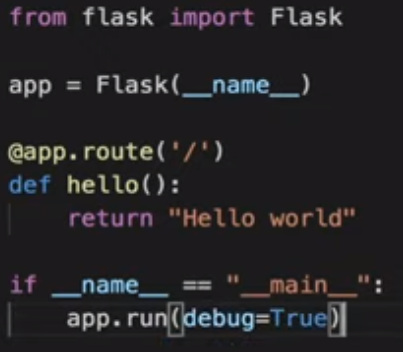
# Flask Web Framework:

## Documentation:

**Link** -- <https://exploreflask.com/en/latest/forms.html#:~:text=validate_on_submit()%20.,we%20defined%20in%20forms.py.>

## Running Simple Flask app:



Route defines the route or the **URL** path.

Whatever is returned by the function is read by the browser (**text, html**).

**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 app.run (port = 5000, debug = **True**)

## Running from command line:

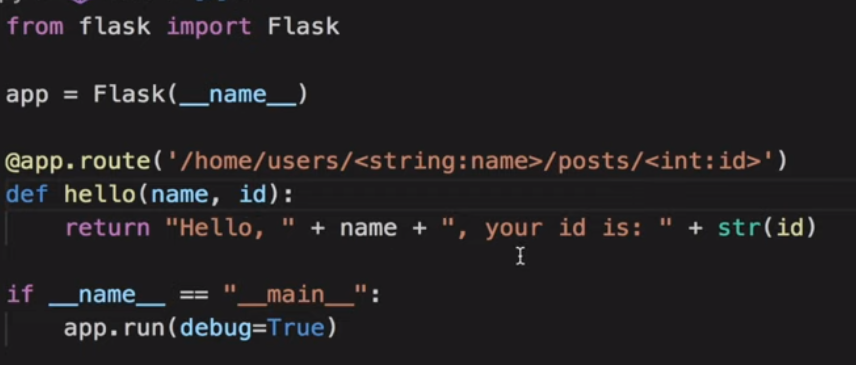
<https://flask.palletsprojects.com/en/1.1.x/quickstart/>

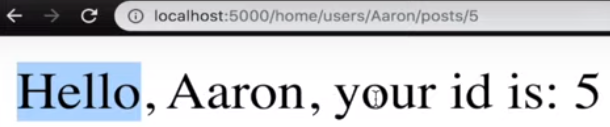
## Conditional title:



## Dynamic URL:

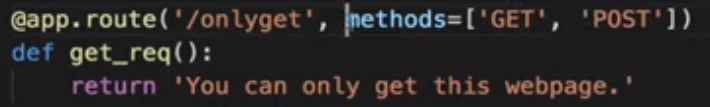
Routing with variables using the <> notation.



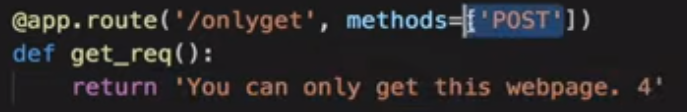


## Methods = [‘Get’, ‘Post’]





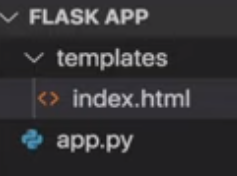
The above 2 examples are valid.



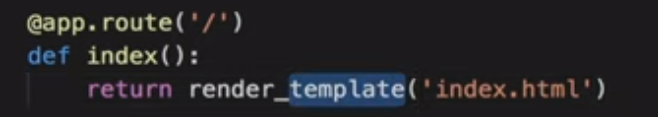
Will give error as whenever we request a webpage we are trying to “**get**” it but here only “**post**” method is allowed.

## Templates:

Create a directory called templates and place your html code here.

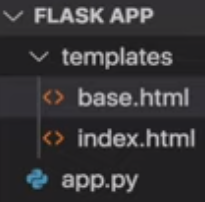


from **flask** import **Flask, render\_template**

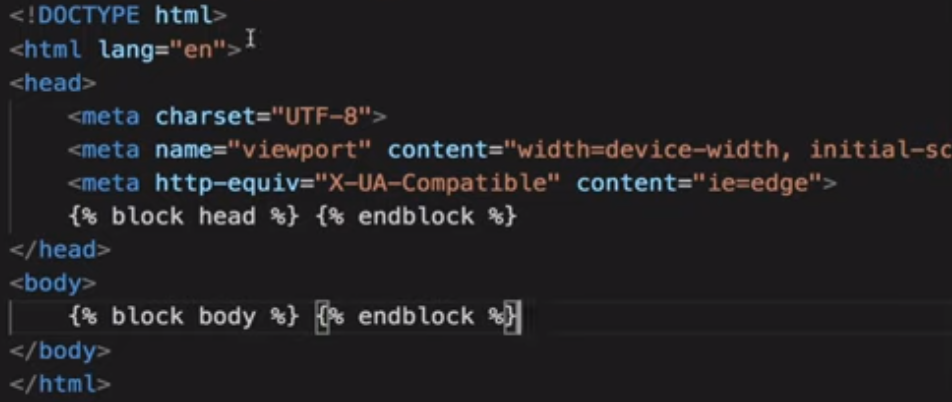


## Template inheritance:

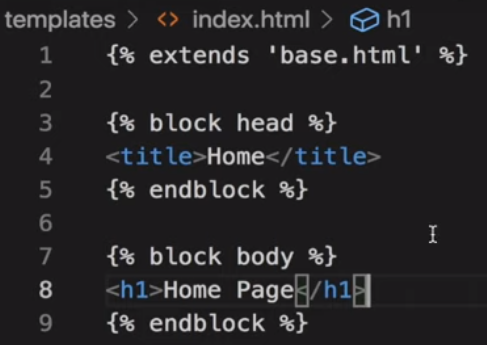
Folder structure:



