

Carbon Accounting and Sustainable Designs in Product Lifecycle Management

Prof. Deepu Philip
Department of Management Sciences

Prof. Amandeep Singh Oberoi
Imagining Laboratory

Dr. Prabal Pratap Singh
Department of Management Sciences

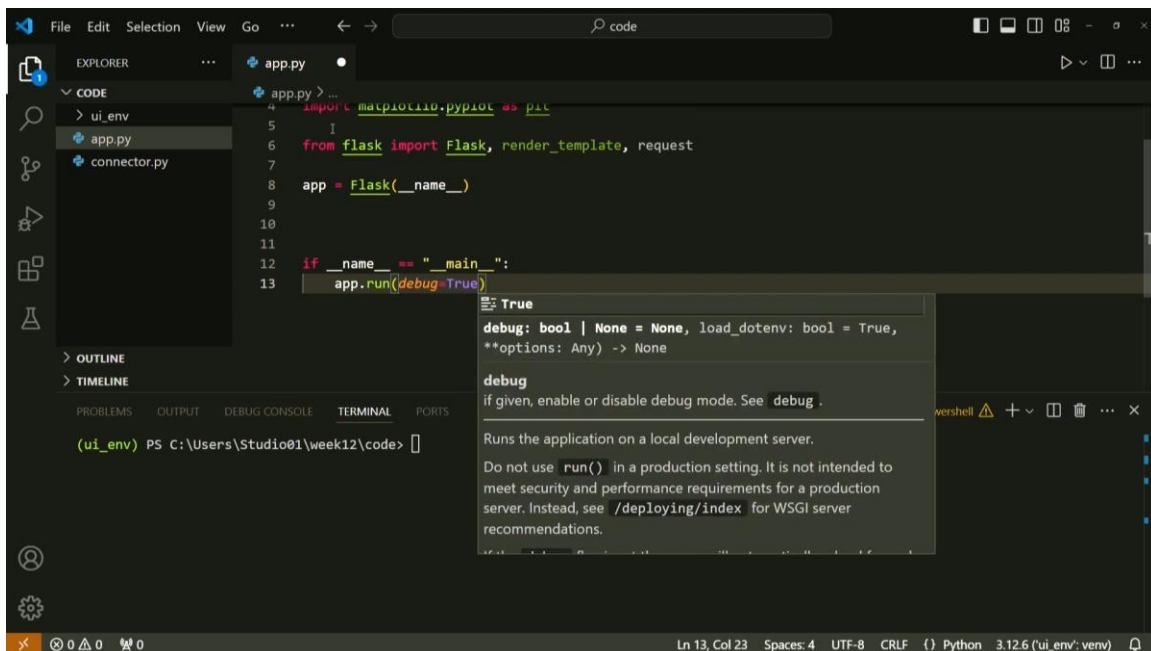
Indian Institute of Technology, Kanpur

Week 12

Lecture 53

Carbon Accounting User Interface (Part-2)

Welcome again to the course. We are trying to develop the user interface for our carbon accounting system. And till now we have covered how to connect our Python script with the MariaDB database server. And how we can fetch or run different kinds of user queries on the MariaDB server.



The screenshot shows a code editor with a Python file named `app.py`. The code is as follows:

```
1 import mysql.connector as mysql
2
3
4
5
6 from flask import Flask, render_template, request
7
8 app = Flask(__name__)
9
10
11
12 if __name__ == "__main__":
13     app.run(debug=True)
```

A tooltip for the `debug=True` parameter is visible, showing the following details:

- `debug: bool | None = None, load_dotenv: bool = True, **options: Any) -> None`
- `debug`: if given, enable or disable debug mode. See `debug`.
- Runs the application on a local development server.
- Do not use `run()` in a production setting. It is not intended to meet security and performance requirements for a production server. Instead, see `/deploying/index` for WSGI server recommendations.

The terminal window at the bottom shows the command prompt:

```
(ui_env) PS C:\Users\Studio01\week12\code>
```

The status bar at the bottom indicates the file is at line 13, column 23, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.12.6, and the virtual environment `(ui_env: venv)`.

So let us now begin by developing the user interface. So for that, let us create a new file named as `app.py`. And we have installed the MariaDB package and Flask package. We need to install few more packages like `pip install numpy pandas and matplotlib`. So once these packages get installed, we can start by utilizing them. So we can write our import statements.

Import the MariaDB package. Then import numpy as `np`. Import pandas as `pd`. And import matplotlib pyplot as `plt`. Also since the flask package is the main web application framework.

So we need to import only few functions from that framework. So we can write `from flask import flask`. This is a function and then we can use `render_template` and also the `request`. So these are the main import statements that we require to develop our UI system. Now the first task to create a flask application is to create a variable named as `app` and try to utilize the imported statement with the name `flask` and use `double_name`.

So this will initialize an app and whenever we need to create a flask application and run it from a script file we also need to create the main check whether we are running the file directly. So to do that we need to write `if name is equal to __underscore main if this is true then run app.run and with the debug is equal to true`. So the main skeleton of our app is covered now we need to create a web page. So to do that we need to again define a new function. Let's say this is a home page so we can write `dev home` and we can provide a doc string the name home page of the app and now we need to use a decorator.

