



Django Login und Registrierung

Einleitung



pythonTM

django

Kapitel III
Login und Registrierung

Lernziele Kapitel III:

Ich kann eine Django Registrierung programmieren.

Ich kann einen Django Login programmieren.

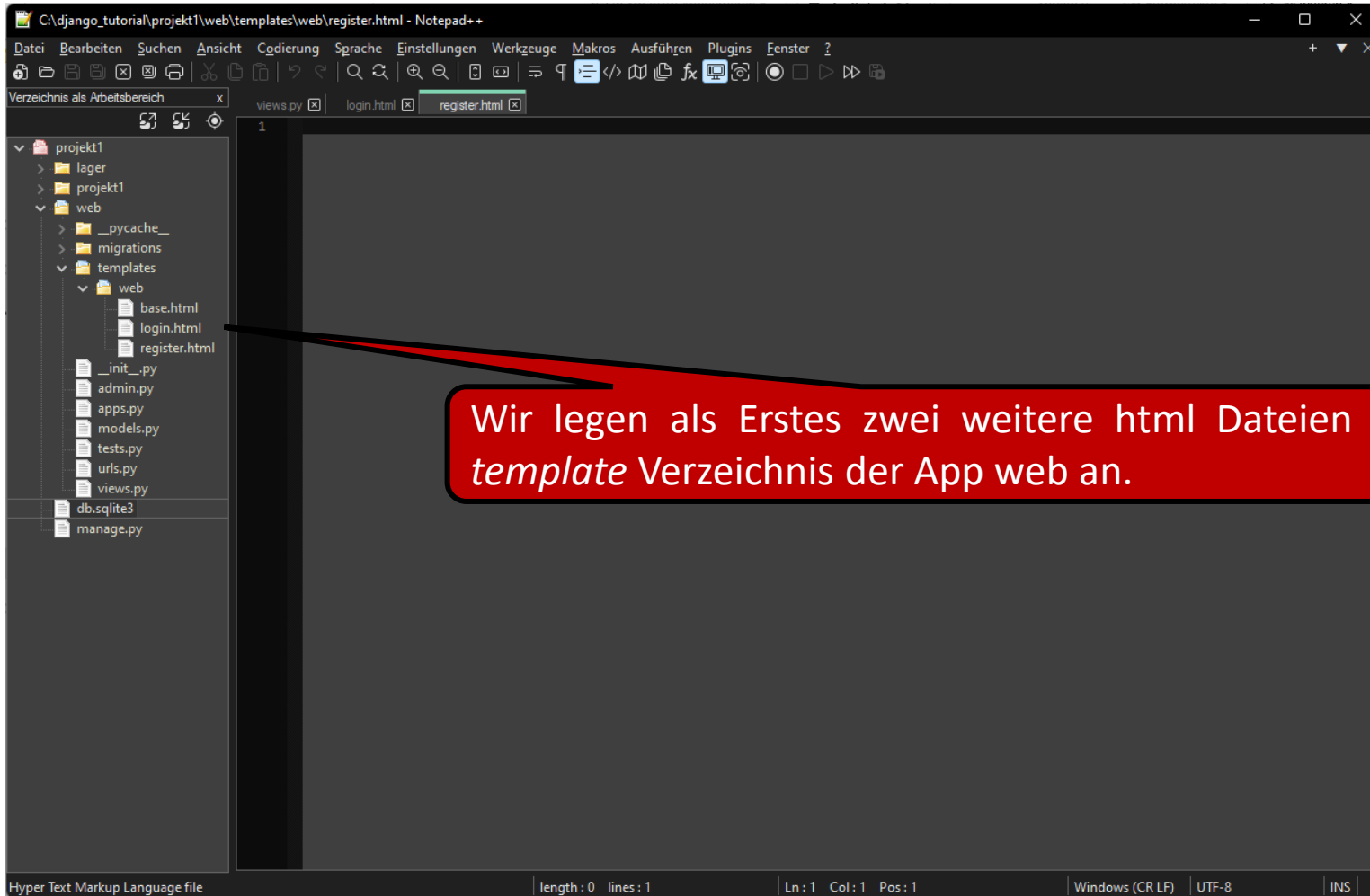
Ich kann einen Django Logout programmieren.

Ich habe verstanden, wofür Django Forms verwendet werden.

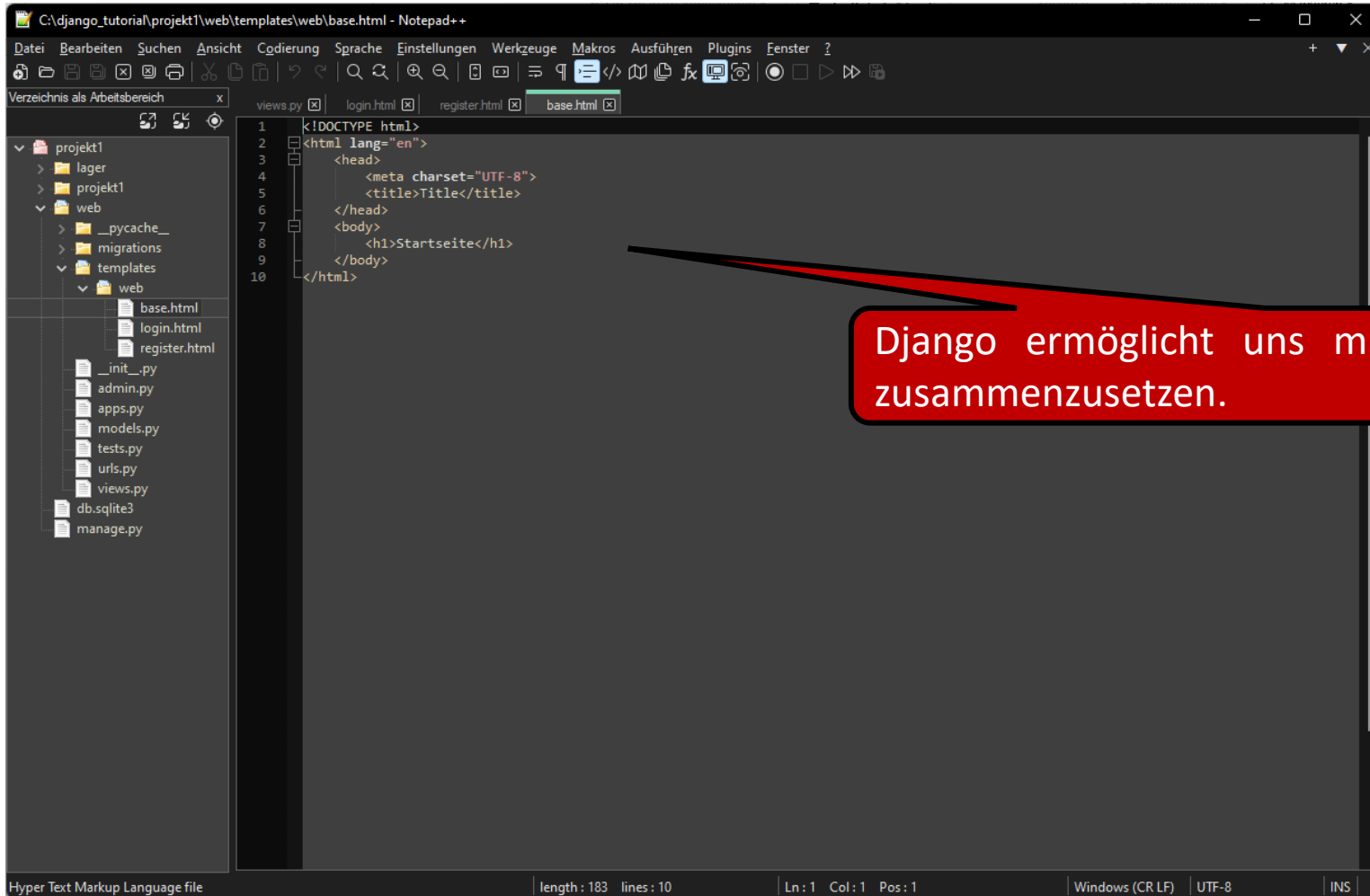
Ich habe verstanden, wie ich Django Tags verwende.

Ich habe verstanden, wie die migration durchführe.

Login und Registrierung



Login und Registrierung



The screenshot shows a Notepad++ window with the title bar 'C:\django_tutorial\projekt1\web\templates\web\base.html - Notepad++'. The menu bar includes 'Datei', 'Bearbeiten', 'Suchen', 'Ansicht', 'Codierung', 'Sprache', 'Einstellungen', 'Werkzeuge', 'Makros', 'Ausführen', 'Plugins', 'Fenster', and '?'. The toolbar contains various icons for file operations and editing. The left sidebar shows a file explorer with the following structure:

- projekt1
 - lager
 - projekt1
 - web
 - __pycache__
 - migrations
 - templates
 - web
 - base.html
 - login.html
 - register.html
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - db.sqlite3
 - manage.py

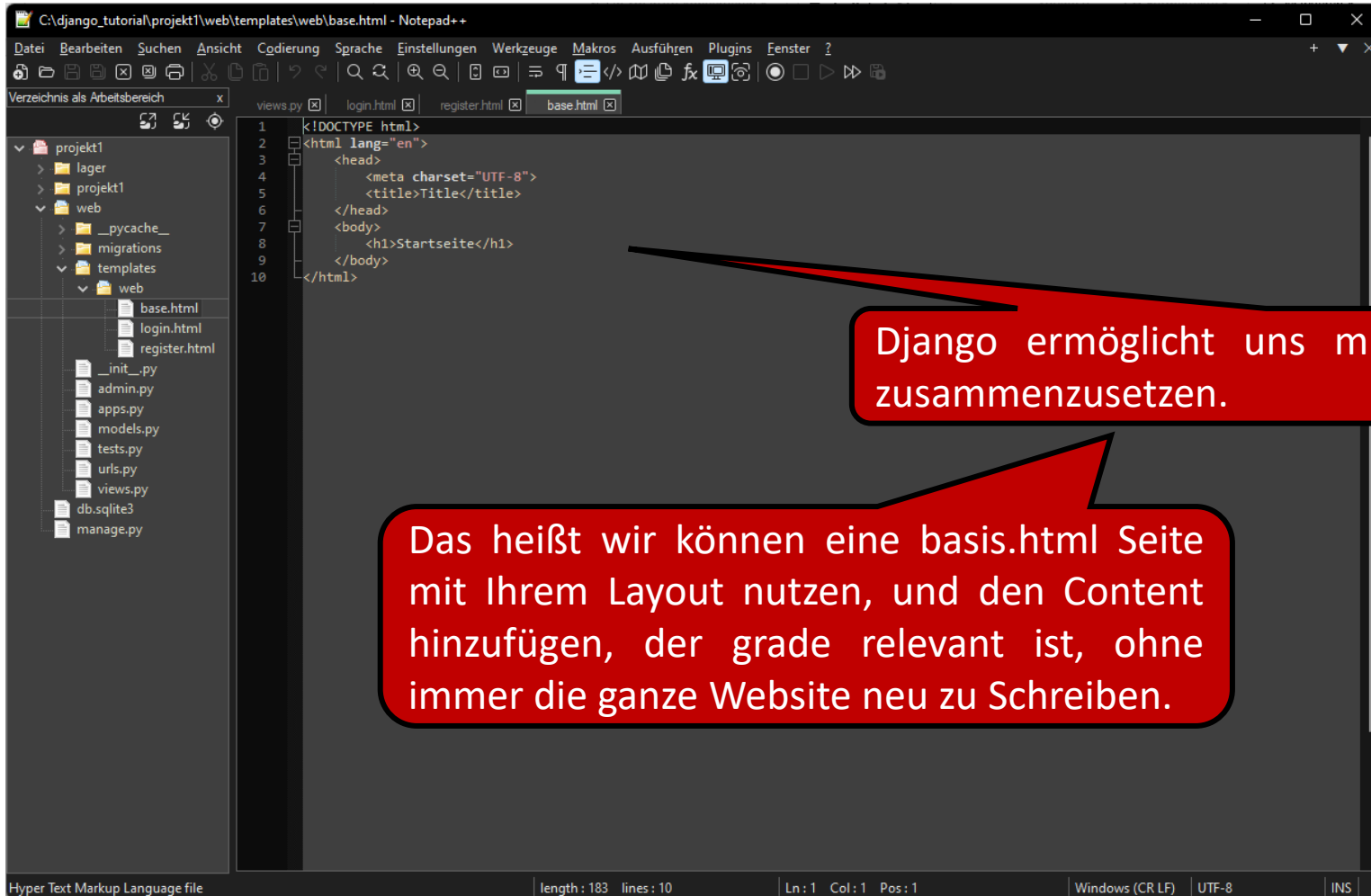
The main editor area shows the content of 'base.html' with line numbers 1 through 10:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6 </head>
7 <body>
8   <h1>Startseite</h1>
9 </body>
10</html>
```

The status bar at the bottom indicates 'Hyper Text Markup Language file', 'length : 183 lines : 10', 'Ln : 1 Col : 1 Pos : 1', 'Windows (CR LF)', 'UTF-8', and 'INS'.

Django ermöglicht uns mittels TAGs die Website zusammenzusetzen.

Login und Registrierung



The screenshot shows a Notepad++ window with the title bar 'C:\django_tutorial\projekt1\web\templates\web\base.html - Notepad++'. The menu bar includes 'Datei', 'Bearbeiten', 'Suchen', 'Ansicht', 'Codierung', 'Sprache', 'Einstellungen', 'Werkzeuge', 'Makros', 'Ausführen', 'Plugins', 'Fenster', and '?'. The toolbar contains various icons for file operations and editing. The left sidebar shows a file explorer with the following structure:

- projekt1
 - lager
 - projekt1
 - web
 - __pycache__
 - migrations
 - templates
 - web
 - base.html
 - login.html
 - register.html
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - db.sqlite3
 - manage.py

The main editor area shows the content of 'base.html' with line numbers 1 through 10:

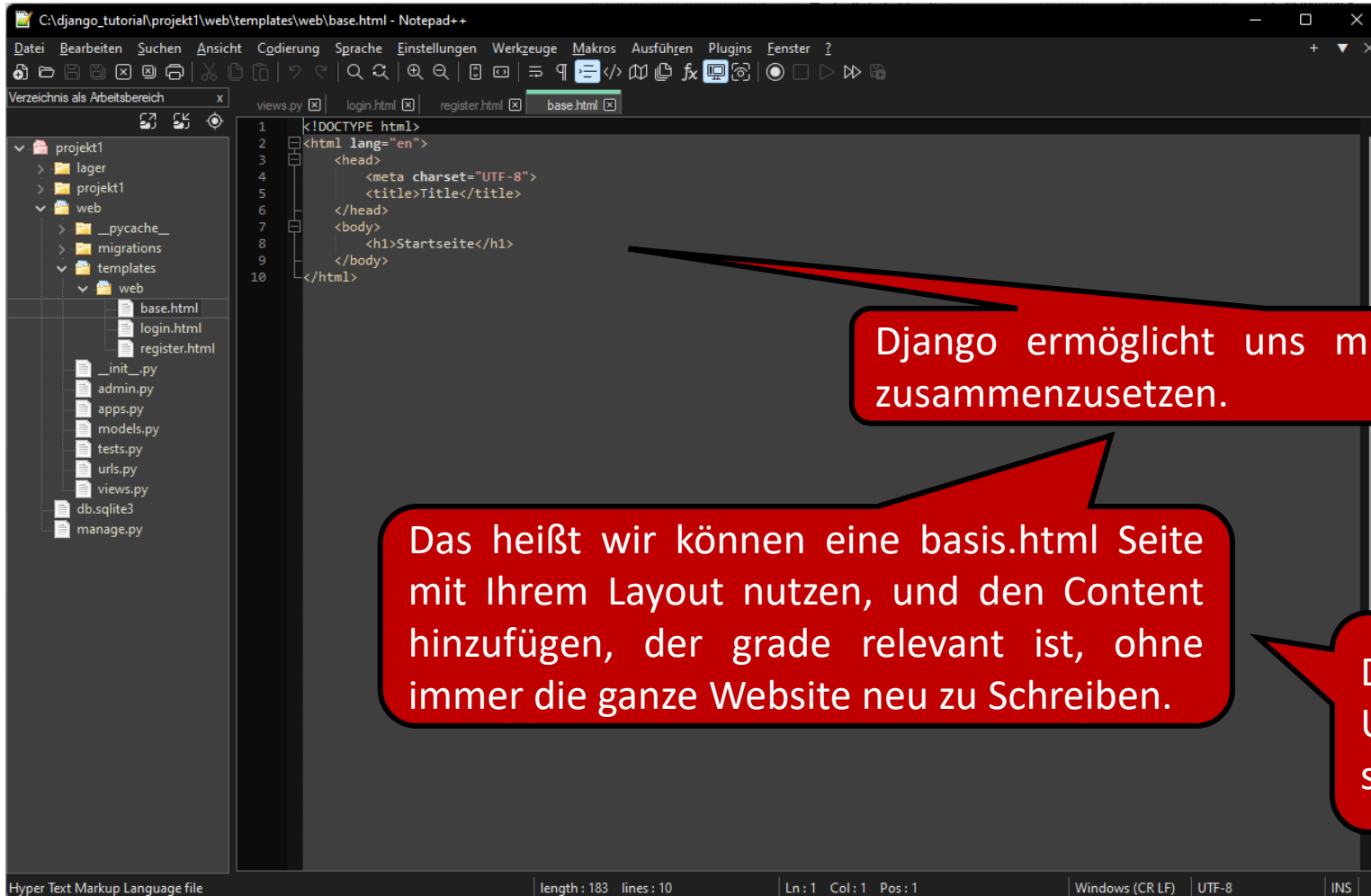
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6 </head>
7 <body>
8   <h1>Startseite</h1>
9 </body>
10</html>
```

At the bottom of the window, the status bar displays: 'Hyper Text Markup Language file', 'length: 183 lines: 10', 'Ln: 1 Col: 1 Pos: 1', 'Windows (CR LF)', 'UTF-8', and 'INS'.

Django ermöglicht uns mittels TAGs die Website zusammenzusetzen.

Das heißt wir können eine basis.html Seite mit Ihrem Layout nutzen, und den Content hinzufügen, der grade relevant ist, ohne immer die ganze Website neu zu Schreiben.