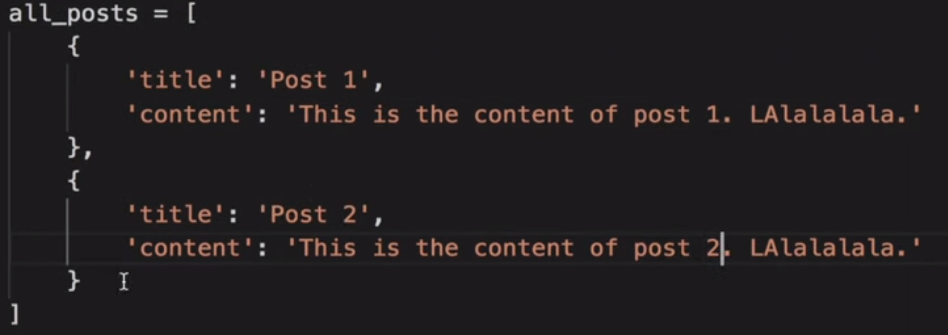
Creating **base.html using ginger syntax {% %}**



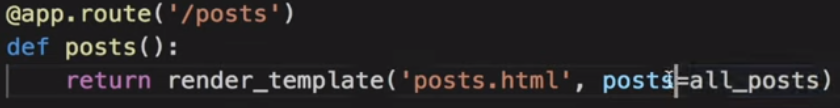
Creating **index.html using ginger syntax {% %}**



## Sending data in variables:



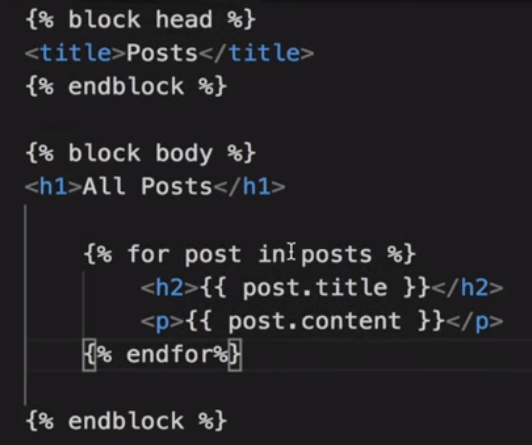
Send data as **list of dictionaries** as it is easy to access.



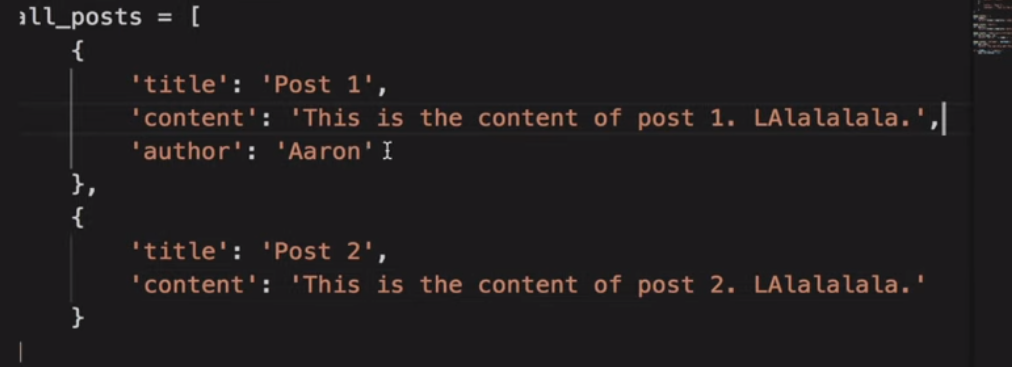
**{{Variables or value but not html tags }}** 🡪 Used for printing data

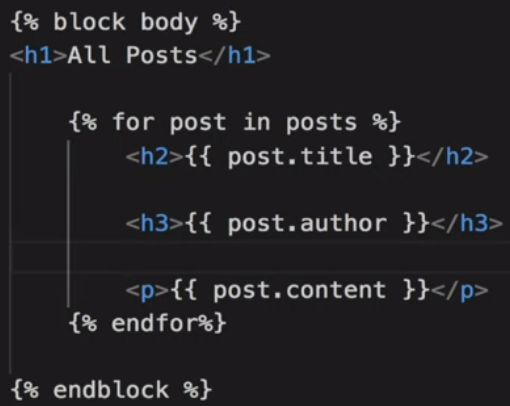
**{% %}** 🡪 Used for putting syntax elements

Create posts.html in templates directory:

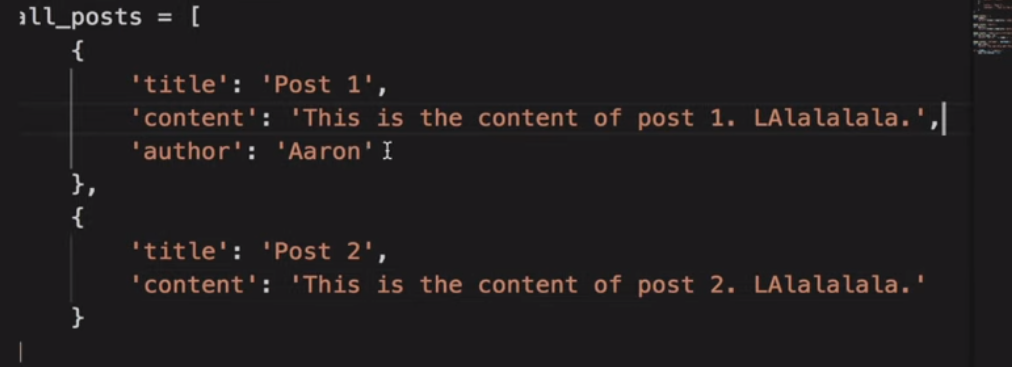


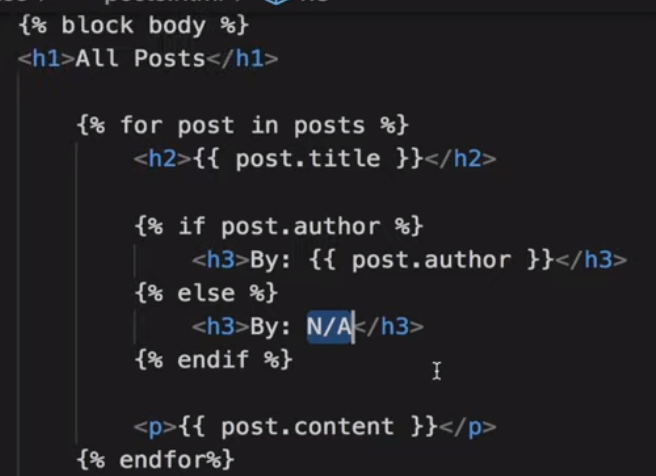
Version 2:





Version 3:





## Render\_template() func:

@app.route('/result',methods = ['POST', 'GET'])

def result():

if request.method == 'POST':

result = request.form

return render\_template("result.html", result = result)

We have to write **result=result** when passing input to render\_template, i.e., key = value

Variable\_name of both key and value must be same if value is stored in a variable. Eg: **result** variable

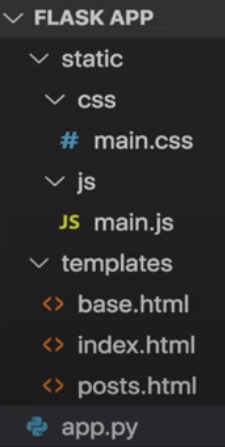
## url\_for():

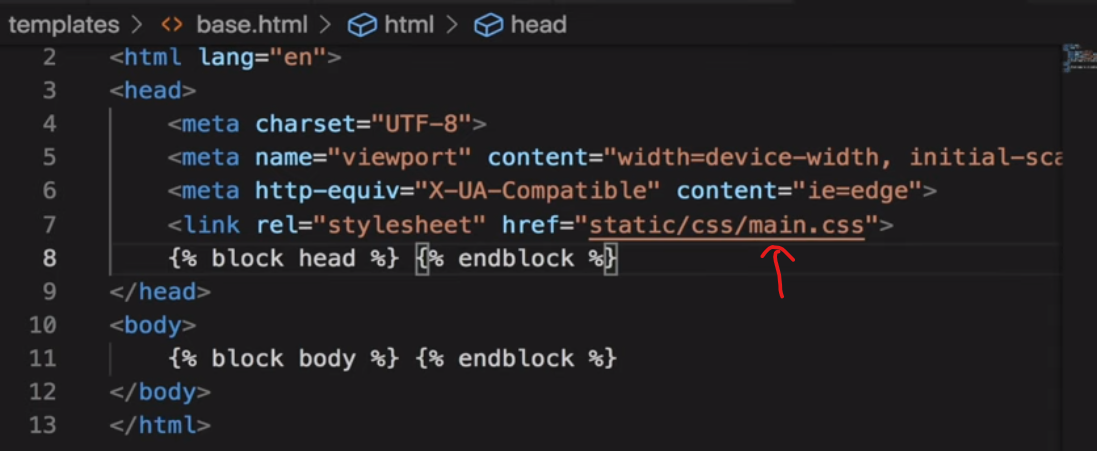
Instead of giving URL give url\_for().

url\_for (‘homepage’) is the name of the function and not the route.

## Static files:

It contains static files like **CSS** and **JS**.





Static files in flask should be linked using **url\_for()**.

Just change the link **href** to the following:



## Wt-forms:

**Must Read --**  <https://www.tutorialspoint.com/flask/flask_wtf.htm>

**Form.py**

**from** flask\_wtf **import** FlaskForm  
**from** wtforms **import** StringField, PasswordField, SubmitField, BooleanField  
**from** wtforms.validators **import** DataRequired, Length, Email, EqualTo

**class** RegistrationForm (FlaskForm):  
 username = StringField(**"Username"**, validators = [DataRequired(), Length(min=5, max=20)])  
 email = StringField(**"Email:"**, validators=[DataRequired(), Email()])  
 password = PasswordField(**"Password"**, validators=[DataRequired(), Length(min=8)])  
 confirm\_password = PasswordField(**"Confirm Password"**, validators=[DataRequired(), EqualTo(**'password'**)])  
 submit = SubmitField (**"Sign Up"**)

**class** LoginForm (FlaskForm):  
 email = StringField(**"Email:"**, validators=[DataRequired(), Email()])  
 password = PasswordField(**"Password"**, validators=[DataRequired(), Length(min=8)])  
 remember = BooleanField(**"Remember Me"**)  
 submit = SubmitField(**"Login"**)

**app.py**

**from** flask **import** Flask, render\_template, flash, redirect

**import** forms  
app = Flask(\_\_name\_\_)  
app.config[**"SECRET\_KEY"**] = **'4224303c1bac94a0e9c38ee7b3d3885e'**

@app.route(**'/register'**, methods = [**'GET'**,**'POST'**])  
**def** register():  
 form = forms.RegistrationForm()  
 **if** form.validate\_on\_submit():  
 flash(**f'username: {**form.username.data**} created successfully'**, **'successs'**)  
 **return** redirect(url\_for(**'homepage'**))  
 **return** render\_template(**"register.html"**,title = **"Registration Page"**, form=form)  
  
@app.route(**'/login'**,methods = [**'GET'**,**'POST'**])  
**def** login():  
 form = forms.LoginForm()  
 **return** render\_template(**"login.html"**,title = **"Login Page"**, form=form)  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 app.run(port = 5000, debug = **True**)

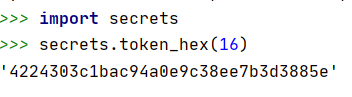
**Register.html:**

See the above mentioned link.

