Product Analytics

DIANE WOODBRIDGE, PH.D

Review

Course Overview

Trello

Virtual Environment

Git Branch

Auto Documentation (Sphinx)

Unit Test

Crontab

Docker

Elastic Beanstalk

Create a script

Contents

Backend and Frontend

Backend and Data Acquisition

- API
- wget
- Crawl
- Collect data from users
  - Create file upload form using Python Flask and WTF.

Things to do before start

1. Pull today's example
2. Update an existing environment
   - `pip install -r requirements.txt`

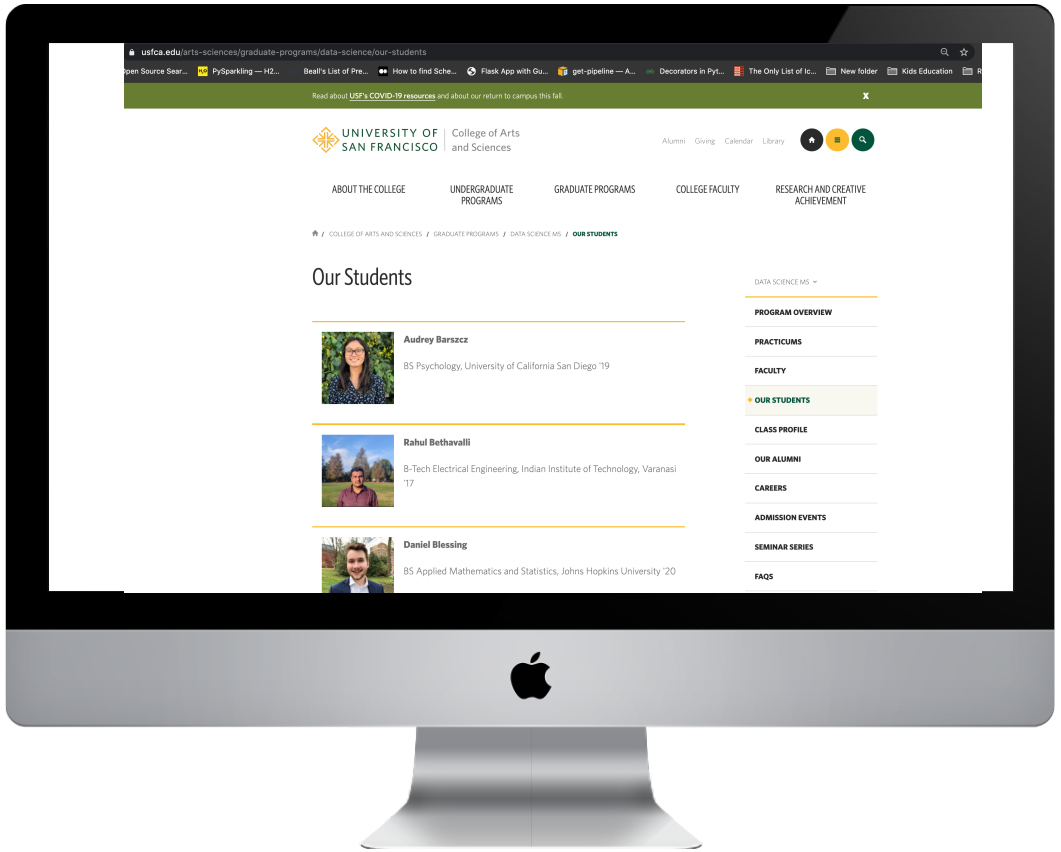
Contents

Backend and Frontend

Backend and Data Acquisition

- API
- wget
- Crawl
- Collect data from users
  - Create file upload form using Python Flask and WTF.

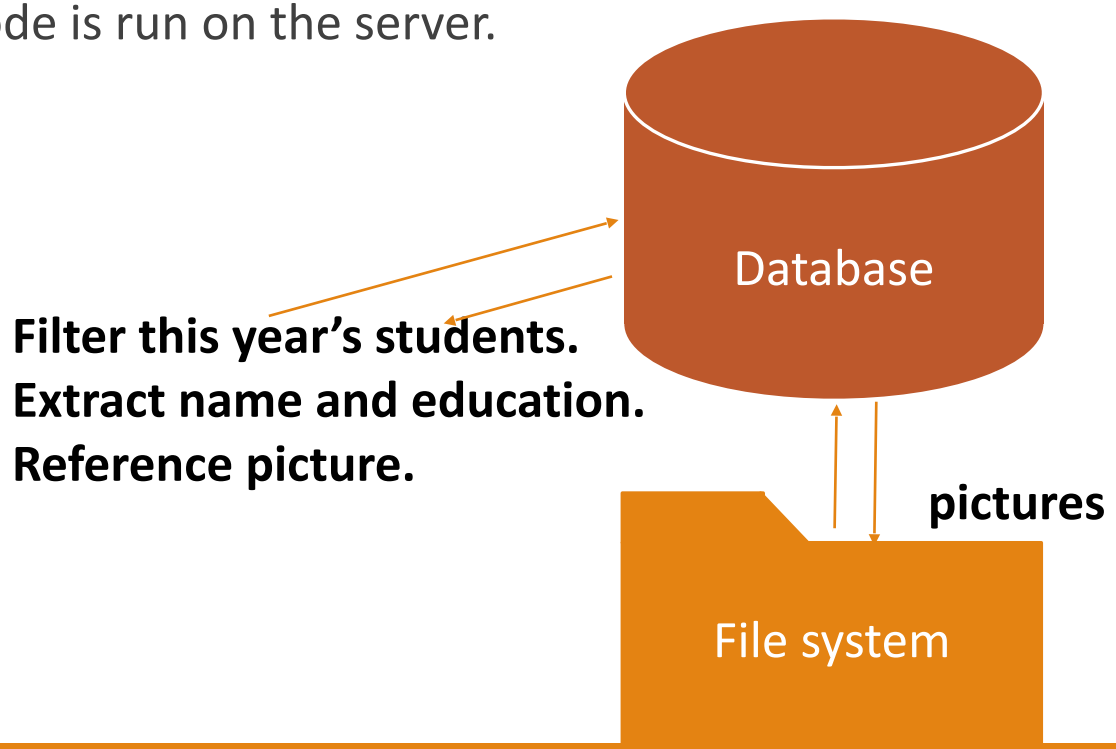
Backend and Frontend



Backend Development

Definition

- The backend of an application is responsible business logic, data acquisition, database interaction, data processing, etc.
  - Backend code is run on the server.