Login und Registrierung



The screenshot shows a Notepad++ window with the file path `C:\django_tutorial\projekt1\web\templates\web\base.html`. The left sidebar displays a file explorer for the `projekt1` directory, showing subdirectories like `lager`, `projekt1`, and `web`, along with files like `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `urls.py`, `views.py`, `db.sqlite3`, and `manage.py`. The main editor area shows the content of `base.html`:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6 </head>
7 <body>
8   <h1>Startseite</h1>
9 </body>
10</html>
```

Django ermöglicht uns mittels TAGs die Website zusammenzusetzen.

Das heißt wir können eine basis.html Seite mit Ihrem Layout nutzen, und den Content hinzufügen, der grade relevant ist, ohne immer die ganze Website neu zu Schreiben.

Django rendert dann die Seite, so dass der User von den TAGs nichts mehr sieht, sondern eine reine html Seite bekommt.

Login und Registrierung

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8   <h1>Lagerverwaltung</h1>
9   {% block authForm %}
10   {% endblock %}
11 </body>
12 </html>
```

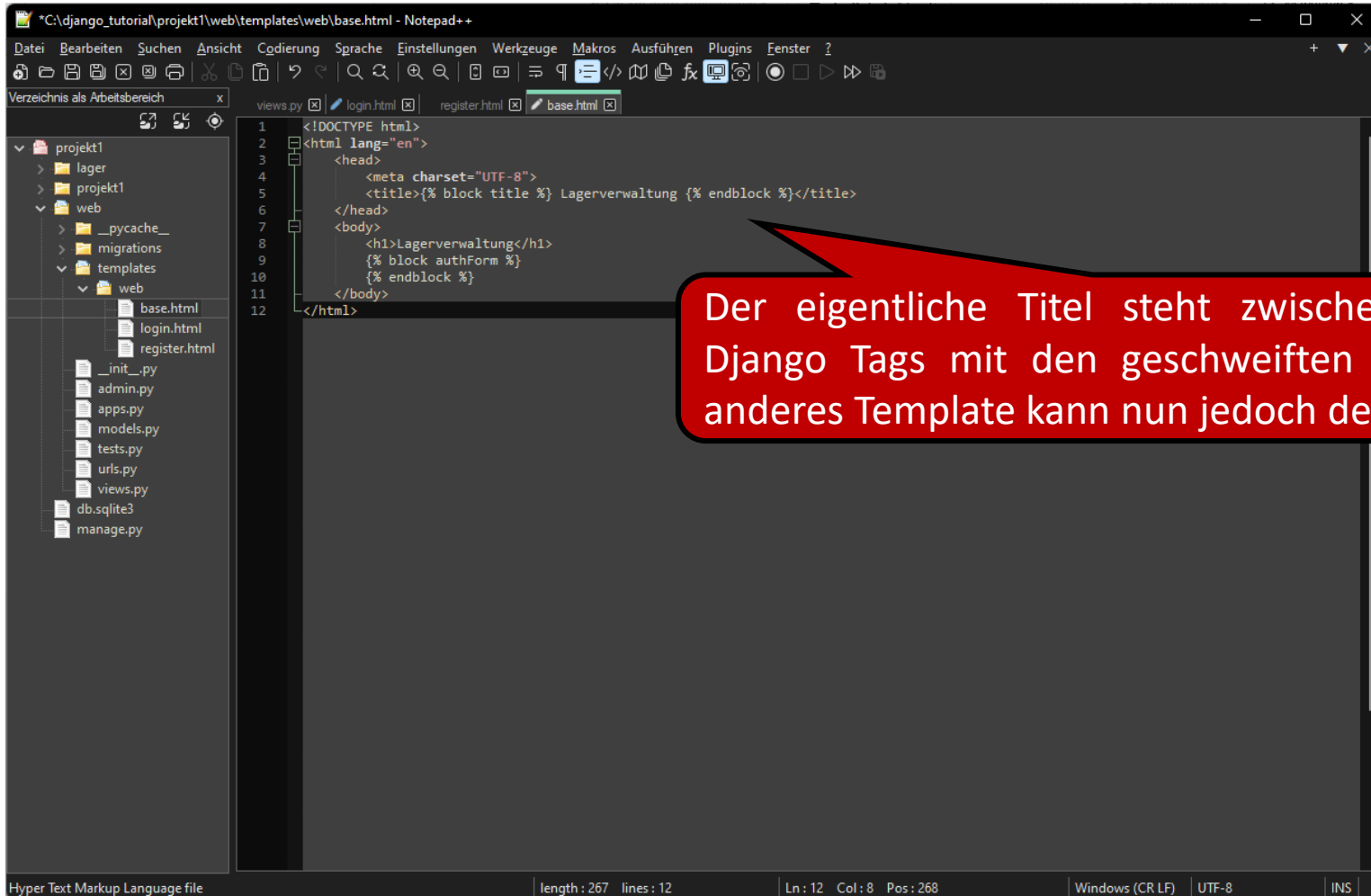
Verzeichnis als Arbeitsbereich

- projekt1
 - lager
 - projekt1
 - web
 - __pycache__
 - migrations
 - templates
 - web
 - base.html
 - login.html
 - register.html
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - db.sqlite3
 - manage.py

Hyper Text Markup Language file | length : 267 lines : 12 | Ln : 12 Col : 8 Pos : 268 | Windows (CR LF) | UTF-8 | INS

Als erstes ändern wir den Titel einmal ab und verwenden hierfür den ersten Django-Block.

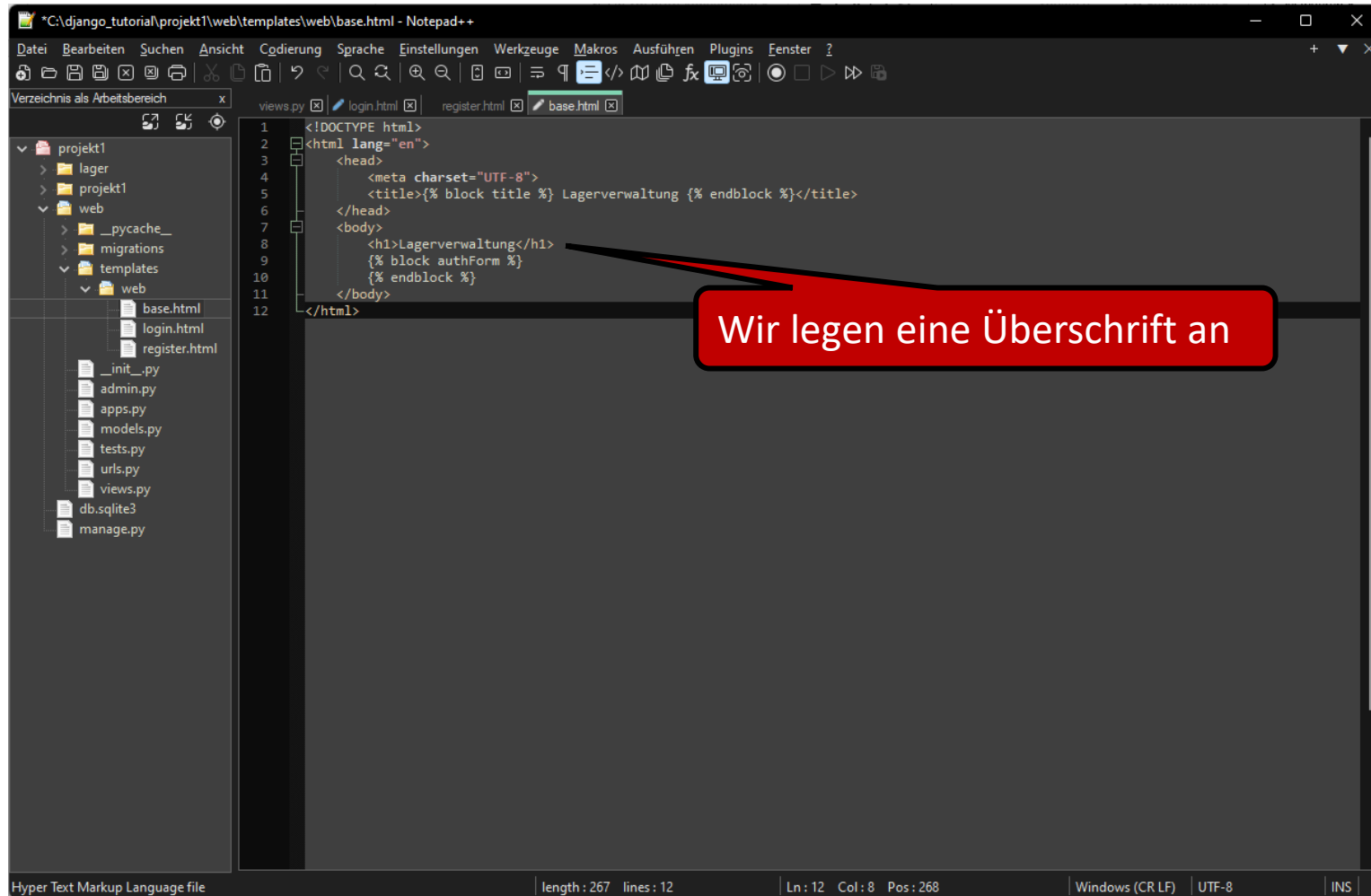
Login und Registrierung



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8   <h1>Lagerverwaltung</h1>
9   {% block authForm %}
10   {% endblock %}
11 </body>
12 </html>
```

Der eigentliche Titel steht zwischen den beiden Django Tags mit den geschweiften Klammern. Ein anderes Template kann nun jedoch den Titel ändern.

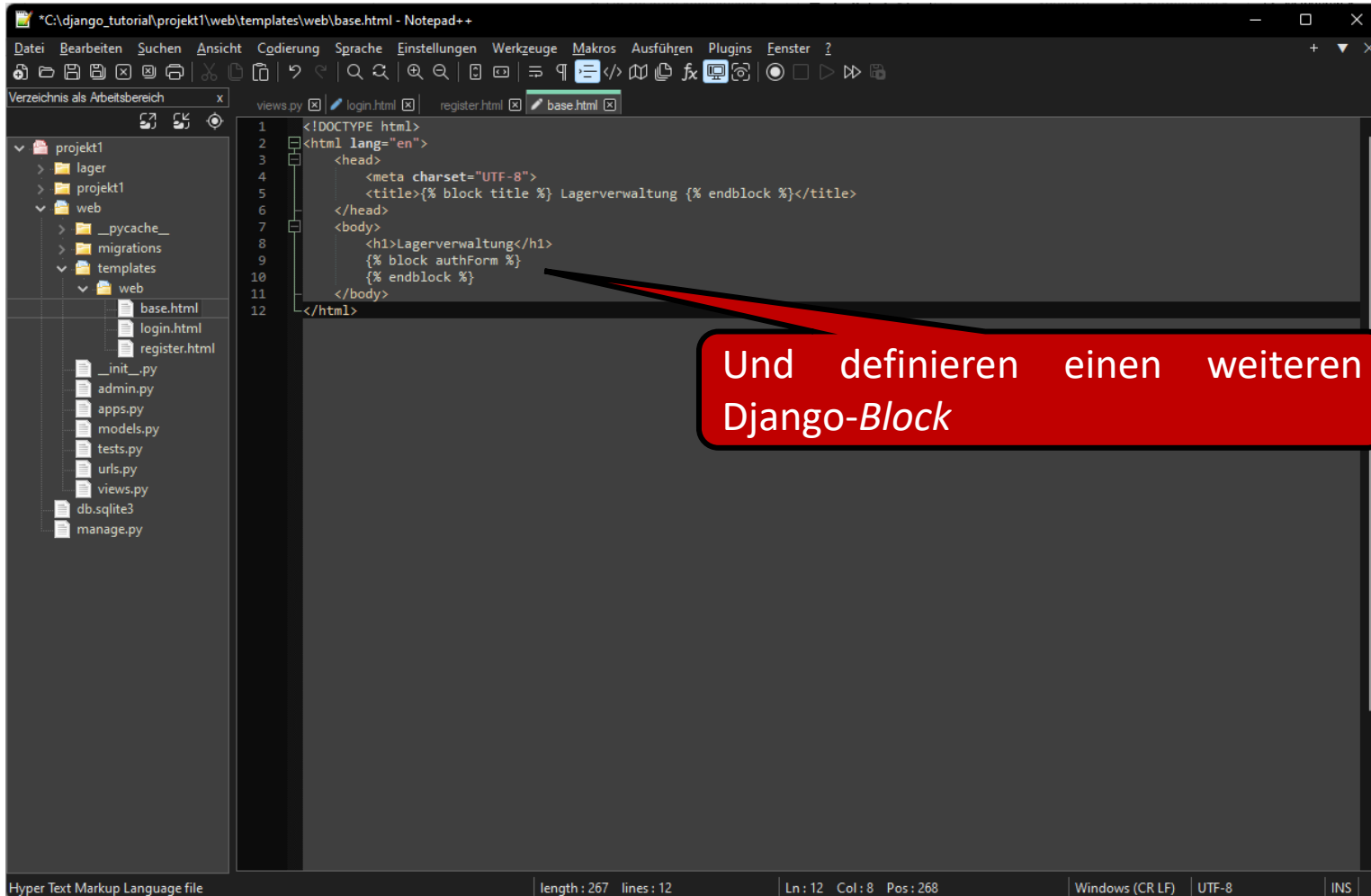
Login und Registrierung



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8   <h1>Lagerverwaltung</h1>
9   {% block authForm %}
10   {% endblock %}
11 </body>
12 </html>
```

Wir legen eine Überschrift an

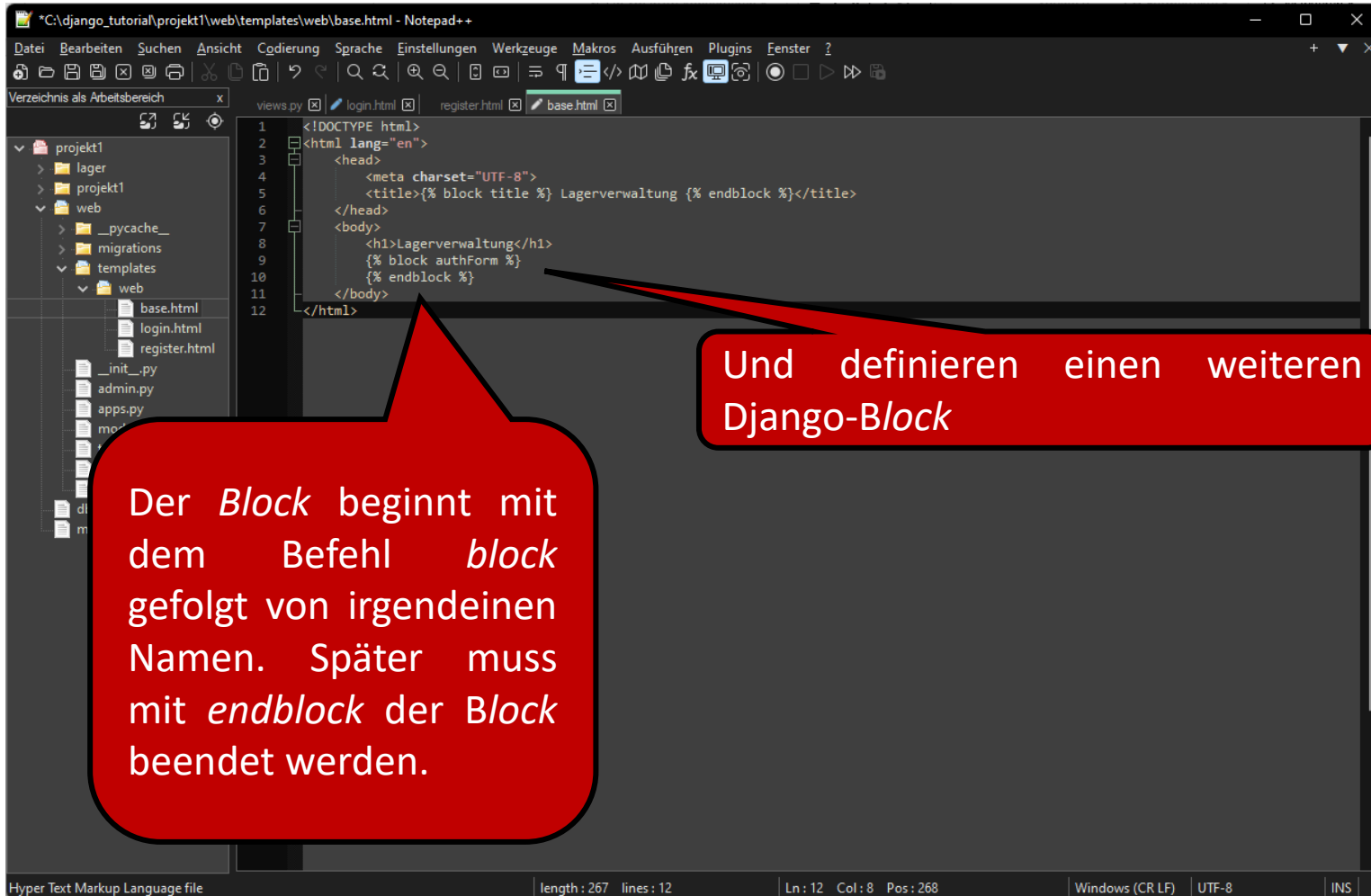
Login und Registrierung



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8 <h1>Lagerverwaltung</h1>
9 {% block authForm %}
10 {% endblock %}
11 </body>
12 </html>
```

Und definieren einen weiteren Django-Block

Login und Registrierung

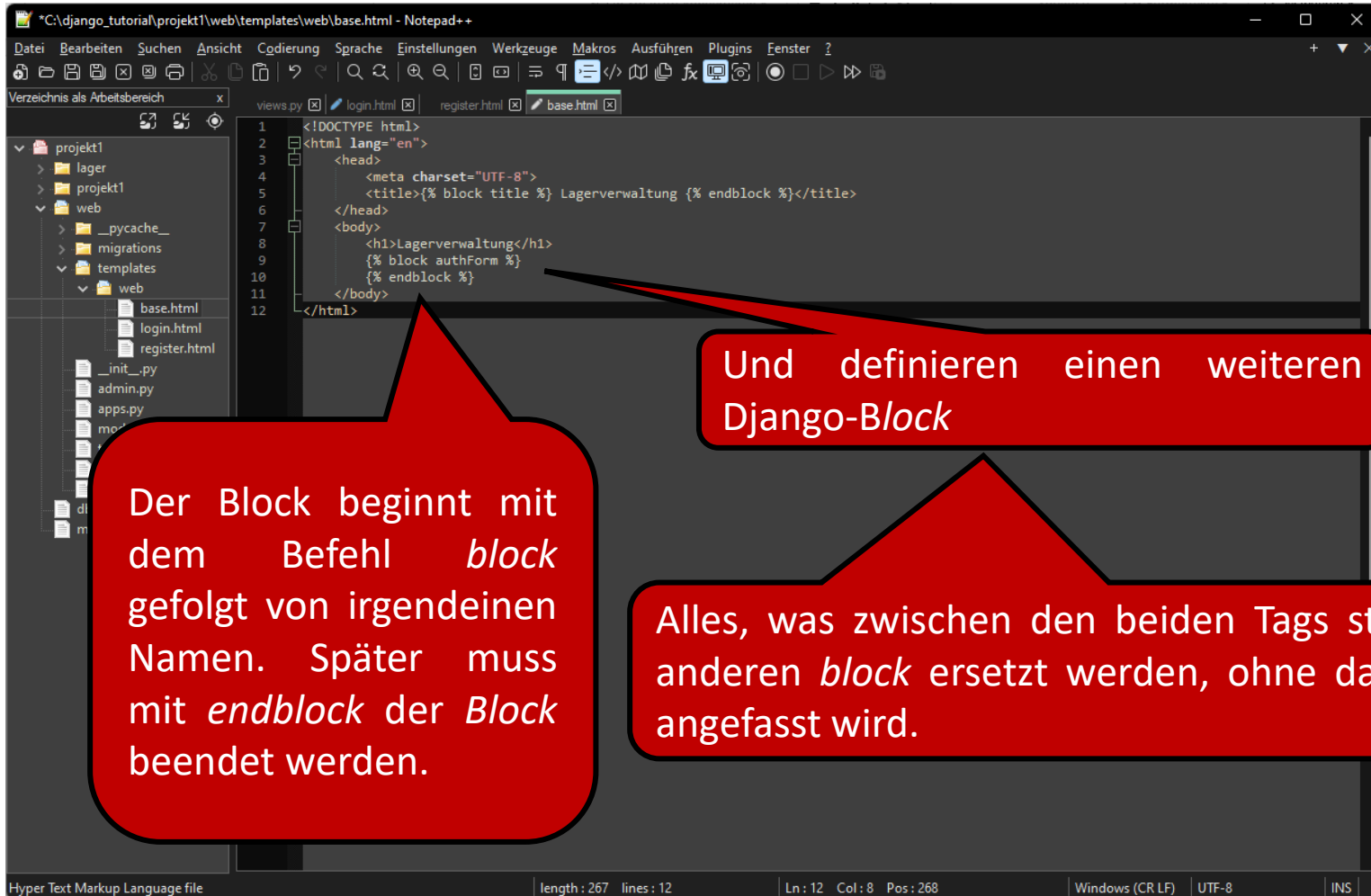


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8   <h1>Lagerverwaltung</h1>
9   {% block authForm %}
10   {% endblock %}
11 </body>
12 </html>
```

Der *Block* beginnt mit dem Befehl *block* gefolgt von irgendeinem Namen. Später muss mit *endblock* der *Block* beendet werden.