**Note:**

1. To access the value of form variables we need to use data attribute

## Secret Key:



**from** flask **import** Flask, render\_template  
  
app = Flask(\_\_name\_\_)  
app.config[**"SECRET\_KEY"**] = **'4224303c1bac94a0e9c38ee7b3d3885e'**

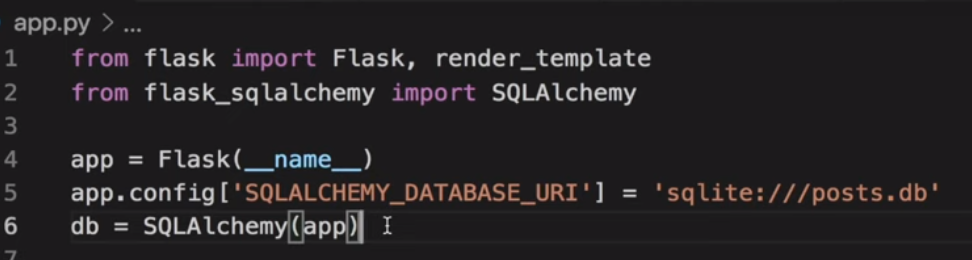
## Database Connectivity:

pip install **flask-sqlalchemy**

from **flask-sqlalchemy** import **SQLAlchemy**

The good thing with **SQLAlchemy** is that it allows us to use any database and it’s as simple as changing the root.

We can also use **PostgreSQL** or **MySQL** instead of **Sqlite**.



**SQLALCHEMY\_DATABASE\_URI** is path to where the database is stored.

**“///”** means relative path wrt **app.py.**

Whereas **“////”** means absolute path.

The path gives the location where the local \*.db file will be stored as sqlite creates a local \*.db file.

 defines the database config.

Creating **Model in model.py**:

Model is a way to structure our data in the database.

**from** app **import** db  
  
**class** User (db.Model):  
 id = db.Columns(db.Integer, primary\_key = **True**)  
 username = db.Columns(db.String(20), unique = **True**, nullable = **False**)  
 email = db.Columns(db.String(120), unique=**True**, nullable=**False**)  
 image\_file = db.Columns(db.String(20), nullable=**False**, default = **"default.jpg"**)  
 password = db.Columns(db.String(60), nullable=**False**)  
  
 **def** \_\_repr\_\_(self):  
 **return f'User: ("{**self.username**}", "{**self.email**}", "{**self.image\_file**}")'**

We can give the class name **anything** and it inherits from **db.Model.**

The default **date\_posted** is:

**from datetime import datetime**

**default = datetime.utcnow**

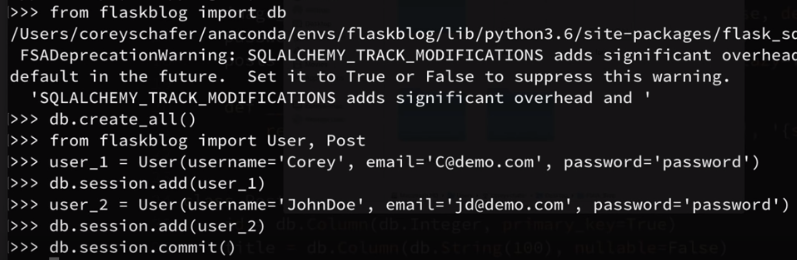
We don’t give parenthesis after the utcnow() function

Now to create the actual database we need to go to python terminal navigate to the **app.py** location

And then do,

**from app import db**

**db.create\_all()**

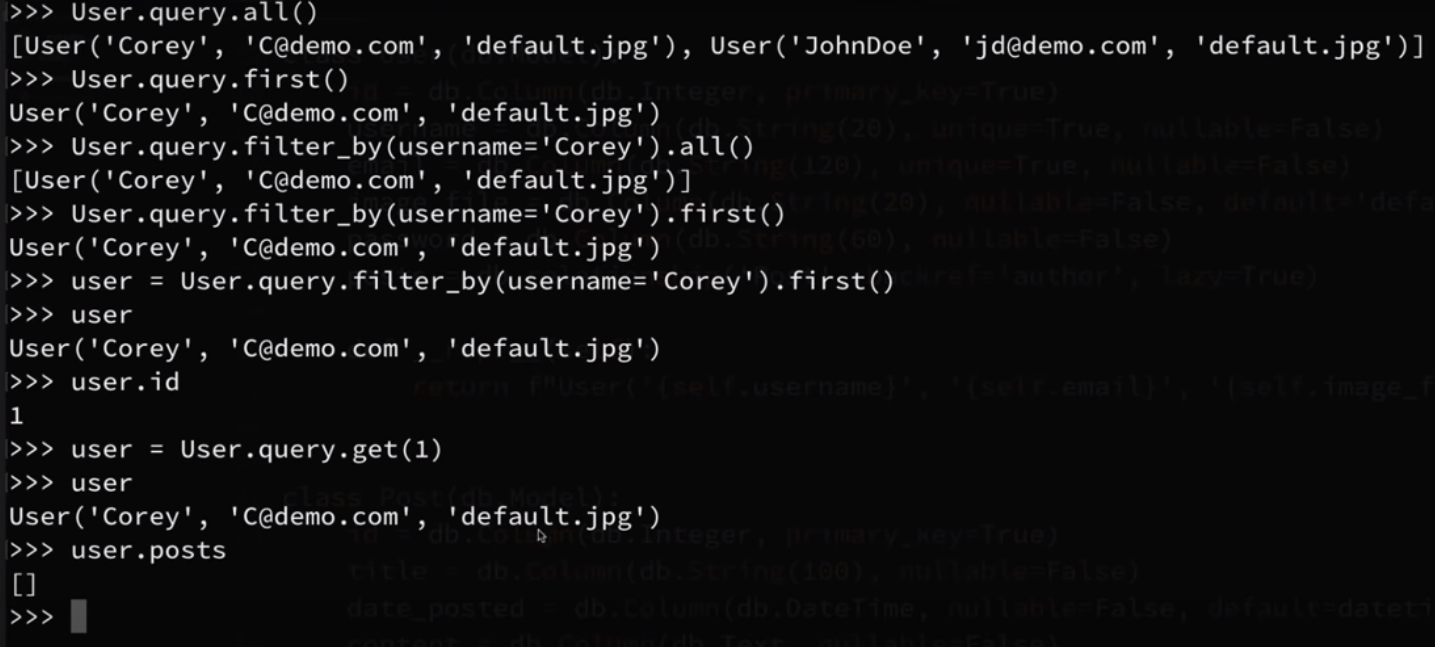


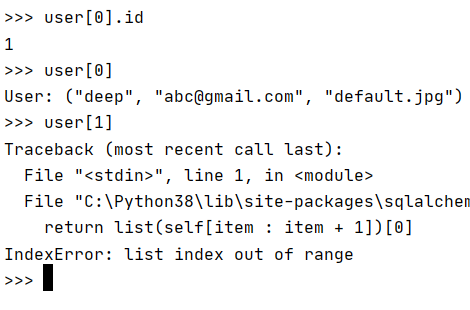
**User.query.all()** returns all rows stored in the database as a list.