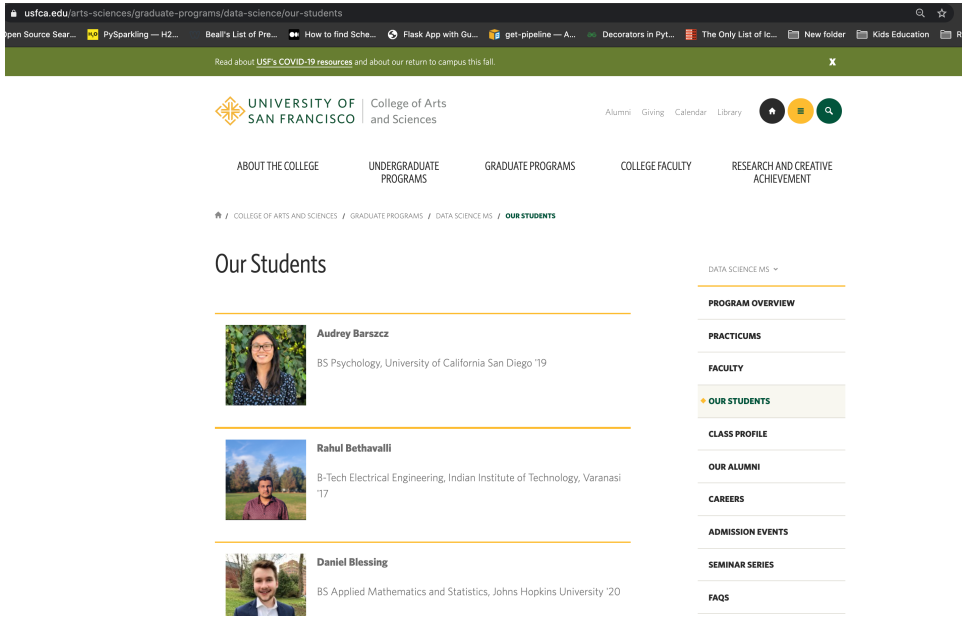


Frontend Development

Definition

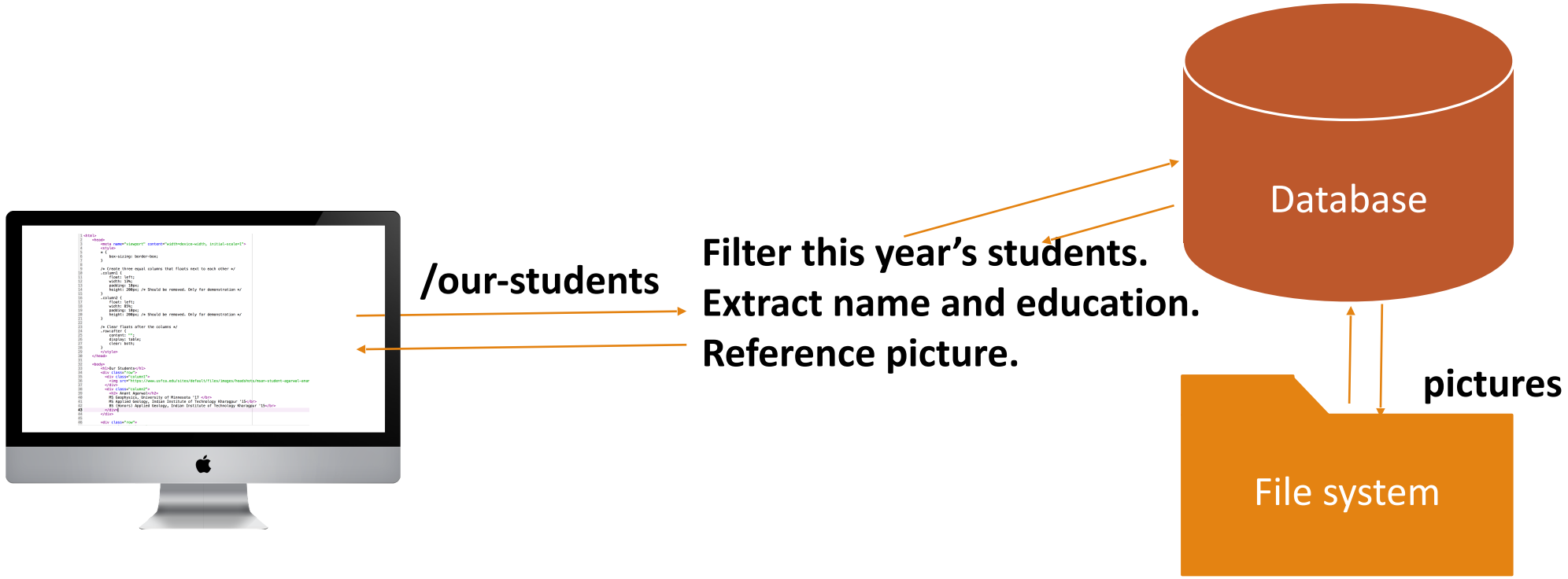
- Using HTML, CSS and JavaScript for users to see and interact with information.
  - A web browser translate the code and display it.

```
<div class="pane-content">

  <div class="field field-name-body field-type-text-with-summary field-label-hidden typography">
    <div class="block_left"><img alt="Akanksha ." class="media-element file-default" data-delta="110" src="https://www.usfca.edu/sites/default/files/styles/rte_150x
height="150" /></div>
<p><strong>Akanksha .</strong></p>
<p>BEng Information Technology, Birla Institute of Technology, Mesra '11</p>
<hr /><div class="block_left"><img alt="Al Abdel Latif" class="media-element file-default" data-delta="136" src="https://www.usfca.edu/sites/default/files/styles/rt
width="150" height="150" /></div>
<p><strong>Al Abdel Latif</strong></p>
<p>BEng Biochemical Engineering, University College London '15 <br />MPhil Scientific Computing, University of Cambridge '16</p>
<hr /><div class="block_left"><img alt="Kamron Afshar" class="media-element file-default" data-delta="122" src="https://www.usfca.edu/sites/default/files/styles/rte
width="150" height="150" /></div>
<p><strong>Kamron Afshar</strong></p>
<p>BS Business Administration, University of Southern California '15</p>
<hr /><div class="block_left"><span><img alt="Zachary Barnes" class="media-element file-default" data-delta="91" src="https://www.usfca.edu/sites/default/files/styl
itok=YTRRzaiJ" width="150" height="150" /></span></div>
<p><strong>Zachary Barnes</strong></p>
<p>BA Computer Science &amp;amp; Music, Knox College '19</p>
<hr /><div class="block_left"><img alt="Shreejaya Bharathan" class="media-element file-default" data-delta="111" src="https://www.usfca.edu/sites/default/files/styl
itok=zIV8abFr" width="150" height="150" /></div>
<p><strong>Shreejaya Bharathan</strong></p>
<p>BEng Civil Engineering, National Institute of Technology, Tiruchirappalli '17</p>
<hr /><div class="block_left"><img alt="Nithish Kumar" class="media-element file-default" data-delta="112" src="https://www.usfca.edu/sites/default/files/styles/rte
width="150" height="150" /></div>
<p><strong>Nithish Bolleddula</strong></p>
<p>BEng Metallurgical Engineering, Indian Institute of Technology (IIT-BHU), Varanasi '17</p>
<hr /><div class="block_left"><img alt="Shane Buchanan" class="media-element file-default" data-delta="92" src="https://www.usfca.edu/sites/default/files/styles/rte
width="150" height="150" /></div>
<p><strong>Shane Buchanan</strong></p>
<p>BA Physics, Colgate University '17</p>
<hr /><div class="block_left"><img alt="Maxwell Calehuff" class="media-element file-default" data-delta="93" src="https://www.usfca.edu/sites/default/files/styles/r
width="150" height="150" /></div>
<p><strong>Max Calehuff</strong></p>
<p>BA Economics, University of California Santa Barbara '13<br /> Juris Doctorate, University of California Davis '17</p>
<hr /><div class="block_left"><img alt="Min Che" class="media-element file-default" data-delta="121" src="https://www.usfca.edu/sites/default/files/styles/rte_150xl
height="150" /></div>
<p><strong>Min Che</strong></p>
<p>BPh Logic, Sun Yat-sen University '12<br /> MS Marketing, Sun Yat-Sen University '14</p>
<hr /><div class="block_left"><img alt="Jiaqi Chen" class="media-element file-default" data-delta="105" src="https://www.usfca.edu/sites/default/files/styles/rte_15
height="150" /></div>
<p><strong>Jiaqi Chen</strong></p>
<p>BBA Accounting, Xiamen University '19</p>
<hr /><div class="block_left"><img alt="Sihan Chen" class="media-element file-default" data-delta="123" src="https://www.usfca.edu/sites/default/files/styles/rte_15
height="150" /></div>
<p><strong>Sihan Chen</strong></p>
<p>BS Physics, Huazhong University of Science and Technology '19</p>
<hr /><div class="block_left"><img alt="Andy Cheon" class="media-element file-default" data-delta="124" src="https://www.usfca.edu/sites/default/files/styles/rte_15
height="150" /></div>
<p><strong>Andy Cheon</strong></p>
<p>BS Bioengineering, University of California Berkeley '15</p>
```



Full Stack Development

Frontend + Backend



Full Stack Development

Frontend + Backend

Contents

Backend and Frontend

Backend and Data Acquisition

- API
- wget
- Crawl
- Collect data from users
  - Create file upload form using Python Flask and WTF.

Data Acquisition

API

wget

Crawl

Collect data from users

Please review/reference Terence's class : https://github.com/parrt/msds692

Contents

Backend and Frontend

Backend and Data Acquisition

- **API**
- wget
- Crawl
- Collect data from users
  - Create file upload form using Python Flask and WTF.

