Hackathon Preparation Workshop

Brought to you by CompSoc and ShefESH

Session Overview

- Git and GitHub
- Python
- SQLite
- Database Security
- Flask

Git and GitHub

What is Git?

- Git is a popular version control system used for tracking code changes, who made them and code collaboration
- Things you can do with git:
  - Create a repository by initialising git on a folder
  - Commit modifications to files by pushing updates
  - Pull the latest version of files to a local copy
  - Revert to previous commits
  - Branch and merge to allow for work on different sections/versions

Installing Git

There are a few different ways of accomplishing this:

- By installing GitHub Desktop
- By installing from the Internet (for Windows/Mac)
- By installing through VS Code GitHub Pull Requests and Issues extension
- Mac specific: Using Homebrew
- Debian/Ubuntu specific: running ‘sudo apt-get install git-all’

Configuring Git

This is an important step to be able to commit file updates as it lets Git know who you are, and is done by running the following in Git Bash (for windows) or terminal (for Mac/Linux):

- git config –global user.name “your username”
- git config -global user.email “your email”

The email should be the same as the email you use/will use for GitHub

Creating a repository

To begin with, create an empty folder and then navigate to it within Bash/Terminal using the ‘cd’ command from COM1001 e.g. cd Documents/folder name

Once you are within the folder, you need to run the command ‘git init’ to initialise Git on that folder

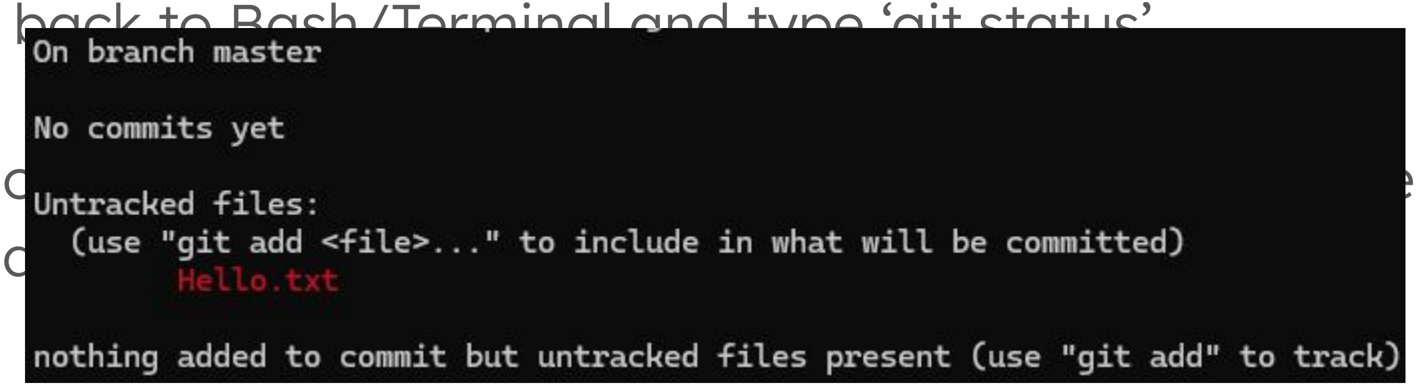
If this is successful, you should get a message returned to you saying ‘Initialized empty Git repository in (place your folder is)’

Adding a file

Within your folder, create and save a new text document using a text editor such as Microsoft Word or Notepad with some information e.g. “Hello World”

Return back to Bash/Terminal and type ‘git status’

This sho e we just add



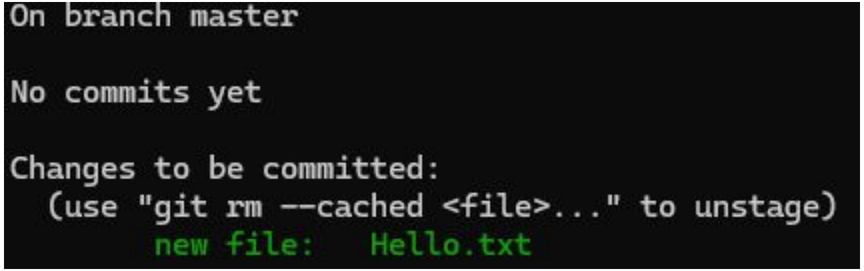
```
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Hello.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Staging a file

Now, we can use the command ‘git add (text file)’ to stage the file, which means that the file is ready to be committed.

If we had multiple files to stage, we can use the command ‘git add --all’ or ‘git add -A’

To check this has the ‘git status’ command again