It prints only the id as we have defined **\_\_repr\_\_()** such that only the **id** is printed but it contains data of the whole table.

**get()** or **get\_or\_404()** returns the model with that particular id.

**db.session.commit()** is required to commit the changes.



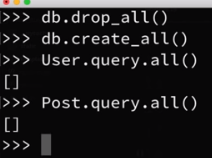




Instead of **first()** we can also use the **list index** like **Post.query.all()[0].**

## Delete post from database:

**Db.drop\_all()** clears the database.



## Circular import problem:

<https://www.youtube.com/watch?v=44PvX0Yv368&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=5>

See till 10 minutes.

See circular-import folder for explanation

## Converting to package:

<https://www.youtube.com/watch?v=44PvX0Yv368&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=5>

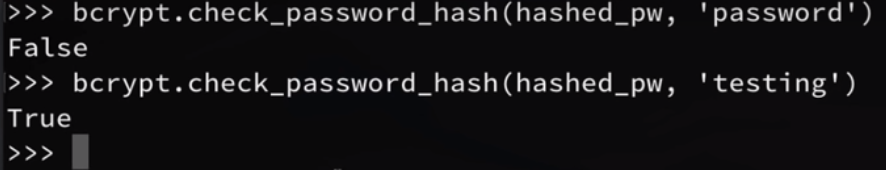
See after 10 minutes.

## Password Hashing:

**Pip install flask-bcrypt**



**Note** – the hash value generated for the same password every time is different.

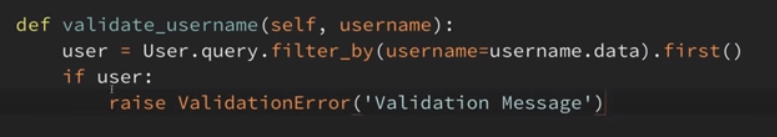


**routes.py**

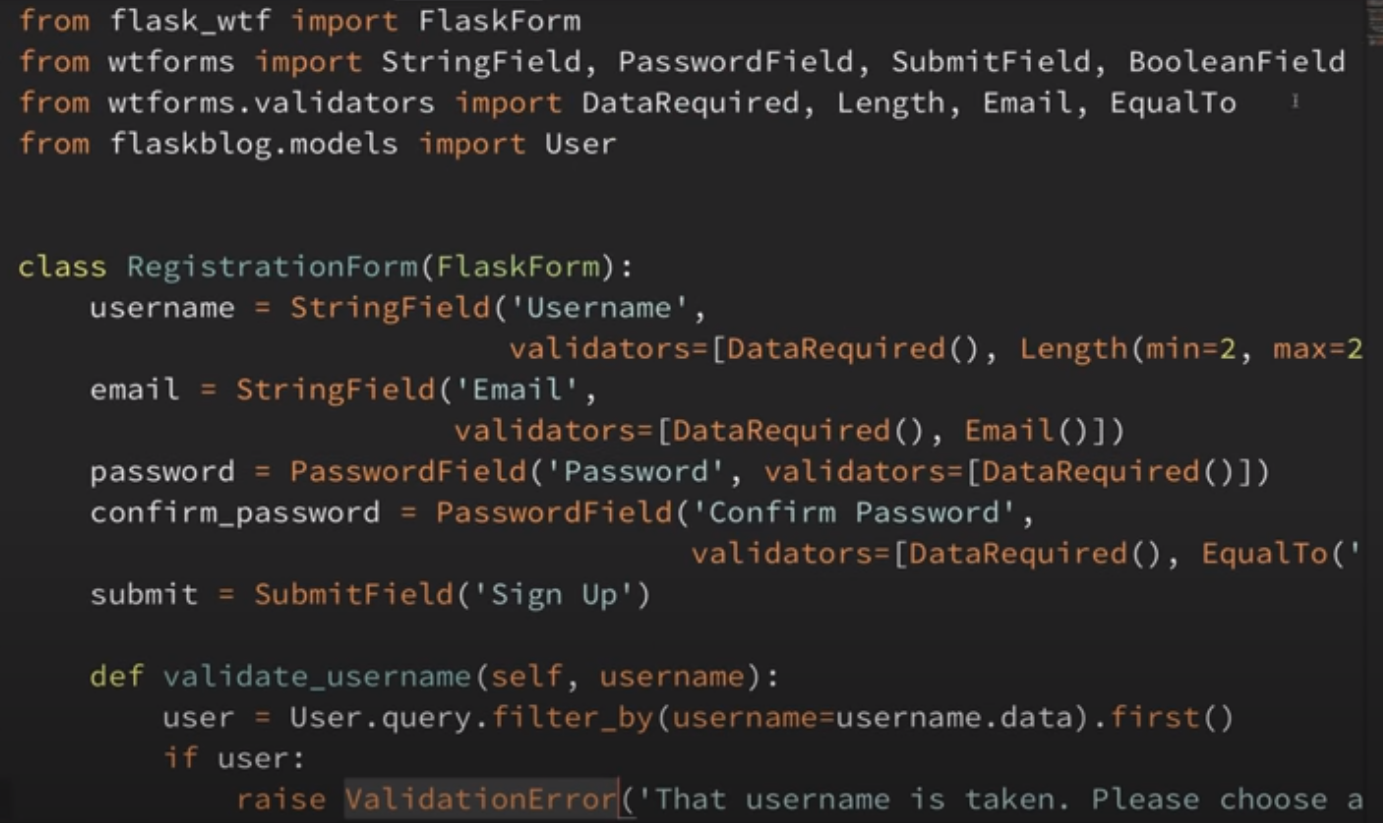
@app.route(**'/register'**, methods=[**'GET'**, **'POST'**])  
**def** register():  
 form = forms.RegistrationForm()  
 **if** form.validate\_on\_submit():  
 hashed\_pass = crypt.generate\_password\_hash(form.password.data).decode(**"utf-8"**)  
 user = dm.User(username= form.username.data, email=form.email.data, password=hashed\_pass)  
 db.session.add(user)  
 db.session.commit()  
 flash(**'Account created successfully! Now you can Log In.'**, **'success'**)  
 **return** redirect(url\_for(**'login'**))  
 **return** render\_template(**'register.html'**, title=**'Register'**, form=form)