Und definieren einen weiteren Django-Block

Login und Registrierung



The screenshot shows a Notepad++ window editing a file at `*C:\django_tutorial\projekt1\web\templates\web\base.html`. The file explorer on the left shows the project structure: `projekt1` > `lager` > `projekt1` > `web` > `templates` > `web`. The editor displays the following HTML code:

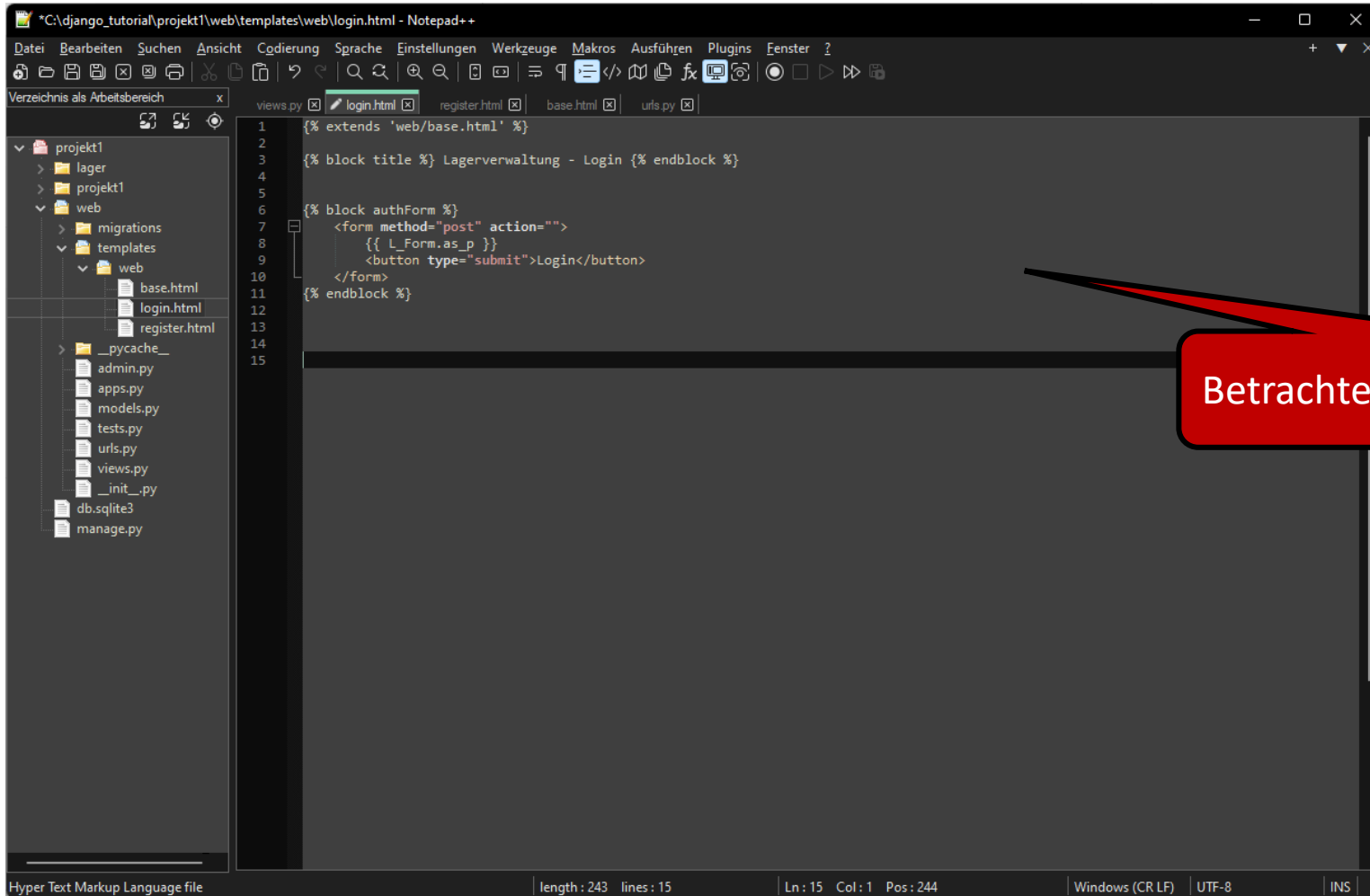
```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6   </head>
7   <body>
8     <h1>Lagerverwaltung</h1>
9     {% block authForm %}
10    {% endblock %}
11  </body>
12 </html>
```

Three red callout boxes provide explanations:

- Left box:** Der Block beginnt mit dem Befehl *block* gefolgt von irgendeinem Namen. Später muss mit *endblock* der *Block* beendet werden.
- Top-right box:** Und definieren einen weiteren Django-Block
- Bottom-right box:** Alles, was zwischen den beiden Tags steht, kann mit einem anderen *block* ersetzt werden, ohne dass die restliche Seite angefasst wird.

The status bar at the bottom indicates: Hyper Text Markup Language file | length : 267 lines : 12 | Ln : 12 Col : 8 Pos : 268 | Windows (CR LF) | UTF-8 | INS

Login und Registrierung



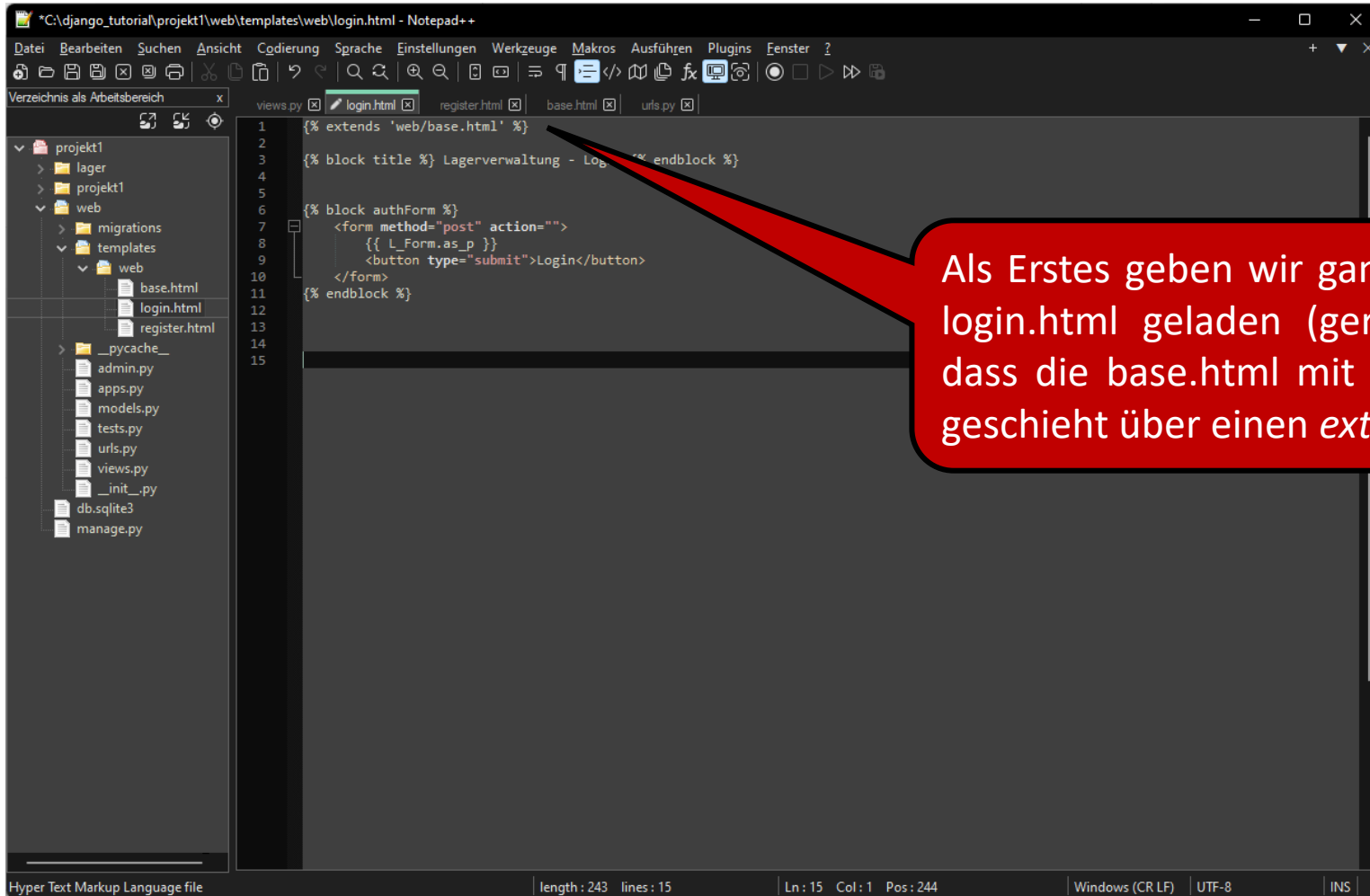
The screenshot shows a Notepad++ window with the file path `*C:\django_tutorial\projekt1\web\templates\web\login.html`. The editor displays the following HTML code:

```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

The file explorer on the left shows the project structure, with the `login.html` file highlighted in the `web/templates/web` directory. A red callout bubble points to the `login.html` file in the explorer and the corresponding code in the editor.

Betrachten wir die *login.html*

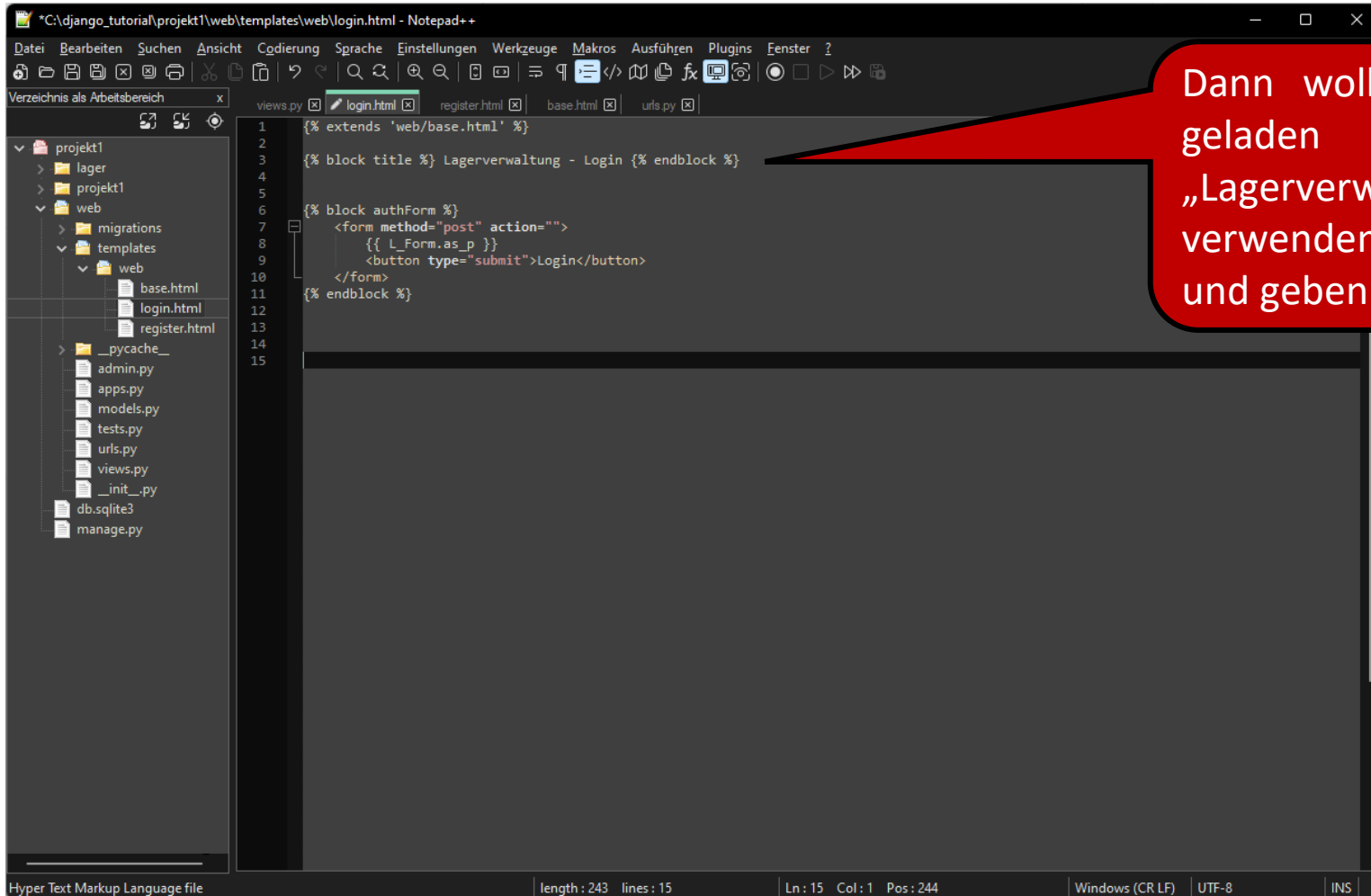
Login und Registrierung



```
1 {% extends 'web/base.html' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

Als Erstes geben wir ganz oben an: Wenn die login.html geladen (gerendert) werden soll, dass die base.html mit verwendet wird. Dies geschieht über einen *extends* Django-Tag.

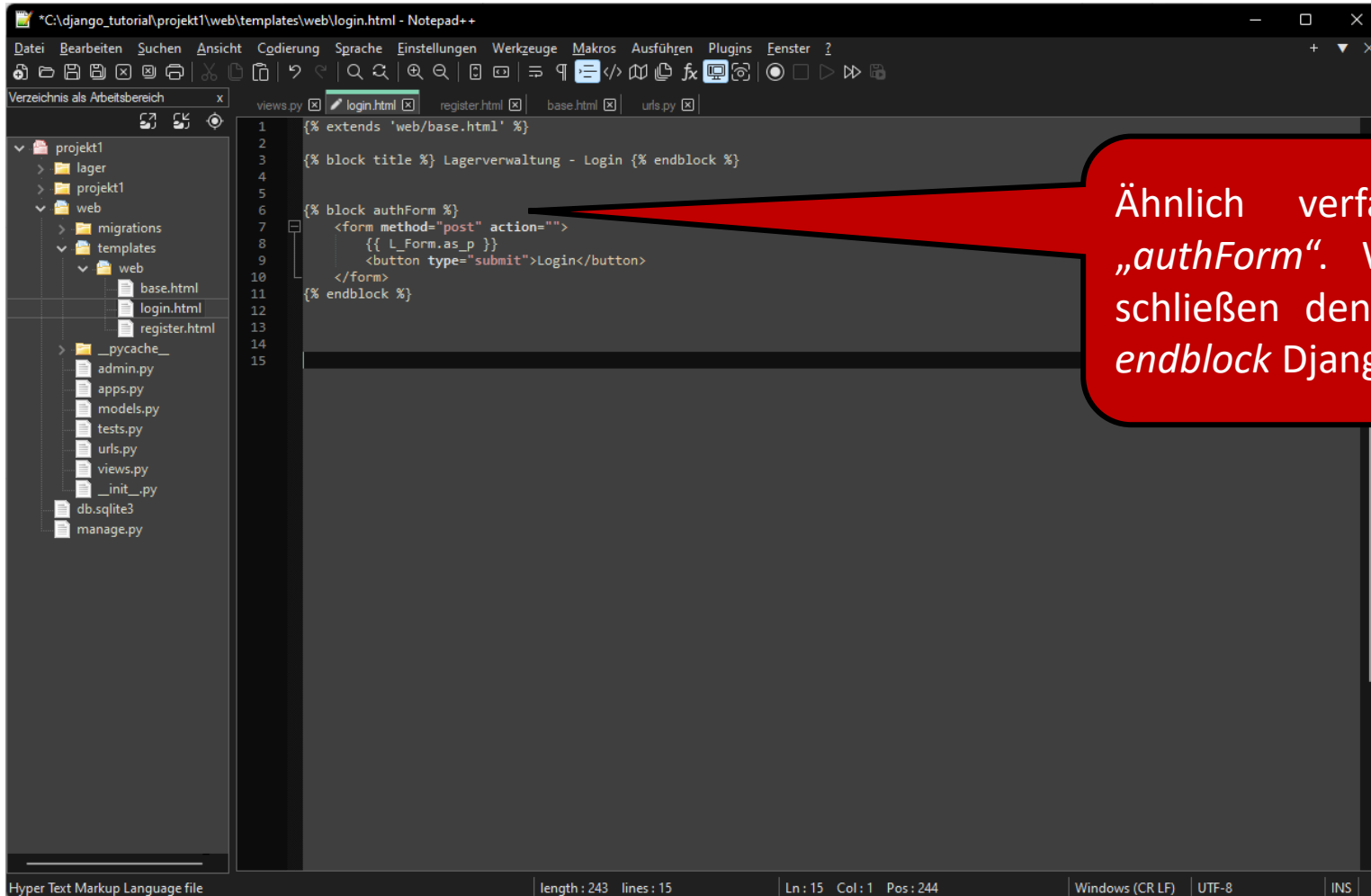
Login und Registrierung



```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7 <form method="post" action="">
8   {{ L_Form.as_p }}
9   <button type="submit">Login</button>
10 </form>
11 {% endblock %}
12
13
14
15
```

Dann wollen wir, dass wenn die login.html geladen wird, das dann der Titel zu „Lagerverwaltung – Login“ geändert wird. Wir verwenden also den Block aus der base.html und geben unseren alternativen Text an.