```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Hello.txt
```

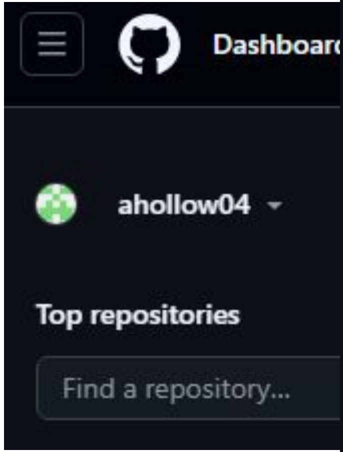
Committing a file

When we commit files, it is important to always include a clear message to help identify to yourself and others what has changed and when.

This is done using the command: ‘git commit -m “Useful message here”’

Pushing to GitHub

To be able t
repository:

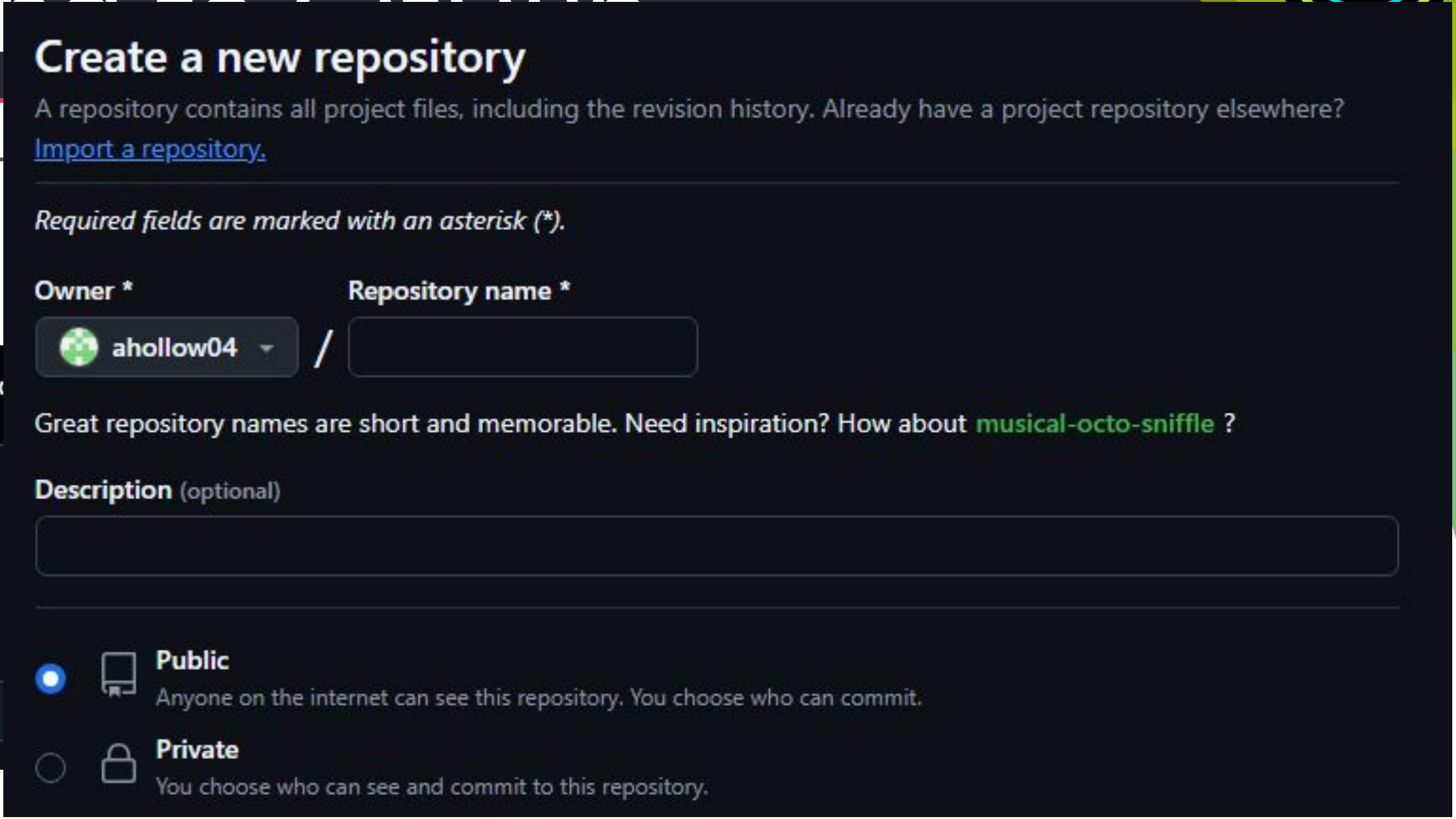


Dashboard

ahollow04

Top repositories

Find a repository...



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * ahollow04 / **Repository name ***

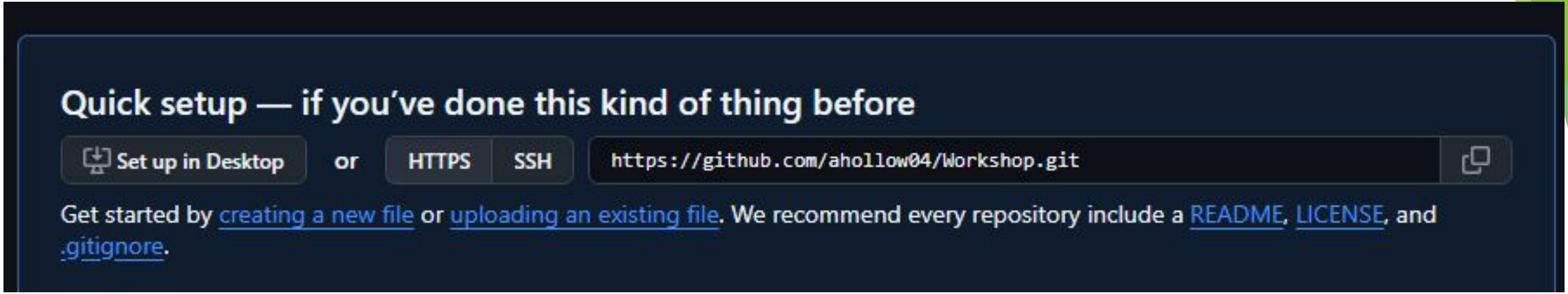
Great repository names are short and memorable. Need inspiration? How about musical-octo-sniffle ?

Description (optional)

● **Public**
Anyone on the internet can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

Pushing to GitHub



Quick setup — if you’ve done this kind of thing before

https://github.com/ahollow04/Workshop.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

You will need to copy the URL and use it in the following command: ‘git remote add origin (paste URL)’

Pushing to GitHub

Now that you have set up a connection between your local Git repository and your online GitHub repository, you can now run the following command: ‘git push --set-upstream origin master’

Since this is the first time you are pushing to GitHub, you need to use ‘--set-upstream’ to identify the default branch you want to push to

If you refresh your GitHub page, you should see that your repository has updated

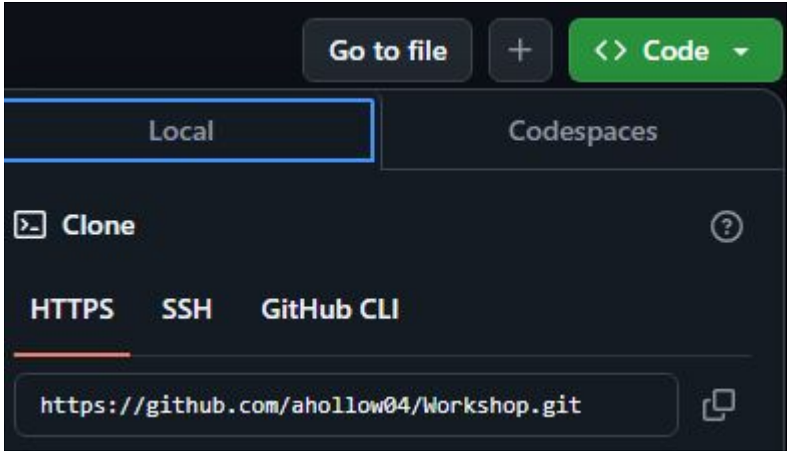
Pulling from GitHub

This is used to update your local version of a repository by using the command ‘git pull origin’

By using pull, we are using both fetch and merge commands behind the scenes, where fetch gets all of the change history and merge combines the current branch with a specified branch.

What if you want to work on an existing repository?

This can be done by cloning an existing repository so that you can work on it locally, using the command: ‘git clone (URL)’ where you can copy the URL from:



https://github.com/ahollow04/Workshop.git

Updates can be committed using:

- git add (file name)
- git commit -m “”
- git push

Branches

These are extremely useful to work on new features of a project in a contained area of the repository, ensuring that any breakages to code only affect the project branch and not the project itself.

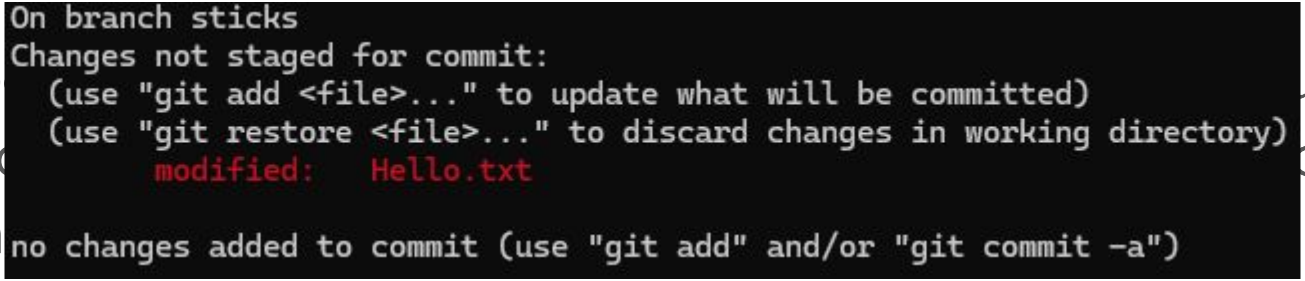
Another benefit of branches is that multiple developers can work on separate tasks at the same time without causing multiple project conflicts.

Pushing a branch to GitHub

In order to create a new branch, we use the following command:
‘git checkout -b (new branch name)’

Afterwards, make a couple of changes to the text file e.g. adding another word (don’t forget to save!)

Now, che us’ - it
is import our
new bran

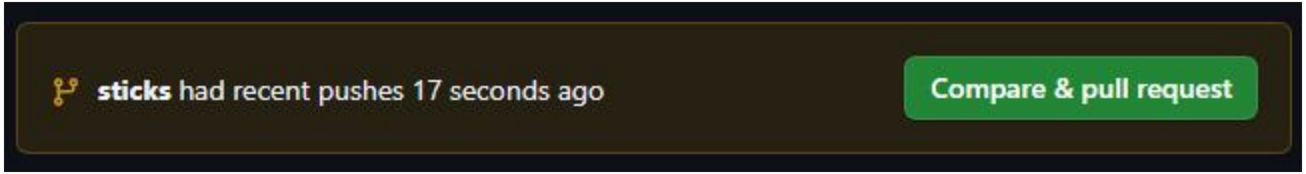


```
On branch sticks
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Pushing a branch to GitHub

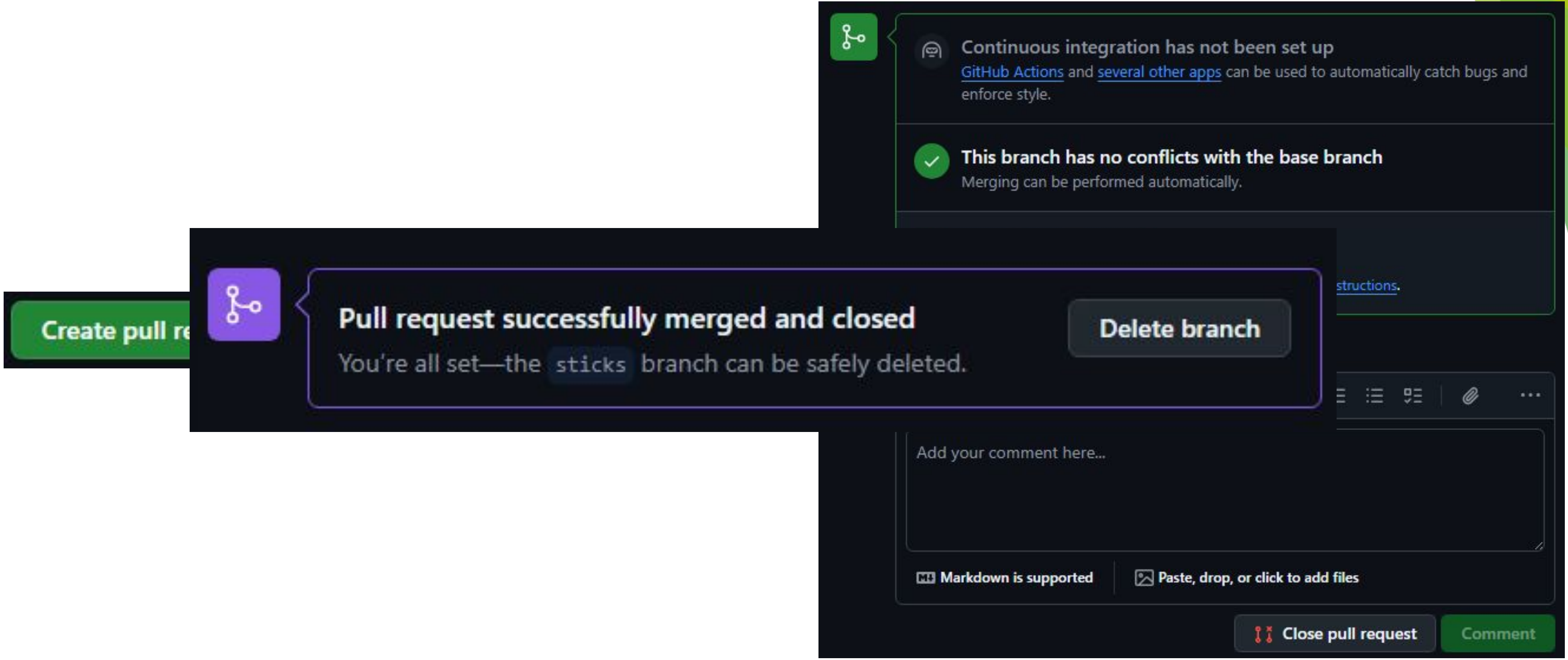
Like before, we can now use ‘git add (file name)’ and ‘git commit -m “(useful message)”’

Now, to push the newly created branch, we use the command ‘git push origin (new branch name)’



sticks had recent pushes 17 seconds ago

Pushing a branch to GitHub



Pull request successfully merged and closed

You're all set—the `sticks` branch can be safely deleted.

Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

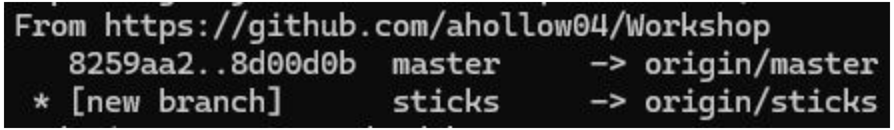
Merging can be performed automatically.

structions.

Add your comment here...

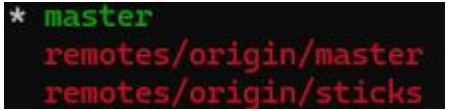
Pulling a branch from GitHub

When a branch has been added to a GitHub repository, it will show up when you run ‘git pull’ e.g.



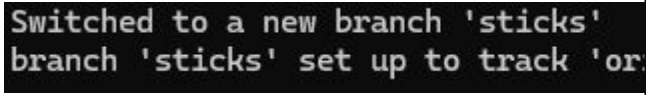
```
From https://github.com/ahollow04/Workshop
   8259aa2..8d00d0b  master     -> origin/master
 * [new branch]      sticks     -> origin/sticks
```

We can find out what branches are available locally and remotely by using ‘git branch -a’

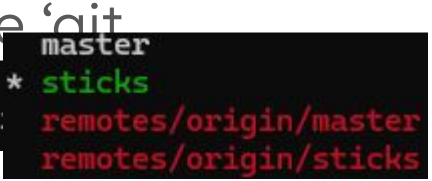


```
* master
  remotes/origin/master
  remotes/origin/sticks
```

And in order access remote branches locally we use ‘git checkout (branch name)’



```
Switched to a new branch 'sticks'
branch 'sticks' set up to track 'or
```



```
  master
* sticks
  remotes/origin/master
  remotes/origin/sticks
```

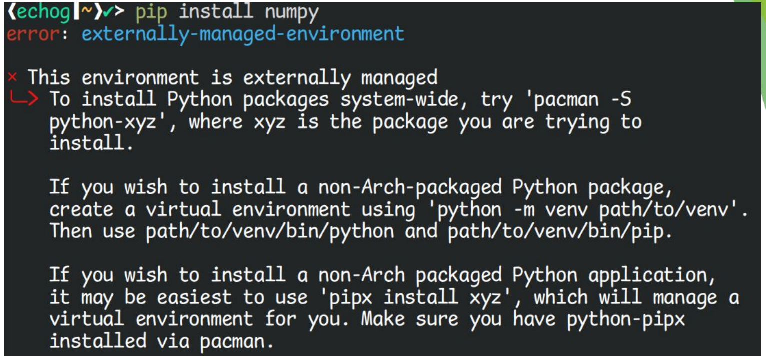
Python

Python

- A language used by almost everyone
- We're assuming that you already have it installed as well as a code editor like VSCode
- Time for a refresher on the basics

Python venv

Sometimes python will throw a fit when you try and pip install. It'll say something about "externally managed environments".



```
⟨echog|~⟩✓> pip install numpy
error: externally-managed-environment

× This environment is externally managed
╰─> To install Python packages system-wide, try 'pacman -S
    python-xyz', where xyz is the package you are trying to
    install.

    If you wish to install a non-Arch-packaged Python package,
    create a virtual environment using 'python -m venv path/to/venv'.
    Then use path/to/venv/bin/python and path/to/venv/bin/pip.

    If you wish to install a non-Arch packaged Python application,
    it may be easiest to use 'pipx install xyz', which will manage a
    virtual environment for you. Make sure you have python-pipx
    installed via pacman.
```

To fix this, we use **v**irtual **env**ironments!

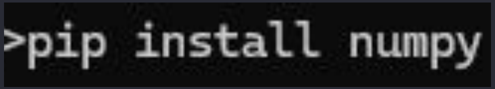
https://docs.python.org/3/library/venv.html

```
Linux: python -m venv /path/to/new/virtual/environment
Windows: python -m venv C:\path\to\new\virtual\environment
```

```
source /path/to/new/virtual/environment/bin/activat
```

To activate it

Numpy



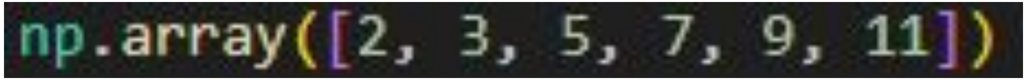
```
>pip install numpy
```

https://www.w3schools.com/python/numpy/default.asp

A Common python library

It adds support for faster multidimensional arrays and mathematical functions

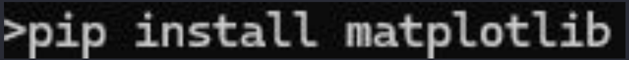
We're going to use it here just for its n-dimensional arrays, as they are faster than Python lists and as such are used for tools like Matplotlib



```
np.array([2, 3, 5, 7, 9, 11])
```

It can also be used for things like logarithms, rounding numbers, trigonometric functions and more!

