INFO3180 Lab 1 (20 Marks)

Due Date: **February 2, 2020 at 11:55pm**

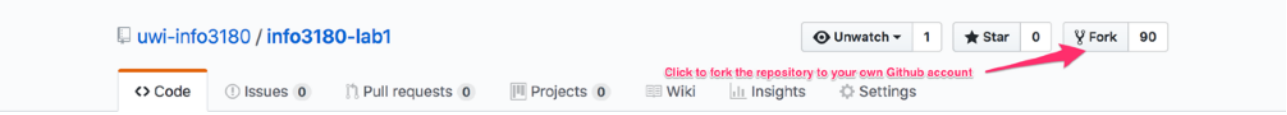
Flask

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.

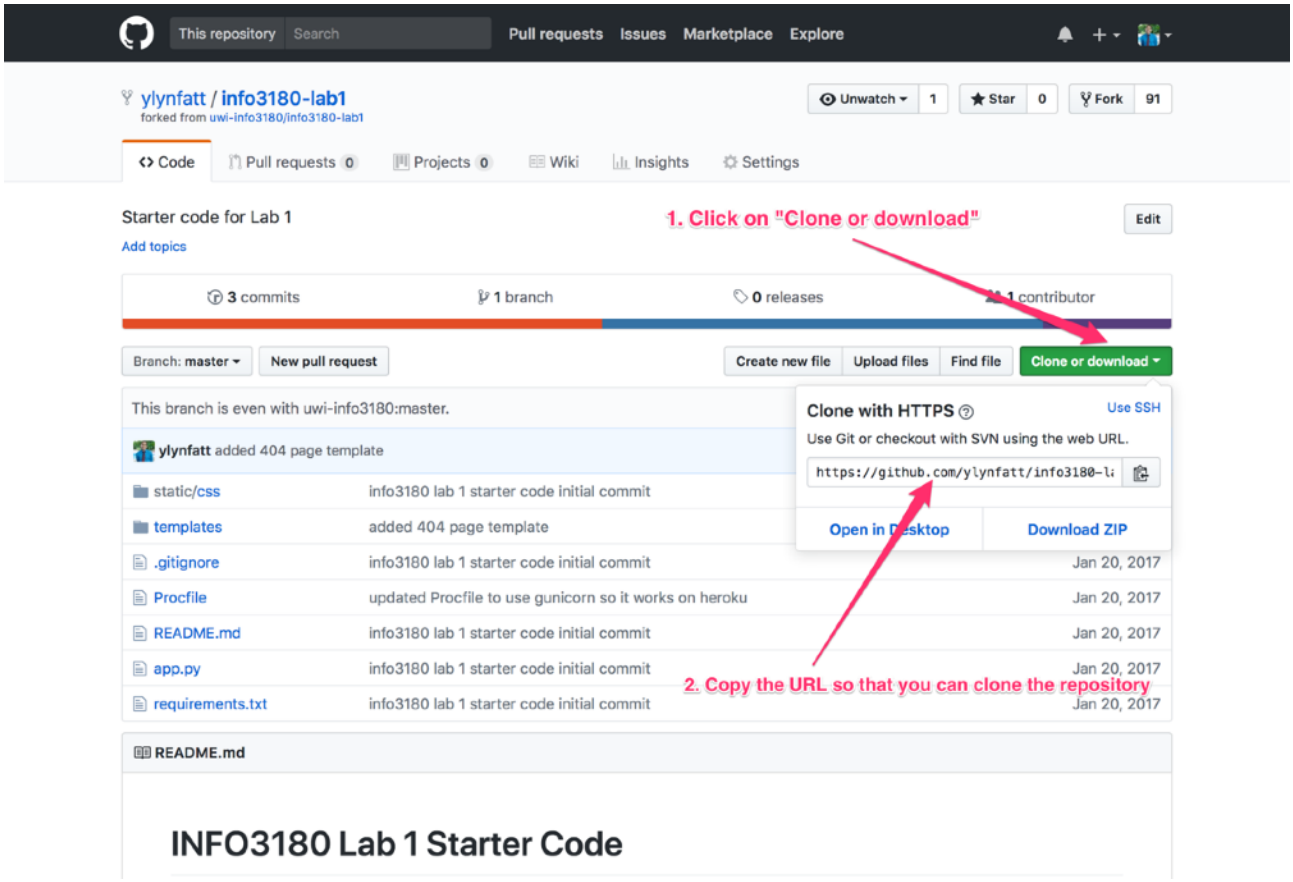
Fork and Clone the repository

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea. In our case for this lab we are using another project on Github as the starting point for this Lab.

