

Advanced Business Decision Support Systems
Professor Deepu Philip
Department of Industrial Engineering and Management Engineering
Indian Institute of Technology, Kanpur
Professor Amandeep Singh
Imagineering Laboratory
Dr. Prabal Pratap Singh
Indian Institute of Technology, Kanpur
Lecture 34
DSS Development using Python

Hello everyone, I welcome you all to the lecture series on Advanced Business Decision Support Systems. I am Doctor Prabal Pratap Singh from IIT Kanpur and we are learning to create Decision Support Systems. And, today we are finally going to develop our own completely working Web-based Decision Support Systems.

- Fundamentals of Python
- Modules
- NumPy

DSS Development using Python
Advanced Business Decision Support Systems

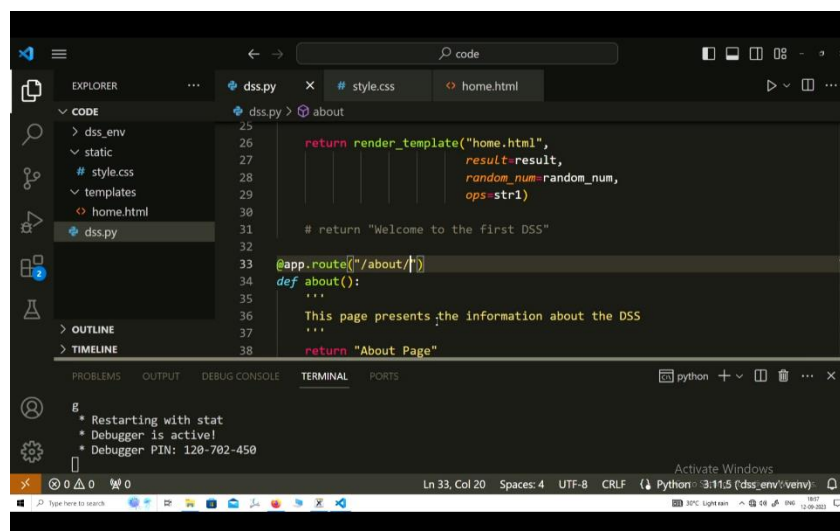


Advanced Business Decision Support System
Dr. Prabal Pratap Singh
Dr. Amandeep Singh
Dr. Prabal Pratap Singh
Indian Institute of Technology Kanpur

So, till now we have covered Fundamentals of Python like variables, statements, different kinds of data types and all these things. Then, in the last lecture we have gone through the Modules like how to create a module, and then import it to a different module and we have also seen a scientific library named as NumPy and we have used this library to generate Pseudo Random Numbers. So, today we will start by creating our own Decision Support System which is available on a browser.

DSS Development

- * Develop a basic DSS system - Decision model is basic
- * A DSS that can
 - Generate Random Number between 1 and 100
 - Decide whether the no. generated is less than 50 → Cube of the number
 - Greater " " → Square of the number
- * To develop the system around this decision task we will use a new library - Flask
 - Install Flask in our environment
 - Initialize our Flask app
 - Create a function that will act as a route of our web-based DSS.
 - Create a main function inside our Python Script →
 - Templates inside our DSS system
 - Static Files inside our DSS system
 - Namely to PSL.



```
25
26     return render_template("home.html",
27                             result=result,
28                             random_num=random_num,
29                             ops=str1)
30
31     # return "Welcome to the first DSS"
32
33 @app.route("/about")
34 def about():
35     ...
36     This page presents the information about the DSS
37     ...
38     return "About Page"
```

So, we will start by developing a basic DSS system. So, our main motive today is to create the system, not to see how complex the task we can perform using this system. So, the decision model which we will be using today is very basic, the decision model is basic. But the learning task today is to create the system.

So, what is the task today? The task is to create a DSS that can generate random numbers between 1 and 100, then it should decide whether the number generated is less than 50 or not. And, if it is 50, then it should perform the cube of that number cube of the number and if it is greater than 50, then it should compute the square of the number. Now, these are the decisions that we are making here and these are the computations that we are performing. So, these decisions and computations will become more complex in the real life scenario, but the development of a system around these decisions and these computations remains the same. So, this is the task and to develop the system around this decision task, we will use a new library which is Flask.