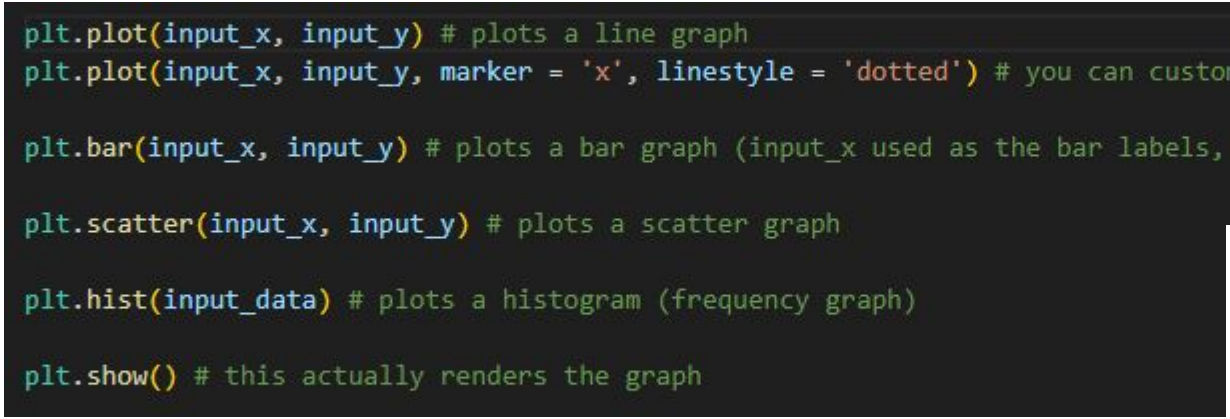
Matplotlib



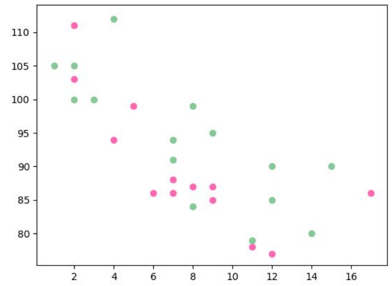
```
>pip install matplotlib
```

https://www.w3schools.com/python/matplotlib_intro.asp

A graph plotting library

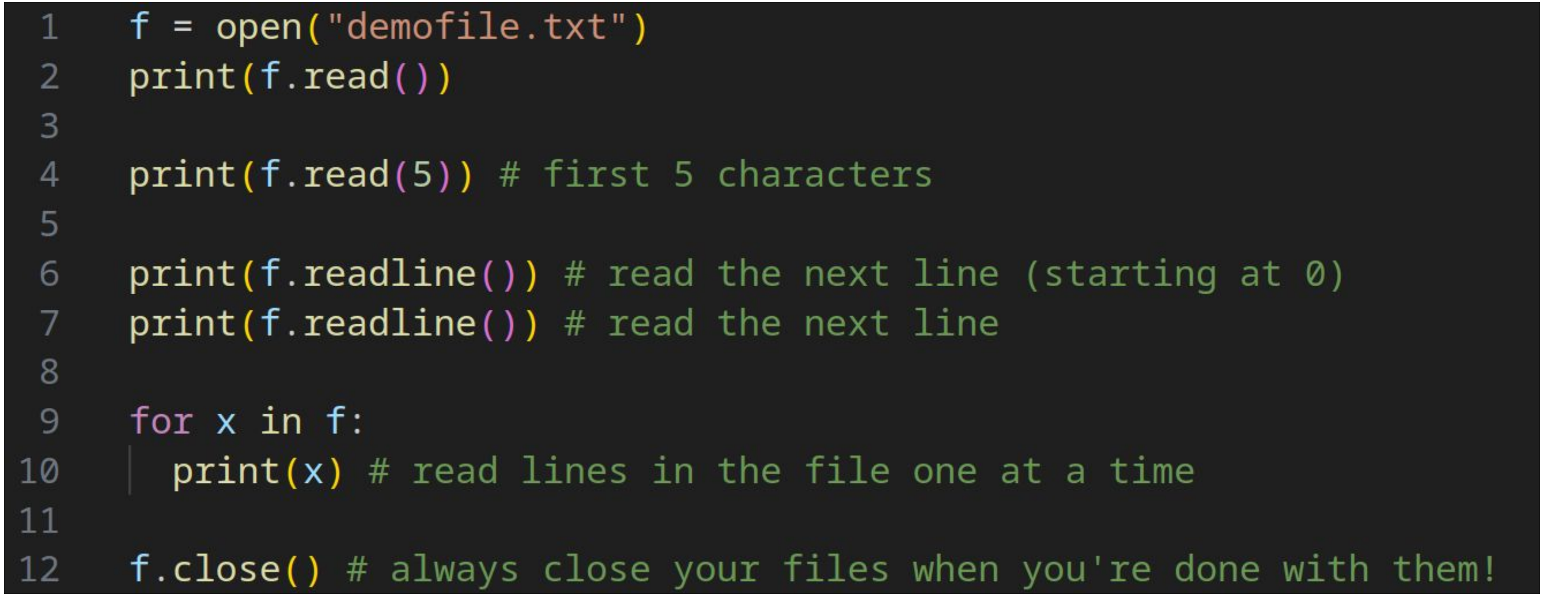


```
plt.plot(input_x, input_y) # plots a line graph
plt.plot(input_x, input_y, marker = 'x', linestyle = 'dotted') # you can customise this even further

plt.bar(input_x, input_y) # plots a bar graph (input_x used as the bar labels, input_y as bar heights)

plt.scatter(input_x, input_y) # plots a scatter graph

plt.hist(input_data) # plots a histogram (frequency graph)

plt.show() # this actually renders the graph
```



File handling

https://www.w3schools.com/python/python_file_handling.asp



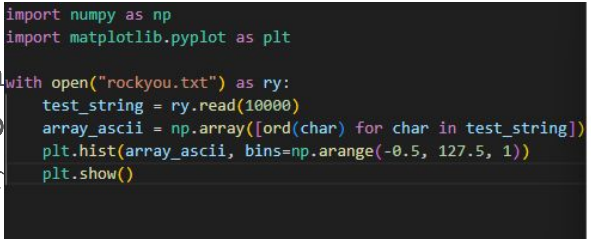
```python
f = open("demofile.txt")
print(f.read())

print(f.read(5)) # first 5 characters

print(f.readline()) # read the next line (starting at 0)
print(f.readline()) # read the next line

for x in f:
    print(x) # read lines in the file one at a time

f.close() # always close your files when you're done with them!
```

A simple program (Exercise)

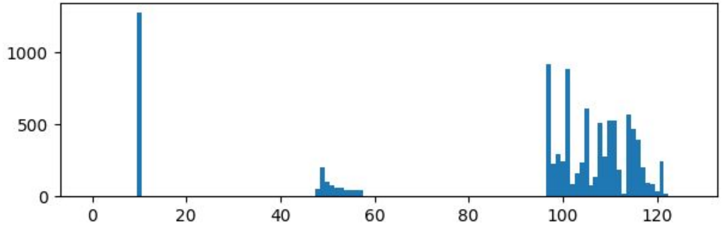
With everything we've just learned, here's a challenge:

Download the file rockyou.txt
Then create a program that reads in all its data (its ok to only use part of it) that plots a histogram (frequency graph) of all characters used, ordered by ASCII codes

Useful:
`ord(ch`
When p
`np.arr`



```
import numpy as np
import matplotlib.pyplot as plt

with open("rockyou.txt") as ry:
    test_string = ry.read(10000)
    array_ascii = np.array([ord(char) for char in test_string])
    plt.hist(array_ascii, bins=np.arange(-0.5, 127.5, 1))
    plt.show()
```



Bonus – if you're doing this in the Git repo you created earlier, commit these additions once you're done

SQLite

What is SQLite

SQLite is a software library that provides a relational database management system and is typically:

- Self-contained
  - This makes SQLite suitable in any environment as it requires minimal support from the OS or external libraries
- Serverless
  - In MySQL or PostgreSQL, a separate server is required for these to run
- Requires zero configuration
- Transactional
  - All actions will either take place completely, or not at all - even if the system crashes

Features unique to SQLite

- SQLite uses dynamic types for tables, so any value can be stored in any column even if it is different from the declared data type
- SQLite allows you to join tables in different databases
- Can create in-memory databases, which are extremely useful for prototyping or testing

When is best?

It is best to use SQLite when you need simplicity, speed and minimal resources, for example:

- Embedded apps
  - Very useful for apps that need to store data locally
- Local storage
  - When you need to store settings/preferences/cached data locally
- Cross-platform apps
- Prototyping and development
  - SQLite doesn’t need to be set up so useful in quick situations
- Internet of Things devices
  - Such as security devices, smart watches and point of sale services like PayPal

SQL Basics

The most important query in SQL is **SELECT**, as this is used to get and return data from a database.

You would use this command like this:

- **SELECT** column1, column2… **FROM** Table_Name;
- **SELECT** * **FROM** Table_Name;

You can filter records via a specified condition by using the **WHERE** query:

- **SELECT** * **FROM** Table_Name **WHERE** condition;
- **SELECT** * **FROM** Order_Table **WHERE** orderID = ‘1’;

SQL Basics

In order to modify existing records in a given table, you can use the query **UPDATE** like this:

- **UPDATE** Table_Name **SET** column1 = newValue1, column2 = newValue2… **WHERE** condition;
- **UPDATE** Order_Table **SET** orderName = ‘Person’ **WHERE** orderID = ‘1’;

Additionally, if we wanted to insert a new record into a table it can be done like this:

- **INSERT INTO** Table_Name (column1, column2…) **VALUES** (value1, value2…);

One thing to note: if you are adding values for every column in a table, you do not need to include (column1, column2…)

SQL Basics

If you wanted to delete a record from a table, you can use the query:

- **DELETE FROM** Table_Name **WHERE** condition;

It is extremely important that you include the **WHERE** query otherwise all records in the specified table will be deleted.

Before any of these queries can be acted upon, you will need to create a database using the following query:

- **CREATE DATABASE** Database_Name;

SQL Basics

Afterwards, you can create a new table using this query:

- **CREATE TABLE** Table_Name (column1 datatype, column2 datatype…);
- **CREATE TABLE** Order_Table (orderID int, orderName varchar(30));

What is varchar()? → This is how you define a string using SQL where the number inside the brackets is the maximum character length.

SQL Basics

Once you have created a table, you can alter it in 3 different ways:

- Adding a column
  - **ALTER TABLE** Table_Name **ADD** columnName datatype;
- Modifying a column
  - **ALTER TABLE** Table_Name **MODIFY COLUMN** columnName datatype;
- Dropping a column
  - **ALTER TABLE** Table_Name **DROP COLUMN** columnName;

SQL Basics

To Delete a Table you would simply just use this:

- **DROP TABLE** *name_of_table*

To delete a database you would use this:

- **DROP DATABASE** databaseName;

These are quite dangerous commands so must be used with caution

SQL Basics

Within SQL there are 2 types of keys:

- Primary
  - This is used to uniquely identify each record in a table and cannot be null
- Foreign
  - Used to identify when a column of one table refers to the primary key of another

Both of these are important to ensure data integrity - primary keys to identify records and foreign keys to ensure only valid primary key data is used

SQL Basics

In order to identify these keys, we can use the following query:

```
CREATE TABLE Table_Name (
	column1 datatype,
	column2 datatype…,
	PRIMARY KEY(columnName),
	CONSTRAINT constraintName FOREIGN KEY (columnName)
	REFERENCES Table_Name(primary_key_name)
);
```

SQL Basics

An example of this:

```
CREATE TABLE Order_Table (
	orderID int,
	orderName varchar(30),
	PRIMARY KEY(orderID),
	CONSTRAINT FK_Items FOREIGN KEY (itemID) REFERENCES
	Item_Table(itemID)
);
```

SQL Basics

Furthermore, you can use the **ALTER TABLE** query to add or remove keys:

- **ALTER TABLE** Table_Name **ADD CONSTRAINT** constraintName
  - **PRIMARY KEY** (columnName);
  - **FOREIGN KEY** (keyName) **REFERENCES** Table_Name(PK_Name);

- **ALTER TABLE** Table_Name **DROP**
  - **PRIMARY KEY**;
  - **FOREIGN KEY** (contraintName);

Database Security

Database Security

What are the main security issues with the database during the hackathon?

