FLASK Notes :)

All the codes are available at: https://github.com/AS-DB/Python

First Flask Application (Hello_world.py)

1>route method is called as decorter which tells flask server which function is assoicited with with this URL.
2>run method will excute our flask application
3>debug=True :while using this inside the app.run() we don't need to stop the server to make changes, while active we can make changes.

Data Types Passing in url (Data_types.py)

1>to pass data types in url we have to follow below syntax

```
app.route('/Datatype:Name')
```

2>Then you can call the Name in function.
3>String is the only data type which do not need to mentioned its data type
4>We have different data types here like int,float,byte,etc...
5> syntax to declare it in app.route("/int:variable")
6>want to declare more data types in it then syntax is: app.route("/int:variable")

Building the URL dynammically (dynammically_url.py)

1> redirect and url_for are imported from Flask to handle redirections and URL building.
2> @app.route('/<guest>') handles string parameters in the URL.
3> The guest1() function dynamically displays the guest's name passed through the URL.
4> @app.route('/user/<name>') handles user-based redirection.
5> If the name parameter is 'admin', the user is redirected to the /admin route.
6> If the name parameter is not 'admin', the user is redirected to the /guest route with the provided name.
7> url_for() dynamically builds the URL for the given function name.

Different Https Methods in Flask (Https.py ,Form.html)

1> redirect, url_for, and request are imported from Flask to handle redirections, URL building, and form data retrieval.
2> @app.route('/login', methods=['POST', 'GET']) allows both POST and GET requests for the /login route.
3> In the POST method, request.form['nm'] retrieves the form data from the submitted HTML form.
4> In the GET method, request.args.get('nm') retrieves the query parameter from the URL.
5> The login() function redirects the user to the /welcome/<guest> route using url_for() with the form data.
6> form.html contains an HTML form that sends data to /login using the POST method.
7> The form action is set to
http://localhost:5000/login, which submits the form data to the Flask server.
8> <input type="text" name="nm"> allows the user to enter their name, which is passed to the server.
9> <input type="submit" value="submit"> creates a submit button for sending the form data.

Template in Flask (Template.py)

1> render_template is imported from Flask to render HTML templates.
2> @app.route('/a') returns HTML content directly using a string with HTML tags.
3> The index1() function returns an HTML string containing <h1>Hello</h1>.
4> @app.route('/') uses render_template() to render the hello.html file.
5> render_template() dynamically loads and displays the HTML file from the templates folder.
6> The templates folder is the default location for storing HTML files in Flask.

Static Files in Flask (Static.py,index.html,hello.js)

1> render_template is imported from Flask to render HTML templates.
2> @app.route('/') serves the index.html file using render_template().
3> The index.html file links to an external JavaScript file (hello.js) using:

```
<script src="{{ url_for('static', filename='hello.js') }}">
```

4> url_for('static', filename='hello.js') dynamically builds the URL for the static JavaScript file.
5> Flask uses a static folder by convention to store static files (CSS, JS, images, etc.).
6> The hello.js file contains a sayHello() function that displays an alert with the message "Hello world".
7> The HTML button triggers the sayHello() function when clicked using:

```
<input type="button" onclick="sayHello()" value="Say Hello"/>
```

Flask Framework Request from Object(student.html,result.html,Request_object.py)

1> render_template and request are imported from Flask to render HTML templates and handle form data.
2> @app.route('/') serves the student.html form using render_template().
3> The student.html file contains a form with fields for Name, IOT, CD, and MPMC scores.
4> The form action sends data to
http://localhost:5000/result using the POST method.
5> @app.route('/result', methods=['POST', 'GET']) handles both POST and GET requests.
6> request.form retrieves the form data submitted by the user.
7> The form data is passed to the result.html template using:

```
return render_template("result.html", result=result)
```

8> The result.html template displays the form data in a table using a Jinja2 for loop:

```
{% for key, value in result.items() %}
```

9> The table uses {{ key }} and {{ value }} to dynamically display the form field names and values.

Cookies in Flask Application(cookie.py,setcookie.html,readcookie.html)

1> render_template, request, and make_response are imported from Flask to render HTML templates, handle form data, and create HTTP responses.
2> timedelta is imported from datetime to set the cookie expiration time.
3> @app.route('/') serves the setcookie.html form using render_template().
4> setcookie.html contains a form that sends the user ID to /setcookie using the POST method.
5> @app.route('/setcookie', methods=['POST', 'GET']) handles POST and GET requests for setting cookies.
6> make_response() creates an HTTP response to send the cookie along with the rendered readcookie.html template.
7> resp.set_cookie('userID', user, max_age=timedelta(days=30)) creates a cookie named userID with the form value and sets its expiration to 30 days.
8> @app.route('/getcookie') retrieves the cookie value using:
name = request.cookies.get('userID')
9> The /getcookie route displays the user ID stored in the cookie using:

```
<h1>Welcome {name}</h1>
```

10> readcookie.html contains a link to /getcookie to read the stored cookie.

Flask Redirects and Errors (redirect.py,login.html)

1> redirect, url_for, render_template, and request are imported from Flask to handle redirections, form data, and template rendering.
2> @app.route('/') serves the login.html form using render_template().
3> login.html contains a form that sends the user ID to /login using the POST method.
4> @app.route('/login', methods=['POST', 'GET']) handles both POST and GET requests.
5> If the username is 'admin', the user is redirected to the /success route.

6> If the username is incorrect, the user is redirected back to the / route.
7> url_for('success') dynamically builds the URL for the /success route.
8> @app.route('/success') displays a success message with the text:

<h1>Logged in successfully!</h1>
9> The form action uses {{ url_for('login') }} to dynamically generate the URL for the /login route.

Flask Redirects and Errors (abort.py)

1> redirect, url_for, render_template, request, and abort are imported from Flask to handle redirections, form data, and aborting requests.
2> @app.route('/') serves the login.html form using render_template().
3> @app.route('/login', methods=['POST', 'GET']) handles both POST and GET requests.
4> If the form is submitted using the POST method and the username is 'admin':
The user is redirected to the /success route.

5> If the username is incorrect, abort(401) returns a 401 Unauthorized error.
6> If the request method is GET, the user is redirected back to the / route.
7> @app.route('/success') displays a success message with the text:

<h1>Logged in successfully!</h1>