Login und Registrierung



```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

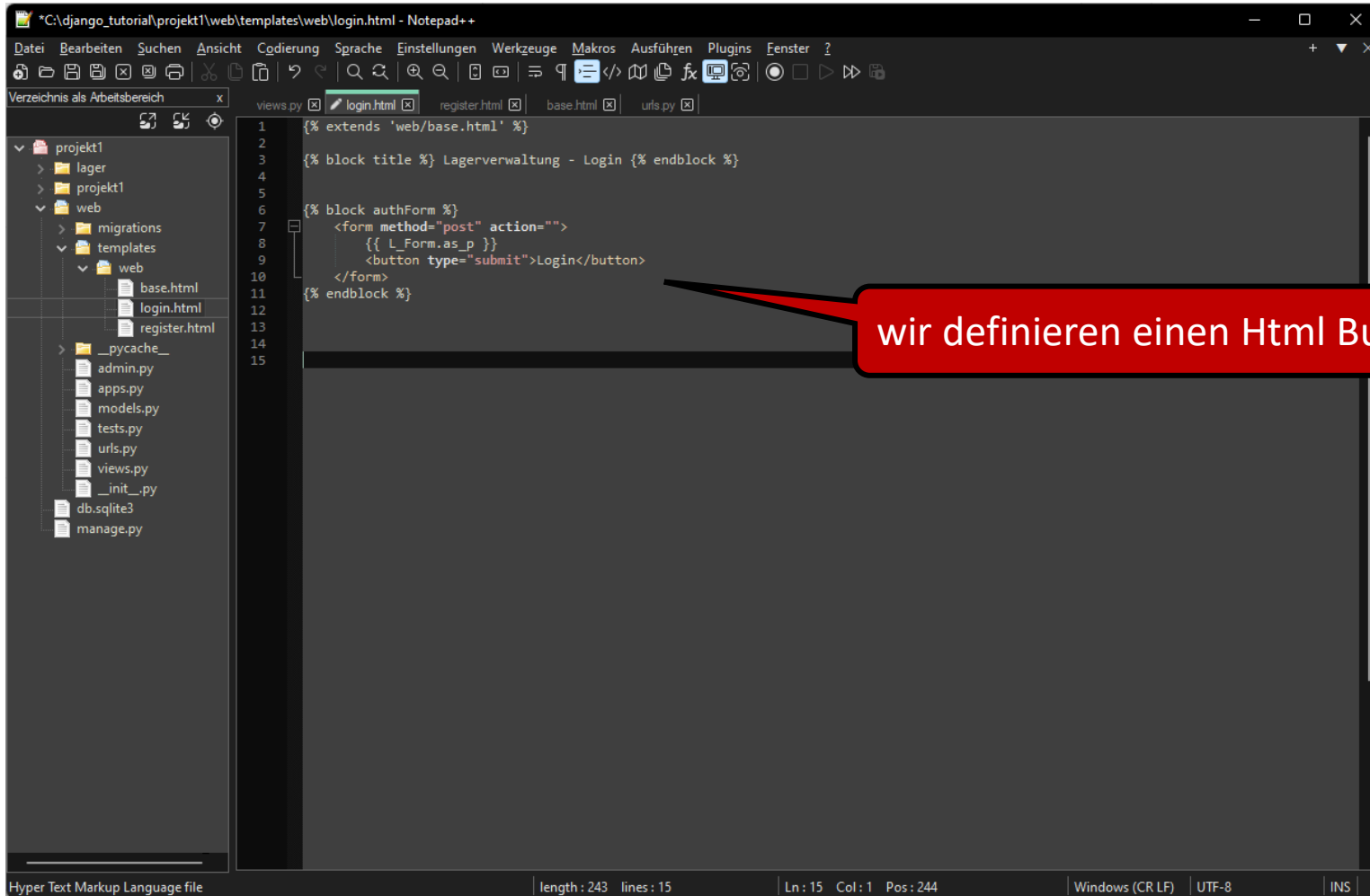
Ähnlich verfahren wir mit dem Block „authForm“. Wir schreiben diesen hin und schließen den weiter unten wieder mit dem endblock Django-Tag.

Login und Registrierung

```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7 <form method="post" action="">
8   {{ L_Form.as_p }}
9   <button type="submit">Login</button>
10 </form>
11 {% endblock %}
12
13
14
15
```

Dazwischen, setzen wir den Inhalt, der für den Block in der base.html verwendet werden soll. Wir definieren eine html form, die einen *Post* senden soll.

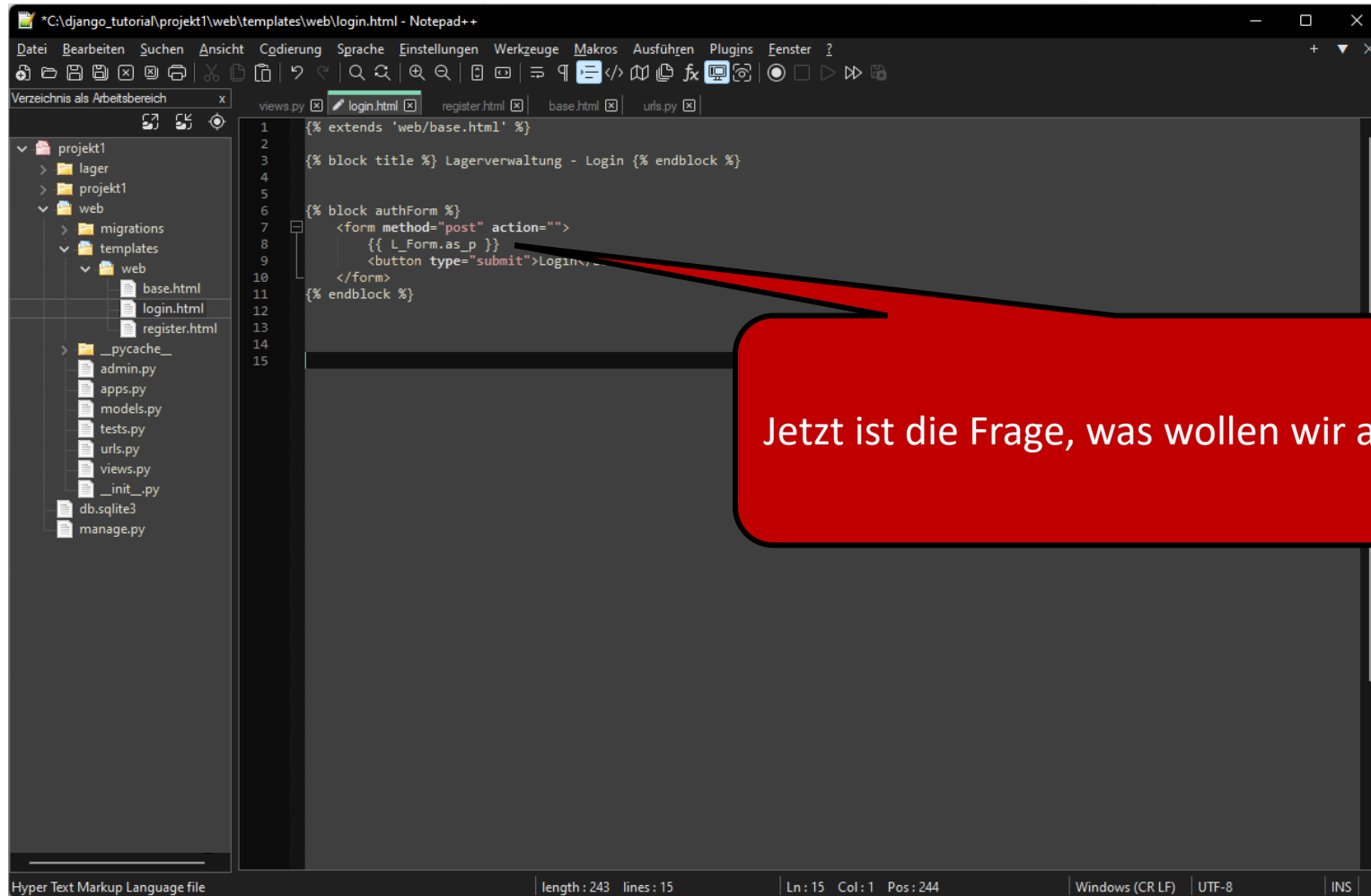
Login und Registrierung



```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

wir definieren einen Html Button, der die Form absenden soll.

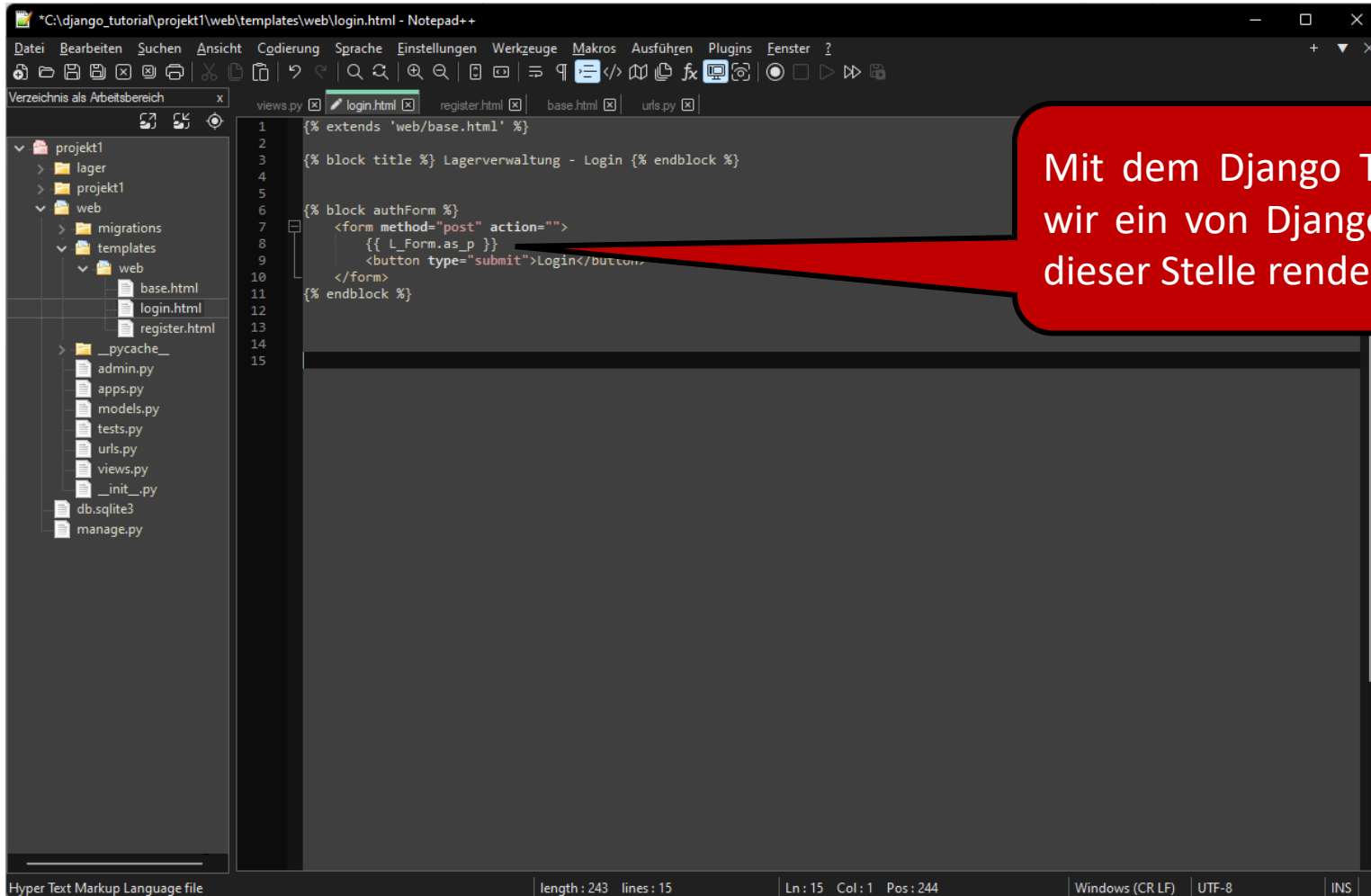
Login und Registrierung



```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

Jetzt ist die Frage, was wollen wir absenden?

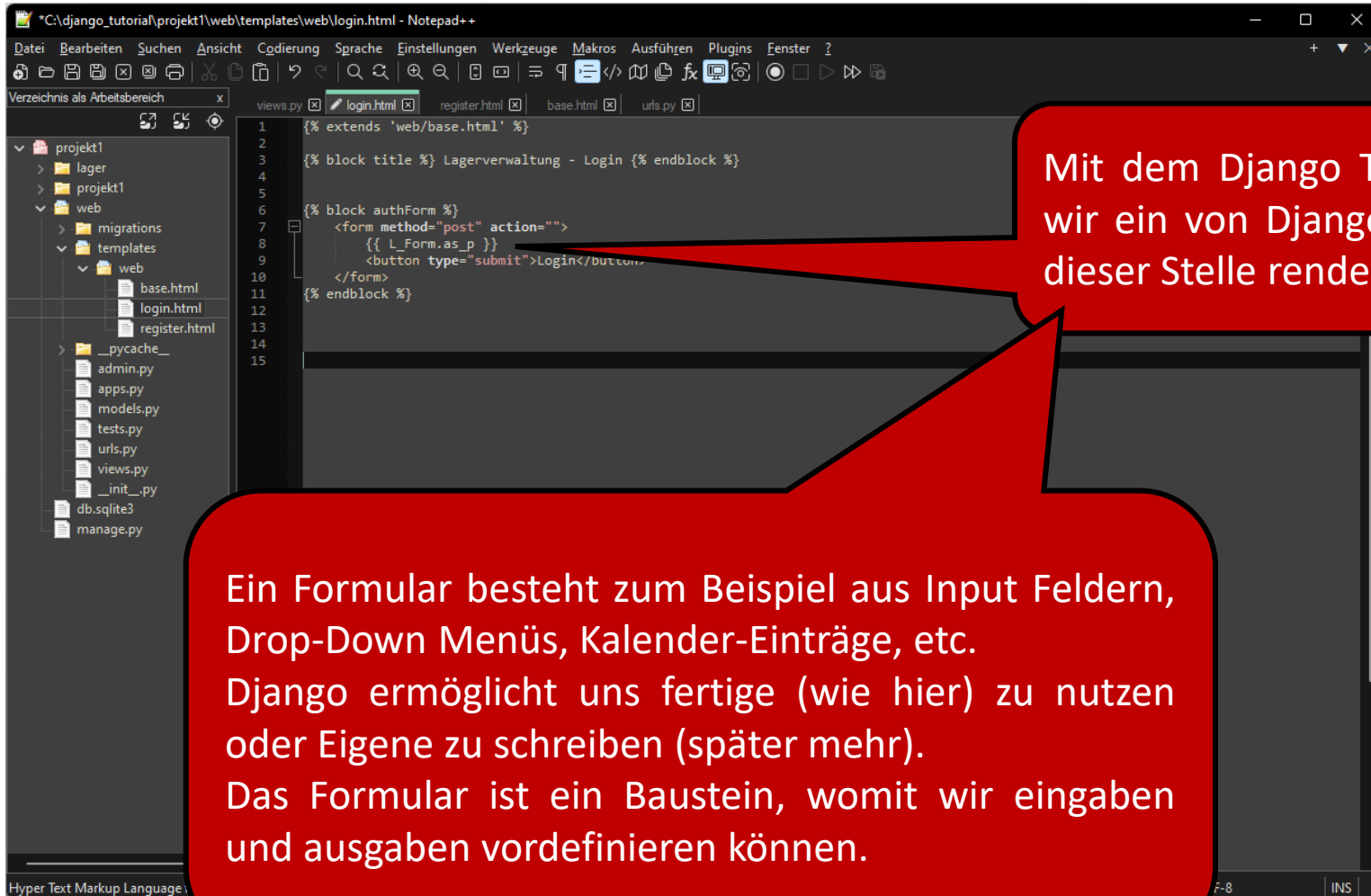
Login und Registrierung



```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

Mit dem Django Tag `{{ L_Form.as_p }}` werden wir ein von Django vor definiertes Formular an dieser Stelle rendern lassen.

Login und Registrierung

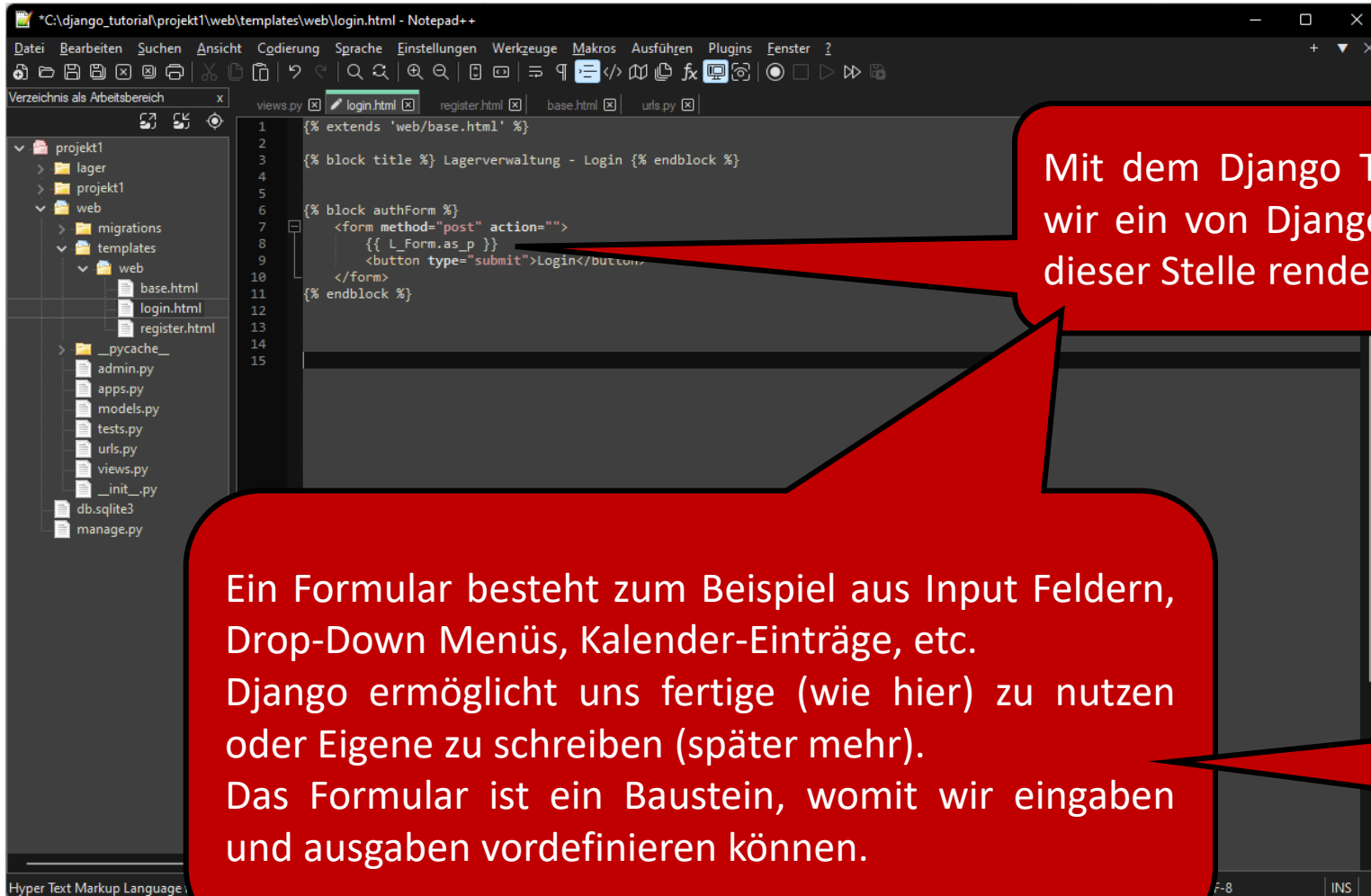


```
1 {% extends 'web/base.html' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

Mit dem Django Tag `{{ L_Form.as_p }}` werden wir ein von Django vor definiertes Formular an dieser Stelle rendern lassen.

Ein Formular besteht zum Beispiel aus Input Feldern, Drop-Down Menüs, Kalender-Einträge, etc. Django ermöglicht uns fertige (wie hier) zu nutzen oder Eigene zu schreiben (später mehr). Das Formular ist ein Baustein, womit wir eingaben und ausgaben vordefinieren können.

Login und Registrierung



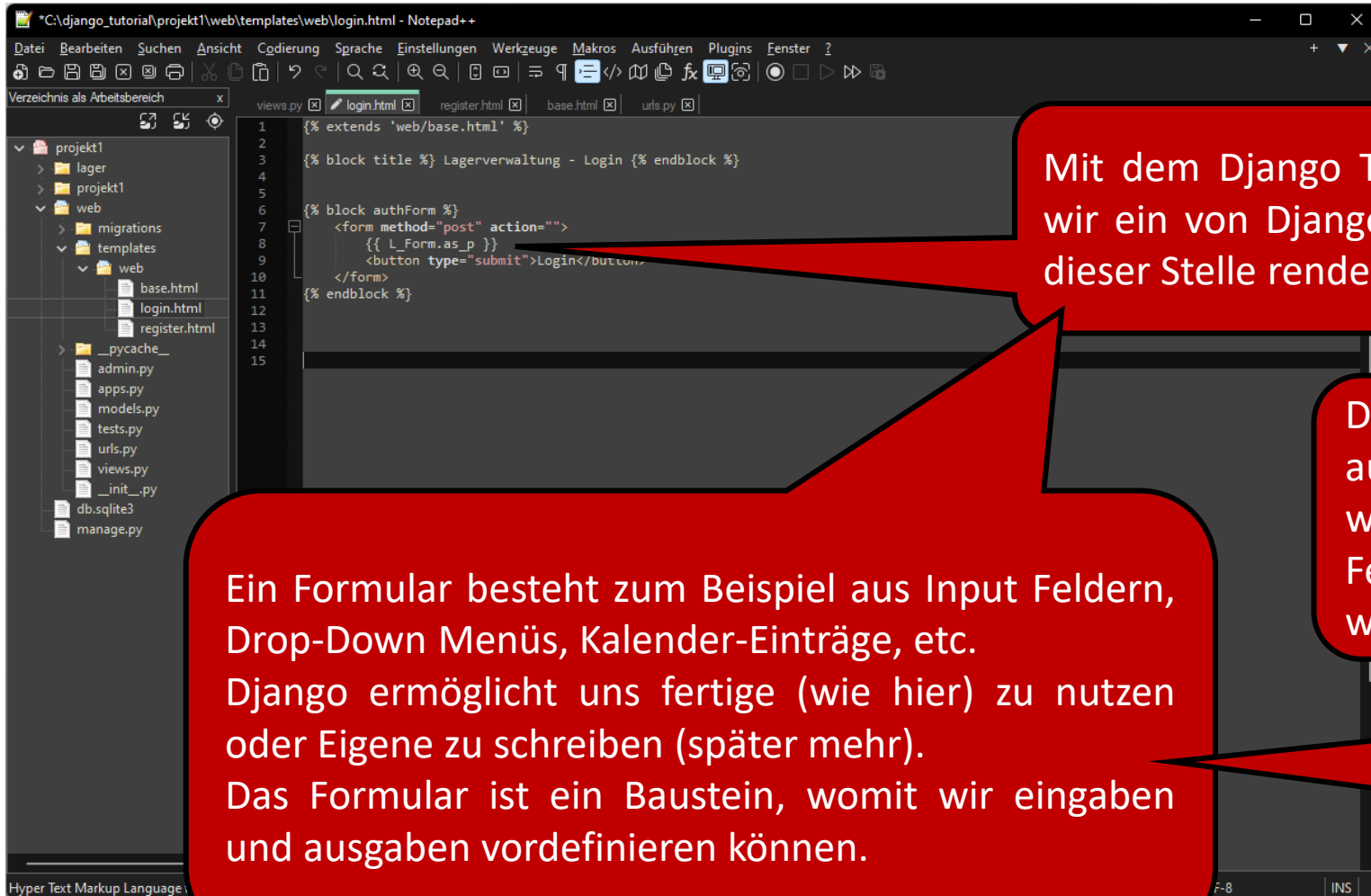
```
1 {% extends 'web/base.html' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