Data Acquisition

API

- Allows to obtain data and uses functions from other businesses/developers.
- Need to register as a user and access with API key.
- May not return all the data.
- May have limits and quota on rates
- Examples
  - Google Map : Maps, Routes and Places APIs
    - URL : https://cloud.google.com/maps-platform/
  - Twitter : Tweets
    - URL : https://developer.twitter.com/
  - Walmart Open API : Product catalog
    - URL : https://developer.walmartlabs.com/
  - Yelp : Content and data from businesses
    - URL : https://www.yelp.com/fusion

Data Acquisition

API

- Example - data_Acquisition.ipynb

```
orig_coord = '37.7909,-122.3925'
dest_coord = '37.7765,-122.4506'
output_file_name = 'output.txt'

output_file = open(output_file_name, "a")
# https://developers.google.com/maps/documentation/javascript/get-api-key
apikey = '[redacted]'
url = "https://maps.googleapis.com/maps/api/distancematrix/json?key={0}&origins={1}&destinations={2}&mode=driving&depar
    str(apikey), str(orig_coord), str(dest_coord))
result = simplejson.load(urllib.request.urlopen(url))
driving_time = result['rows'][0]['elements'][0]['duration_in_traffic']['text']

output_file.write(str(datetime.datetime.now()) + "\n")
output_file.write(result['origin_addresses'][0] + "\n")
output_file.write(result['destination_addresses'][0] + "\n")
output_file.write(driving_time + "\n\n")

output_file.close()
```

```
https://maps.googleapis.com/maps/api/distancematrix/json?key=[redacted]&origins=37.7909,
-122.3925&destinations=37.7765,-122.4506&mode=driving&departure_time=now&language=en-EN&sensor=false
```

Contents

Backend and Frontend

Backend and Data Acquisition

- API
- **wget**
- Crawl
- Collect data from users
  - Create file upload form using Python Flask and WTF.

Data Acquisition

wget

- Download a file from a website or webserver usnig URL.
- Examples
  - Data.gov : U.S. Government's open data
    - URL : https://www.data.gov/
  - UCI Machine Learning Repository : Various data set for machine learning
    - URL : https://archive.ics.uci.edu
  - PhysioBank : Physiological data
    - URL : https://physionet.org/physiobank/database/

Data Acquisition

wget

- Example - data_Acquisition.ipynb

```
(ProductAnalytics_Env) (base) ML-ITS-901885:2020_MSDS603 dwoodbridge$ pip install xlrd
```

```
import wget
import xlrd # Extract data from Excel spreadsheets

# Change in Food Price Index Forecast
url = "https://www.ers.usda.gov/webdocs/DataFiles/50673/CPIforecast.xlsx?v=7651"
filename = wget.download(url)
xl_workbook = xlrd.open_workbook(filename)
xl_sheet = xl_workbook.sheet_by_index(0)
num_cols = xl_sheet.ncols
for row_idx in range(0, xl_sheet.nrows): # Iterate through rows
    for col_idx in range(0, num_cols):  # Iterate through columns
        cell_obj = xl_sheet.cell(row_idx, col_idx)  # Retrieve the cell value.
        print(cell_obj)
```

```
text:'Changes in Food Price Indexes, 2017 through 2019'
empty:''
empty:''
empty:''
```

Contents

Backend and Frontend

Backend and Data Acquisition

- API
- wget
- **Crawl**
- Collect data from users
  - Create file upload form using Python Flask and WTF.

Data Acquisition

Crawl

- Create a bot that systematically browses webpages and stores contents.
- Requires web browser driver (ex. Chrome Driver), selenium, etc.

```
pip install selenium
```

```
(ProductAnalytics_Env) (base) ML-ITS-901885:Week2 dwoodbridge$ pip install selenium
Collecting selenium
  Downloading selenium-3.141.0-py2.py3-none-any.whl (904 kB)
     |████████████████████████████████| 904 kB 775 kB/s
Requirement already satisfied: urllib3 in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from selenium) (1.25.8)
Installing collected packages: selenium
Successfully installed selenium-3.141.0
(ProductAnalytics_Env) (base) ML-ITS-901885:Week2 dwoodbridge$ jupyter-notebook
[I 19:49:07.438 NotebookApp] Serving notebooks from local directory: /Users/dwoodbridge/Class/2020_MSDS603/Example/Week2
[I 19:49:07.438 NotebookApp] The Jupyter Notebook is running at:
[I 19:49:07.439 NotebookApp] http://localhost:8888/?token=8772c5953034cbdf5bf9ee6e42826ff9fc79dac61b461bf3
[I 19:49:07.439 NotebookApp]  or http://127.0.0.1:8888/?token=8772c5953034cbdf5bf9ee6e42826ff9fc79dac61b461bf3
[I 19:49:07.439 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:49:07.451 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///Users/dwoodbridge/Library/Jupyter/runtime/nbserver-13647-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=8772c5953034cbdf5bf9ee6e42826ff9fc79dac61b461bf3
     or http://127.0.0.1:8888/?token=8772c5953034cbdf5bf9ee6e42826ff9fc79dac61b461bf3
```

Data Acquisition

Crawl

- Example - data_Acquisition.ipynb

```
from selenium import webdriver
url = "https://www.usfca.edu/arts-sciences/graduate-programs/data-science/our-students"
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--headless")

# you can download driver from http://chromedriver.chromium.org/downloads
browser = webdriver.Chrome("./chromedriver_mac", options=chrome_options)

browser.get(url)
print(browser.page_source)
```

```
<!DOCTYPE html><!--[if lt IE 9 ]>    <html class="lt-ie9 no-js"  lang="en" dir="ltr"> <![endif]--><!--[if gte IE 9
><!--><html xmlns="http://www.w3.org/1999/xhtml" class="js flexbox flexboxlegacy canvas canvastext webgl no-touch
eolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla multiplebgs ba
kgroundsize borderimage borderradius boxshadow textshadow opacity cssanimations csscolumns cssgradients cssreflect
ons csstransforms csstransforms3d csstransitions fontface generatedcontent video audio localstorage sessionstorage
webworkers applicationcache svg inlinesvg smil svgclippaths js no-touch boxshadow opacity cssgradients csstransfor
s video svg" lang="en" dir="ltr" style=""><!--<![endif]--><head>
    <meta charset="utf-8" />
```

Contents

Backend and Frontend

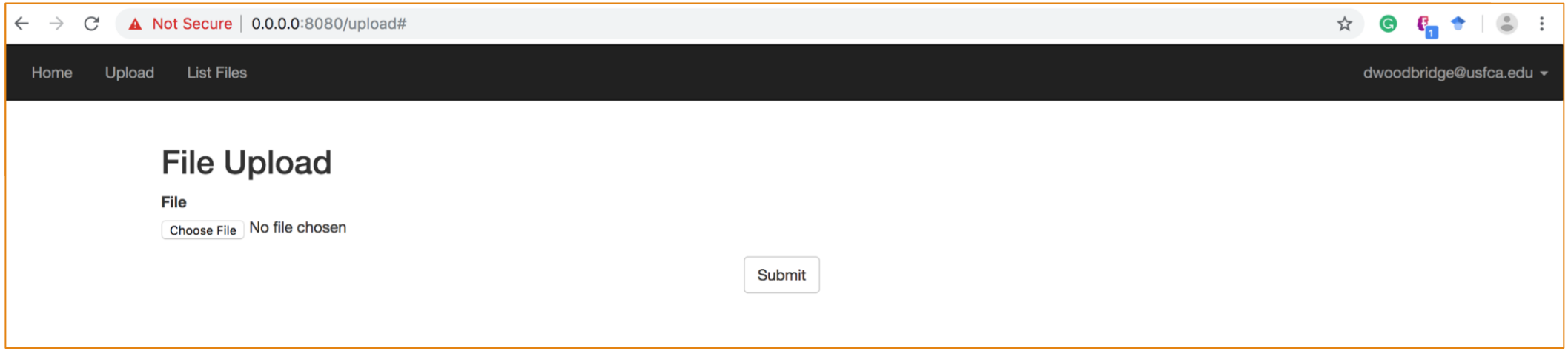
Backend and Data Acquisition

- API
- wget
- Crawl
- **Collect data from users**
  - **Create file upload form using Python Flask and WTF.**