- Dodgy inputs
- Teammates
- Anything else?

Database Security

Sanitation:

- Unlikely to have malicious input
- Much more likely to be poorly formed

Solutions:

- ORM - will help make sure datatypes are correct
- SQLite - datatypes are more like “suggestions”
  - Can store Strings in Int columns

Database Security

Rogue teammates

- Unlikely to be malicious
- Delete tables, add bad data, break links between tables

Solutions

- SQLite - literally just a file, can keep a backup elsewhere
- Keep your SQL to create your table on hand!!! Don’t just use a CLI and hope nothing breaks

Flask

Flask

What is Flask?

- A web framework for Python
- Simple and lightweight

- Allows you a lot of control over exactly how it will function
- Packaged with a webserver so can be ran on your device (Only for use in testing)

Flask

The basics

Setup

Define routes

Return HTML

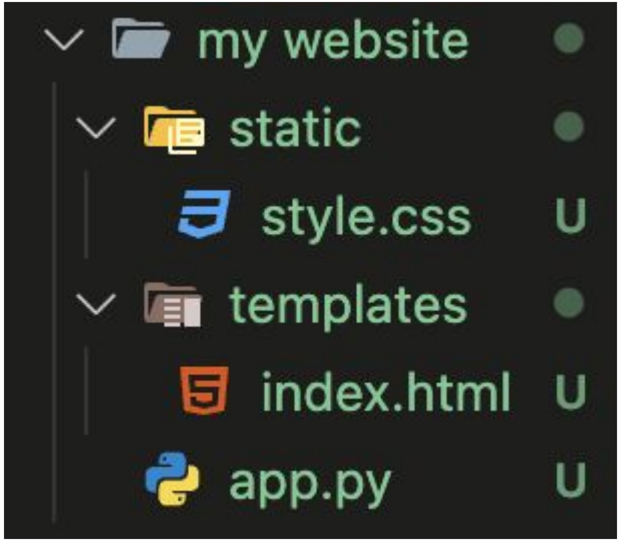
GET / POST requests

HTML templates with custom data

Blueprints

Flask - Setup

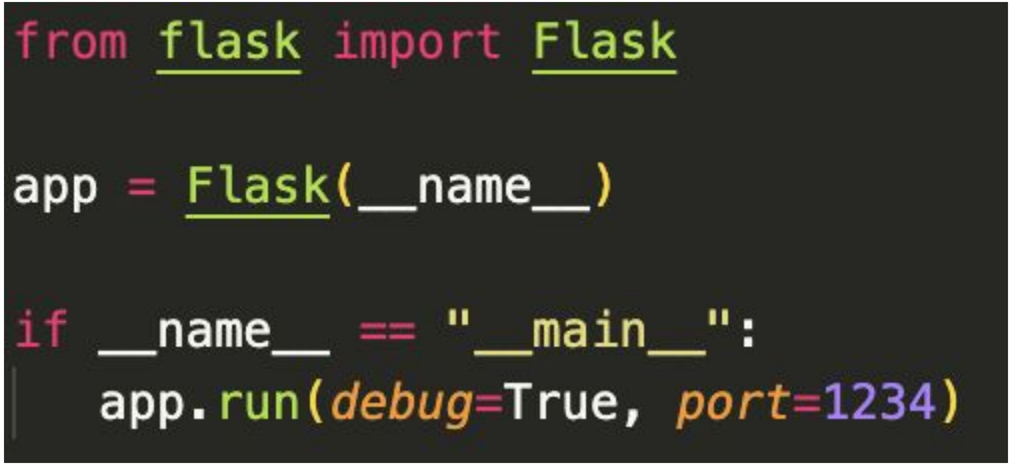
Basics folder setup



```
my website
  static
    style.css      U
  templates
    index.html     U
  app.py           U
```

Flask - Setup

Basics of a flask webapp:



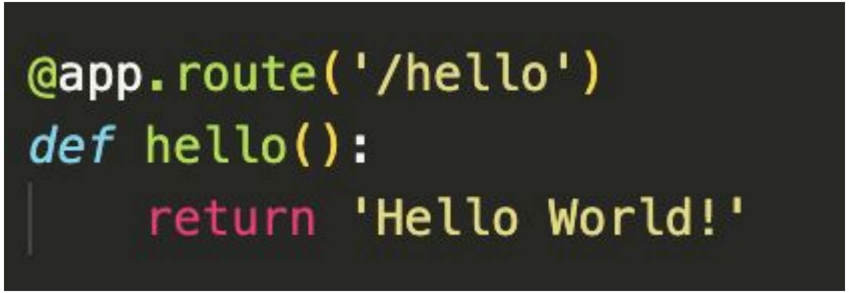
```python
from flask import Flask

app = Flask(__name__)

if __name__ == "__main__":
    app.run(debug=True, port=1234)
```

Flask - Routes

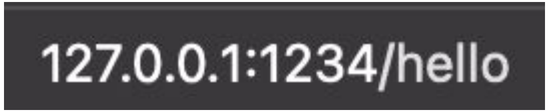
Routes

Basic part of any webapp

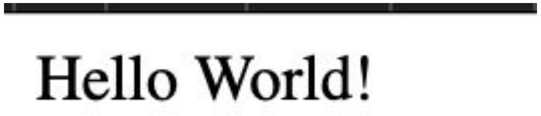
Links to different parts of the site



```
@app.route('/hello')
def hello():
    return 'Hello World!'
```



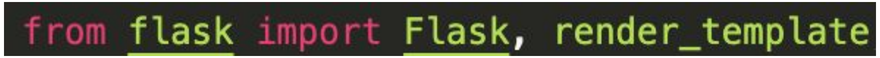
127.0.0.1:1234/hello



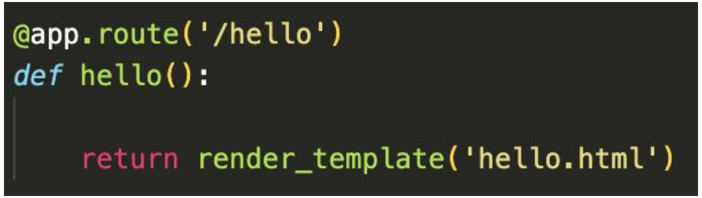
Hello World!

Flask - HTML

You can also return html pages so that your website looks nicer!
Requires that ‘render_template’ be imported from Flask

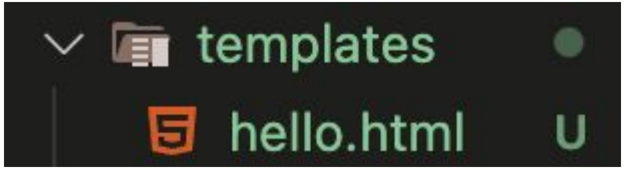


```
from flask import Flask, render_template
```

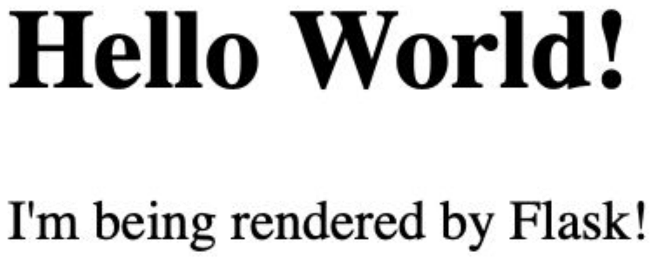


```
@app.route('/hello')
def hello():

    return render_template('hello.html')
```

All HTML files must be placed within a folder title ‘Templates’



```
∨ templates        ●
    hello.html     U
```



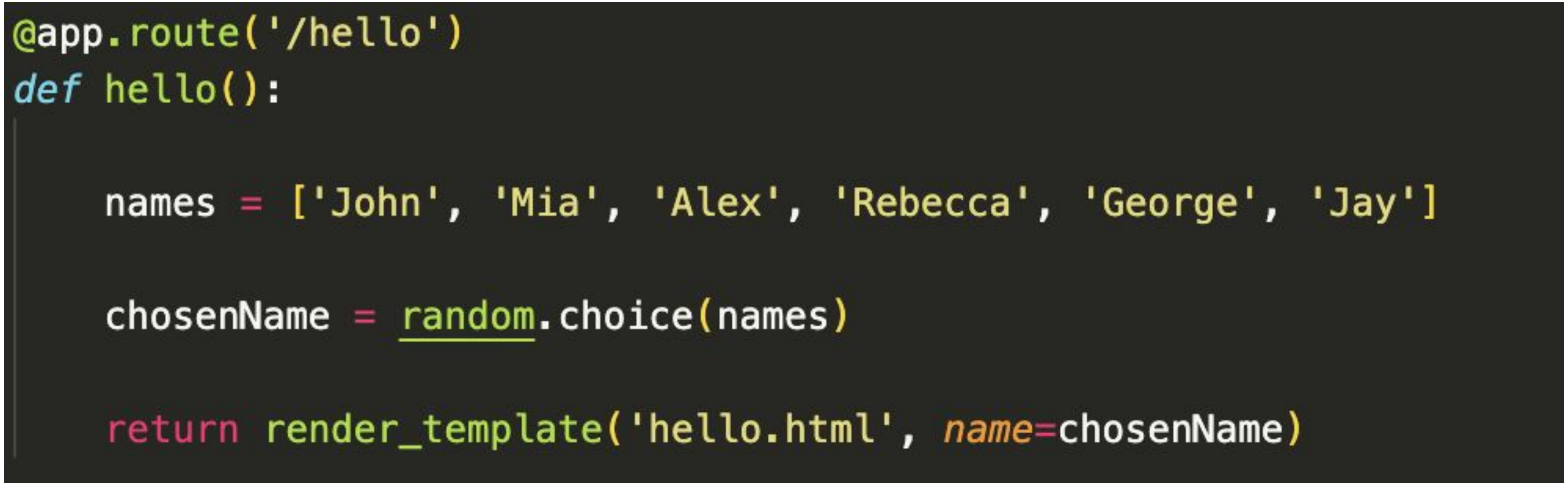
Hello World!