## User Authentication:

<https://www.youtube.com/watch?v=CSHx6eCkmv0&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=6>

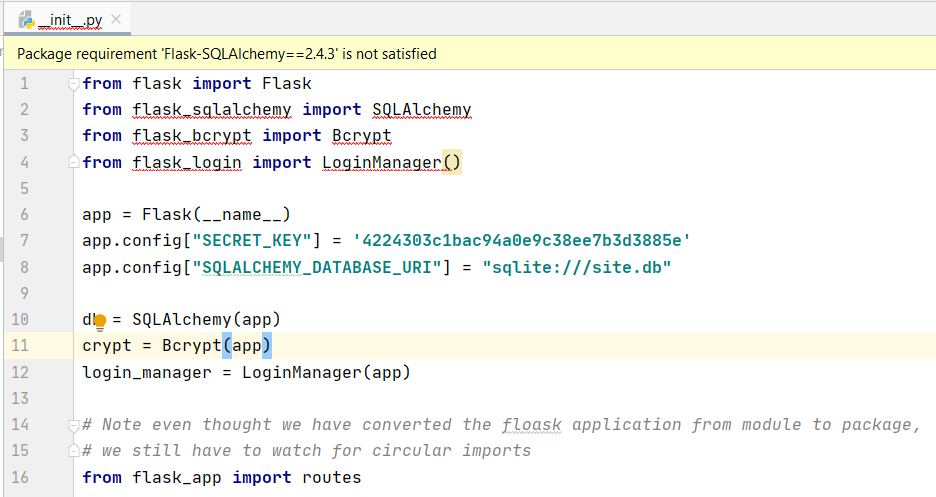


**forms.py**

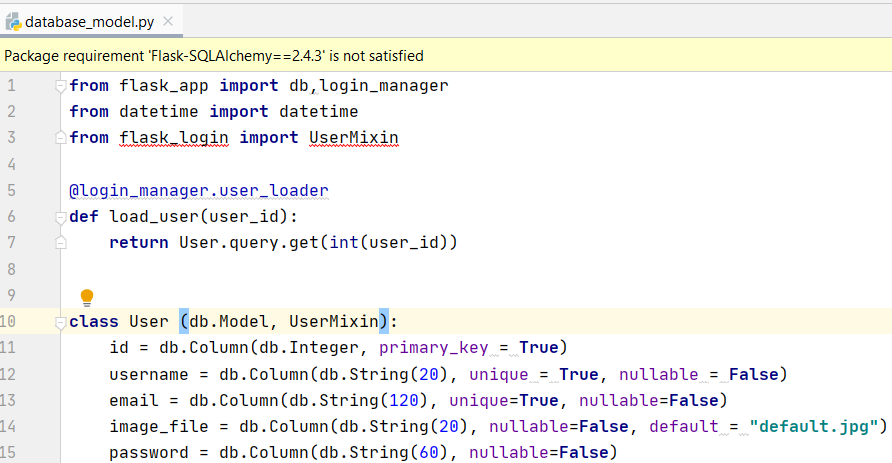


### Flask-login:

Pip install flask-login



**Database\_model.py**



**Routes.py:**

**from** flask **import** render\_template, flash, redirect, url\_for  
**from** flask\_bcrypt **import** check\_password\_hash  
**from** flask\_app **import** app, db, crypt, forms, database\_model **as** dm  
**from** flask\_login **import** login\_user, current\_user, logout\_user  
  
  
post\_data\_1 = [{**'title'**: **"Blog-1"**, **'desc'**: **'the first blog'**, **'author'**: **'deep'**, **'date\_posted'**: **'12th june'**},  
 {**'title'**: **"Blog-2"**, **'desc'**: **'the 2nd blog'**, **'author'**: **'alien'**, **'date\_posted'**: **'11th june'**}]  
  
  
@app.route(**'/'**)  
@app.route(**'/home'**)  
**def** home():  
 **return** render\_template(**"index.html"**, post\_data\_1 = post\_data\_1)  
  
  
@app.route(**'/about'**)  
**def** about():  
 **return** render\_template(**"about.html"**, title = **"About Page"**)  
  
  
@app.route(**'/register'**, methods=[**'GET'**, **'POST'**])  
**def** register():  
 *# if user is already logged in then it will not allow  
 # the user to sign up again.* **if** current\_user.is\_authenticated:  
 **return** redirect(url\_for(**'home'**))  
 form = forms.RegistrationForm()  
 **if** form.validate\_on\_submit():  
 hashed\_pass = crypt.generate\_password\_hash(form.password.data).decode(**"utf-8"**)  
 user = dm.User(username= form.username.data, email=form.email.data, password=hashed\_pass)  
 db.session.add(user)  
 db.session.commit()  
 flash(**'Account created successfully! Now you can Log In.'**, **'success'**)  
 **return** redirect(url\_for(**'login'**))  
 **return** render\_template(**'register.html'**, title=**'Register'**, form=form)  
  
  
@app.route(**'/login'**, methods=[**'GET'**,**'POST'**])  
**def** login():  
 *# if user is already logged in then it will not allow  
 # the user to login again.* **if** current\_user.is\_authenticated:  
 **return** redirect(url\_for(**'home'**))  
 form = forms.LoginForm()  
 **if** form.validate\_on\_submit():  
 user = forms.User.query.filter\_by(email = form.email.data).first()  
 **if** user **and** check\_password\_hash(user.password, form.password.data):  
 login\_user(user, remember = form.remember.data)  
 flash(**'You are logged in successfully'**, **'success'**)  
 **return** redirect(url\_for(**'home'**))  
 **else**:  
 flash(**'Wrong email or password entered. Try Again'**, **'danger'**)  
 **return** render\_template(**"login.html"**,title = **"Login Page"**, form=form)  
  