Data Acquisition

Collect data from users

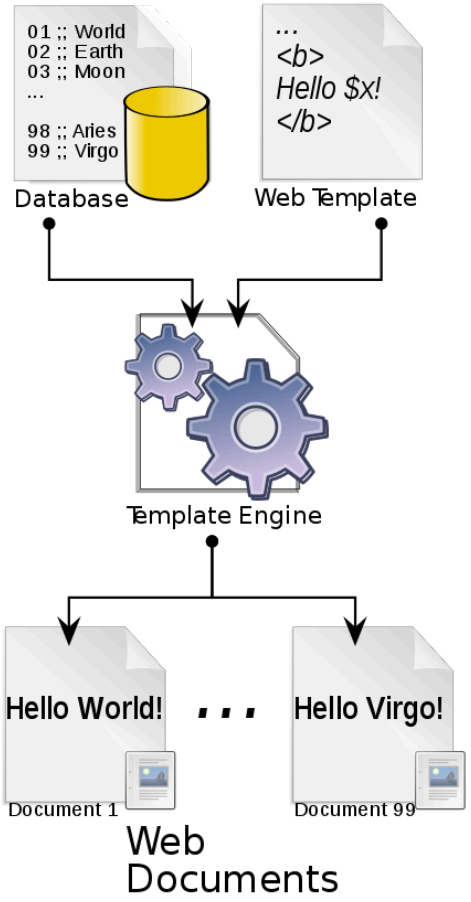
- Survey answers : Using button, checkbox, text entry, etc.
- File upload: Upload files from the user's machine.



Python Flask

A web framework for Python based on Werkzeug, Jinja 2.

- Support the development of web applications.
- Werkzeug : Web Server Gateway Interface (WSGI, interface between web servers and web applications) including routing, debugging, request and response objects, HTTP utilities, cookie handling, etc.
- Jinja 2 : Web template engine support for the automatic generation of custom web pages.



Recap.. Terence covered this ☺ https://github.com/parrt/msan692/blob/master/notes/flask.md

Python Flask

Installation

```
pip install flask
```

```
(ProductAnalytics_Env) (base) ML-ITS-901885:2020_MSDS603 dwoodbridge$ pip install flask
Collecting flask
  Using cached Flask-1.1.1-py2.py3-none-any.whl (94 kB)
Collecting Jinja2>=2.10.1
  Using cached Jinja2-2.11.1-py2.py3-none-any.whl (126 kB)
Collecting Werkzeug>=0.15
  Using cached Werkzeug-1.0.0-py2.py3-none-any.whl (298 kB)
Collecting click>=5.1
  Downloading click-7.1.1-py2.py3-none-any.whl (82 kB)
     |████████████████████████████████| 82 kB 952 kB/s
Collecting itsdangerous>=0.24
  Using cached itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting MarkupSafe>=0.23
  Using cached MarkupSafe-1.1.1-cp37-cp37m-macosx_10_6_intel.whl (18 kB)
Installing collected packages: MarkupSafe, Jinja2, Werkzeug, click, itsdangerous, flask
Successfully installed Jinja2-2.11.1 MarkupSafe-1.1.1 Werkzeug-1.0.0 click-7.1.1 flask-1.1.1 itsdangerous-1.1.0
(ProductAnalytics_Env) (base) ML-ITS-901885:2020_MSDS603 dwoodbridge$ pip install sqlalchemy
```

Python Flask

Example File Structure

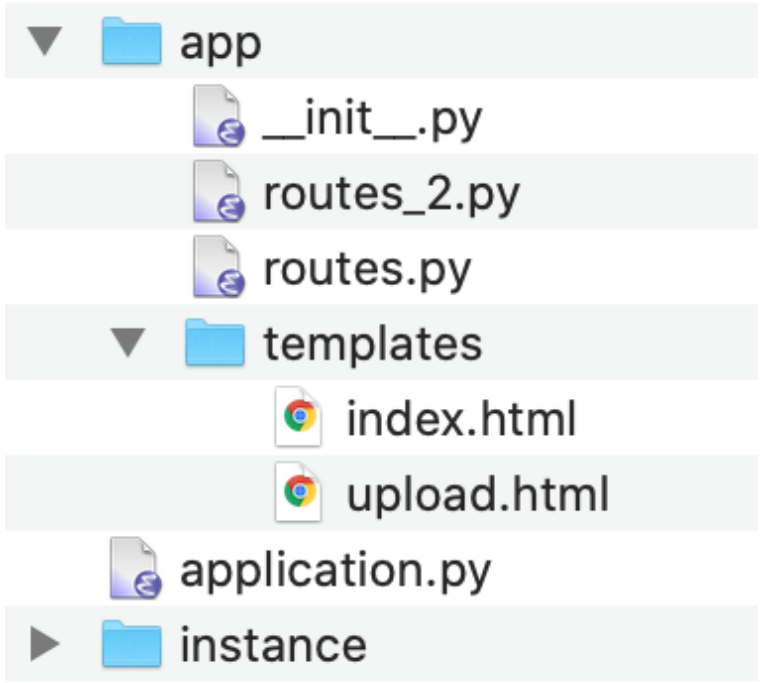
app/ – __init__.py

– routes.py

– templates/

instance/

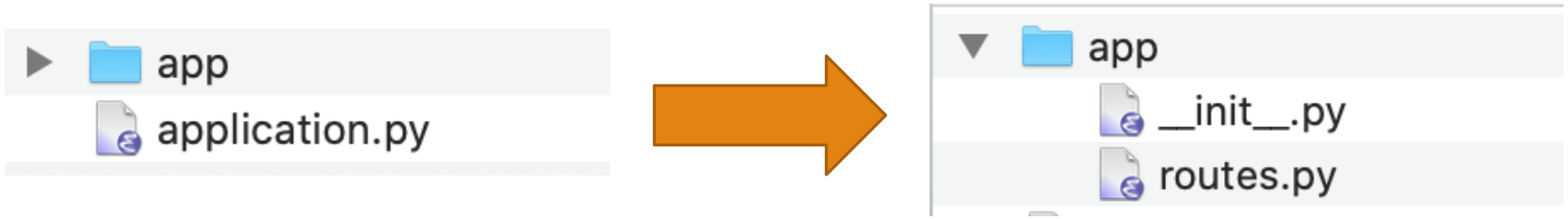
application.py



Python Flask

Initialization

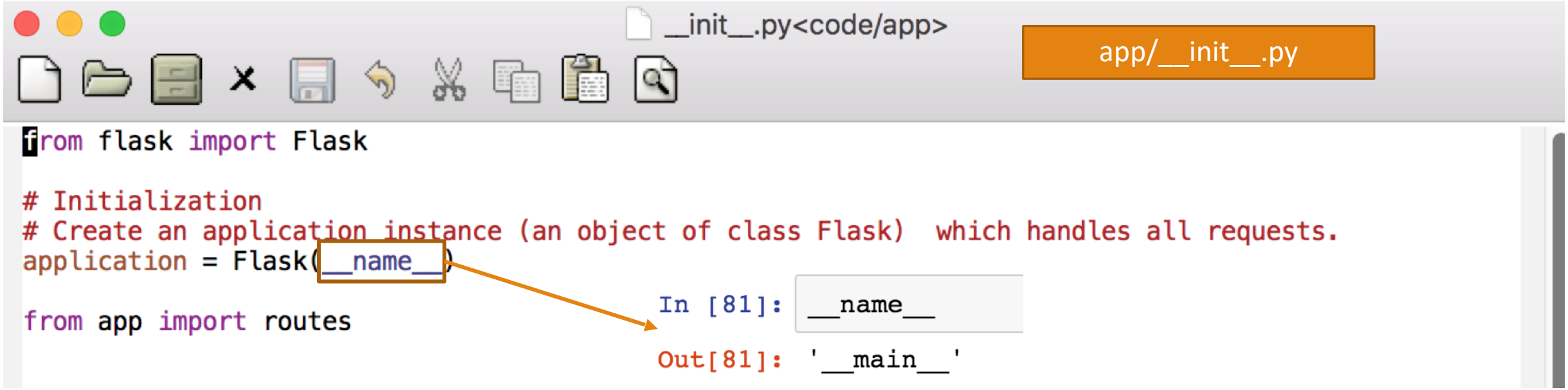
- An application instance needs to be initialized for passing all requests from clients.
  - The application instance is an object of Flask.
    - For Flask class constructor, it requires the name of the main module.