So, Flask is a micro framework available in python libraries and we will first install it in our environment, then we will try to initialize our Flask app. So, from here we will start coding in a python script, and then the first task is to initialize our Flask app, we will see how we can do it, and then we will try to create a function that will act as a root of our Web-Based DSS. So, we will see what is the root of our Web-Based DSS, when we start coding today. After that, we will also try to create a main function inside our python script. So, this main function will act similar to what we have, the main function, in our C or C++ code.

So, these kinds of function they will ask the interpreter to start the execution from this function and whatever you write in this function, the interpreter will start executing the statements, and then we will also see how to use templates, these are HTML templates, inside our DSS system and how to use static files. Static files are your JavaScript or style sheets like CSS files, how to use these static files inside our DSS system. So, how to generate and code these static files? We have already seen in our previous basic Decision Support System course available on NPTEL and today we will start coding and using those static files to modify our Decision Support System.

The last thing is we will also use NumPy because our task is to generate these random numbers. So, we have to use this library to generate Pseudo Random Numbers. So, let us start by opening our Integrated Development Environment, that is IDE. So, this is my code directory which is blank. Now, let me open this with Visual Studio code. So, as a customary thing, we will first create our environment for today for this project of creating our basic Decision Support System. So, you can see here that the environment is created, now we need to activate this. So, it is active now.

You can see here that this is the environment which our command prompt will use from now. Now, today we need two different external libraries from the extensive Python library ecosystem which is the NumPy which we already installed in our last lecture also. So, to install it again, we will write the same command. Python hyphen m pip install NumPy. This will install the NumPy.

And, the main package for today, that is flask, we need to install that package as well. So, now it is installed. Now, we can start writing our code today. So, the very first thing is to import a module that we need today which is NumPy and we are using it using an alias called np.

The next thing which we want to use today is flask. So, we are importing. We can import flask like this also or the better way which we discussed last time is if we just want to use a particular function from an external library, then we can directly name that function and import it. So, from flask we need to import only the flask function. So, we have completed our imports and now we need to initialize the DSS application. So, let us initialize a variable name app, and in that we can initialize our main application by using the flask function which we have just imported, and then here we are writing something which we have not yet discussed.

This is the predefined named variable which is available in all the python scripts we write. So, what this statement tells is that in the flask function send the argument of whatever the value of name is, send that value as an argument to this function flask. So, it will by default complete all the basic necessary things

that this flask application needs, like setting up the name, setting up the directory and all those things. So, it will set this directory as its source path. The next thing is to create a main function which will route our Web Decision Support System to the home page.

So, in summary we are actually creating the home page by defining this function. So, let us write the name of this function as home without any arguments and let us provide a doc string which says the home page of the DSS. So, let it return only a single statement. So, when this application will run, this function is the route of the Web- Based DSS and what it will do is, it will return this string. So, this string will get printed on the browser.

So, we will see how it will happen, but before that this is a simple function. now. But, to expand the capabilities of this function there is an advanced property in python which is known as decorators. So, what decorators do? They try to extend the capabilities of a particular function that you are giving them as input. So, in flask applications, there are different decorators available in the library. We are using a prebuilt decorator named as route. So, app.route. And, the argument we need to provide here is the actual URL which you want the user to move to find the output of this function.

So, since this is the home page, we will use only the slash operator. So, anybody, who will just start this app, will first move to this route directory and this route directory will return this string to the user. Now, this is all that our app can do till now, but to run this app we have to also provide a main function here and this we can do by using 'if name'. So, this is a special variable which we have already discussed and 'if name' = main. So, whenever you run your programs if you directly run a particular script, then the value of this variable becomes this.

So, if I am writing in my terminal that python DSS.py. So, this predefined underscore, underscore, name, underscore, underscore will automatically get the value of this one. So, if this happens, I am writing this conditional statement that whenever this happens I need to run, 'app.run' with an argument debug = true. So, we will see what all these things will do. So, the basic structure of our application is complete and we can run this file.

So, the name of this file is DSS.py. So, I have already written here that run python, using python this file DSS.py. So, let us see the output first. So, when I wrote python DSS.py, the flask library which we just installed, it started computing what to do. So, the first output it gave is, it is serving flask app DSS. Then, it showed that the debug mode is on. What is debug mode I will tell you, but it is also giving this warning, this is a development server.

So, whatever we are doing this is not the actual production server that we are using here. So, flask identified this and it is giving this warning that do not use it in a production deployment. So, until unless our application is complete, we will keep on working with this development server, then it is saying that the app is running on this URL. So, this is a localhost at port 5000. So, this is the default port, but we can change the port if we want, by changing it by providing the port and the local domain name in this 'app.run' argument function.