Mit dem Django Tag `{{ L_Form.as_p }}` werden wir ein von Django vor definiertes Formular an dieser Stelle rendern lassen.

Ein Formular besteht zum Beispiel aus Input Feldern, Drop-Down Menüs, Kalender-Einträge, etc. Django ermöglicht uns fertige (wie hier) zu nutzen oder Eigene zu schreiben (später mehr). Das Formular ist ein Baustein, womit wir eingaben und ausgaben vordefinieren können.

Django ermöglicht uns sogar, ein Validation-check der eingegebenen Daten vorzunehmen!

Login und Registrierung



```
1 {% extends 'web/base.html' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {{ L_Form.as_p }}
9         <button type="submit">Login</button>
10    </form>
11 {% endblock %}
12
13
14
15
```

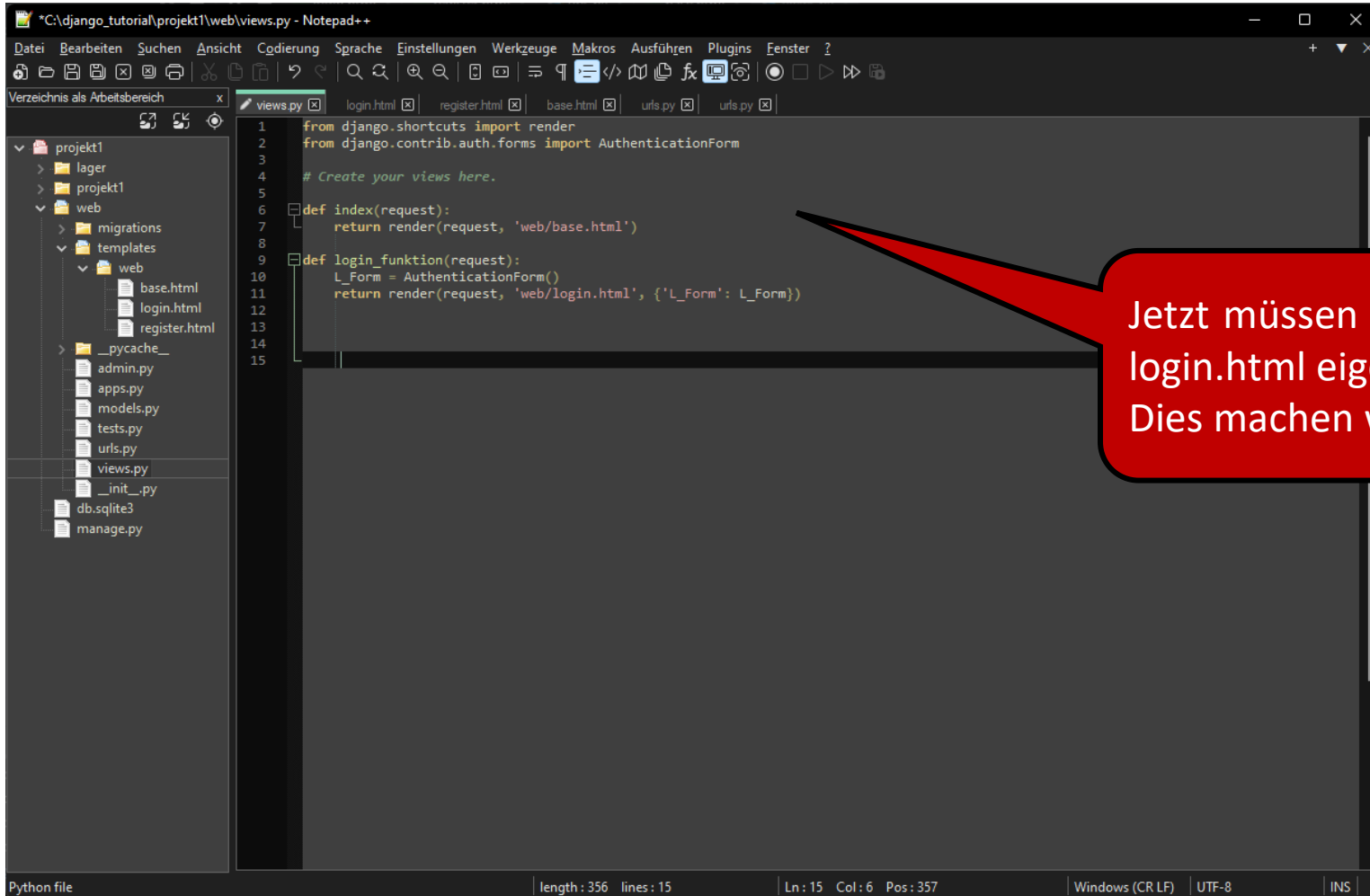
Mit dem Django Tag `{{ L_Form.as_p }}` werden wir ein von Django vordefiniertes Formular an dieser Stelle rendern lassen.

Der Name `L_Form` ist dabei von mir ausgewählt und muss noch übergeben werden. Das `.as_p` sagt Django das die Felder als Paragraph ausgegeben werden sollen.

Ein Formular besteht zum Beispiel aus Input Feldern, Drop-Down Menüs, Kalender-Einträge, etc. Django ermöglicht uns fertige (wie hier) zu nutzen oder Eigene zu schreiben (später mehr). Das Formular ist ein Baustein, womit wir eingaben und ausgaben vordefinieren können.

Django ermöglicht uns sogar, ein Validation-check der eingegebenen Daten vorzunehmen!

Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13
14
15
```

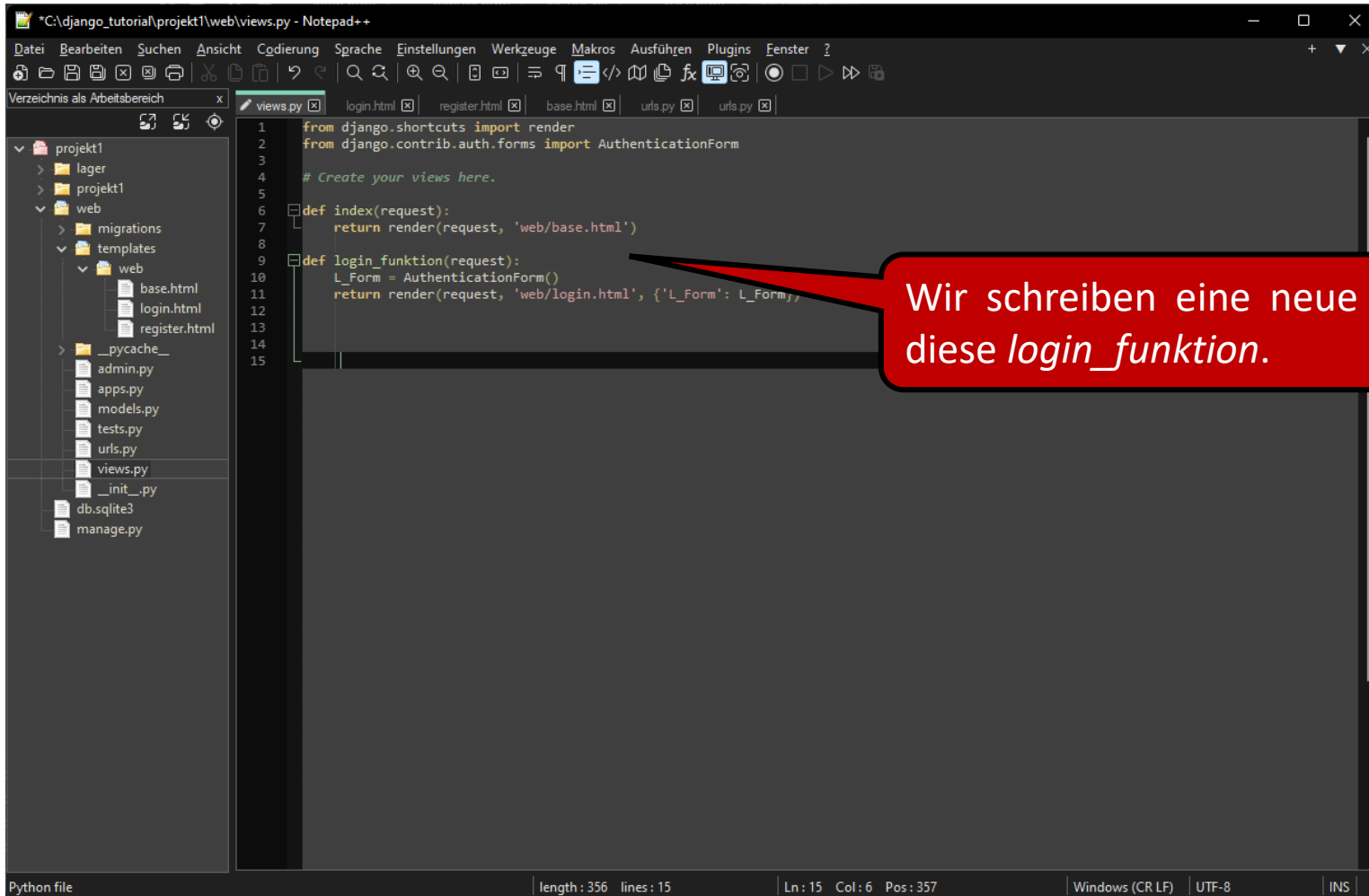
Verzeichnis als Arbeitsbereich

- projekt1
 - lager
 - projekt1
 - web
 - migrations
 - templates
 - web
 - base.html
 - login.html
 - register.html
 - __pycache__
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - __init__.py
 - db.sqlite3
 - manage.py

Python file | length: 356 lines: 15 | Ln: 15 Col: 6 Pos: 357 | Windows (CR LF) | UTF-8 | INS

Jetzt müssen wir Django noch sagen, wie er die login.html eigentlich rendern soll. Dies machen wir in der *views.py*

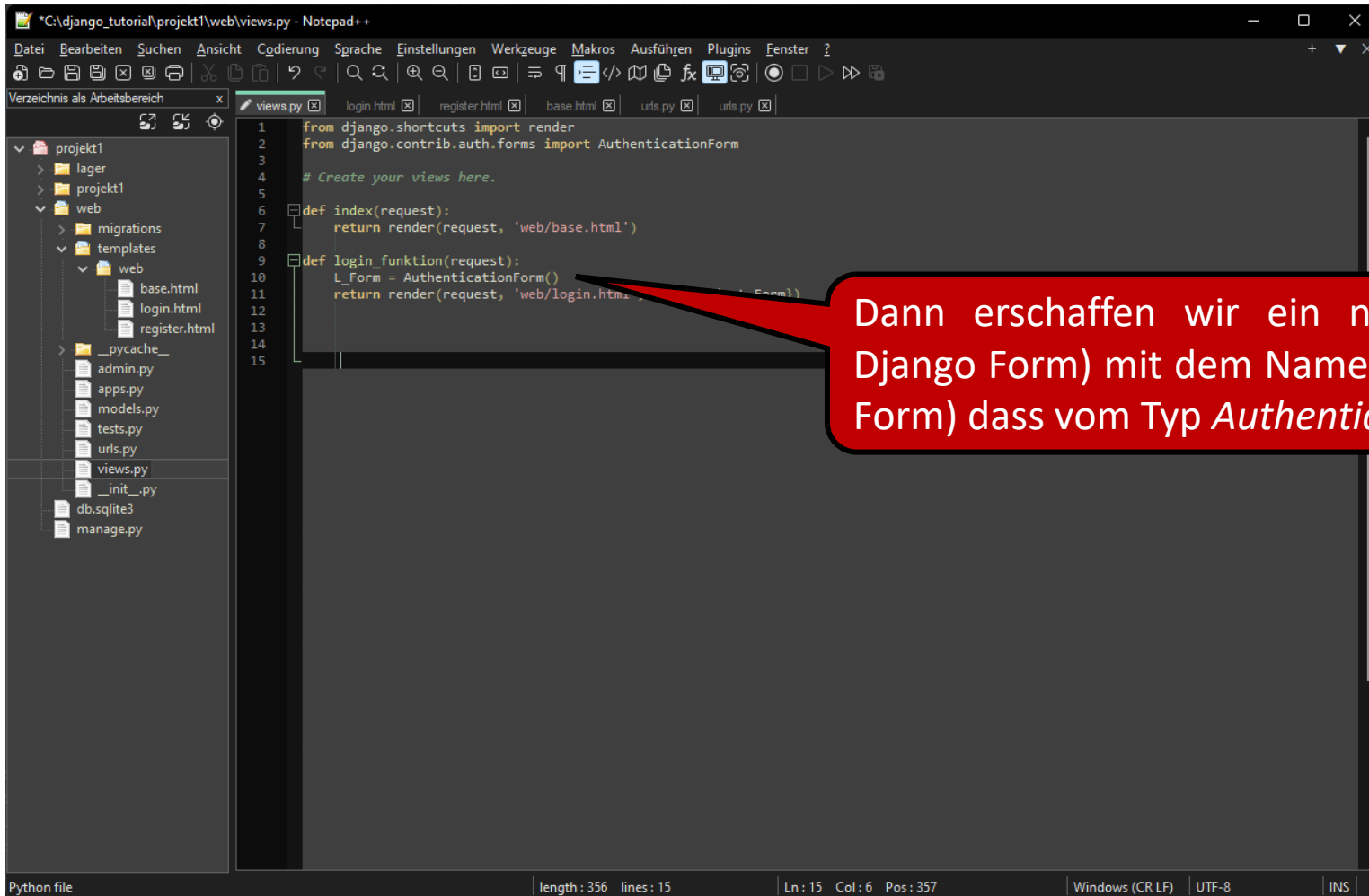
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form},)
12
13
14
15
```

Wir schreiben eine neue Funktion und nennen diese *login_funktion*.

Login und Registrierung

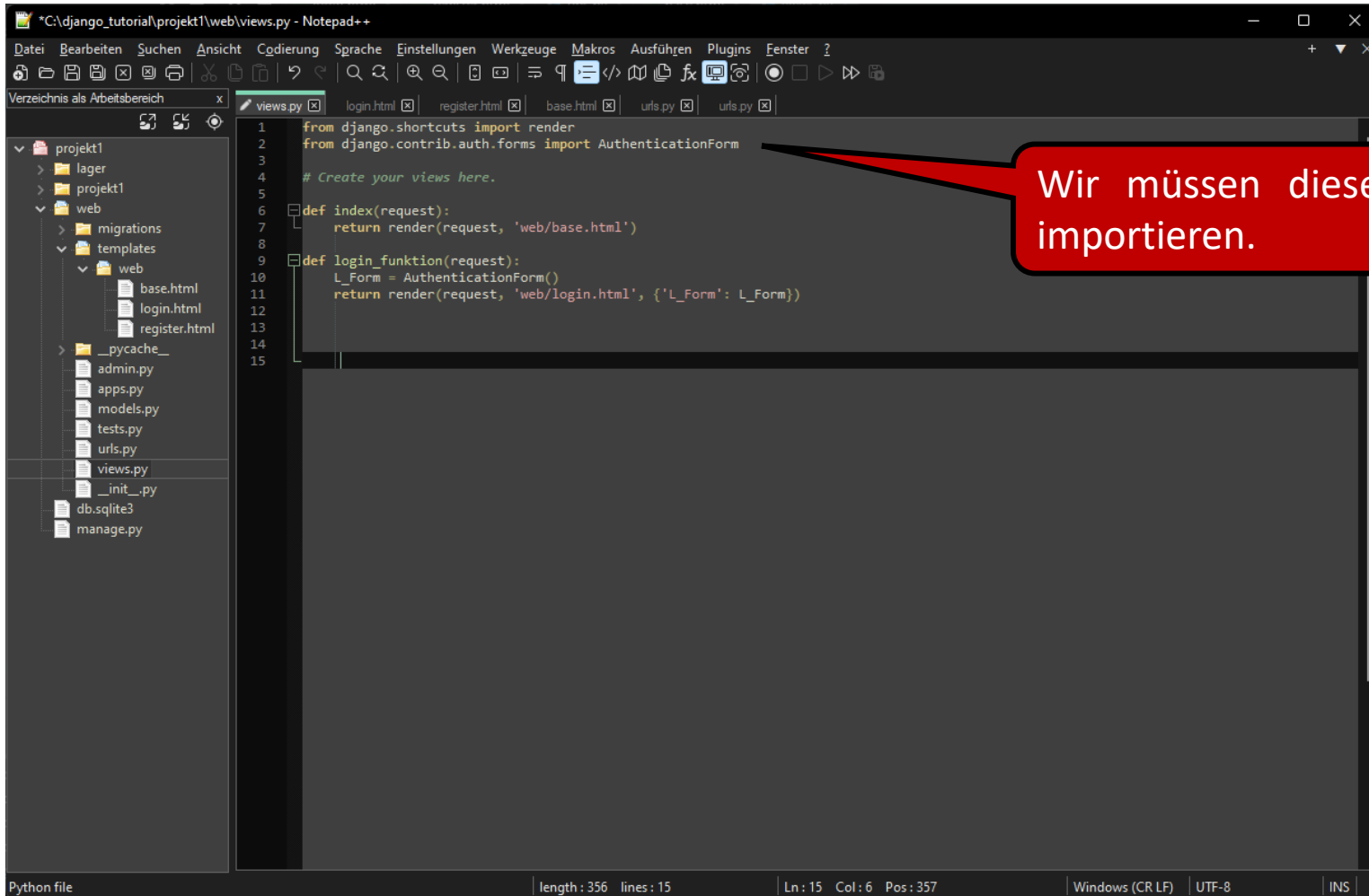


```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'form': L_Form})
12
13
14
15
```

Python file | length: 356 lines: 15 | Ln: 15 Col: 6 Pos: 357 | Windows (CR LF) | UTF-8 | INS

Dann erschaffen wir ein neues Objekt (eine Django Form) mit dem Namen *L_Form* (für Login Form) dass vom Typ *AuthenticationForm* ist.