Python Flask

Initialization

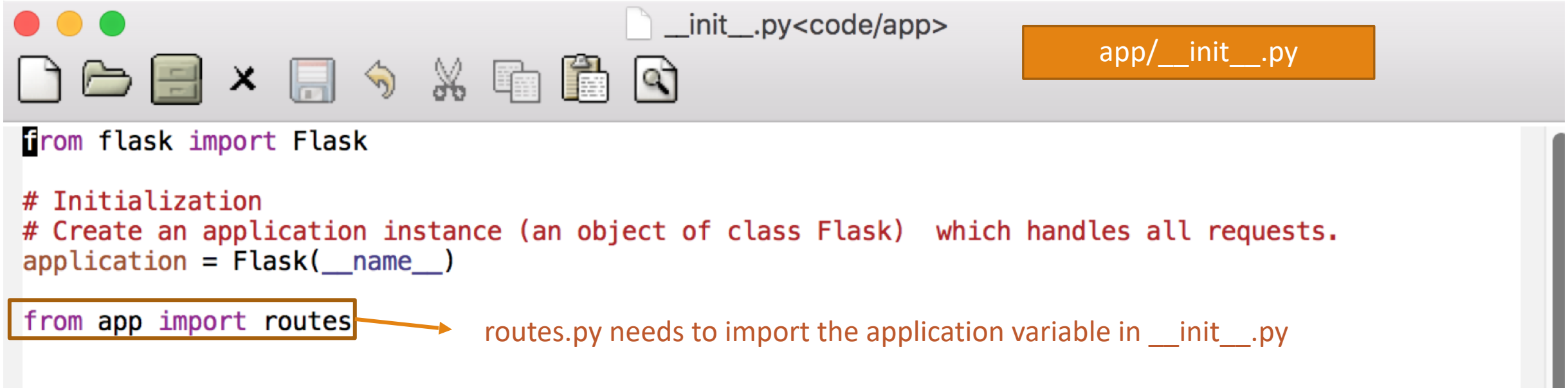
- An application instance needs to be initialized for passing all requests from clients.
  - The application instance is an object of Flask.
    - For Flask class constructor, it requires the name of the main module.



Python Flask

Initialization

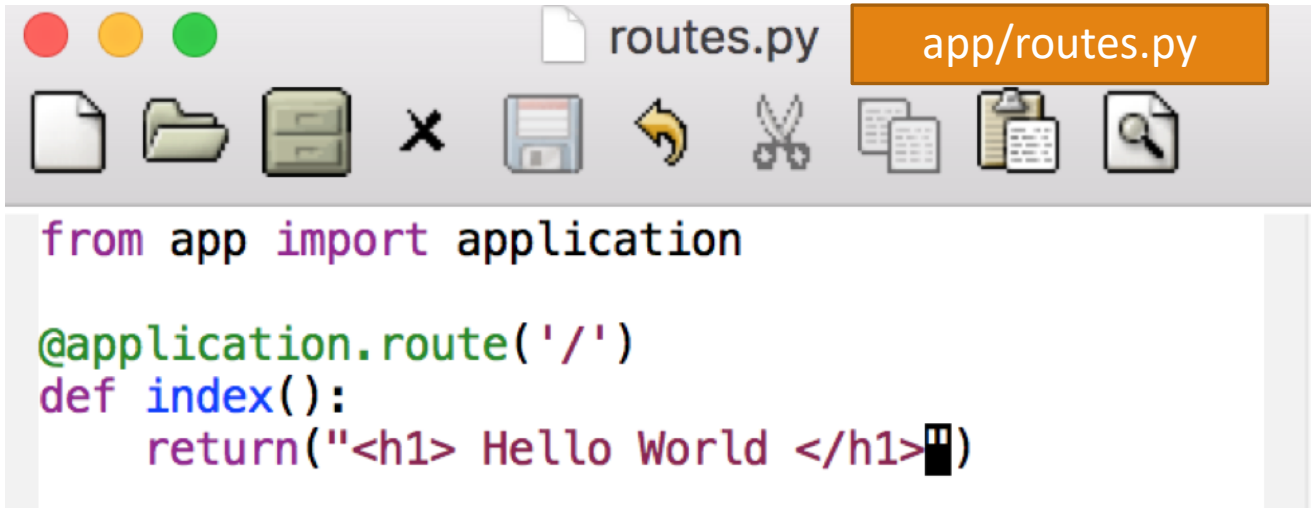
- An application instance needs to be initialized for passing all requests from clients.
  - The application instance is an object of Flask.
    - For Flask class constructor, it requires the name of the main module.



Python Flask

Routes and View Functions

- Flask maps URLs to Python functions using route.
  - @application.route*(route)* defines a route and registers a function.



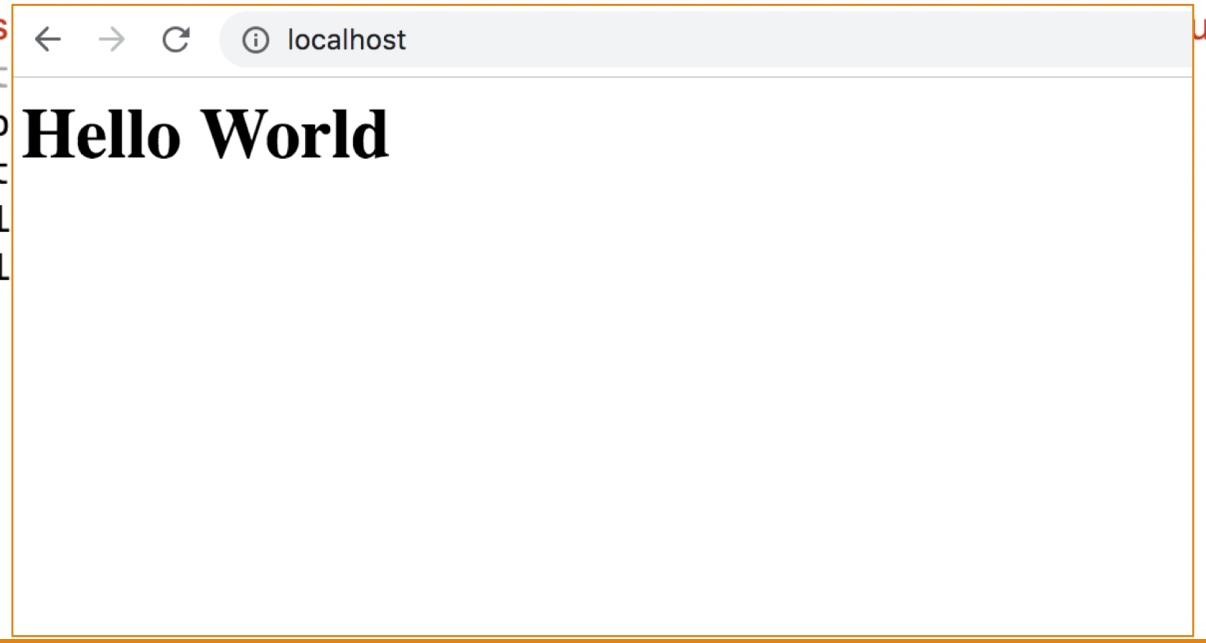
Decorators for Functions and Methods https://www.python.org/dev/peps/pep-0318/

Python Flask

Server Startup

- Goal : Enables a command, “flask run” to execute the application.