Also, it is telling that if you want to close this app, then you can use control +c to quit, and then the debugger is active and a pin has been generated. Now, we can copy this and let us see what it will show in our browser. So, you can see if I zoom in, it is showing welcome to DSS.

So, the URL is this one, this is available on localhost at port 5000, and this is the output of the complete application which we have written here. Now, if you change this value, welcome to the first DSS, and reload this page then you can see that it has been changed here, but if you don't run this with debug = true, then it will not show this change until unless you restart your complete app.

So, these are few things that debug = true argument gives you as an added functionality. Also, if you keep on writing your code and you get some error, then this debug = true will show you all the errors that the python interpreter can tell you at this page only.

So, this is the basic app and you can do anything on this DSS application which you have learned in our previous course by using HTML, CSS and JavaScript or other languages and now we will try to create the next version of the DSS by performing this task which we have seen earlier that we need to generate a random number between 100 and then decide whether the number generated is less than 50 or not. So, let us start coding that again. So, to do that, we first need to generate our random number.

So, we have already imported our scientific library NumPy with an alias of np. So, to generate a random integer, we need to use 'np.random.randint'. And, the low value is 1, high is 100 and step is 1. So, we can skip that. Now, let us store this generated random number into a variable name random underscore num. Now, comes our very basic decision model. And, to write that we will use our 'if statements'.

So, 'if random underscore num' is less than equal to 50. So, we need to do two tasks. First, let us write a variable string 1 and store it, that we are calculating the square and compute the square of this also. So, the result = we are storing the result in this and, here we need to write the cube because the task was to calculate the cube if the number is less than equal to 50 and the result should be a random underscore num to the power of 3. So, I am using the exponent operator here, but what if, this is not the case.

So, let us write the as statement as well and we will write string 1 = square and result equals random underscore num to the power of 2. We can also print this result and we will see where this print statement will work because this print statement will not directly print the content to your browser, that is your DSS. So, let us see what it will do. Now, let us run the system again. If I reload this, nothing has changed, but you can see here that some number has been printed in here which is 9409.

So, what is this number? Let us print this string also. So, let us reload it again and now you can see that a number is printed 5041 and it is showing that square is the string inside str1. That means the number was greater than 50, that is why this block of code was executed by the interpreter. But nothing is changing here. Why? because we are not telling anything to the flask DSS system that we need to change something here also. So, to do that, either we can print these things using this statement or what we can do is, we can create an HTML file. And then, try to expand the capabilities of our DSS, and then use the

markup languages and provide the output in a neat form. So, to do that, there is one more function available in this flask library which is called `render_underscore_template`.

Now, we have imported this as well. But, to render a template, we first need to create a template. So, how to create a template? You need to go to your source directory where this `DSS.py` is available. Now, create a new directory here, that is a new folder and write templates. So, your flask application will automatically understand that this is the directory for templates. Why? Because while initializing, we wrote this variable as its argument. So, whatever is happening by default where this variable is playing its role.

So, your flask application will look for this directory, wherever this `DSS.py` is available. Now, in the templates directory, you can create a new file and call it `home.HTML`. And, to write a basic structure of HTML you can write. This is the autocomplete feature available in this vs code environment and here we can write DSS. So, this is the title of our Web-Based DSS and we can write here. So, now this is the structure of our home page. And, to deliver this home page by this application, we already imported this `render_template` function.

Now, we have to use this `render_template` function. Initially, we were using this return statement but let me comment on this, so that we can see what we are changing now. So, let us write our new return statement by using that `render_template` function and provide the first argument to this `render_template` function as the name of that template.

So, you do not need to provide the complete directory location, and then the file location. You only need to provide the `home.HTML`, that is the file name. And, now what this function will do is, it will route all the contents of this function. And, whatever this function will return, it will return it as a template. In which template? In the `home.HTML` template. So, let us restart this.

Actually, we do not need to restart, but it is best to restart, since we have already used `debug = true`. So, usually this development server will try to restart whenever there is some change or modification in this file. So, in the `home.HTML` template, which is here, we have written in our body section that write this welcome to DSS string in the center using the `h1` tag. So, this is what is happening here now in our DSS system. So, nothing else has changed, only this string is coming up.

Now, how to add all these decisions, we are taking, how to add all these things in our template. So, to do that, we can send these variables to this template using this `render_template` function and you can write any variable name here and store these variables which you want to transfer to your template using that. So, let me just use the same name. So, I need to transfer string `l` and what I need to transfer in it is, this string value, this `str1`. Also, I need to transfer this other variable, that is the result.