I'm being rendered by Flask!

Flask - HTML cont

We can now render HTML pages

But how do we send data from the server to the pages?

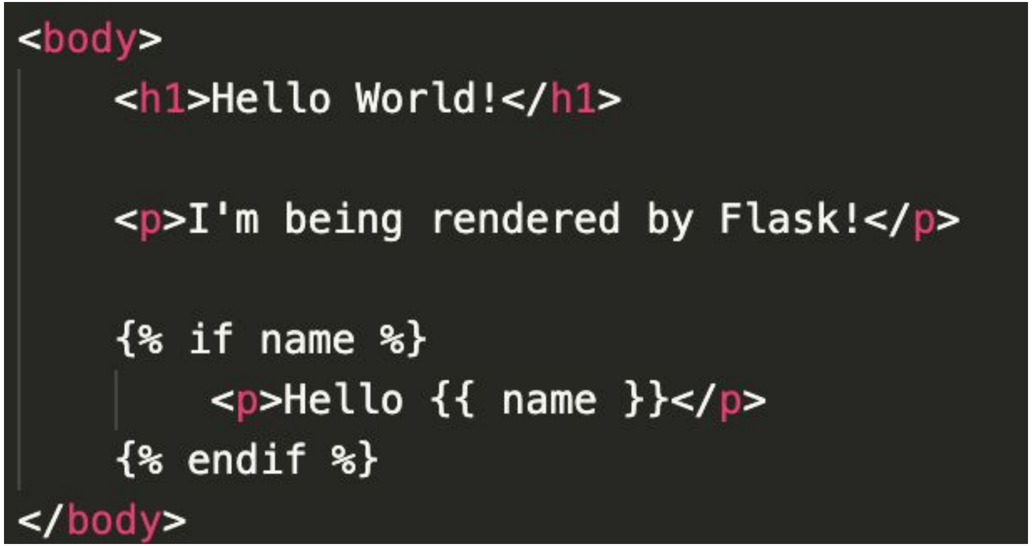


```
@app.route('/hello')
def hello():

    names = ['John', 'Mia', 'Alex', 'Rebecca', 'George', 'Jay']

    chosenName = random.choice(names)

    return render_template('hello.html', name=chosenName)
```

Flask - HTML cont

We can now render HTML pages

But how do we send data from the server to the pages?

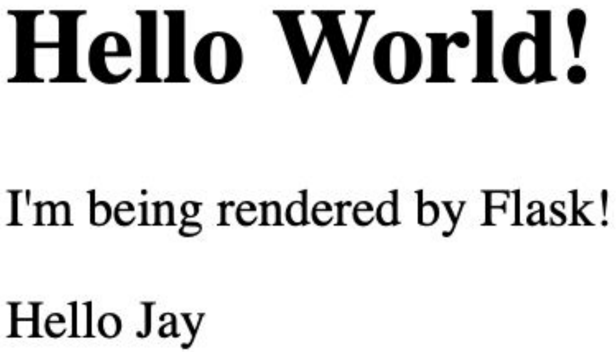


```
<body>
    <h1>Hello World!</h1>

    <p>I'm being rendered by Flask!</p>

    {% if name %}
        <p>Hello {{ name }}</p>
    {% endif %}
</body>
```

Flask - HTML cont

We can now render HTML pages

But how do we send data from the server to the pages?



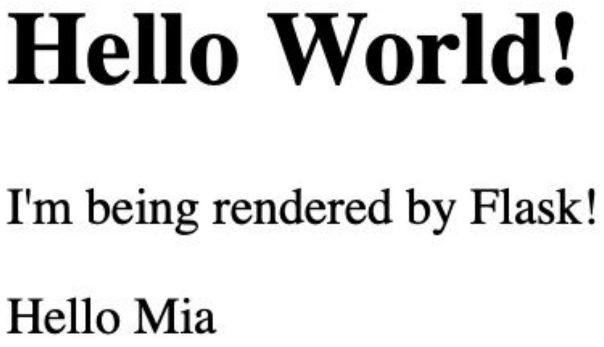
Hello World!

I'm being rendered by Flask!

Hello Jay

Flask - HTML cont

We can now render HTML pages
But how do we send data from the server to the pages?



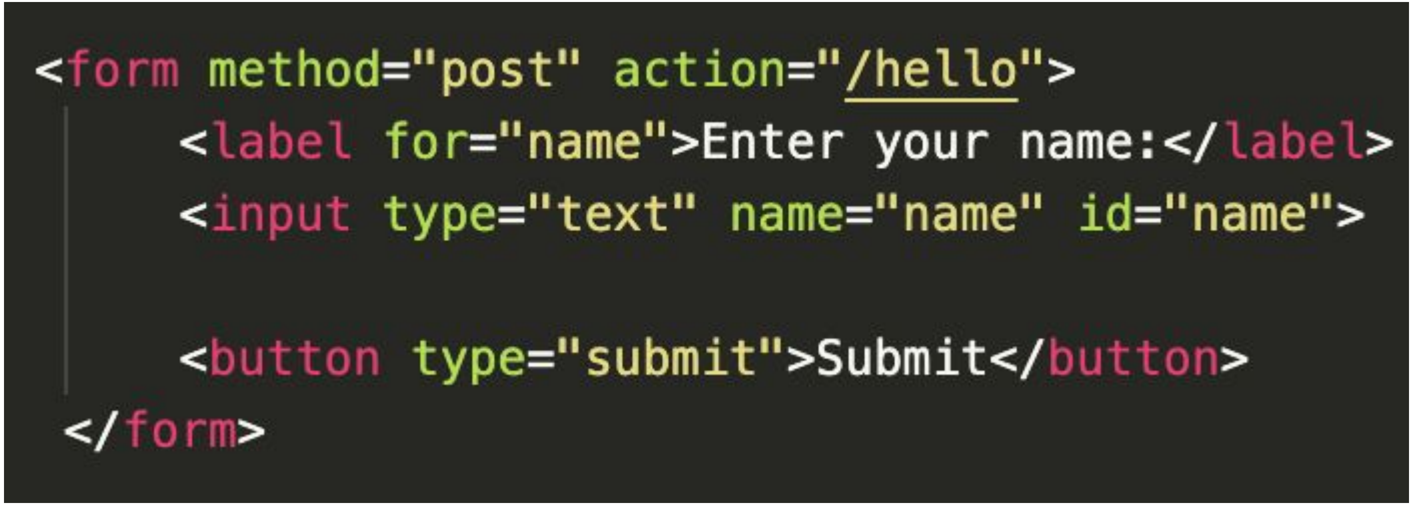
Hello World!

I'm being rendered by Flask!

Hello Mia

Flask - POST/GET

So far all we’ve done is get content - we need a way for users to send content to the server. First we need to make a form in our html



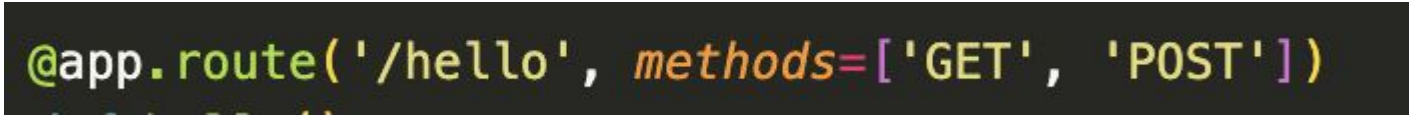
```
<form method="post" action="/hello">
    <label for="name">Enter your name:</label>
    <input type="text" name="name" id="name">

    <button type="submit">Submit</button>
 </form>
```

Flask - POST/GET

Then we need to specify the request methods our different routes can use.