  
*# Creating log\_out route.*@app.route(**'/logout'**)  
**def** logout():  
 logout\_user()  
 **return** redirect(url\_for(**'home'**))

## Toggling display of buttons of page using if statement:

We just need to edit the **layout.html** file (in our case **basic\_page\_structure.html**)

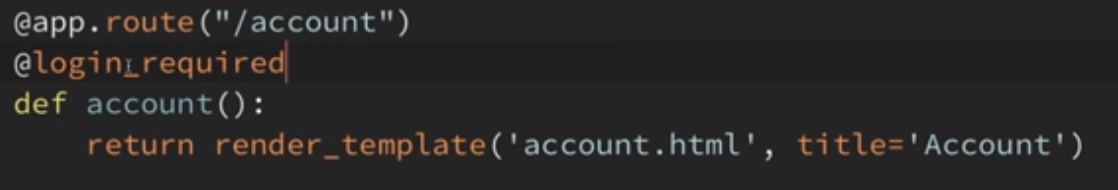
**Note:**

Nothing needs to be imported in any of the html files.

{% if current\_user.is\_authenticated %}  
 <**a class="btn btn-outline-success my-2 my-sm-0 mr-4 bg-light" href="{{ url\_for('logout') }}"**>Log Out</**a**>  
{% else %}  
 <**a class="btn btn-outline-success my-2 my-sm-0 mr-4 bg-light" href="{{ url\_for('login') }}"**>Log In</**a**>  
 <**a class="btn btn-outline-success my-2 my-sm-0 bg-light" href="{{ url\_for('register') }}"**>Sign Up</**a**>  
{% endif %}

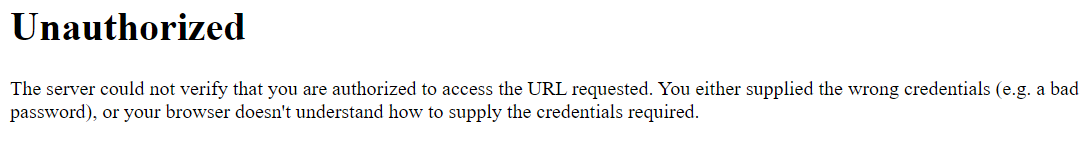
## Restricting link based on login:

**From flask\_login import login\_required**



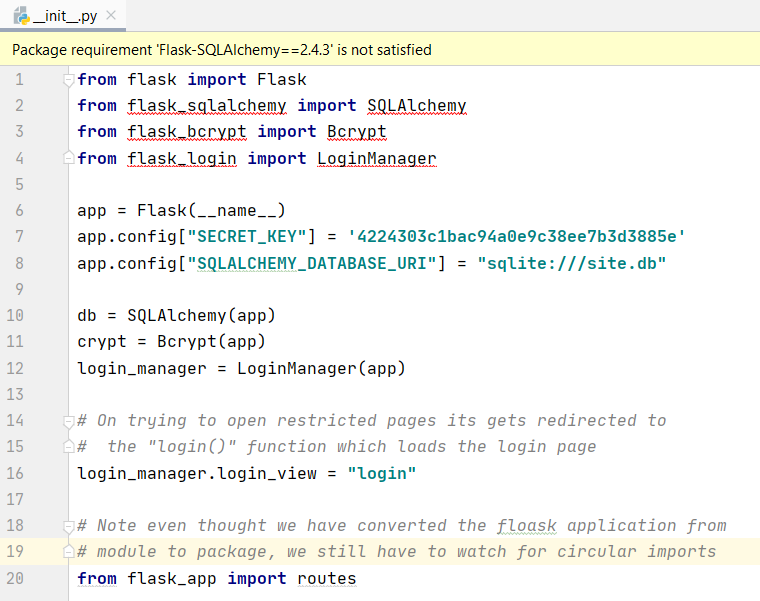
Try accessing - **/account** without logging in

Gives error on trying to access restricted page:



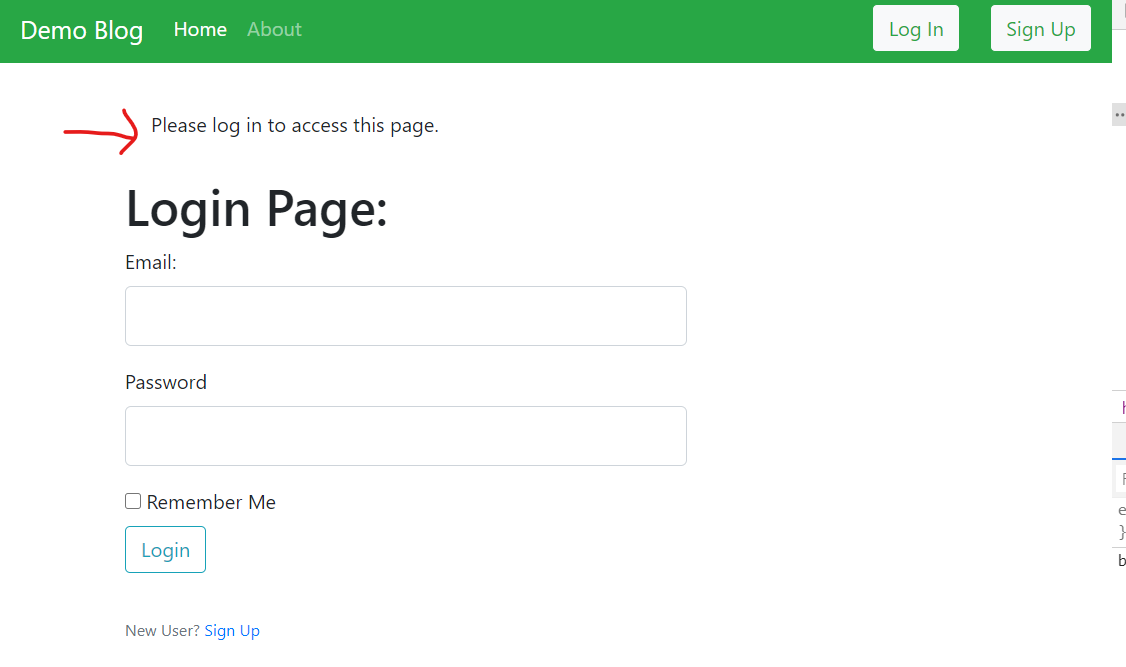
But we want to be redirected to the **login page** instead.