So these decorators are available in this last library so we can use `app route`. So this decorator will try to route all the requests that are going for this home page under slash and currently we are using only the methods as `get`. So we can write `get` now what this home page will show you. So let us say just so this is a very basic web application that will show you a home page on the slash directory and it will print carbon accounting system right. So let us try to run this application and see how this application works so we can write.

So the output in the terminal shows that this web application is running on this address. So just copy this address and try to open a web browser. Now you can see here that it is showing carbon accounting system, right. So we have initialized our app successfully. And now we will try to build on this app.

So that we can utilize our database connection and try to manipulate or add or retrieve the data set from the database and show various different kinds of statistics to the user, right. So let us now come again to the app. And this we need to create again the connection with the database. So we can utilize the function we have defined here. Now we can also use the import statement to directly import this connectorpy.

But since we are going to modify this function as well for our current carbon counting system. We are trying to we will be copying this system connector system completely. So this is the connector system. Now to make changes to this version of our web application we need to develop different kinds of html pages and styling sheets. So to do that we need to first create a new directory with the name templates.

So whenever this flask application run it will try to search this predefined templates directory. And anything inside this directory will be shown as and when required as per the definitions defined in the python script. So let us now try to create a new file with the name about.html okay. And it is currently blank and to create a new web page in this application we need to create a new function. So we will create def about the application.

And this will also route so we can use and we will try to create a new page with the name about. And we can this function will return a template. So to return a template instead of a simple string like this we need to utilize the imported render template function okay. So we can right here render_template. And this is a function that will take different kinds of parameters.

So you can simply state the name of the file that is having the information about this application. So we can write about.html here and if you just write about page in this about.html. You can see that the application will try to fetch this information available in this about.html file. So go back again to your carbon accounting system and try to write okay. So now this slash about is also a web page available in your web application.

Now let me try to modify this page by creating different kinds of CSS. So we will try to use the bootstrap CSS here. So we will import the bootstrap CSS CDN but and we will also create the base home page by utilizing index.HTML. So when we are trying to create the user interface, we first need to show different kinds of components to the user of the system. So we will try to create the skeleton of that structure and whatever the dynamic content that we are trying to modify, we will fetch through the database.

So the first task is to create the skeleton. So let me fill the basic skeleton of the user interface by utilizing the Bootstrap CSS. So now you can see that we have created the basic skeleton of the webpage for the homepage. And it includes head that is utilizing the CDN of Bootstrap CSS and the title of the webpage is Carbon Accounting DSS. And we are creating a navbar on that webpage by utilizing the Bootstrap framework.

Also there are only two different links that are available on the navbar that is home and about right. And this is the javascript for utilizing this bootstrap successfully, okay. So now we can modify our home page and utilize this index.html by following the render template function. And we can write index. So, now if we try to load our web application again, we will see the skeleton of the page that we have just imported.

So, this is how our system will look like. So these are just the UI elements which we have created but the actual meat of this web application we will code in the next steps. Again if you try to click on about it will show the about page but it is not showing the actual skeleton because the about.html is still not having all these elements that we have created here. So what we can do is. We can copy this code and paste it again in our about.html.

And we can write about the custom accounting system. And now if you try to load it again this about page is also modified. Now let us try to develop the main user queries and try to change how things will get into from the user to the database and backwards. So to do that we first need to take the queries from the user dynamically right. So for performing these actions these types of actions till.

Now what we are doing is we are creating our script and showing what the what are the changes that are happening on the web application. But to interact with the web application, the user needs different kinds of buttons, different kinds of user information. So to do that, we need to first create forms on the web page. And that way we can deliver the user queries from the Python language to the MariaDB server. So again, we need to modify the skeleton of the project by utilizing this index.html.

So wherever your navbar code ends you need to start creating the div element and the class of this element will be. We are utilizing the bootstrap css. So the class of the element is container suit we are not hard coding all of these information we are just you using whatever that is available. So the next division is in rows we have created a new row. And now we will create two different columns on the home page.

So to do that we will again create a div and this will have a class of colm 6. So we are dividing it into two that is why we are using six and we can write the heading as we choose emission sources. So this part of the form will provide different kinds of selection option to the user. So that they can choose the emission sources for showing it on the web application. So let us try to create a form for doing this and the method let us remove the action and the method of this form is post.

Now since the method is post we need to add this functionality here as well in our base home page decorator okay. And what this form will contain it will have a label for country_select first the form will provide you a selection for the country. So now we have started creating our form that will be useful for providing different kinds of queries to the system using the web application. So we are creating the label for that form and this the first thing that this form will try to provide is the selection for the countries.

So the class for this label is form hyphen label these are predefined classes in the bootstrap css and we are utilizing those classes directly we can hard code these classes as well. This is the most useful way to create different kinds of web applications. Now the label is created. We need to utilize the select tag and we will use the class for this tag as form select the id is country_select and name. This is important why because the name of the variable that this web application will try to find using this form is this name that we are writing here so this is country select.