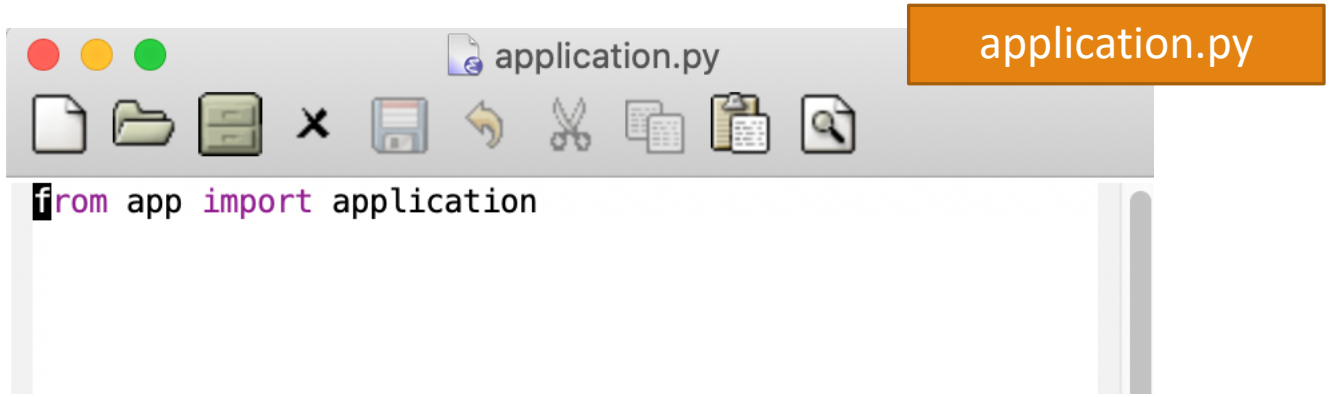
```
(ProductAnalytics_Env) (base) ML-ITS-901885:code dwoodbridge$ flask run
 * Environment: production
   WARNING: This                                              uction deployment.
   Use a product
 * Debug mode: o
 * Running on ht
127.0.0.1 - - [1
127.0.0.1 - - [1
```



Python Flask

Server Startup

1. Create a .py defines the Flask application instance.



Python Flask

Server Startup

2. Install python-dotenv and configure .flaskenv file.
   - Allow register environment variables that can be automatically imported when running "`flask run`" command.
     - `flask run :` Start the development server.
   - Command : `pip install python-dotenv`

```
(ProductAnalytics_Env) (base) ML-ITS-901885:code dwoodbridge$ pip install python-dotenv

nv-0.12.0-py2.py3-none-any.whl (17 kB)
ages: python-dotenv
thon-dotenv-0.12.0
```

Example

Run application.py on port 80 by creating and configuring .flaskenv.

- By default, HTTP uses port **80** and HTTPS uses port **443.**

Python Flask

Server Startup

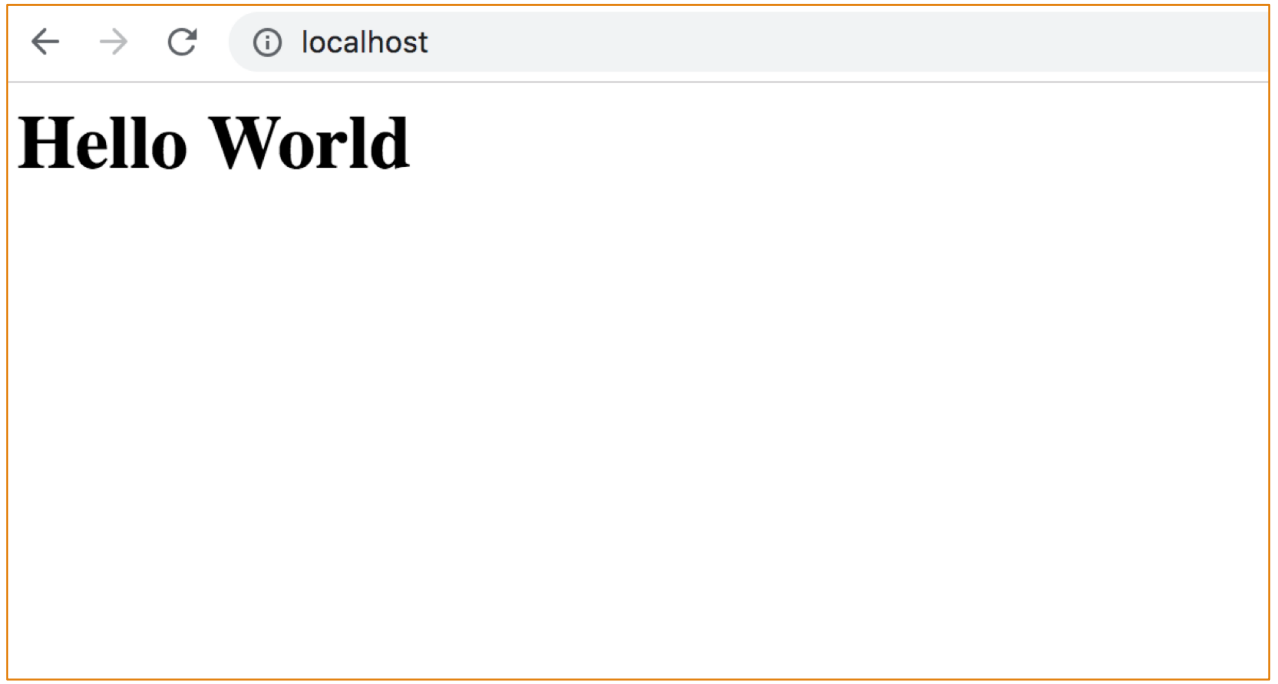
3. Run the application
   - Command : `flask run / sudo flask run`

```
(MSDS603) ML-ITS-603436:code dwoodbridge$ sudo flask run
Password:
 * Serving Flask app "run_app.py"
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:80/ (Press CTRL+C to quit)
```

Python Flask

Server Startup

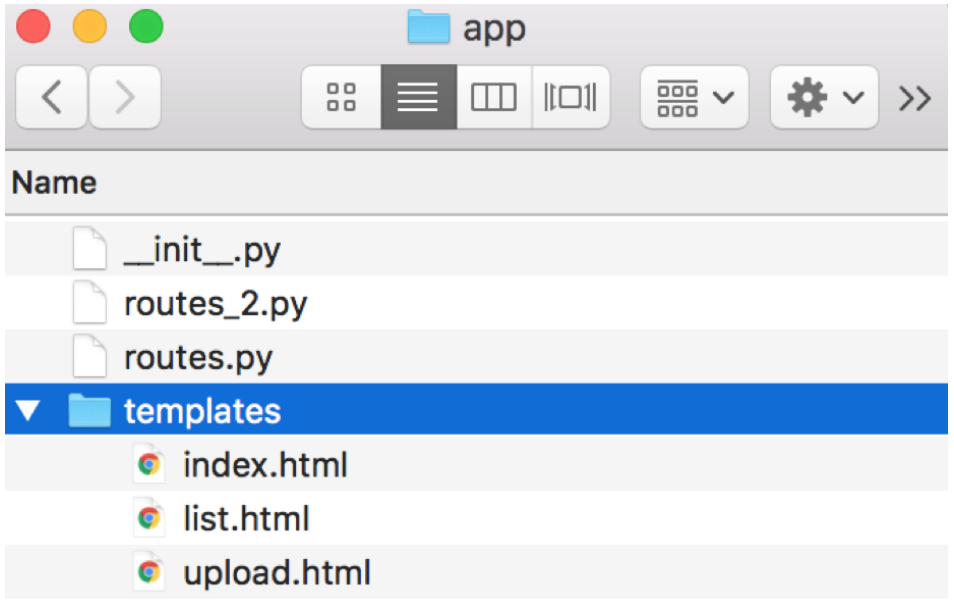
3. Run the application
   - Command : `flask run / sudo flask run`