View - GET

Send - POST

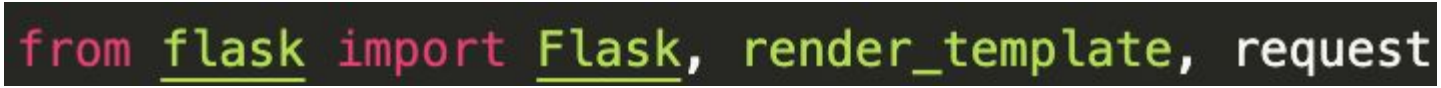


```
@app.route('/hello', methods=['GET', 'POST'])
```

Flask - POST/GET

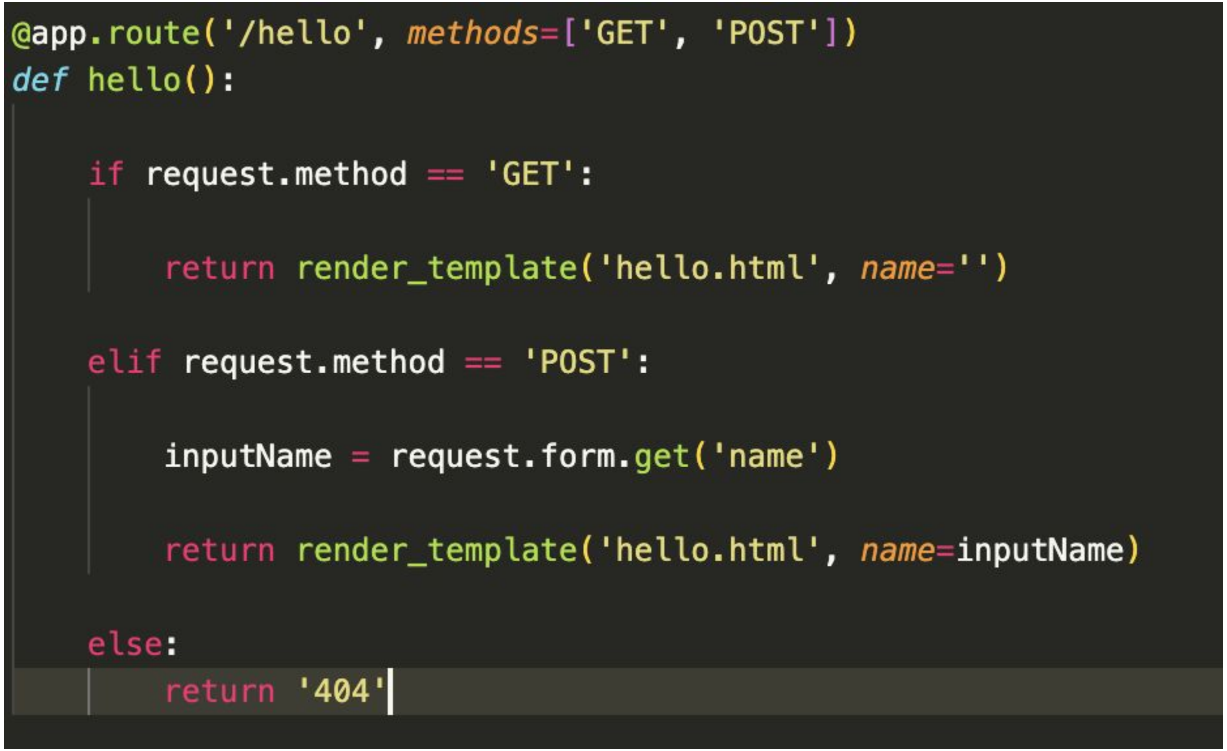
Now we need to handle these on the server

We need to import ‘request’ from flask



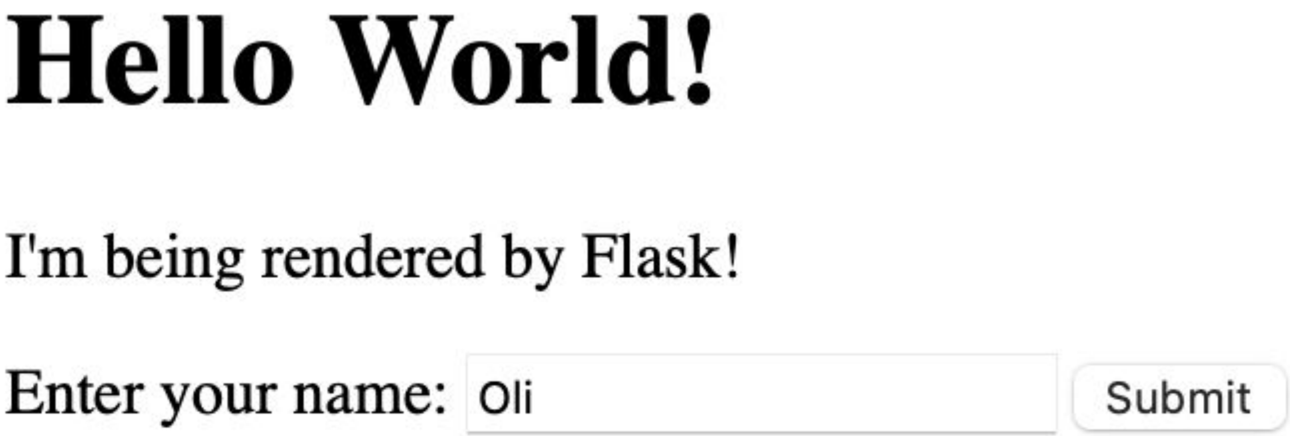
```
from flask import Flask, render_template, request
```

Flask - POST/GET



```
@app.route('/hello', methods=['GET', 'POST'])
def hello():

    if request.method == 'GET':

        return render_template('hello.html', name='')

    elif request.method == 'POST':

        inputName = request.form.get('name')

        return render_template('hello.html', name=inputName)

    else:
        return '404'
```

Flask - POST/GET

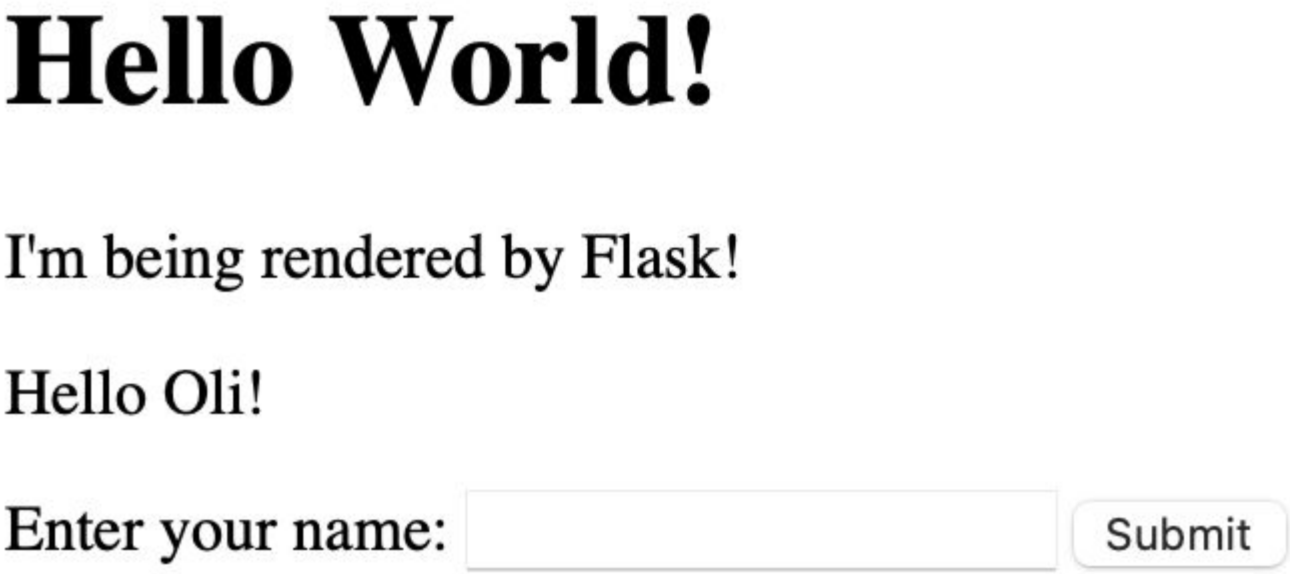


Hello World!

I'm being rendered by Flask!

Enter your name: Oli

Flask - POST/GET



Hello World!

I'm being rendered by Flask!

Hello Oli!

