首先我们用到了保存全局变量的g属性：

g: global

g对象解释： 就是为了保存用户一些自定义参数

g对象是专用用来保存用户的数据的。

g对象在一次请求中，全局可以调用。

1. Separate code blocks and explanations for each of the python functions

We write three python functions for database management in a file app.py.

1. We define get\_message\_db () to handle creating the database of messages.
2. Firstly, we check whether there is a database called message\_db in the g attribute of the app. If not, then connect to that database.

(2) We check whether a table called messages exists in message\_db, if there is no table available, we create the table messages with three columns: ID as an integer, handle as text, and message as text.

(3)return the connection g.message\_db.

1. The second function we created is called insert\_message() which should handle inserting a user message into the database of messages.

(1) We extract the message and the handle from request if they exist. We should ensure submit.html template creates these fields from user input.

(2) Then we connect the database using cursor.

(3) To ensure that the ID number of each message is unique, we set the ID number equal to one plus the current number of rows.

(4) Write a SQL command in a string called g.sql with variable input value, then perform the insertion with cursor.exectute(g.sql).

(5) Run db.commit() to ensure the row insertion has been saved.

1. The third function we created is random\_messages(n), which fetches n random messages from the message\_db.
2. We connect the database using cursor.
3. Execute sql command to extract the message from message database in random order.
4. Get n rows from the result.
5. Close the cursor and the database.
6. Return the result.
7. This is the fourth function I created. The function is straightforward, as it gives us the main page of the Webapp.

**I put navigation links(insert and view links at the top of the screen) inside a template called base.html provided by the professor. Then I had the main.html, submit.html, insert.html templates extend base.html.**

1. The fifth function we created is called view (). The function creates a viewable webpage.
2. Call the template view.html and fetch the number of messages from the data we inputted.
3. We will fetch a function called random\_messges(n) from db\_app.py to pick n messages from the database randomly.
4. If we submitted nothing, the program outputs error.
5. The last function we defined is called submit ().
6. From template submit.html, we extract handle and message from user input.
7. If they exist, we will fetch a function called insert\_message from db\_app.py.
8. The insert\_message () function will insert a user message into the database of messages.
9. Discussion of a template file I created in my app

I choose to discuss the template file view.html. The file view.html presents us with the random messages chosen if we click on a button.

1. First, we display the line of text “please input how many messages do you want to view (1-5)?” on the website.
2. Then we design the input box with the type of number so that 只允许输入数字，而且数值的范围从1 to 5.
3. 当我们点击“submit”按钮，会根据我们输入的数值n，调用app.py中的函数random\_messages(n)，从messages表中返回n条记录，并记录到g.result中。
4. return render\_template('view.html')返回到view.html, 在页面的下端显示n条记录，方法是做一个循环，依次取出g.result中的每一条记录送入r，r[0]是message，r[1]是handle，用斜体显示。

We take advantage of the fact the Jinja tags support looping and indexing of objects, so we looped over the messages.

If r is a tuple of user handle and message, r[0]contains the message and r[1] contains the handle.

1. 如果出错，则显示出错信息。

All in all, we create a header, tell the user to submit some interesting message, and finish the html by giving some font and size.