

# INFO3180 Lab 1 (10 Marks)

Due Date: **4 February, 2018**

**Note:** Because Python and all the relevant development tools are already installed at <http://c9.io>, it is recommended that you use c9.io for this lab, you will need to sign up for an account. At a later date you can setup a similar environment on your computer.

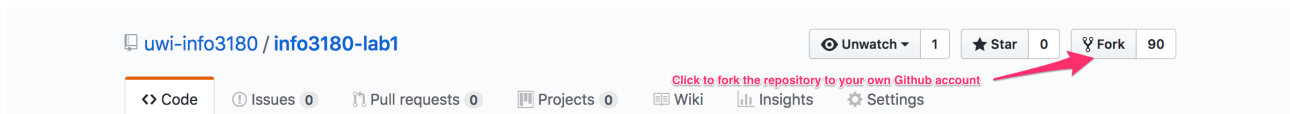
## Flask

In this exercise you will install flask, create some routes, customize a template and then deploy it to Heroku, so ensure you sign up for a Heroku account.


## Fork and Clone the repository

Login to your Github account and fork the following repository:




<https://github.com/uwi-info3180/info3180-lab1/>










Next, to clone it to your local machine or to Cloud9 you will first need to get the URL of the new repository that you just forked:







 This repository


Pull requestsIssuesMarketplaceExplore

 **yllynfatt / info3180-lab1**  
forked from uwi-info3180/info3180-lab1

 Unwatch  1  Star  0  Fork  91

 Code  Pull requests  Projects  Wiki  Insights  Settings


Starter code for Lab 1 1. Click on "Clone or download" 

Add topics

3 commits1 branch0 releases1 contributor

Branch: masterNew pull requestCreate new fileUpload filesFind fileClone or download


This branch is even with uwi-info3180:master.

 ylynfatt added 404 page template

|                  |  |              |
|------------------|--|--------------|
| static/css       | info3180 lab 1 starter code initial commit             |              |
| templates        | added 404 page template                                |              |
| .gitignore       | info3180 lab 1 starter code initial commit             | Jan 20, 2017 |
| Procfile         | updated Procfile to use gunicorn so it works on heroku | Jan 20, 2017 |
| README.md        | info3180 lab 1 starter code initial commit             | Jan 20, 2017 |
| app.py           | info3180 lab 1 starter code initial commit             | Jan 20, 2017 |
| requirements.txt | info3180 lab 1 starter code initial commit             | Jan 20, 2017 |

Clone with HTTPSUse SSH

Use Git or checkout with SVN using the web URL.



Open in DesktopDownload ZIP

README.md

INFO3180 Lab 1 Starter Code

If working with Cloud9 then do the following to clone the repository to the C9 workspace:

The screenshot shows the 'Create a new workspace' form in AWS Cloud9. Five red arrows point to specific fields with numbered instructions:

- 1. Enter a name for your workspace**: Points to the 'Workspace name' field, which contains 'info3180-lab1'.
- 2. If given the option to select a team, ensure you set it to "Don't Set A team for this workspace"**: Points to the 'Team' dropdown menu, which is set to 'Don't set a team for this workspace'.
- 3. Make your workspace Public.**: Points to the 'Public' radio button under the 'Hosted workspace' tab.
- 4. Enter the Github URL for the repository you would like to clone to your workspace.**: Points to the 'Clone from Git or Mercurial URL (optional)' text input field, which contains the placeholder 'e.g. ajaxorg/ace or git@github.com:ajaxorg/ace.git'.
- 5. Ensure you select "Python" as your workspace template.**: Points to the 'Python' template card in the 'Choose a template' section.

The form also includes a 'Description' field, a 'Hosted workspace' tab, and a grid of template cards for various languages and frameworks like HTML5, Node.js, PHP, Python, Django, Ruby, C++, Wordpress, Rails Tutorial, Blank, and Harvard's CS50. A green 'Create workspace' button is at the bottom.

or if working on your own local computer then do the following from the command line/prompt in a folder of your choice. Ensure you change **{yourusername}** to your actual Github username:

```
git clone https://github.com/{yourusername}/info3180-lab1
info3180-lab1
```

### **Step 1 - Setup and activate a virtualenv**

Navigate to the folder with the code you just cloned and setup a virtual environment.

```
cd info3180-lab1 #not needed if you are working on Cloud9
virtualenv venv
```

Activate the environment

```
source venv/bin/activate
```

### ***Step 3 - Install Flask***

Install Flask by running the following command:

```
pip install Flask
```

You'll see output similar to this:

```
(venv) → lab1 pip install Flask
Collecting Flask
  Using cached Flask-0.12-py2.py3-none-any.whl
Collecting Jinja2>=2.4 (from Flask)
  Using cached Jinja2-2.9.4-py2.py3-none-any.whl
Collecting Werkzeug>=0.7 (from Flask)
  Using cached Werkzeug-0.11.15-py2.py3-none-any.whl
Collecting click>=2.0 (from Flask)
  Using cached click-6.7-py2.py3-none-any.whl
Collecting itsdangerous>=0.21 (from Flask)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->Flask)
Installing collected packages: MarkupSafe, Jinja2, Werkzeug, click, itsdangerous, Flask
Successfully installed Flask-0.12 Jinja2-2.9.4 MarkupSafe-0.23 Werkzeug-0.11.15 click-6.7 itsdangerous-0.24
```

### ***Step 3 - Create a Route and View Function***

The main application code is located at **app.py**. You will notice that we have already imported the Flask library and initialized the application:

```
from flask import Flask, render_template
app = Flask(__name__)
```

However, we have not yet created a route and it's respective *view function*. So let's do this now. Our first route will be '/' and will represent our home page. We do this by using the **@app.route** decorator from Flask. Next we'll create a view function called '**home**' and we'll simply return the message '**My home page**'.

```
@app.route('/')  
def home():  
    return 'My home page'
```

You will also need to edit the last line and change it to say:

```
app.run(debug=True, host="0.0.0.0", port=8080)
```

This tells Flask how to start the development server. The host IP will be 0.0.0.0 which will listen for connections on any available IP address that the computer/server has open for connections. It will also be set to use port 8080 and run in debug mode.

