Linux Operating System. We recommend Ubuntu. Even though it is foreign to most people, we recommend downloading Oracle VM and using this guide here.
2. After the VM is set up, there is more software needed to be installed. This can be done in two ways:
   - Using a requirements file
   - Manually installing Flask, Docker, and any other dependencies

3. Before bigger applications can be installed, python3 and pip (a package management used to install and manage python packages). These can be done by opening a terminal in the VM and typing in:

```
sudo apt install python3
Sudo apt install python3-pip
```

4. From there, Docker and Flask can be installed. To do so with a requirements file, create a requirements.txt file in the terminal using the “touch” command, and enter the following in the file:

- flask
- flask_restful
- docker

Then type the command:

```
pip3 install -r requirements.txt
```

Of course, if one doesn’t want to use a requirements.txt, each dependency can be installed using pip3 install *dependency*

Flask

1. Flask is a simple web server that is written in and uses python. It allows for fast and easy web development without the need of bigger web servers like nginx or apache. It is primarily involved with the frontend, i.e. what the user sees. More information will be discussed in our design choices, but to get started we recommend this link

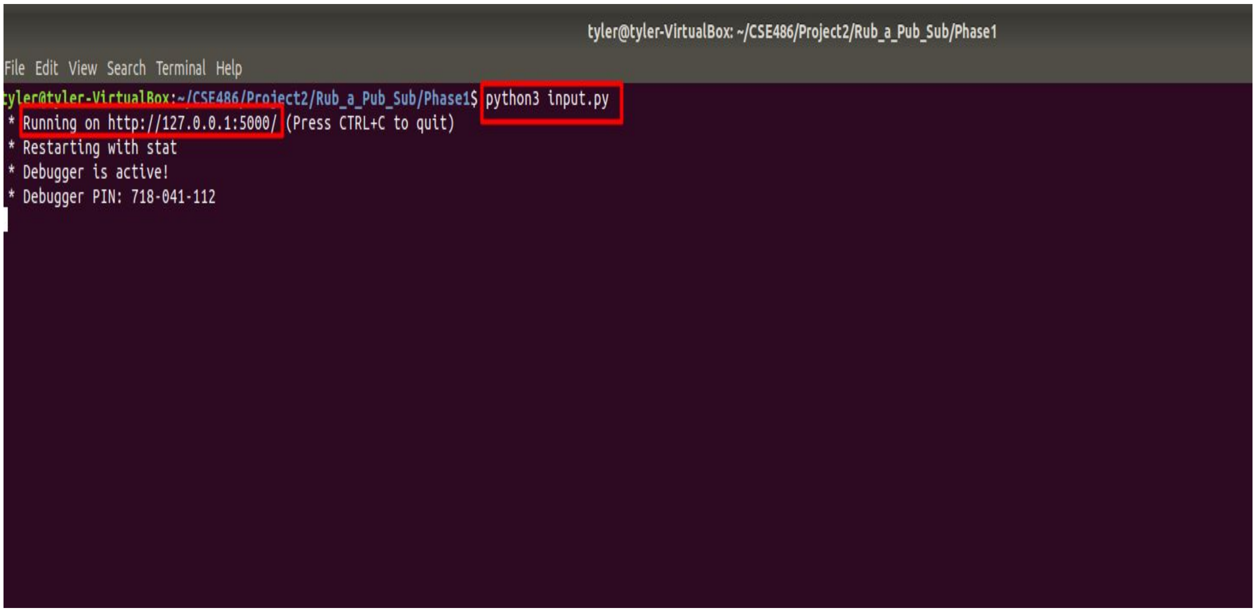
Docker

1. For Phase1, we run a python application that the user inputs through docker instead of having a python compiler directly doing so. The user itself will not directly be involved with doing anything docker related - the purpose is mainly to be done on the backend. More is discussed in our design choices document, but to get started we recommend this link