Login to your Github account and fork the repository by visiting the link below and clicking on the "**Fork**" button:
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 to your account:



If working with Cloud9 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 virtual environment

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

```
cd info3180-lab1
python -m venv venv
or
python3 -m venv venv (if you have both Python 2 and Python 3)
```

```
or
python3.5 -m venv venv (if using Cloud9)
```

Activate the environment

```
source venv/bin/activate
or
.\venv\Scripts\activate (if using Windows)
```

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)
```

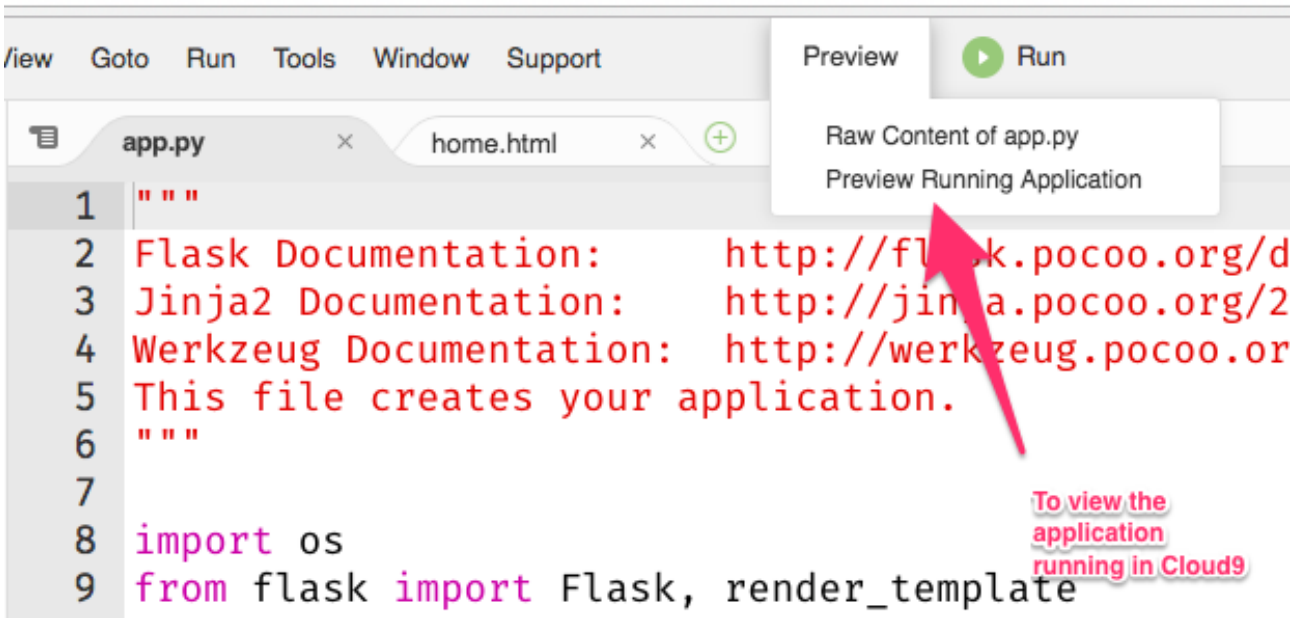
This tells Flask how to start the development server. The host IP will be **0.0.0.0** which will listen for connections on any available IP address that the computer/server has open for connections. It will also be set to use port **8080** and run in **debug mode**.

Now launch the app using the following command

```
python app.py (or you may need to use python3 or python3.5)
```

You will see output similar to this:

If using Cloud9, 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 (this is the preferred URL on Windows).

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

Step 4 - Push your code to your Github repository

Now that you've created your first route, 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 5 - Create another route and create a template to render.

We will now create another route and its 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 the HTML code below. Ensure that you put your name within the level one heading (**<h1></h1>)** and write a short paragraph within a paragraph tag (** **) 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 paragraph.

Step 6 - Push your code to your Github repository

Now that you've created your second route, 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 7 - 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/.

Note: If you need to make any further changes to your code, you do not need to re-run **`heroku apps:create`**. You should be able to just make your changes, then add and commit them to your local git repository and then push to both Github and Heroku again by doing **`git push origin master`** and then **`git push heroku master`**.

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

Grading

1. Defined the route and created associated view function for home page which returns the string **'My home page'.** (3 marks)
2. Modified the app.run() command to start the web server using host 0.0.0.0 on port 8080 and enabled debug mode. Also the flask development server should start successfully and the running web application should be able to be viewed in the browser without any errors. (2 marks)
3. Defined the route and created associated view function for the About page. The view function should utilize the appropriate function to render the template for the about page. (5 marks)
4. The template for the about page should be created in the **templates** folder with HTML specified in the example and your name should be in the **<h1></h1>** heading and paragraph about yourself in a ** ** tag. (5 marks)
5. You should be able to successfully view the about page in web browser. (2 marks)
6. The Application should be successfully deployed to Heroku and you should be able to view all of the above at URL submitted. (3 marks)