Python Flask

Jinja2

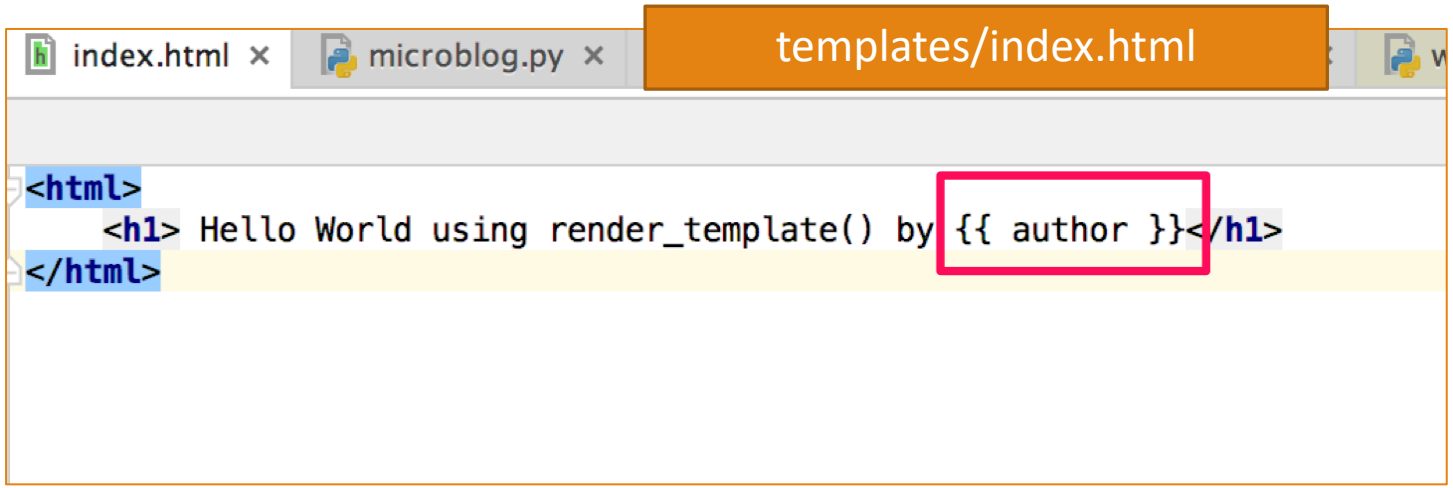
- Web template engine support for the automatic generation of custom web pages.
- Rendering templates
  - Find templates (.html, etc.) in *'templates'* folder located inside the application folder.



Python Flask

Jinja2

- **Variables**
  - {{***name***}} references a variable.
    - It is a special placeholder that tells the template engine that the value of ***name*** should be obtained dynamically at the time the template is rendered.



Example

Update routes.py so that index.html will be rendered when the web site is called or the 'index' page is called.

- Make sure that 'author' is your name.

Python Flask

Jinja2

- **Control structures**
  - Conditional Statements :
    {% if *condition* %}
    something
    {% else %}
    something
    {% endif %}

  - Include an existing template :
    {% include '*template*' %}

Flask-WTF

WTF

- Provide web forms and handle “Post” requests from clients.
- Installation
  - `$ pip install flask-wtf`
- Format
  - `FormField('label', validators=[FormValidators])`

```
(ProductAnalytics_Env) (ProductAnalytics) ML-ITS-901885:documentation dwoodbridge$ pip install flask-wtf
Requirement already satisfied: flask-wtf in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (0.14.3)
Requirement already satisfied: itsdangerous in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from flask-wtf) (1.1.0)
Requirement already satisfied: WTForms in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from flask-wtf) (2.2.1)
Requirement already satisfied: Flask in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from flask-wtf) (1.1.1)
Requirement already satisfied: Jinja2>=2.10.1 in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from Flask->flask-wtf) (2.11.1)
Requirement already satisfied: click>=5.1 in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from Flask->flask-wtf) (7.1.1)
Requirement already satisfied: Werkzeug>=0.15 in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from Flask->flask-wtf) (1.0.0)
Requirement already satisfied: MarkupSafe>=0.23 in /Users/dwoodbridge/Class/2020_MSDS603/ProductAnalytics_Env/lib/python3.7/site-packages (from Jinja2>=2.10.1->Flask->flask-wt
```

https://flask-wtf.readthedocs.io/

Flask-WTF

Form Field

Field Type	Description
StringField	Text field (<input type = 'text'> in HTML)
TextAreaField	Multiple-line text field
PasswordField	Password Field
DateField	Text field that accepts date value
IntegerField	Text field which accepts an integer
SelectField	Drop-down list of choices
FileField	File upload field
SubmitField	Form submission button
More	

https://flask-wtf.readthedocs.io/

Flask-WTF

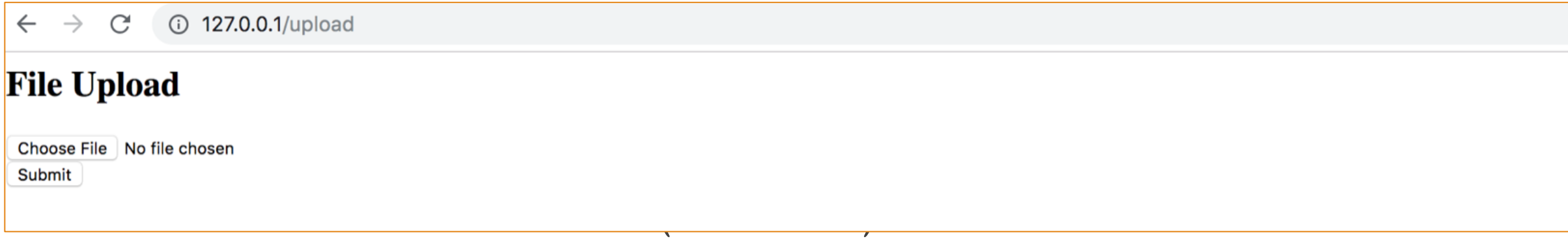
Form Validator

Validator	Description
Email	Validates an email address
Length	Validates the length of the string entered
DataRequired	Validates that the field is not empty
FileRequired	Validates the data is FileStorage object
Regexp	Validates the data against a regex
URL	Validates a URL
AnyOf	Validates that input is one of the given values.
More	

https://flask-wtf.readthedocs.io/

Flask-WTF

Each web form should be a class which inherits from class *FlaskForm*.



https://flask-wtf.readthedocs.io/

When we are designing an "upload" page, what are the form fields we need to use?

Flask-WTF

Each web form should be a class which inherits from class *FlaskForm*.

app/routes_2.py

```
from flask_wtf import FlaskForm

class UploadFileForm(FlaskForm):
      file_selector = FileField('File', validators=[FileRequired()])
      submit = SubmitField('Submit')
```

Will talk more about "Class and Object Oriented Programming" Next week ☺

https://flask-wtf.readthedocs.io/

Flask-WTF

Each web form should be a class which inherits from class *FlaskForm.*

app/routes_2.py

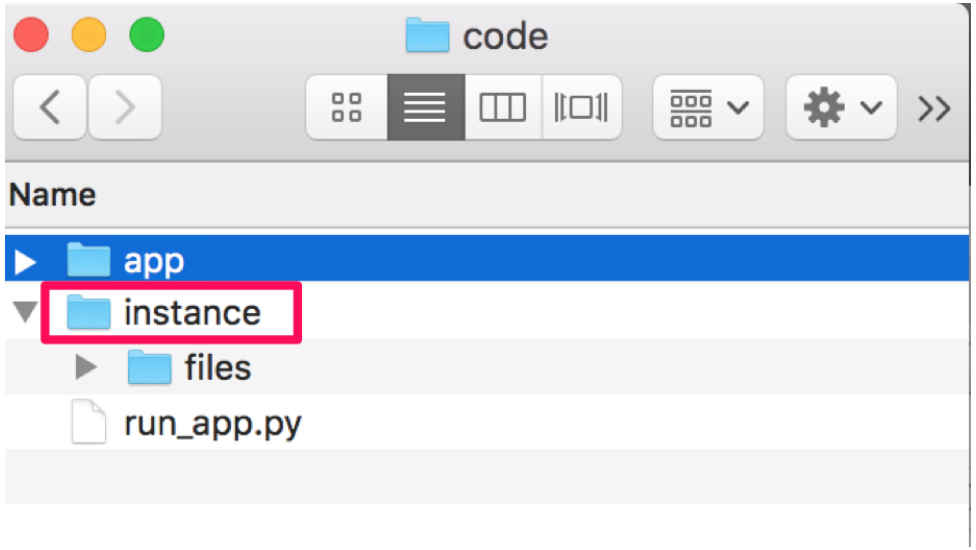
```
file = UploadFileForm()  # file : UploadFileForm class instance
if file.validate_on_submit(): # Check if it is a POST request and if it is valid.
    f = file.file_selector.data  # f : Data of FileField
    filename = f.filename  # filename : filename of FileField

    file_dir_path = os.path.join(application.instance_path, 'files')
    file_path = os.path.join(file_dir_path, filename)
    f.save(file_path) # Save file to file_path (instance/ + 'files' + filename)

    return redirect(url_for('index'))  # Redirect to / (/index) page.
return render_template('upload.html', form=file)
```