Login und Registrierung

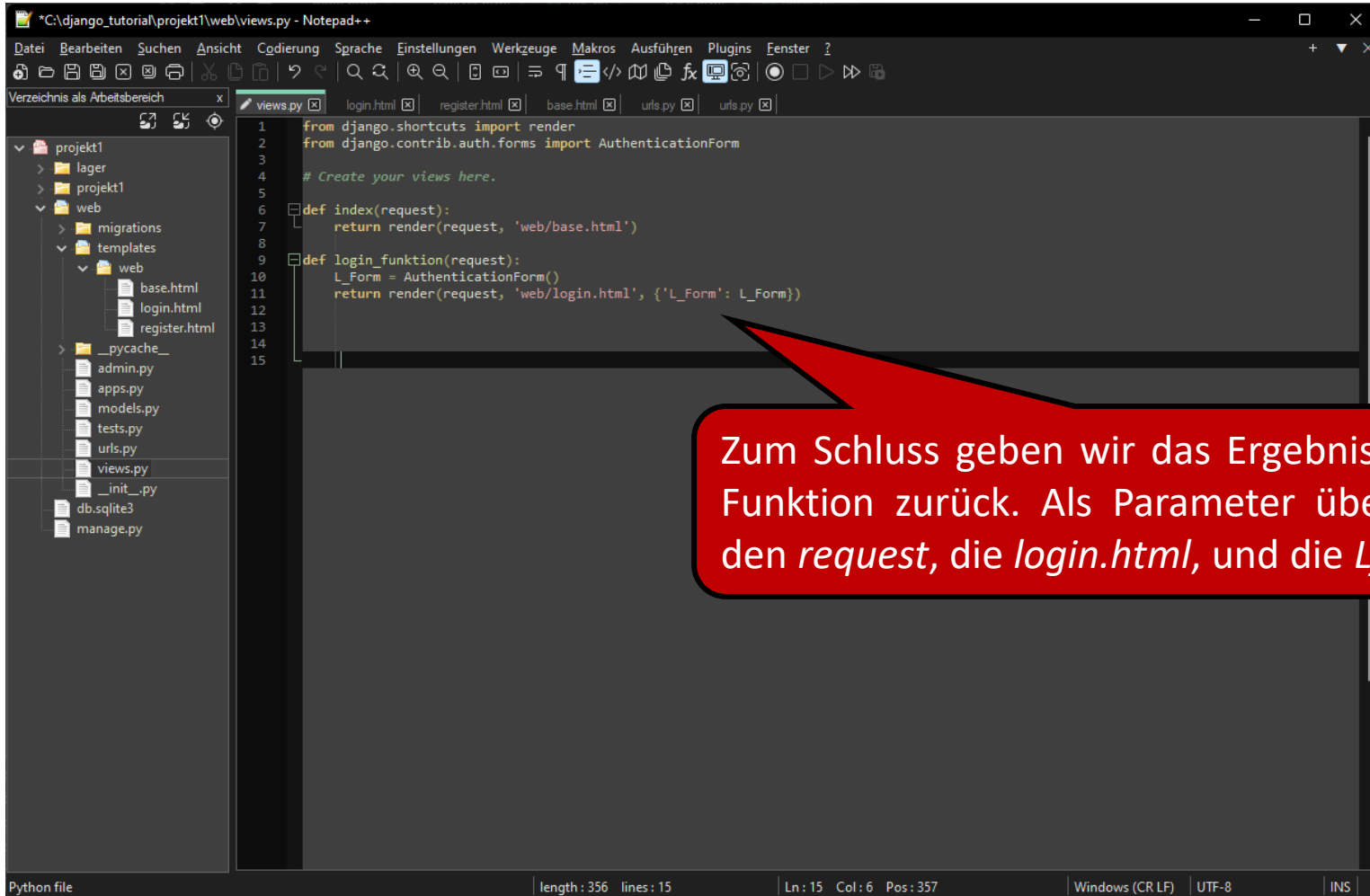


```
*C:\django_tutorial\projekt1\web\views.py - Notepad++
Datei Bearbeiten Suchen Ansicht Codierung Sprache Einstellungen Werkzeuge Makros Ausführen Plugins Fenster ?
Verzeichnis als Arbeitsbereich x
views.py login.html register.html base.html urls.py urls.py
1 from django.shortcuts import render
2 from django.contrib.auth.forms import AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13
14
15
```

Wir müssen diese fertige Django Form noch importieren.

Python file | length: 356 lines: 15 | Ln: 15 Col: 6 Pos: 357 | Windows (CR LF) | UTF-8 | INS

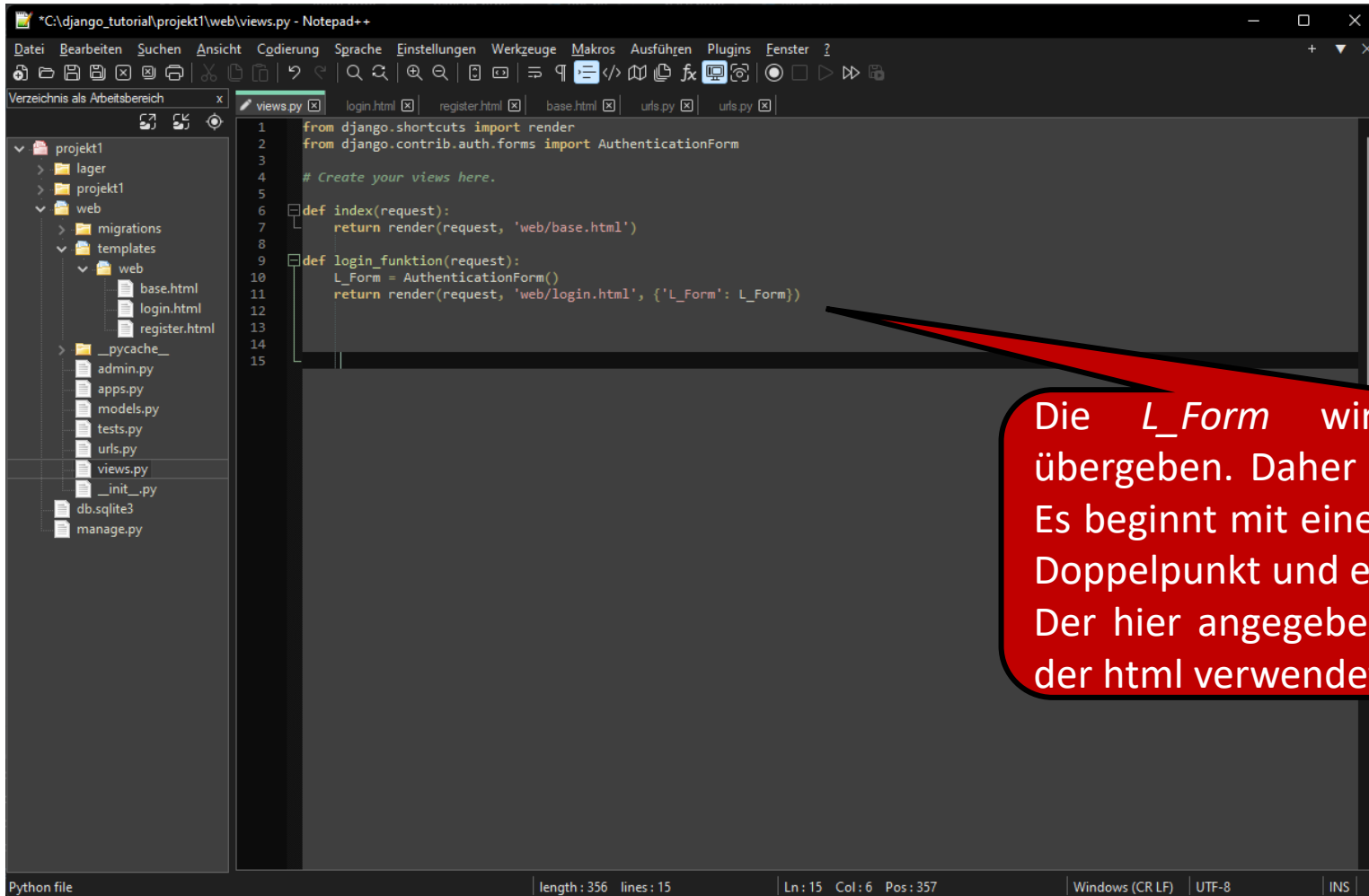
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13
14
15
```

Zum Schluss geben wir das Ergebnis der *render* Funktion zurück. Als Parameter übergeben wir den *request*, die *login.html*, und die *L_Form*.

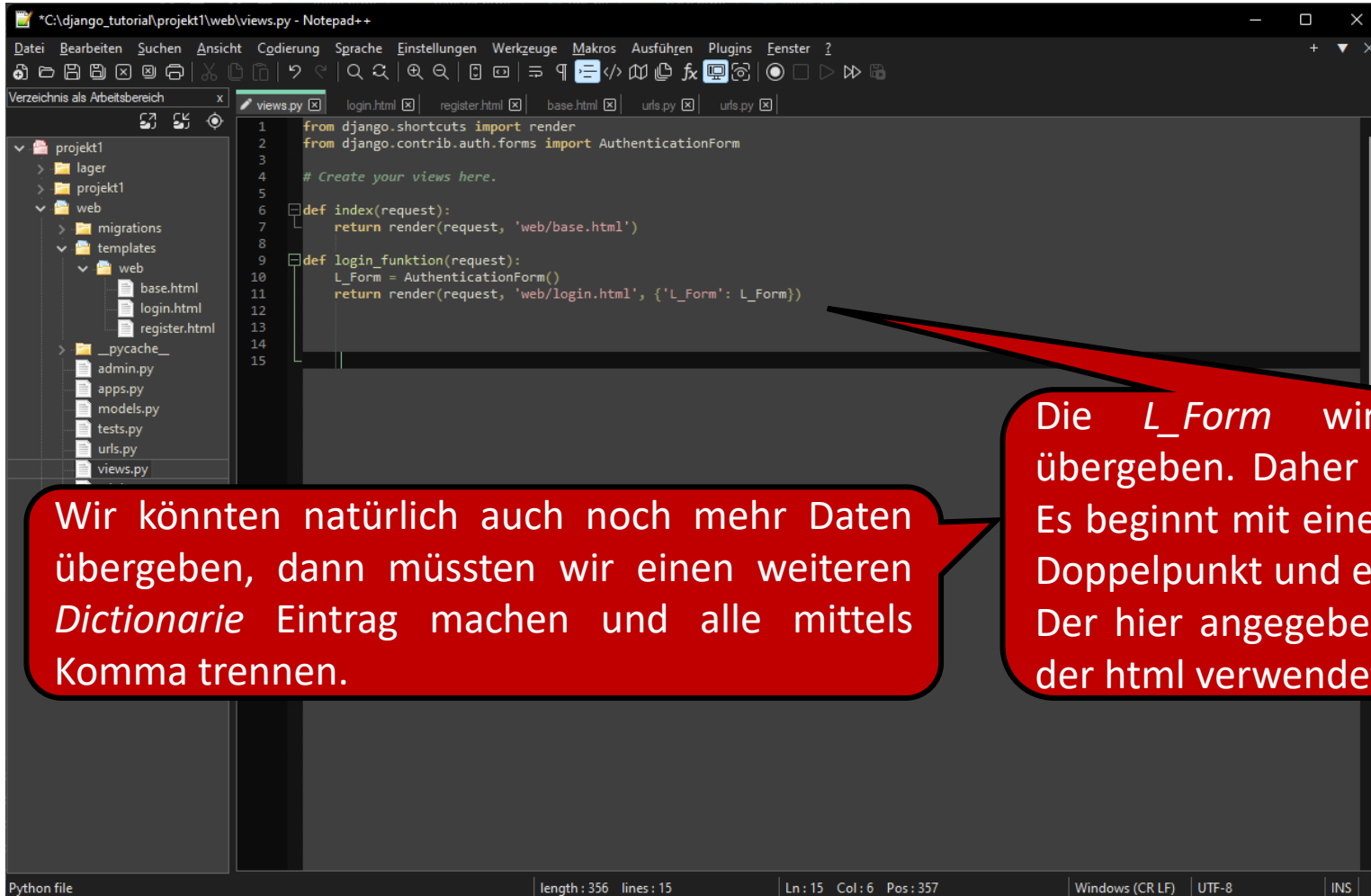
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13
14
15
```

Die *L_Form* wird dabei als *Dictionarie* übergeben. Daher die geschweiften Klammern. Es beginnt mit einem Namen gefolgt von einem Doppelpunkt und es kommen dann die Daten. Der hier angegebene Name wird dann auch in der html verwendet.

Login und Registrierung



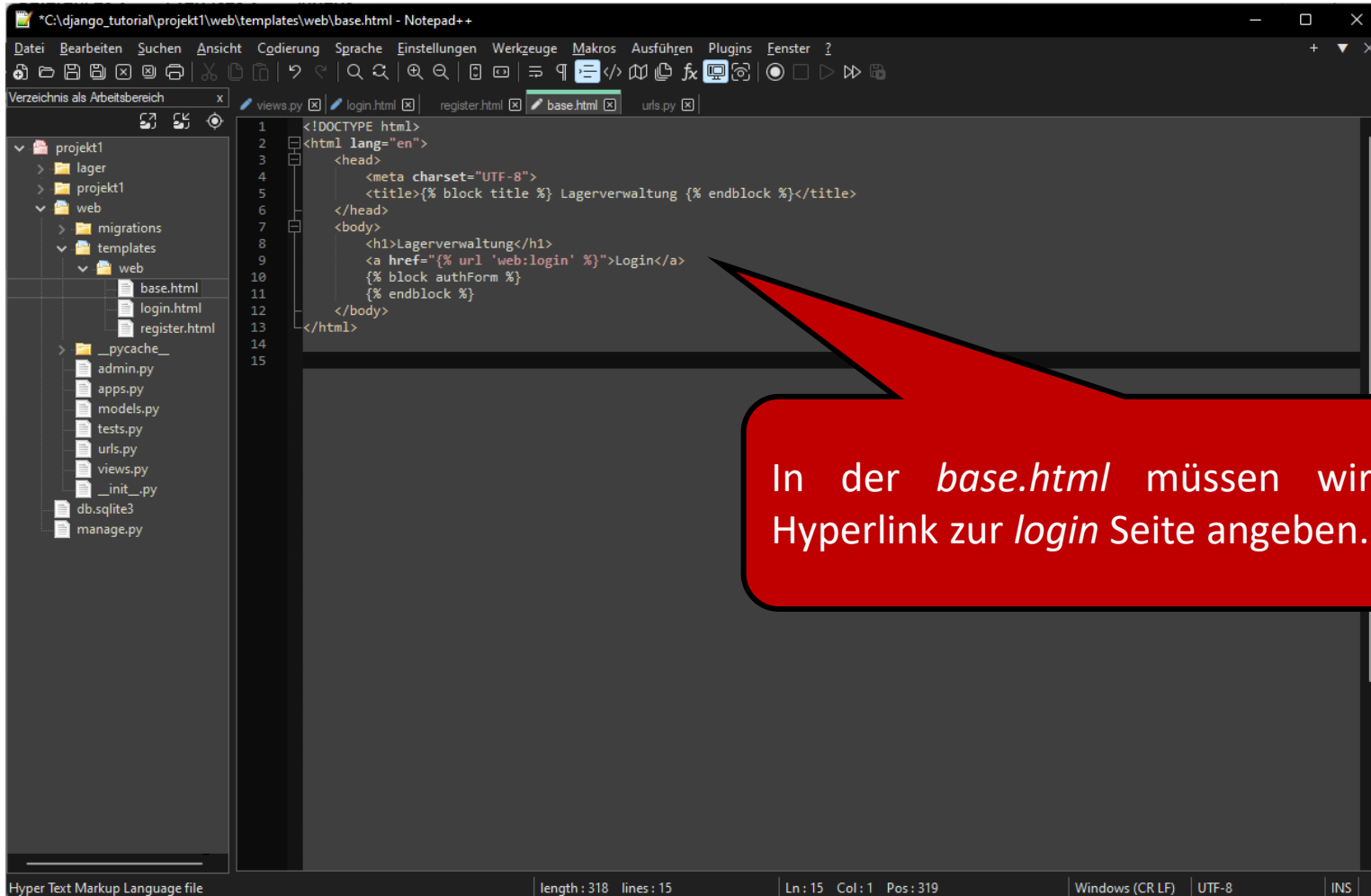
```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13
14
15
```

Python file | length: 356 | lines: 15 | Ln: 15 | Col: 6 | Pos: 357 | Windows (CR LF) | UTF-8 | INS

Wir könnten natürlich auch noch mehr Daten übergeben, dann müssten wir einen weiteren *Dictionary* Eintrag machen und alle mittels Komma trennen.

Die *L_Form* wird dabei als *Dictionary* übergeben. Daher die geschweiften Klammern. Es beginnt mit einem Namen gefolgt von einem Doppelpunkt und es kommen dann die Daten. Der hier angegebene Name wird dann auch in der html verwendet.