Enter your name:

Flask - Exercise

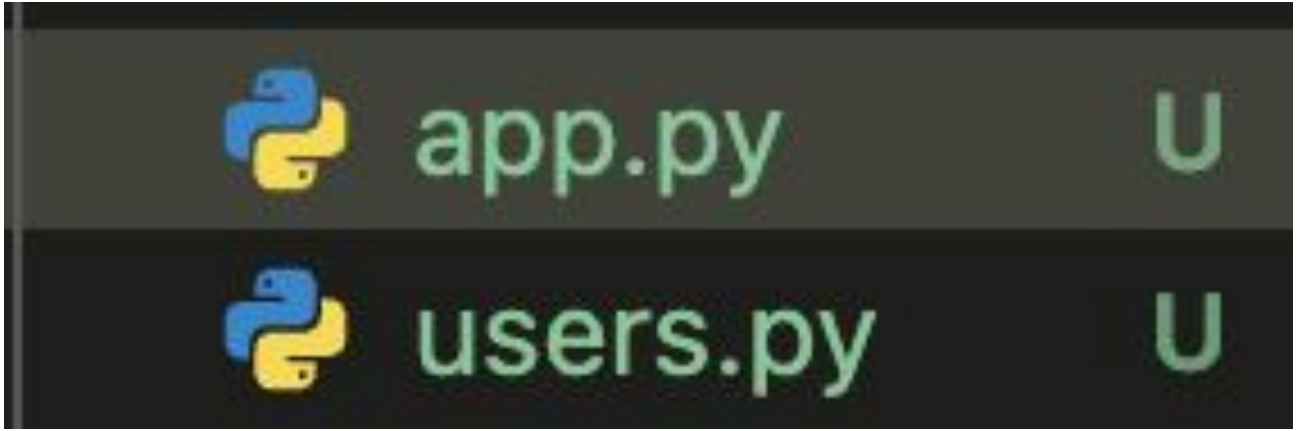
Try making a simple website that

- Takes the users name and age
- Returns a greeting
- Calculates how many days it will be until their next birthday and how old they will be turning

You will need to install and import ‘datetime’ module

Flask - Blueprints

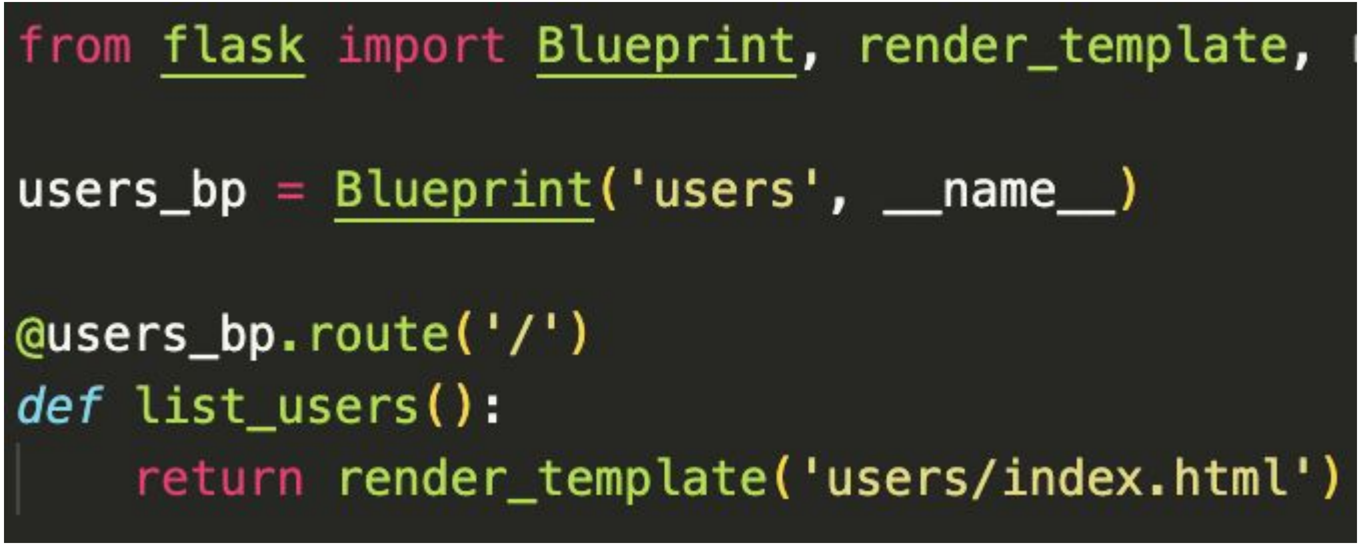
We can create blueprints to separate routes into better defined categories and keep our files much more organised



app.py U

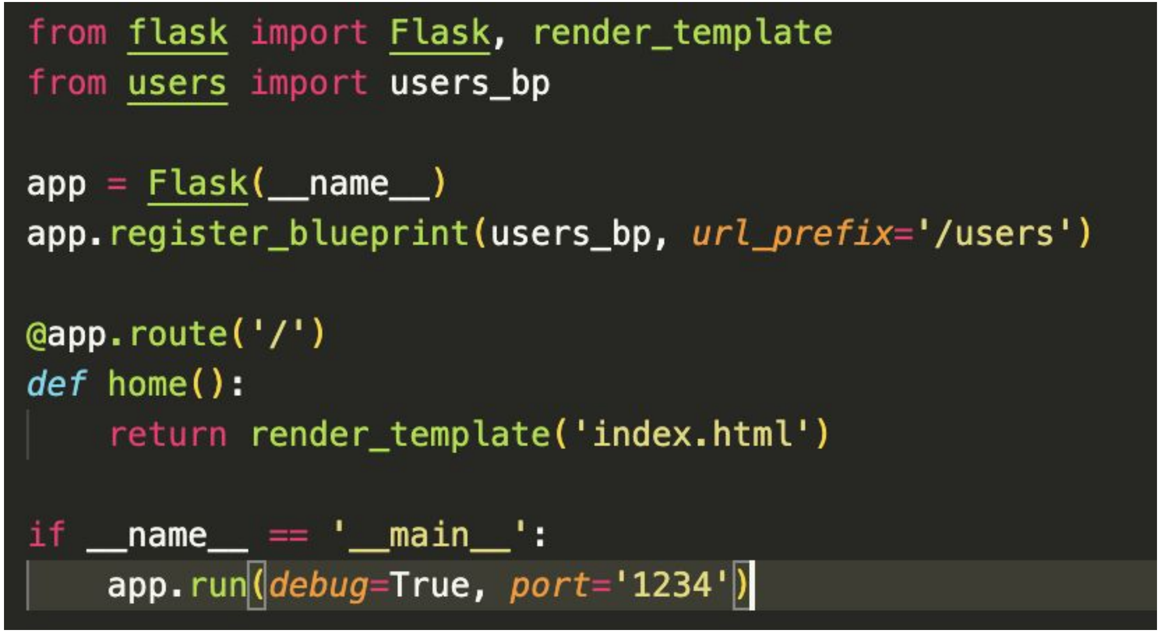
users.py U

Flask - Blueprints



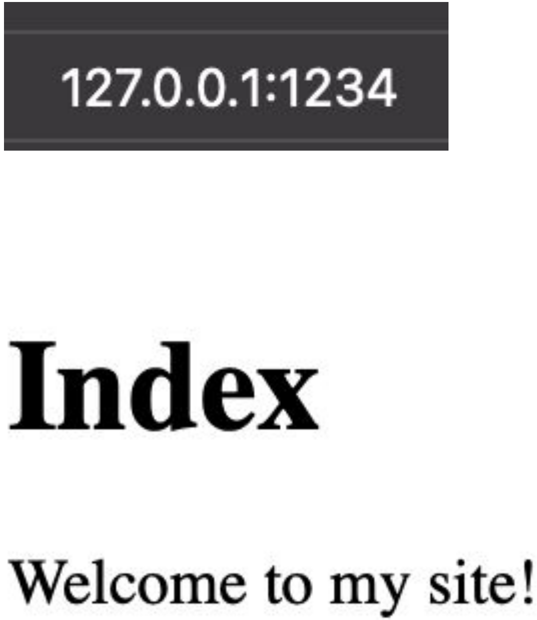
```
from flask import Blueprint, render_template,

users_bp = Blueprint('users', __name__)

@users_bp.route('/')
def list_users():
    return render_template('users/index.html')
```

Flask - Blueprints



```python
from flask import Flask, render_template
from users import users_bp

app = Flask(__name__)
app.register_blueprint(users_bp, url_prefix='/users')

@app.route('/')
def home():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True, port='1234')
```

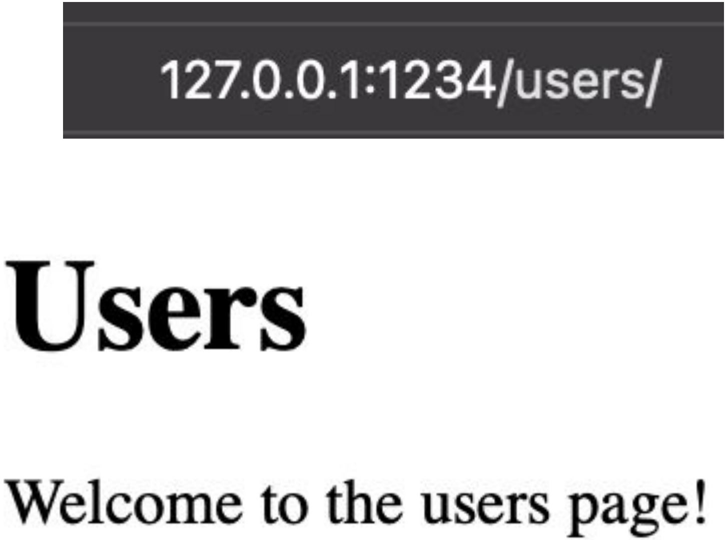
Flask - Blueprints



127.0.0.1:1234

Index

Welcome to my site!



127.0.0.1:1234/users/

Users

Welcome to the users page!

Flask - Wrapup

Those are the basics of Flask
In bigger pr

Flask has other cool features built in that you can read more about on their website: https://flask.palletsprojects.com/

Anything is possible in Flask!

Thank you for attending!

We look forward to seeing you at the Hackathon!