Interacting With the UI

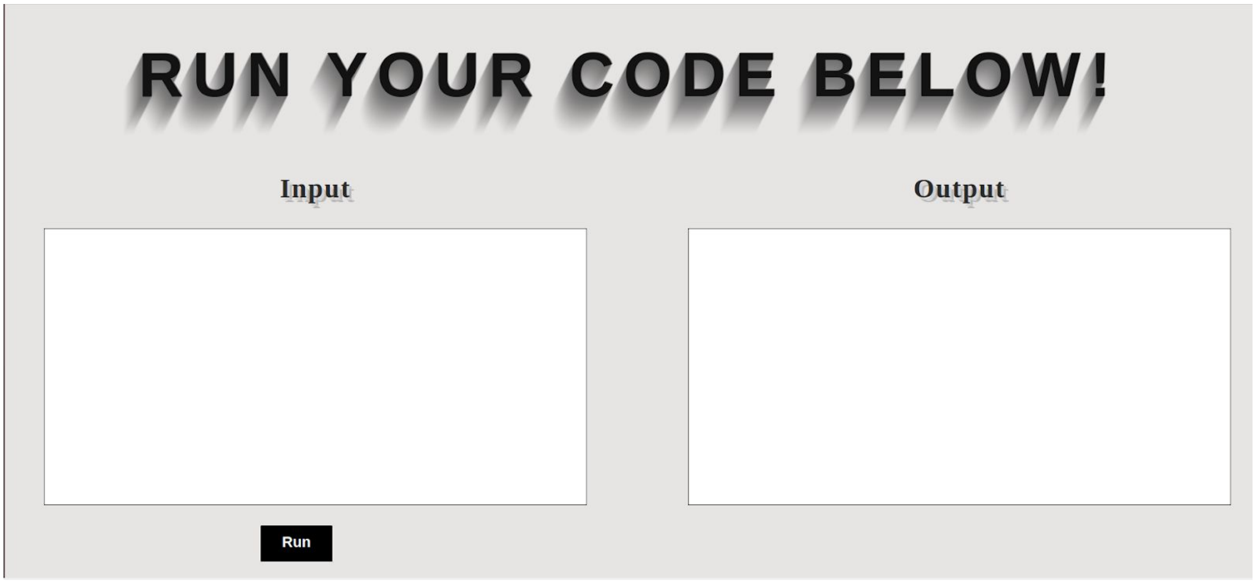
1. Now that everything is installed, we can get started. Flask and Docker (front-end and back-end) interact with one another to create a simple application that, when a user puts in valid python code, will run the application through docker and post the result in an output box.