https://flask-wtf.readthedocs.io/

Flask-WTF

`application.instance_path`

- Refers to a new concept called the “instance folder”.
  - The instance folder is designed to not be under version control and be deployment specific.
    - It’s a good place to store files that change at runtime.



http://flask.pocoo.org/docs/1.0/config/

Flask-WTF

Each web form should be a class which inherits from class *FlaskForm*.

app/routes_2.py

```
file = UploadFileForm()  # file : UploadFileForm class instance
if file.validate_on_submit():# Check if it is a POST request and if it is valid.
    f = file.file_selector.data  # f : Data of FileField
    filename = f.filename  # filename : filename of FileField

    file_dir_path = os.path.join(application.instance_path, 'files')
    file_path = os.path.join(file_dir_path, filename)
    f.save(file_path) # Save file to file_path (instance/ + 'files' + filename)

    return redirect(url_for('index'))  # Redirect to / (/index) page.
return render_template('upload.html', form=file)
```

https://flask-wtf.readthedocs.io/

Flask-WTF

You can pass wtf variables to the rendered .html file.

templates/upload.html

```
<div class="container">
        <form method="POST" enctype="multipart/form-data”>
        <h2>File Upload</h2>
        {{ form.csrf_token }} <!-- render the form's CSRF field like normal. -->
        {{ form.file_selector }}
        <br/>
        {{ form.submit }}
        </form>
</div>
```

app/__init__.py

```
application = Flask(__name__)
application.secret_key = os.urandom(24)

from app import routes_2 # Added at the b
```

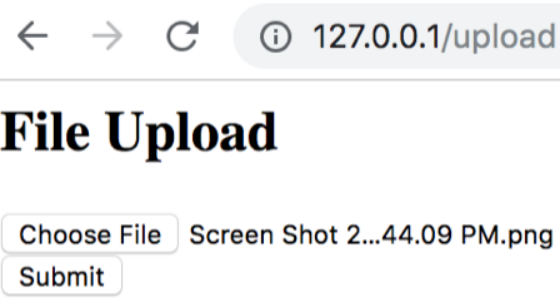
app/routes_2.py

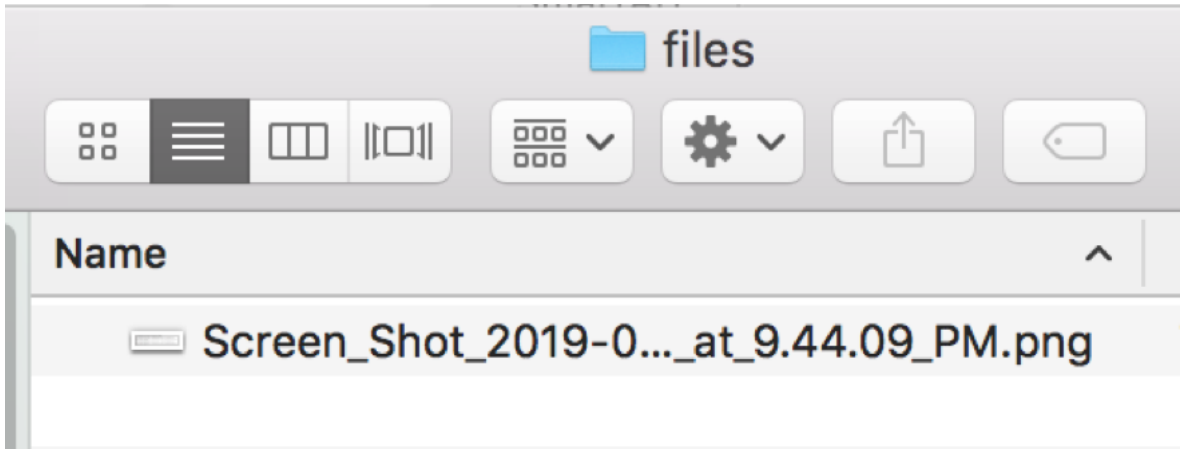
```
class UploadFileForm(FlaskForm):
    file_selector = FileField('File', validators=[FileRequired()])
    submit = SubmitField('Submit')
```

https://en.wikipedia.org/wiki/Cross-site_request_forgery
https://flask-wtf.readthedocs.io/en/stable/csrf.html

Flask-WTF

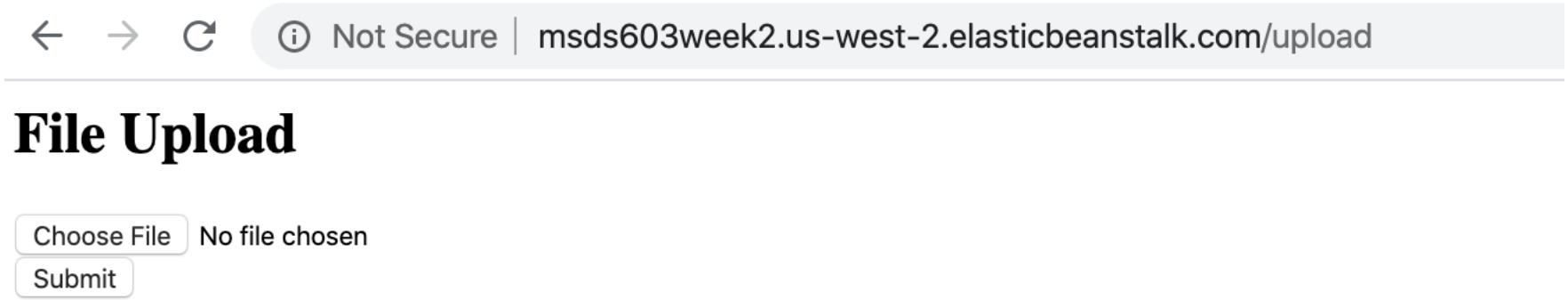
Each web form should be a class which inherits from class *FlaskForm*.





Example

Deploy on Elastic Beanstlk using Docker



Data Acquisition and Storage

Brainstorm what you need to collect for your application.

- Things to consider.
  - What are the main data that you need to collect?
  - Are the data sets that you need even available? (Or you can collect on time?)
    - Farmers real-time crop data in Petaluma?
    - HIPPA regulated patient health record?
    - IRB approval required data?
- Techniques
  - Data acquisition (MSAN692 Data Acquisition)
    - Open data set, API calls, Crawling, etc.
  - Data storage
    - SQL (MSDS 691 Relational Databases)
    - NoSQL (MSDS 697 Distributed Data Systems)

Contents

Backend and Frontend

Backend and Data Acquisition

- API
- wget
- Crawl
- Collect data from users
  - Create file upload form using Python Flask and WTF.

Comments (What you liked/disliked so far? What should I do for you?)

Reference

Flask SQLAlchemy, http://flask-sqlalchemy.pocoo.org/2.3/

Werkzeug Security Helpers, http://werkzeug.pocoo.org/docs/0.14/utils/#module-werkzeug.security

Flask Login, https://flask-login.readthedocs.io

Kenneth Reitz. *The Hitchhiker's Guide to Python*. http://docs.python-guide.org/en/latest/

Grinberg, Miguel. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.