And we can write area hyphen label as default select. We can utilize the word wrap functionality in our code editor so that things don't go beyond the screen limits. And we can close this selected. Now there are 194 countries in this data set and we need to provide the user to select any of these countries. So we need to write 194 options instead of writing those 194 options.

Let us just copy those as we have predefined these options. So we have these many countries that are available in a mariadb database and all of these options are will be available in the user interface. So let us try to check how this will look like in the web application. So this is how you can select okay so we have created only one select option from the on the web application using the form tag right now. Let us try to create different select option.

So we can again initialize a new div element and the class for this will be form select. Class will be mb3. Again the label is the sector select. This part of the form will try to

provide the option for selecting the sectors that are available in the database. And class remains the same.

Form label the name for this select is sector select and id remains the same. We can change the id but let us keep it the same value. And there are different kinds of options that are available now the last part that will be available for the user to select is this the type of gas that has been reported. So guess select ID is also gas select. And aria-label is equal to default gas.

Select example options for gas are very limited in the database. So these are the only options like. So either we have an entry in the database with all GST that is it includes the summary information or the carbon dioxide or methane N2 or the F gases. So until this our form will now have three different kinds of options for the user. So select a sector, select type of gas and select a country.

We need to correct this, right. Now we need to create a submit button so we can place it in center and try to utilize the button tag. The type of this button is submit, ID is also submit. Value and name that is required to access it in our python script, okay. And let us also provide a bootstrap class btn and btn-primary.

So we need to create a new development here as well p3 border. So now we are trying to develop the other column on our homepage and let us write column two. This was column one. So we can, in this column, we will try to show whatever the user has selected from the available options. So we can write this under heading.

And here also we will try to add data generated from script. So we will populate this section afterwards after we create our application completely. Now we can see that this is the user selection and we need to create this border as well for this so to do that we should create a new div. And it should have a class with the same p3 border bg light k. So we also we need to add this class as well otherwise it will not look the same as it is looking for the first option.

So now the basic structure is complete and we can now see that we can select any of these values from here. And we can submit this information to the Python script or the web application that is being run using the Flask application. And now let us try to capture whatever information the user will send from this form to the Python script and run the queries on the database. So to do that, we need to heavily modify the homepage because whatever we are doing, we are doing it on the homepage of the carbon

accounting system. So now I have told you that there are two different kinds of methods that can be used in this application that is get and post.

So to receive or show the basic information or strings we can utilize the get method. But when we are trying to utilize the form functionality, we need to use the POST method. And whenever the application runs on the browser, it will either run as a simple request to the web server or by providing the user input to the web server. So whenever we try to push the submit button, The communication between the user interface and the web server will require the POST method.

So, we need to check whether we are using the GET method or POST method while communicating from the user interface. So, to do that, we need to write basic if statement if request.method is equal to POST. If the method is post then only perform the perform the operations required on the user input. The first thing is we can capture all the information that has been sent by the user. So we can create another variable form results, form res is equal to request.form.

Now you may think where this request variable is coming from. So this is the variable that we have included from the flask package and we are utilizing this here. So now we have all the form data that has been sent by the UI into the form res variable and we can select country_name. We are creating a new variable that will store the name of the country that has been sent by the user and it is available in the form_res.get and country_select. So this country_select is the name of this section of the form.

Now the name of this section will be utilized in the script of the ui. So similarly we will have the emission source name as well source name is equal to form.res form_res.get source_select. Similarly the name of the guess that the user is trying to fetch the data for from the database. So form_s.get_guess select. Now we have the information that has been provided by the user using the form of on the user interface.

So we can store all this data into a python dictionary by creating a new variable name as user_selection_dict. This will be useful when we try to showcase all the information that has been selected on the user interface again. We can create the key as country for this dictionary and store the country name. Then we can create another key with the name source and store source name. Then we will utilize the guess and store the guess name, okay.

So our dictionary is ready now we can also print this dictionary to validate that we are receiving these information from the form. So if now we try to push the submit button. We will send this information like ch4 energy and select any country. Let's say india so we have clicked the submit button and something has happened. Now let us check whether we are receiving this information.

So in the terminal window you can see that once we sent a get request and submitted the data. Through POST, we received country as IND, source as none, and gas as CH4. So, we are not receiving the source information. Let us now try to debug why we are not receiving the source information. So, source name is source select.

So, we need to check the source, okay. So we have defined the name here as the sector select. But instead of using sector select, we are using source select. So we can write here sector select. And now we should get this information back to our system.

So let us choose something else. Agriculture format. And you can see here now that country is COD, source is agriculture, gas is CHO. So what we have done is we created a connection script to the MariaDB server. Then we created the skeleton of the user interface.

After creating the skeleton of the user interface, we developed a form on the UI of the web application. And with this UI, we have successfully transferred the user selection to the web application in the terminal. But our main aim is to fetch the data from the database that has already been created in the mariadb dbms. And try to get the information that has been asked by the user through the user interface. So now in the next section we will try to create the queries by utilizing the information provided by the user and try to fetch the information from the database.

Thank you.