2. The first step is to run our flask application. To do so, follow the screenshot:

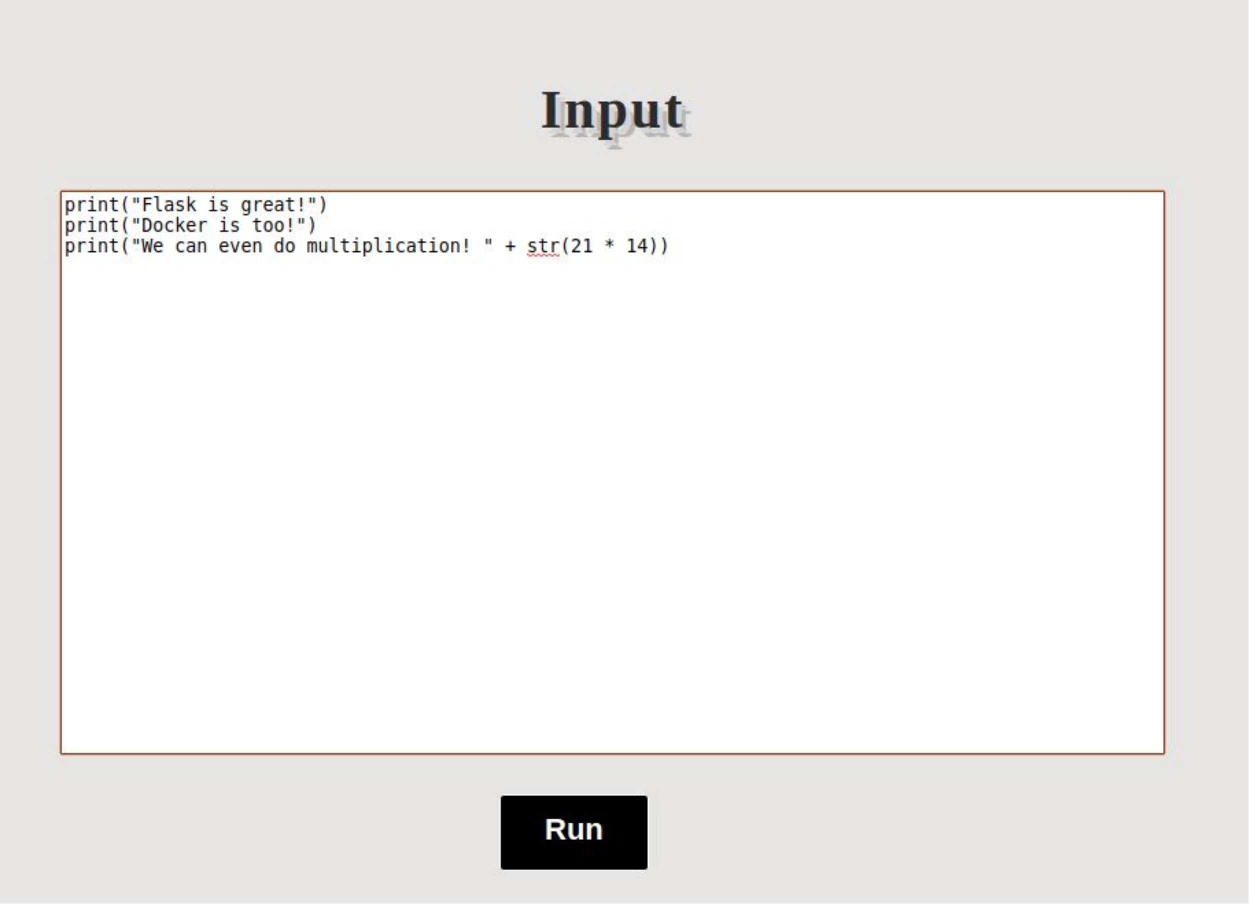

tyler@tyler-VirtualBox: ~/CSE486/Project2/Rub_a_Pub_Sub/Phase1
File Edit View Search Terminal Help

```
tyler@tyler-VirtualBox:~/CSE486/Project2/Rub_a_Pub_Sub/Phase1$ python3 input.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 718-041-112
```

3. Notice how the server is running on http://127.0.0.1:5000. This is another way of saying that it is running on our machine's localhost, on port 5000. This simulates an application running on the web without others being able to access it. Now that our app is successfully running, open a web browser and type in http://localhost:5000

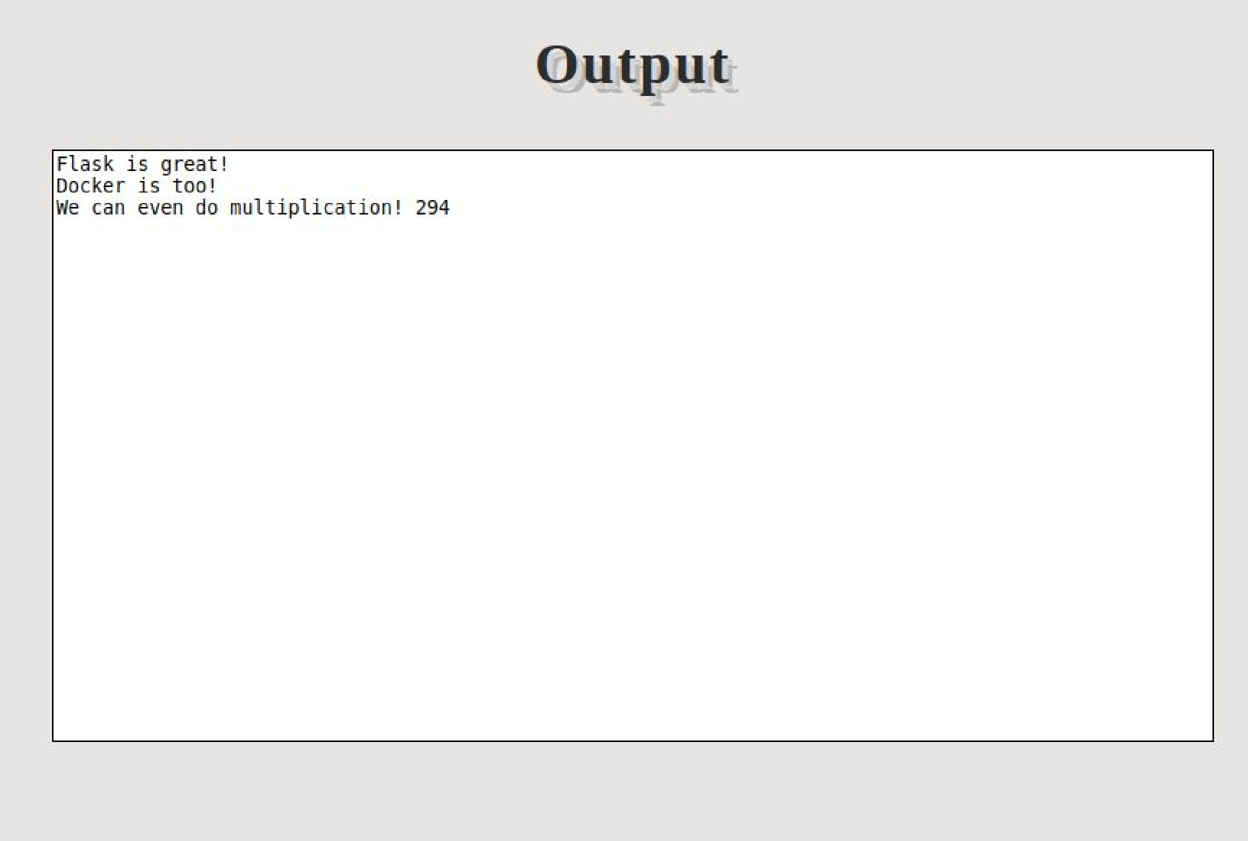


4. You will see the image above. Any valid python can be entered into the input and the application will run through docker. Let's see a simple example:



```
print("Flask is great!")
print("Docker is too!")
print("We can even do multiplication! " + str(21 * 14))
```