And, I am storing this into the result. So, this orange part, this is the variable name that is available in your template and this white colored variable, this is the actual variable that is available in this python script which we are printing here. So, whenever we try to use a print statement in this flask application, it will print in our development server, like this cube 21952. Otherwise, if we want to print anything on

our DSS, then we need to transfer that variable to our template using this method. So, now the application knows that we are transferring this, but how to capture this in our template? We need to add one more thing here. So, let us write a basic paragraph and say the operation performed was `str1`, the result is.

So, now this is a new syntax which we have not discussed earlier because this is available only in the flask application. So, flask provides this templating engine known as Jinja, and this Jinja templating engine can take the dynamic variables that we are transferring to our template and capture this using these punctuation marks.

So, between two curly braces, we can write the variable name that we are transferring from our script to our template. So, whatever the variable name we write here, it will print the value that was transferred from this return statement. So, we have saved both the files, now let us see, how it will work.

I am reloading the page. Now, you can see that the operation performed was square and the result is 3844. As many times you reload, the function will keep on running differently and will provide new output every time. So, now it is a cube and it is 97366. Similarly, it has changed. So, it is working fine. Now, our decision is happening correctly and our result is also getting printed on our DSS system.

The next thing is how to use static files to beautify or to make the system useful for any user to understand easily. So, to do that, until now, let me summarize what we have done. So, we started with a blank script and imported our two main libraries that we need today, which were NumPy and Flask. Then, from Flask we imported two important functions, that is, this Flask function and the render template function.

After that, we initialized our application using, inside this `app` variable, you can write anything here, I am using this `app`. Now, using this `app` variable, we first created a basic function which we learned in our earlier lectures. And, in that function, we initially returned a very basic string and saw that it was successfully printing that string on our DSS.

The next thing was that we developed our decision model which was to check whether a randomly generated number was less than or equal to 50 or not. And based on that, we performed a basic computation of cubing that number.

Otherwise, if it was greater than 50, then we squared that number. Then, we returned that computation and this string to the template. So, to use a template, we first created our directory. Inside that directory, we created our HTML file and provided the name of that HTML file in this render template, and further we also transferred our variables that were dynamically generated using the same render template function.

Now, the next thing is to create static files, so to do that, we need to create one new directory which is static, and we can write here a new file named `style.CSS`. You can write any name here, but the folder name should remain the same, static. The file name could be anything.

Now, let us create the complete structure of this home.HTML. So, to do that, we have already created our header. Next is, we will create a new element named div, and give it a class which we have not yet initialized but we will do it in our style.CSS, and name this class as container.

In this basic container, we will create two more divs. The first is, let us say, having the class named left containers and the next is having a class named right. Now, we can populate the contents of this container as well, so we can just inform the user what this DSS is all about, so that is our task. So, this DSS will generate a random number between one and hundred, it will check whether the number is less than fifty or not. If the number is less than equal to fifty, then DSS will compute a queue of the number. If the number is larger than DSS will compute the square of the number. So, this is the information that our DSS will show.

Now, on the right section, we will produce our output which is, let us start a paragraph and write the random number generated is random underscore num, and then also inform the user the operation on the number is num underscore op. Let us write result here.

So, now we are trying to inform the user that the random number generated is inside this random num. So, we need to transfer this variable as well here. So, we can write random underscore num = random underscore num, also we need to tell this operation. So, we can write ops = str1 and we can remove this because we are not using it anymore. So, now our render template will transfer the result variable, random number variable, and the operation we are performing, and we can remove these things of course, this is also no use now.

Now, we are going to run this once, and see whether we are getting any error or not. So, now it is showing that this is the task and the random number generated is 27 and the cube on the number is 19683. So, we can write a cube of. But, it is still not styled well. We have created our containers here like right or left, but they are not showing any difference.

So, we need to style these containers also. So, to do that, we have already created this style.CSS, and now we will write the definition of all these containers. So, as we have already discussed in our previous course that to style a class variable in CSS, we need to start it with a.operator. So, now let us write the CSS properties here. So, first we start by using the display property with the flex value and add the justify content to it, and use the space around option. After that, we will also say that the flex direction should be row and flex wrap should also be row.

The row gap between these containers should be 2 times rem value, and the column gap should also be 2 times rem. All these values are dependent on the programmer and what type of display they want to show to their user. So, I am just using it since we are just creating a very basic DSS. So, I am just using very basic values, align content and provide the value as centers.

Now, our container is developed, but we have two more containers inside this container which was named as left and right. So, for the left column, let us write the DSS properties. So, let us say, border is 2px dashed black padding + 2px dashed black padding 2rem, aligned content is center for the right. Container,

which contains the output of our decision model. So, let us change the background color for this as beige provides a border to these containers. 2px dashed gray provides a border radius as 7px, provides padding the same as the left container which is 2rem and aligns all the content to centers.

So, our style is complete, now let us reason our code. So, we reloaded our application, but nothing changed. Even after providing all the styles, why? Because, we are not telling our template that we need to use a different style for this page. So, to do that, we need to use this link argument here and we need to tell that it is a style sheet and the URL for this is. So, if we are just using a basic web development system, then we can provide the actual path here, but since we are developing the system through a different library named as flask.

So, we need to provide this dynamic location of this file. So, to do anything dynamic, we are performing in our template, we need to use the Jinjan templating syntax. So, we will use these curly braces and write this new function URL underscore for which is available in flask applications and we write that the name of our stylesheet is we are performing the name of the stylesheet and the location inside this URL underscore for function. The directory under which this is placed is the static and the file name is the style.CSS.

Now, let us run this again and reload the page. So, now you can see that our system has changed. Now, it is neatly showing that this was our task that this DSS will generate a random number between 1 and 100. So, when we loaded this page the current random number generated is 3 and the cube of the number is 27. If we reload it again it will keep on reloading the style on the template and your decision models output.

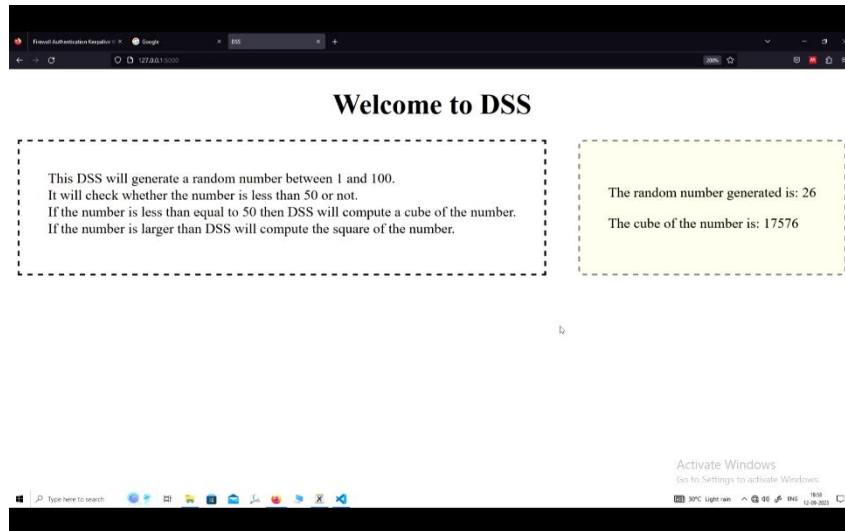
So, let us reload it again. So, now it is 10, 1000, now 24, 91. So, now you can see here that when the number is greater than 50, then this text is also changing. So, now it is showing that the square of the number is this. If we let us reload it again. Now, since it is less than 50. So, the number is 35 and the text is the cube of the number.

So, we can manipulate anything on this system. and now this is the root web page because we are writing this. So, if you write google.com, now this is the base URL of this, but we can also write about here also or any other URL that is available inside this website. So, we can also perform this operation in our DSS as well. So, to do that, we need to add one new function here, like this.

Let us say we are writing def about. So, we are creating a new function for our about page and the dobsting should tell the programmer that this page presents the information about the DSS and let us return only the address of the single string as earlier which is about page. And to route this, we need to create an app.route and here the URL should change from slash to about slash.

Now, we have two pages in our complete DSS. So, let us see since debug mode has already reloaded this. So, let us go to a new page. Now, you can see that this is the new page available on the same URL that

is localhost at the port 5000 and here you can add any information which you want to write about this our page and if you just remove this, then your actual DSS is also available here.



-The final Result

So, this way you can create your own Decision Support Systems. And, in the next few lectures, we will try to develop a complete Decision Support Systems for inventory systems and other decision trees which was already discussed by professor Philip in this course and we will try to see how to manipulate and tweak our code to get some significant insights from our Decision Support Systems. So, we will meet in the next lecture. Thank you.