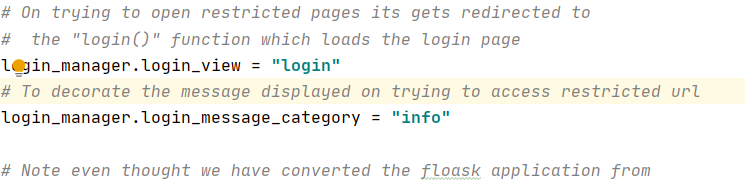
## Redirecting to the login page if not authorised:

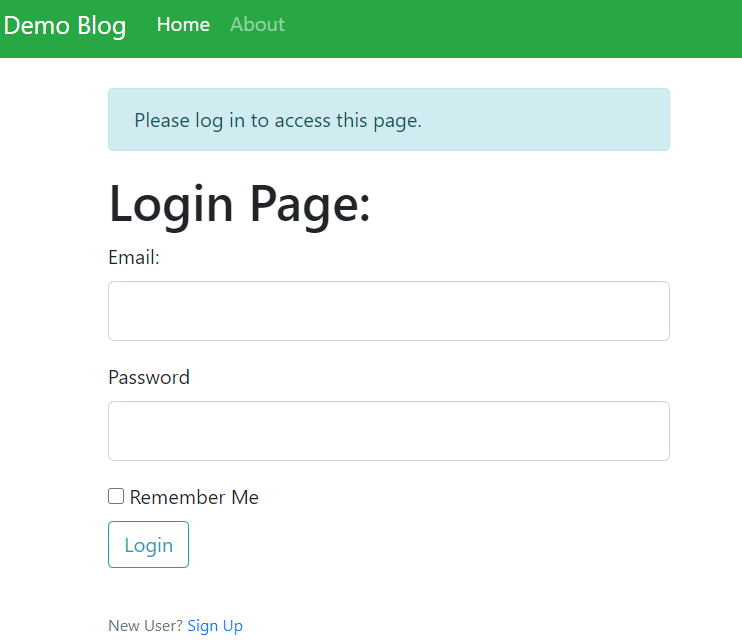


Try accessing - **/account** without logging in.

Now on trying to access restricted page , we get the following message and get redirected to the login page.



Now to decorate the message, we need to add this line followed by the bootstrap class name for styling.  




## Fixing redirects:

On getting a restricted url and on logging it, it will always take us to the home page as redirect is set to the “home” function.

But we want it to redirect to the page which it couldn’t access.

Eg: account page

Go to **routes.py:**

And edit the login function as follows:

**from** flask **import** request

…

…

…

**if** user **and** check\_password\_hash(user.password, form.password.data):  
 login\_user(user, remember = form.remember.data)  
 flash(**'You are logged in successfully'**, **'success'**)  
  
 next\_page = request.args.get('next', None)  
 if next\_page:  
 return redirect(url\_for(next\_page[1:]))  
 else:  
 return redirect(url\_for('home'))

**else**:  
 flash(**'Wrong email or password entered. Try Again'**, **'danger'**)

**request.args** returns a dictionary but we use the **get()** function as it will not give error if **key = ‘next’** is not present.

## Creating Form Update:

**Routes.py**

@app.route(**'/account'**, methods=[**'GET'**,**'POST'**])  
@login\_required  
**def** account():  
 form = forms.UpdateAccountForm()  
 image\_url = url\_for(**'static'**, filename = **"profile\_pics/"** + current\_user.image\_file )  
 **if** form.validate\_on\_submit():  
 **if** current\_user.username != form.username.data **or** current\_user.email != form.email.data:  
 current\_user.username = form.username.data  
 current\_user.email = form.email.data  
 db.session.commit()  
 flash(**'User details updated successfully.'**, **'success'**)  
 **return** redirect(url\_for(**'account'**))  
  
 *# to fill the updated form with values already present* **elif** request.method == **'GET'**:  
 form.username.data = current\_user.username  
 form.email.data = current\_user.email  
 **return** render\_template(**"account.html"**, title=**"Personal Profile"**, image\_url = image\_url, form = form)

**Forms.py**

**class** UpdateAccountForm (FlaskForm):  
 username = StringField(**"Username \*"**, validators = [DataRequired(), Length(min=3, max=20)])  
 email = StringField(**"Email \*"**, validators=[DataRequired(), Email()])  
 submit = SubmitField(**"Update"**)  
 cancel = SubmitField(**"Cancel"**)  
  
 **def** validate\_username (self, username):  
 **if** current\_user.username == username.data:  
 **if** User.query.filter\_by(username= username.data).first():  
 **raise** ValidationError(**'Username already taken. Please enter another username.'**)  
  
 **def** validate\_email (self, email):  
 **if** current\_user.email == email.data:  
 **if** User.query.filter\_by(email=email.data).first():  
 **raise** ValidationError(**'Email Id already in use. Please enter another Email Id or Log In to your account.'**

Also update **account.html.**

## Updating Profile Picture & auto-scale image:

See video from 22 mins

<https://www.youtube.com/watch?v=803Ei2Sq-Zs&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=7>

## Create, Update, Delete Post:

<https://www.youtube.com/watch?v=u0oDDZrDz9U&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=8>

## Pagination:

<https://www.youtube.com/watch?v=PSWf2TjTGNY&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=9>

## Email and password reset:

<https://www.youtube.com/watch?v=vutyTx7IaAI&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=10>

## Rest of the playlist:

<https://www.youtube.com/playlist?list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH>