**Meerkat Software Training**

**Web Programming in Python**

**Exercise: Content and Styling**

**Primary Target:** A good place to look for tutorials and docs related to web programming is the site: [www.w3schools.com](http://www.w3schools.com/). The aim of this exercise is to become as comfortable with basic HTML and CSS files. This is all about specifying content and styling.

1. Work through the first stages of the HTML5 tutorial found at: [www.w3schools.com/html](http://www.w3schools.com/html)

HTML Introduction

HTML Basic

HTML Elements

HTML Attributes

HTML Headings

HTML Paragraphs

HTML CSS

HTML Links

2 .Work through the first stages of the CSS tutorial found at: [www.w3schools.com/css](http://www.w3schools.com/css)

CSS Introduction

CSS Syntax

CSS How To

CSS Colors

CSS Backgrounds

CSS Borders

CSS Margins

CSS Padding

CSS Height/Width

CSS Box Model

**Advanced target:** The CSS in Meerkat Frontend is assembled from SASS files. Read about SASS here: [sass-lang.com/guide](http://sass-lang.com/guide). SASS is a preprocessor that generates standard .css files from .scss files. Among many useful features, it allows us to abstract values (e.g. colours, sizes etc...) into variables. The SASS is compiled into CSS as part of our build process (we will discuss this in the next talk). You can find the .scss files in:

[meerkat\_frontend/meerkat\_frontend/src/sass].

1. Read about SASS using the link above.
2. Browse the SASS files for Meerkat Frontend and see if you can make sense of them.
3. Use developer tools to find the ID/Class of some elements, make some changes and see if they are visible in a web browser. (You will need to run the commands [gulp clean] and then [gulp] in the Frontend docker container to compile the sass)

**Meerkat Software Training**

**Web Programming in Python**

**Exercise: A Meerkat Frontend Case Study**

In this exercise we will step through the process of adding a simple new web page to Meerkat Frontend. The web page will show some location specific details for a specified location. The location will be specified using a location ID number as part of the url. The location ID for Petra is 12, so you will add a web page at the url [/location/12] that returns some information about Petra.

**Basic Target:** Work through the following steps (as far as possible)...

1. Start up meerkat frontend docker container.
2. Set up a simple Jinja2 template.

Create a file **meerkat\_frontend/meerkat\_frontend/templates/location.html**.

Initialise the file as a simple HTML page by copying the contents of the blank template found at:

[meerkat\_frontend/meerkat\_frontend/templates/template.html]

1. Create a python flask page function.

Use the following tutorial to create a python flask page function at the end of the file [meerkat\_frontend/\_\_init\_\_.py]:

[http://flask.pocoo.org/docs/0.12/quickstart/#rendering-templates](http://flask.pocoo.org/docs/0.12/quickstart/" \l "rendering-templates)

You will find that errors are thrown if you don’t send the template some data. You’ll need to send it which epi week we are in and some configuration info:

week=c.api('/epi\_week'),

content=current\_app.config['SHARED\_CONFIG']

1. View the work so far.

One of the docker containers that we run in our development environment is a web server. It allows you to make http requests to your own computer at the [localhost] domain name, or the ip address [127.0.0.1]. To view the changes we've made so far, we'll need to restart the Frontend docker container by running the command [docker-compose restart frontend]. Then open a web browser and type [127.0.0.1] into the address bar. If all goes well you should see a web page showing details about Petra, if an error occurs, use the command [docker-compose logs frontend].

1. Add some sensible titles and text content to the page.
2. We have written a function that gets data from meerkat\_api using the server-side python. The function is in [meerkat\_frontend/meerkat\_frontend/common.py]. This python module is imported into \_\_init\_\_.py at the top of the file with the variable name “c”. To call it type: c.api(‘api\_url’). The URL that will give you information about locations is: ‘/locations’. Type [127.0.0.1/api/locations] into the web browser address bar and you should see the data that the function will import as a python dictionary. Initially, call the function within your flask page function (step 3) and pass it to the jinja2 template. Just print the data out in the web page to begin with. The tutorial from step three should help you with this.
3. View the work so far.
4. Now we write the code to allow the user to specify a location ID number in the URL and then print the name out for that location ID in the page. You will need to add a variable to the URL for the flask page function that will hold the location ID number. Again, the tutorial from step 3 can help with this. Then you will need to get the location from the API data. Finally you will need to pass the name to the Jinja2 template and print the name out.
5. Add some styling.

The CSS is compiled from SASS files. Read about SASS here: [sass-lang.com/guide](http://sass-lang.com/guide). SASS is a preprocessor that generates standard .css files from .scss files. Among many useful features, it allows us to put values (e.g. colours, sizes etc...) into variables that can be used throughout the css. It also lets us split the CSS across multiple files that are imported into a single file [main.scss]. SASS has to be compiled into CSS before viewing. This is done when running Gulp. Find the SASS files in:

[meerkat\_frontend/meerkat\_frontend/src/sass]

1. View the work so far.

Remember that you will have to restart the front end docker container. However, this time you will also have to compile the SASS. This is done as part of the gulp build process, so you will need to run the commands [gulp clean] and then [gulp] in the Frontend docker container.

**Advanced Target:** Change the page design to match the other pages in Meerkat Frontend.

The file [meerkat\_frontend/meerkat\_frontend/templates/error.html] is a simple example of how to create a page that matches the design of the rest of the site. Merge the contents of this file into your newly created JINJA2 template so that your location page has a visual design that matches the visual design of the rest of the site.