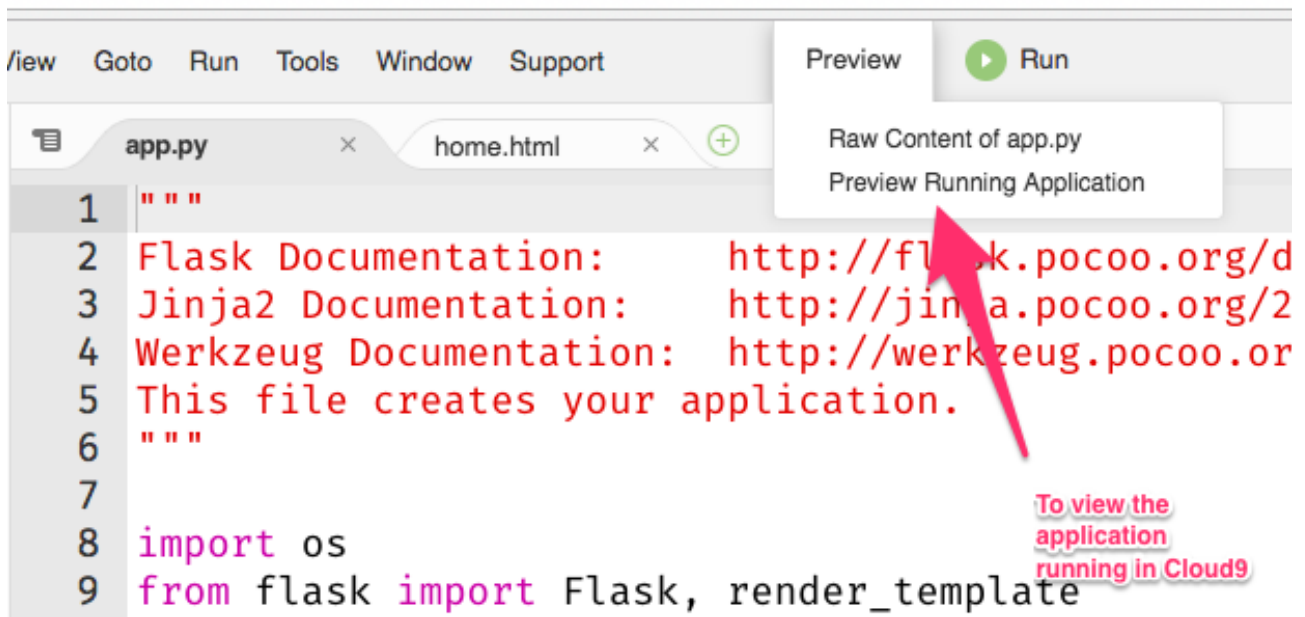
Now launch the app using the following command

```
python app.py
```

You will see output similar to this:

```
(venv) → lab1 python app.py  
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)  
* Restarting with stat  
* Debugger is active!  
* Debugger pin code: 264-267-630
```

Visit your application by using the Preview > "Preview Running Application" in Cloud9



or if you are working on your local machine then browse to <http://0.0.0.0:8080> or <http://localhost:8080> .

You should simply see '**My home page**' being displayed.

#### ***Step 4 - Create another route and create a template to render.***

We will now create another route and it's respective view function, but this time we will render a Jinja2 template using the **render\_template()** method. The new route is **'/about'** and the view function should be called **'about'**:

```
@app.route('/about')
def about():
    return render_template('about.html')
```

Now create a new template file in your **templates** directory called **'about.html'**. In that file, create a basic HTML page with at least a heading (**<h1></h1>**) with your name and a paragraph (**<p></p>**) about yourself.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Website - About</title>
  <link rel="stylesheet" href="/static/css/app.css">
</head>
<body>
  <h1>Place Your Name Here</h1>
  <p>Write a short paragraph about yourself. here</p>
</body>
</html>
```

Now browse to your new route in your web browser to see your new page. Feel free to customize this some more if you'd like but ensure you have the `<h1>` heading and the `<p>` paragraph.

### ***Step 5 - Push your code to your Github repository***

Now that you've created your two routes, it's time to add, commit and push your code to your Github repository.

```
git add .
git commit -m 'created my first Flask app'
git push origin master
```

### ***Step 6 - Deploy the application to Heroku***

Finally, we can now deploy your application to Heroku. Heroku is a cloud based hosting platform for you to deploy and run web apps. If you haven't already done so, ensure that you sign up for an account on the Heroku website (<https://heroku.com>) and then do the following. **Note:** You will also need the Heroku CLI. This should already be installed on

Cloud9. If you are instead working on your local machine, then you must install the CLI tool. See instructions at <https://devcenter.heroku.com/articles/heroku-cli> .

```
heroku login  
heroku apps:create  
git push heroku master
```

When you run the **heroku apps:create** command it will give you a unique app name. Then once you push your code to Heroku, you should now be able to view your web application on Heroku at <https://{yourappname}.herokuapp.com/>.

## Submission

Submit your code via the "Lab 1 Submission" link on OurVLE. You should submit the following links:

1. Github repository URL for your Flask Exercise e.g. <https://github.com/{yourusername}/info3180-lab1>
2. URL for your Heroku app e.g. <https://{yourappname}.herokuapp.com>