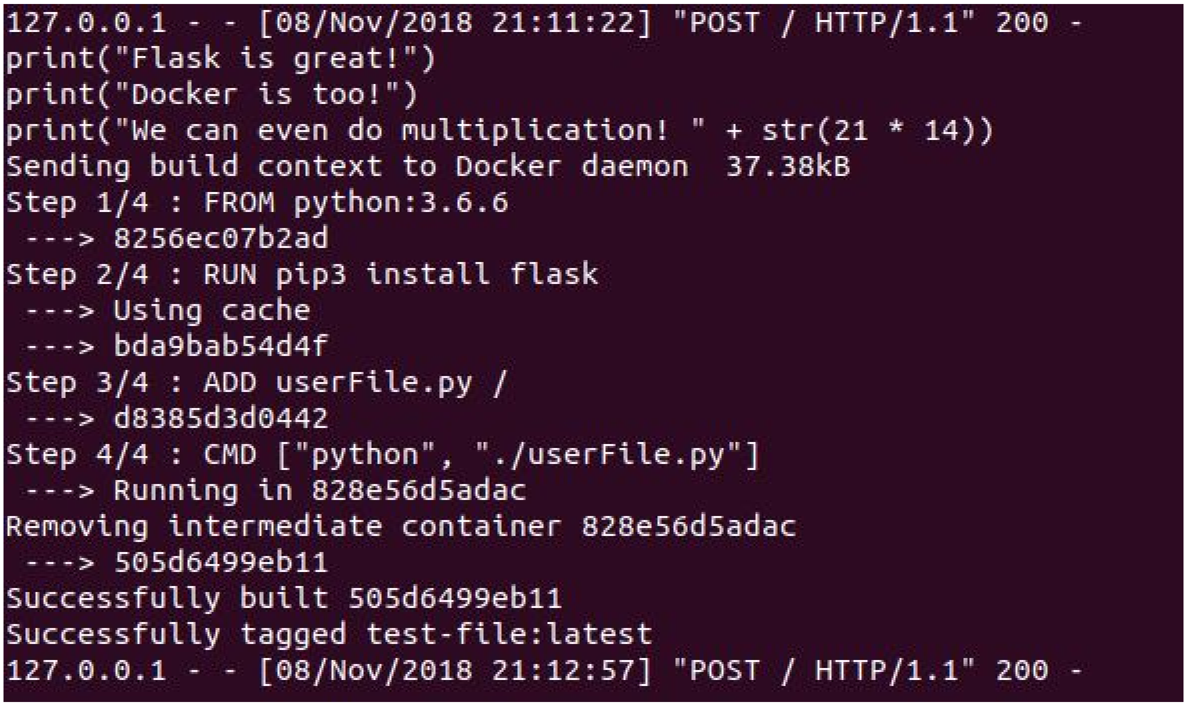
After inputting your code, click the "Run" button.



Output

```
Flask is great!
Docker is too!
We can even do multiplication! 294
```

5. There's a lot of stuff going on under the hood. If we look at the terminal, we can see the Flask application running a docker image that is created on the backend.



```
127.0.0.1 - - [08/Nov/2018 21:11:22] "POST / HTTP/1.1" 200 -
print("Flask is great!")
print("Docker is too!")
print("We can even do multiplication! " + str(21 * 14))
Sending build context to Docker daemon  37.38kB
Step 1/4 : FROM python:3.6.6
 ---> 8256ec07b2ad
Step 2/4 : RUN pip3 install flask
 ---> Using cache
 ---> bda9bab54d4f
Step 3/4 : ADD userFile.py /
 ---> d8385d3d0442
Step 4/4 : CMD ["python", "./userFile.py"]
 ---> Running in 828e56d5adac
Removing intermediate container 828e56d5adac
 ---> 505d6499eb11
Successfully built 505d6499eb11
Successfully tagged test-file:latest
127.0.0.1 - - [08/Nov/2018 21:12:57] "POST / HTTP/1.1" 200 -
```