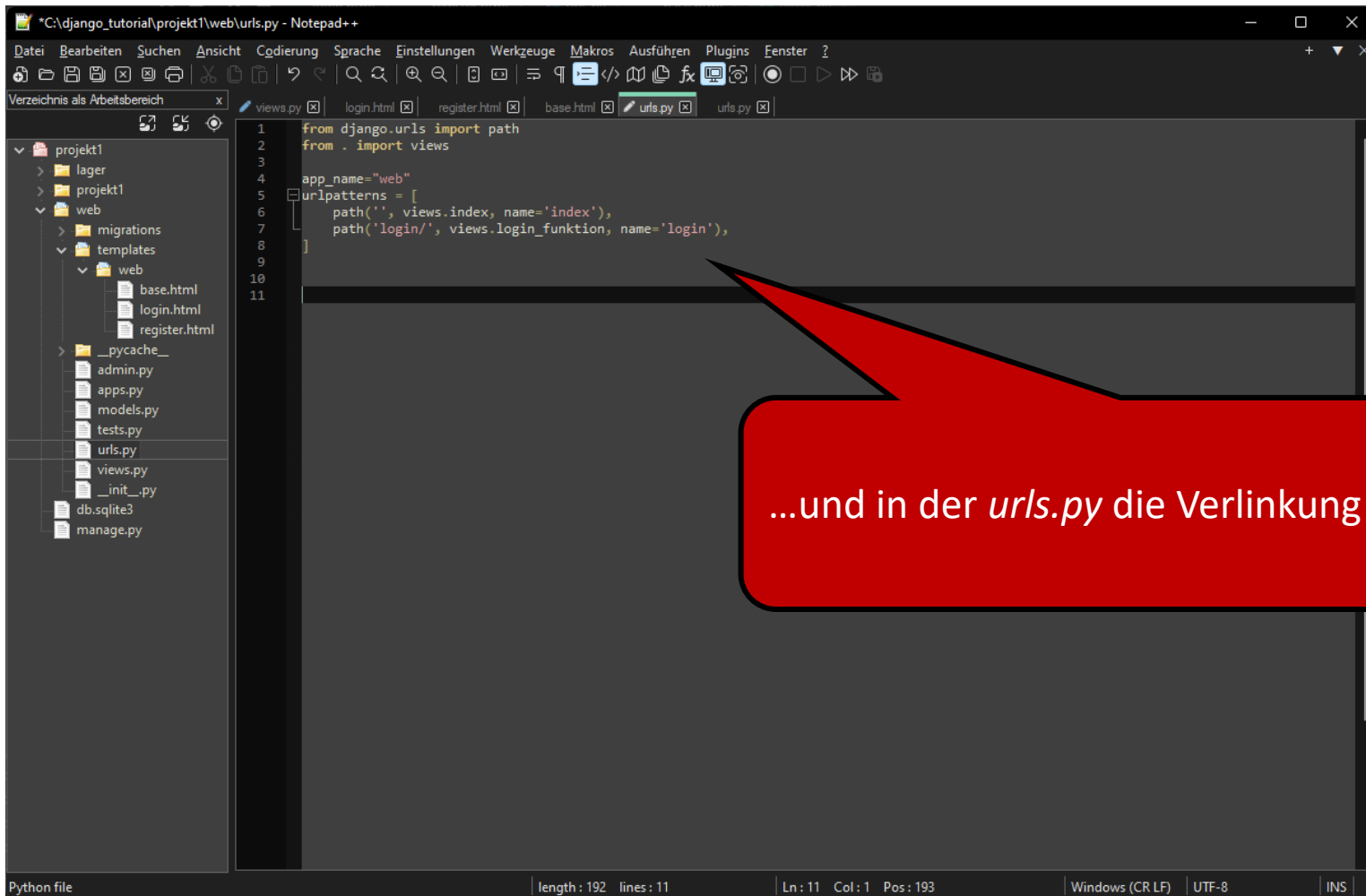
Login und Registrierung



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8   <h1>Lagerverwaltung</h1>
9   <a href="{% url 'web:login' %}">Login</a>
10   {% block authForm %}
11   {% endblock %}
12 </body>
13 </html>
14
15
```

In der *base.html* müssen wir noch einen Hyperlink zur *login* Seite angeben...

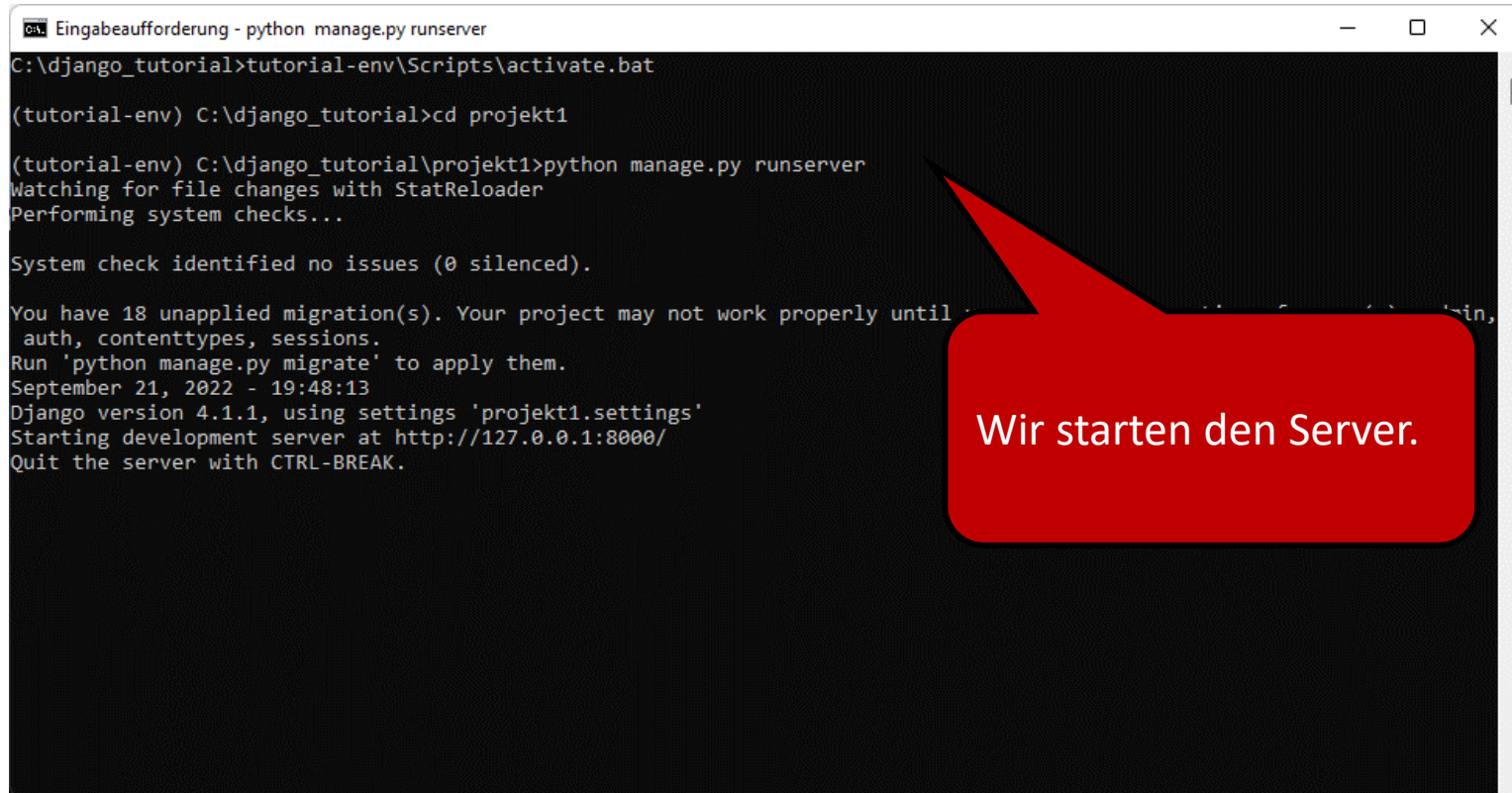
Login und Registrierung



```
1 from django.urls import path
2 from . import views
3
4 app_name="web"
5 urlpatterns = [
6     path('', views.index, name='index'),
7     path('login/', views.login_funktion, name='login'),
8 ]
9
10
11
```

...und in der *urls.py* die Verlinkung zur Funktion.

Login und Registrierung



```
C:\django_tutorial>tutorial-env\Scripts\activate.bat
(tutorial-env) C:\django_tutorial>cd projekt1
(tutorial-env) C:\django_tutorial\projekt1>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until
  auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
September 21, 2022 - 19:48:13
Django version 4.1.1, using settings 'projekt1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Wir starten den Server.

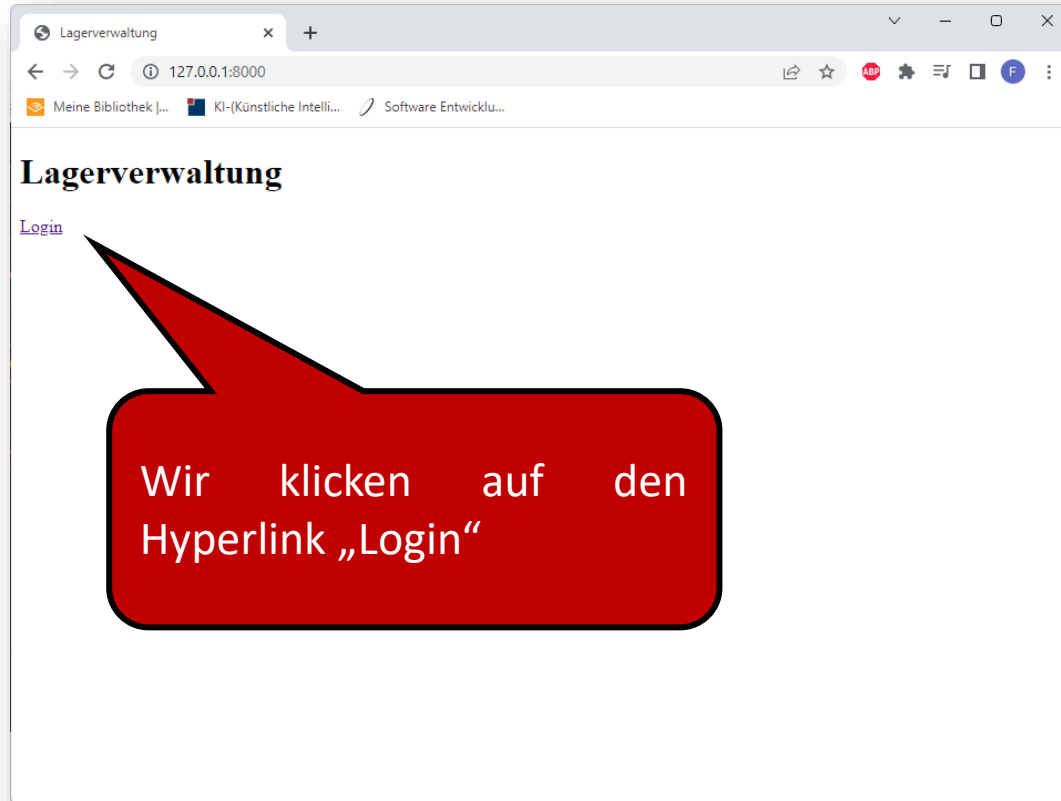
Login und Registrierung



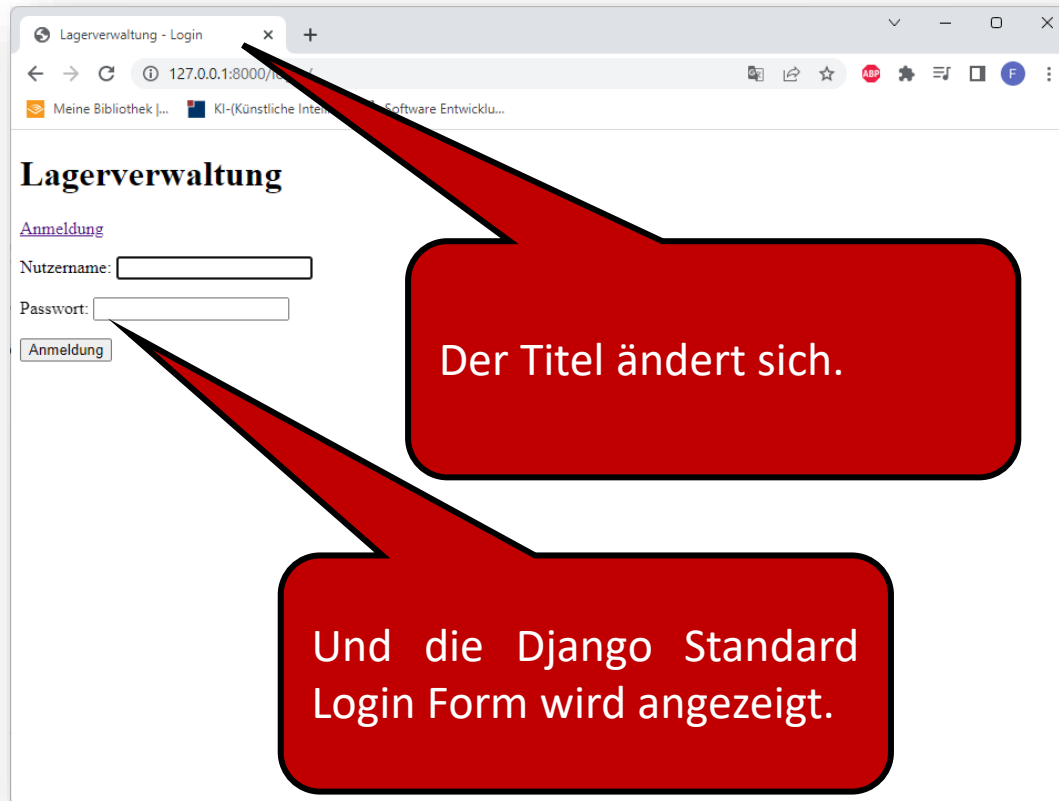
Login und Registrierung



Login und Registrierung



Login und Registrierung



Login und Registrierung

Lagerverwaltung - Login

127.0.0.1:8000/login/

Meine Bibliothek],... KI-{Künstliche Intelli... Software Entwicklu...

Lagerverwaltung

[Anmeldung](#)

Nutzername:

Passwort:

Anmeldung

Wenn wir irgendetwas angeben...

Login und Registrierung

Lagerverwaltung - Login

127.0.0.1:8000/login/

Meine Bibliothek [...]

KI-(Künstliche Intelli...

Software Entwicklu...

Lagerverwaltung

[Anmeldung](#)

Nutzername:

Passwort:

Anmeldung

...kommt die Fehlermeldung (Debug-Ausgabe), dass der CSRF-Token fehlt.

403 Verboten

127.0.0.1:8000/login/

Meine Bibliothek [...]

KI-(Künstliche Intelli...

Software Entwicklu...

Verboten (403)

CSRF-Verifizierung fehlgeschlagen. Anfrage abgebrochen.

Sie sehen diese Nachricht, weil diese Website beim Senden von Formularen ein CSRF-Cookie erfordert. Dieses Cookie ist aus Sicherheitsgründen erforderlich, um sicherzustellen, dass Ihr Browser nicht von Dritten gekapert wird.

Wenn Sie Ihren Browser so konfiguriert haben, dass Cookies deaktiviert werden, aktivieren Sie sie bitte erneut, zumindest für diese Website oder für Anfragen „gleichen Ursprungs“.

Hilfe

Angegebener Grund für das Scheitern:
CSRF-Cookie nicht gesetzt.

Im Allgemeinen kann dies auftreten, wenn eine echte Cross-Site-Request-Fälschung vorliegt oder wenn [der CSRF-Mechanismus von Django](#) nicht korrekt verwendet wurde. Für POST-Formulare müssen Sie Folgendes sicherstellen:

- Ihr Browser akzeptiert Cookies.
- Die Ansichtsfunktion übergibt an die Methode `request` des Templates `render`.
- In der Vorlage gibt es `{% csrf_token %}` in jedem POST-Formular ein Vorlagen-Tag, das auf eine interne URL abzielt.
- Wenn Sie nicht verwenden `csrfViewMiddleware`, müssen Sie `csrf_protect` für alle Ansichten verwenden, die das `csrf_token` Template-Tag verwenden, sowie für diejenigen, die die POST-Daten akzeptieren.
- Das Formular hat ein gültiges CSRF-Token. Nachdem Sie sich in einem anderen Browser-Tab angemeldet oder nach einer Anmeldung auf die Schaltfläche „Zurück“ geklickt haben, müssen Sie möglicherweise die Seite mit dem Formular neu laden, da der Token nach einer Anmeldung rotiert wird.

Sie sehen den Hilfebereich dieser Seite, weil Sie ihn `DEBUG = True` in Ihrer Django-Einstellungsdatei haben. Ändern Sie dies in `False`, und nur die anfängliche Fehlermeldung wird angezeigt.

Sie können diese Seite mit der Einstellung `CSRF_FAILURE_VIEW` anpassen.

Login und Registrierung

Lagerverwaltung - Login

127.0.0.1:8000/login/

Meine Bibliothek [...]

KI-(Künstliche Intelli...

Software Entwicklu...

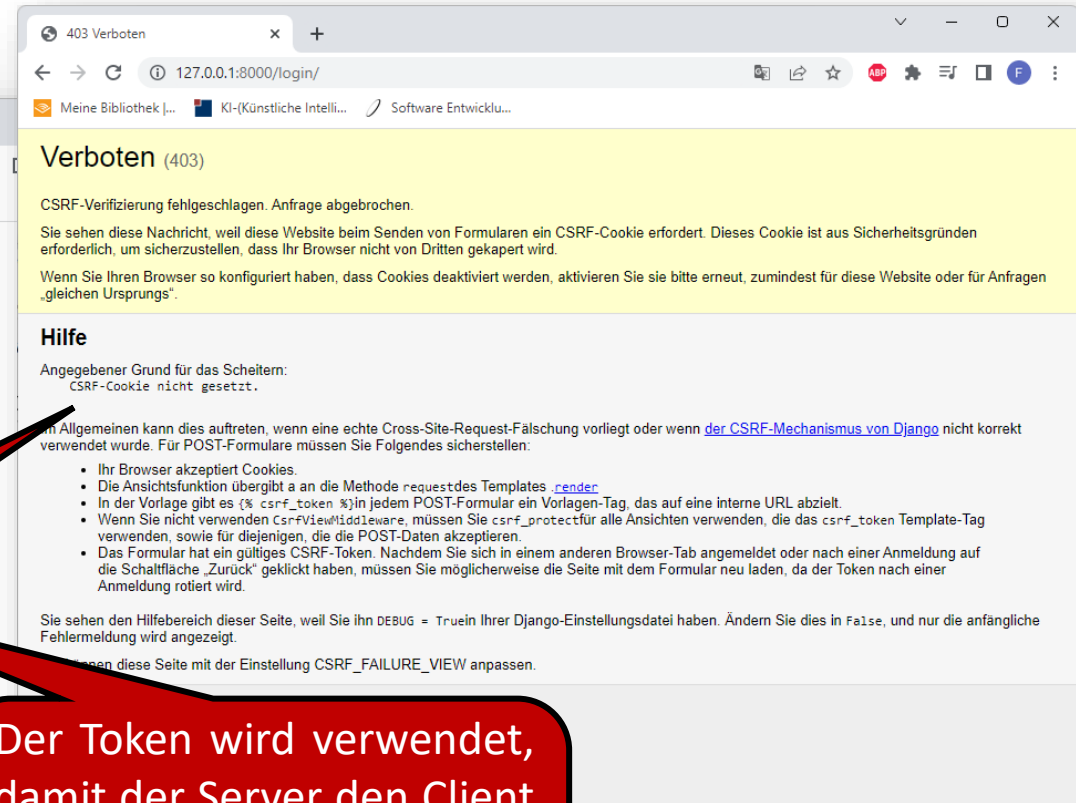
Lagerverwaltung

[Anmeldung](#)

Nutzername:

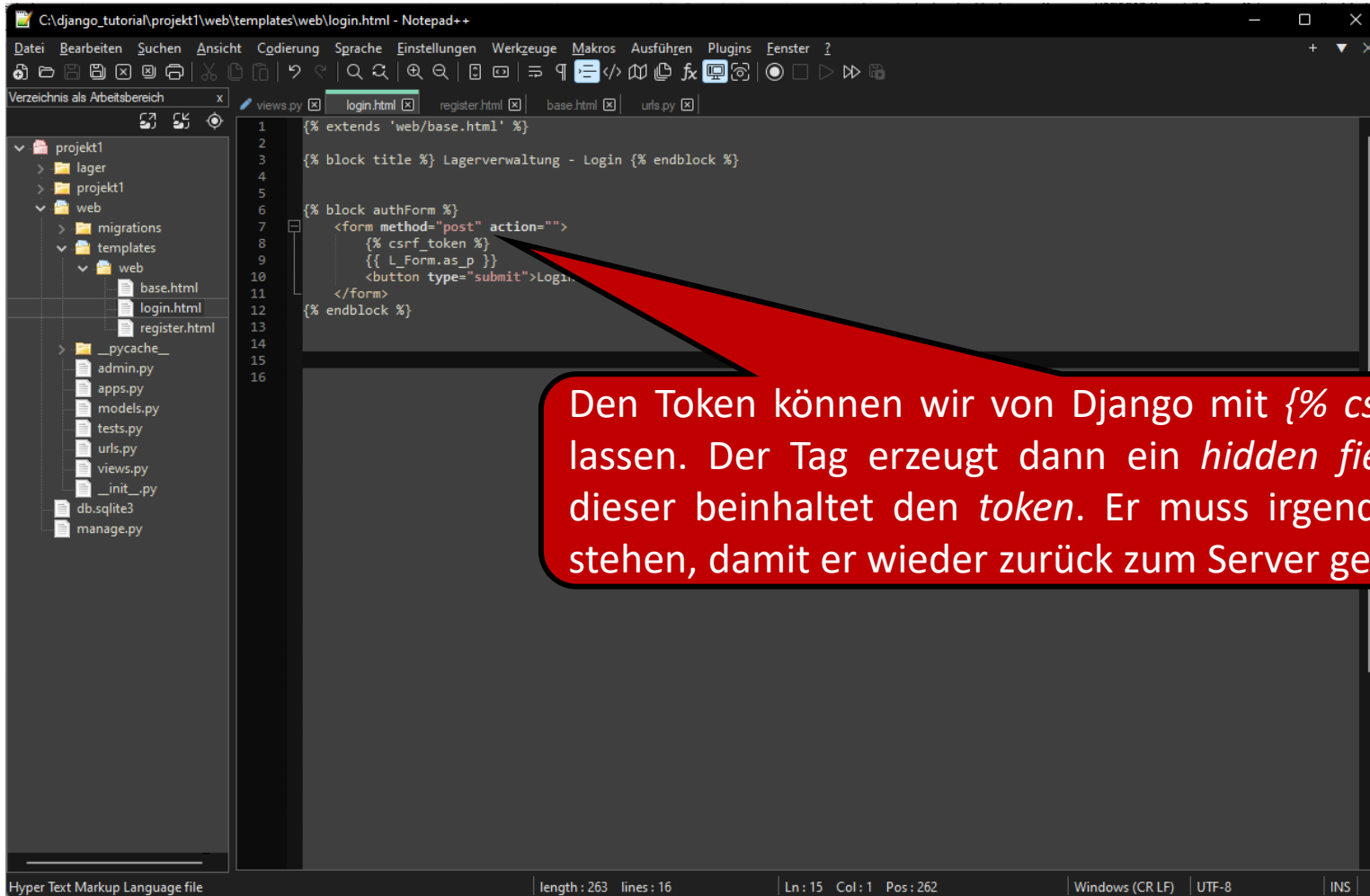
Passwort:

...kommt die Fehlermeldung (Debug-Ausgabe), dass der CSRF-Token fehlt.



Der Token wird verwendet, damit der Server den Client zuordnen kann. (Wiedererkennen kann). (Später mehr)

Login und Registrierung



The screenshot shows a Notepad++ window with the file path `C:\django_tutorial\projekt1\web\templates\web\login.html`. The editor displays the following Django template code:

```
1 {% extends 'web/base.html1' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7     <form method="post" action="">
8         {% csrf_token %}
9         {{ L_Form.as_p }}
10        <button type="submit">Login
11    </form>
12 {% endblock %}
13
14
15
16
```

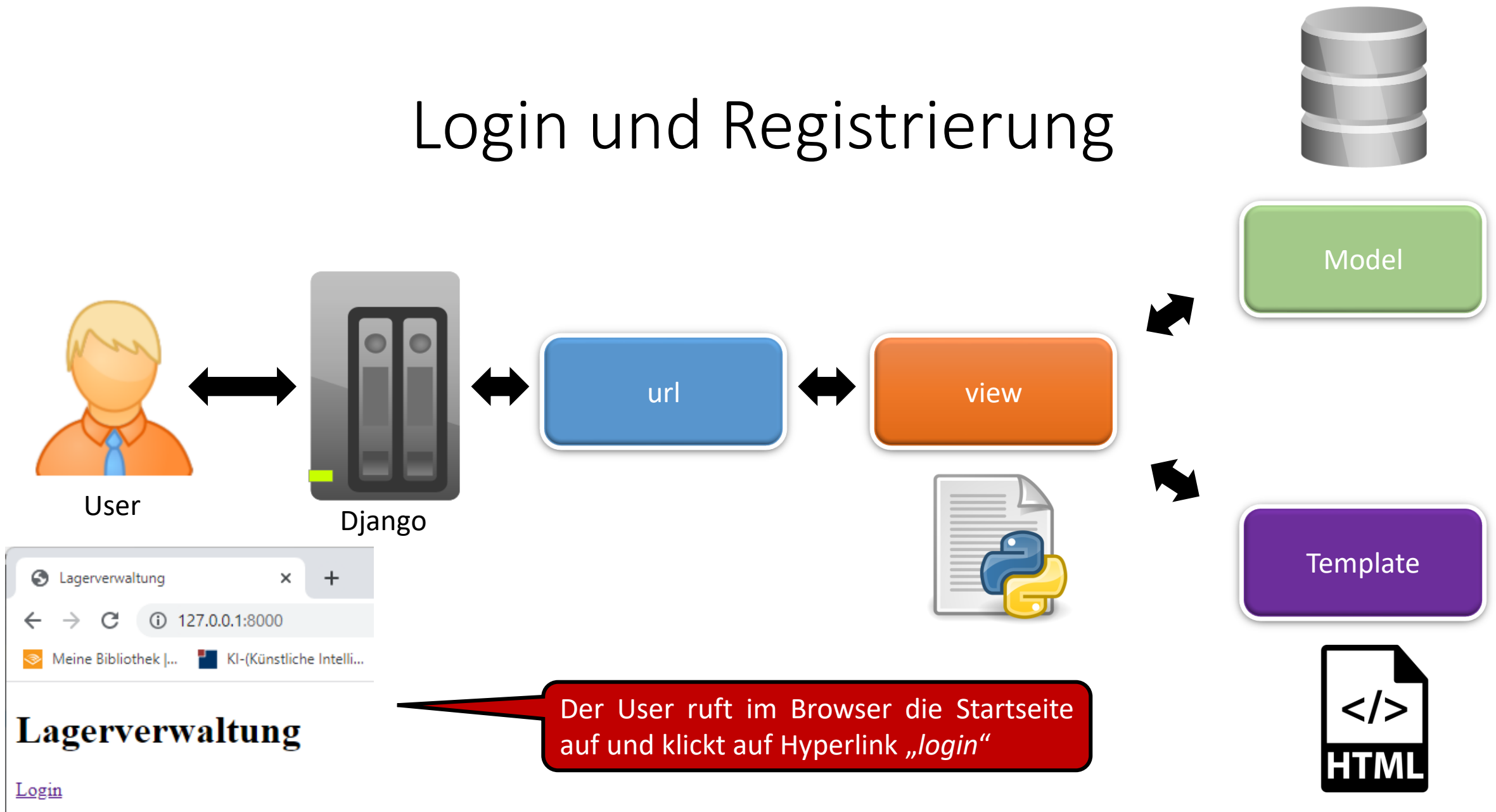
A red callout bubble points to the `{% csrf_token %}` tag on line 8. The text inside the bubble explains its purpose in Django.

Den Token können wir von Django mit `{% csrf_token %}` erzeugen lassen. Der Tag erzeugt dann ein *hidden field* in der *html* Seite, dieser beinhaltet den *token*. Er muss irgendwo in der *html form* stehen, damit er wieder zurück zum Server gegeben wird.

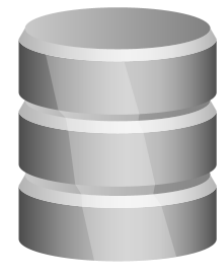
Login und Registrierung



Login und Registrierung



Login und Registrierung

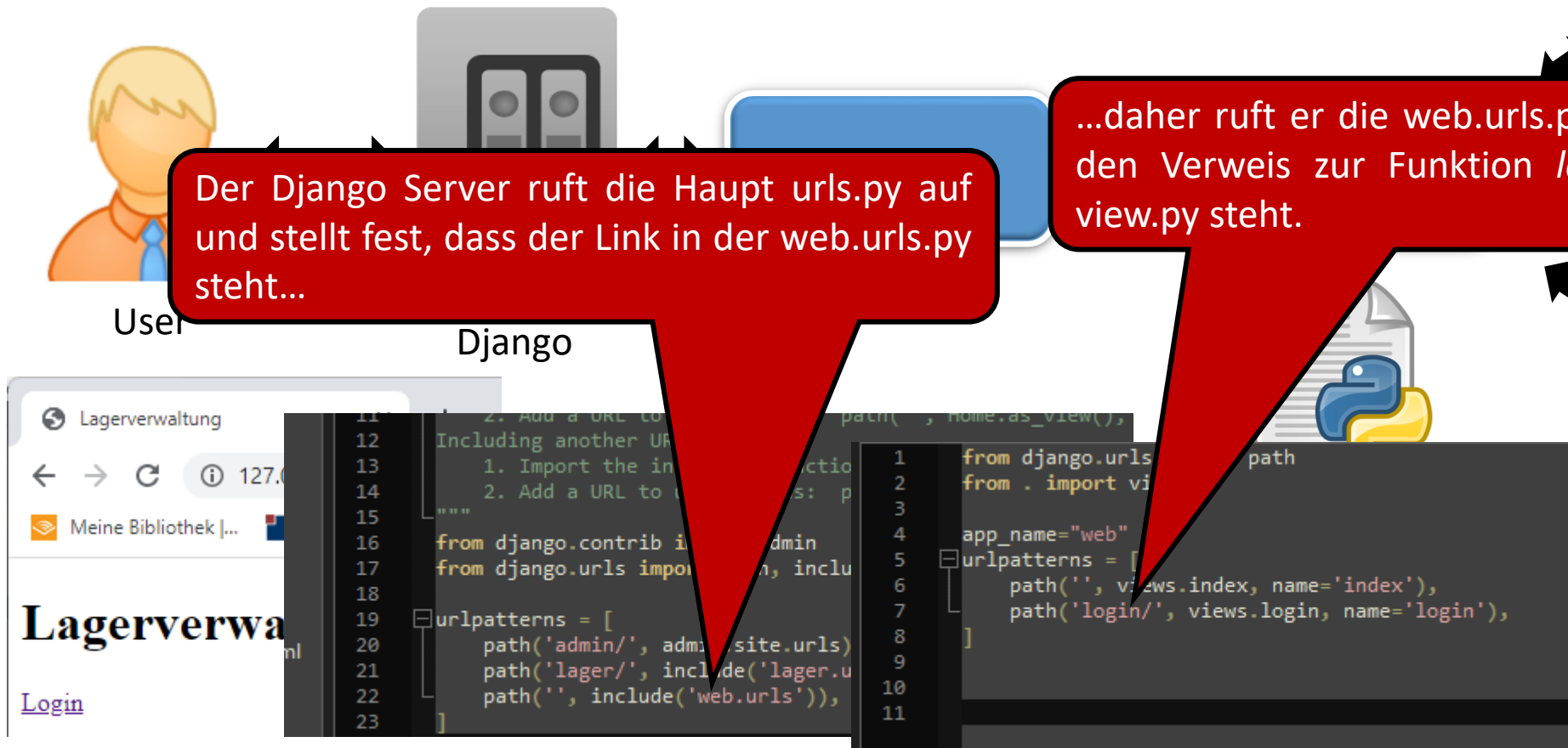


Model

...daher ruft er die web.urls.py auf und findet den Verweis zur Funktion *login*, die in der view.py steht.

Der Django Server ruft die Haupt urls.py auf und stellt fest, dass der Link in der web.urls.py steht...

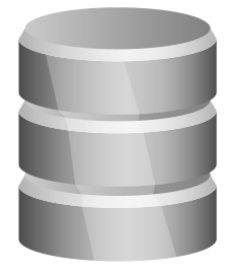
Template



Login und Registrierung

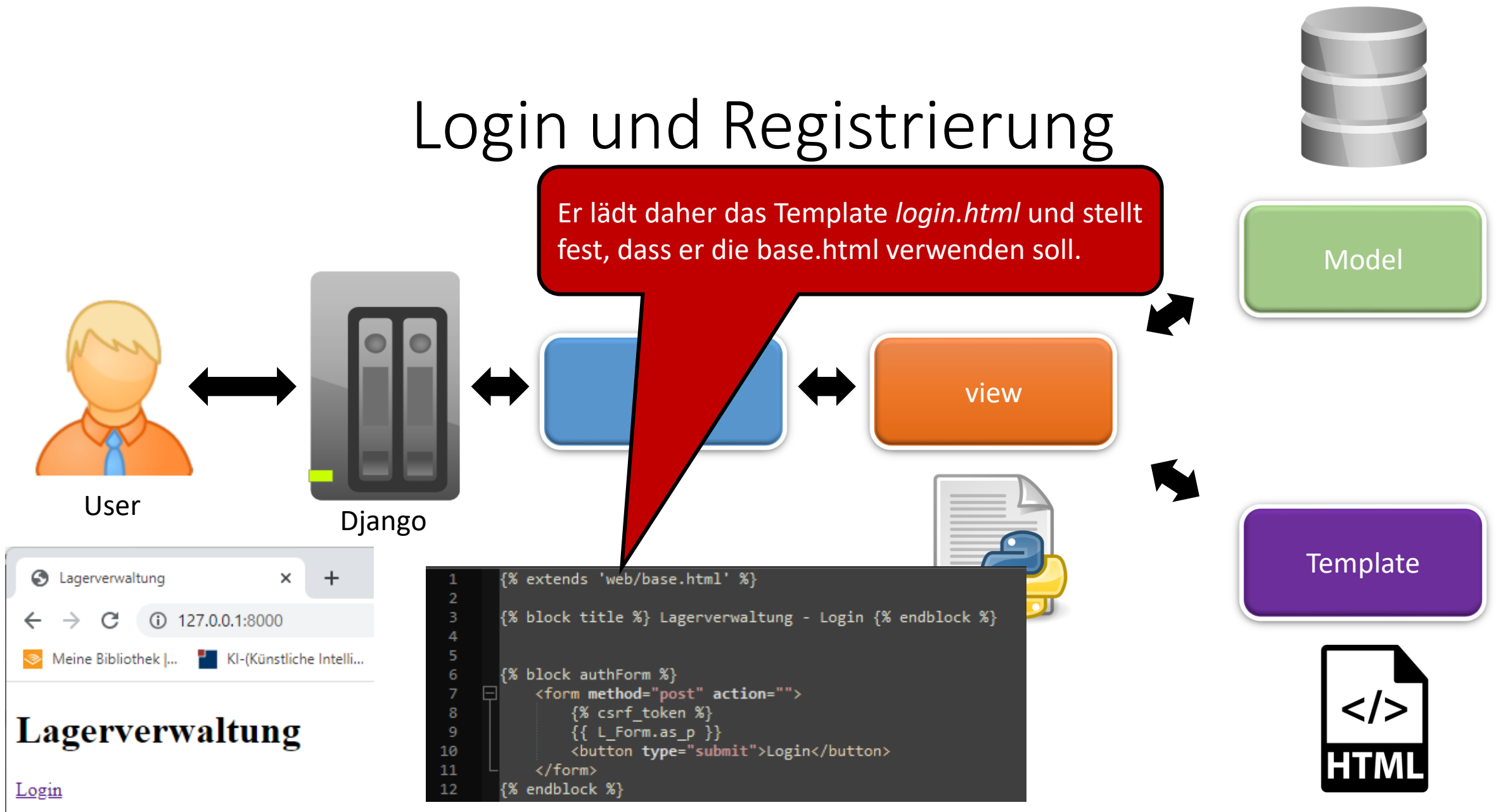


Login und Registrierung

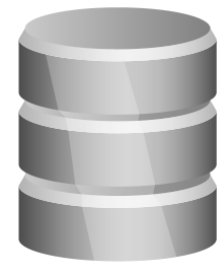


Login und Registrierung

Er lädt daher das Template *login.html* und stellt fest, dass er die *base.html* verwenden soll.



Login und Registrierung



Also lädt er das Template base.html und ersetzt die Blöcke mit den Einträgen aus login.html.



User



Django



url

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6   </head>
7   <body>
8     <h1>Lagerverwaltung</h1>
9     <a href="{% url 'web:login' %}">Login</a>
10    {% block authForm %}
11    {% endblock %}
12  </body>
13 </html>
```

Model

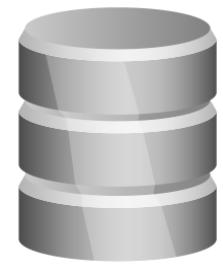
Template



```
1 {% extends 'web/base.html' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5
6 {% block authForm %}
7   <form method="post" action="">
8     {% csrf_token %}
9     {{ L_Form.as_p }}
10    <button type="submit">Login</button>
11  </form>
12 {% endblock %}
```



Login und Registrierung



Also lädt er das Template `base.html` und ersetzt die Blöcke mit den Einträgen aus `login.html`.

Anschließend rendert er noch den `csrf_token` und die `L_Form` der User bekommt also das `html-Skript...`

url

User

Django

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6   </head>
7   <body>
8     <h1>Lagerverwaltung</h1>
9     <a href="{% url 'web:login' %}">Login</a>
10    {% block authForm %}
11    {% endblock %}
12  </body>
13 </html>
```

Model

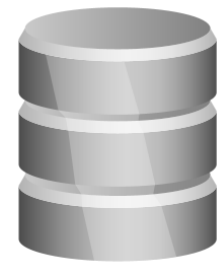
Template



```
1 extends 'web/base.html' %}
2
3 {% block title %} Lagerverwaltung - Login {% endblock %}
4
5 {% block authForm %}
6   <form method="post" action="">
7     {% csrf_token %}
8     {{ L_Form.as_p }}
9     <button type="submit">Login</button>
10  </form>
11  {% endblock %}
```



Login und Registrierung



Das wie hier aussieht.



User



Django



```
view-source:127.0.0.1:8000/login/

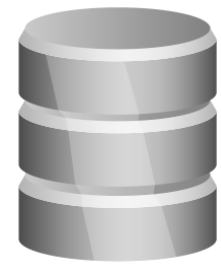
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title> Lagerverwaltung - Login </title>
  </head>
  <body>
    <h1>Lagerverwaltung</h1>
    <a href="/login/">Login</a>
    <!--<a href="/register/">Registrierung</a-->

    <form method="post" action="">
      <input type="hidden" name="csrfmiddlewaretoken" value="1xGI9e2kqAfeqHI8IOVo5x80DM0mNNfhZVs76017DunGLr7ATqrOfTvRJS0YCJAA">
      <p>
        <label for="id_username">Username:</label>
        <input type="text" name="username" autofocus autocapitalize="none" autocomplete="username" maxlength="150" required id="id_username">
      </p>

      <p>
        <label for="id_password">Password:</label>
        <input type="password" name="password" autocomplete="current-password" required id="id_password">
      </p>

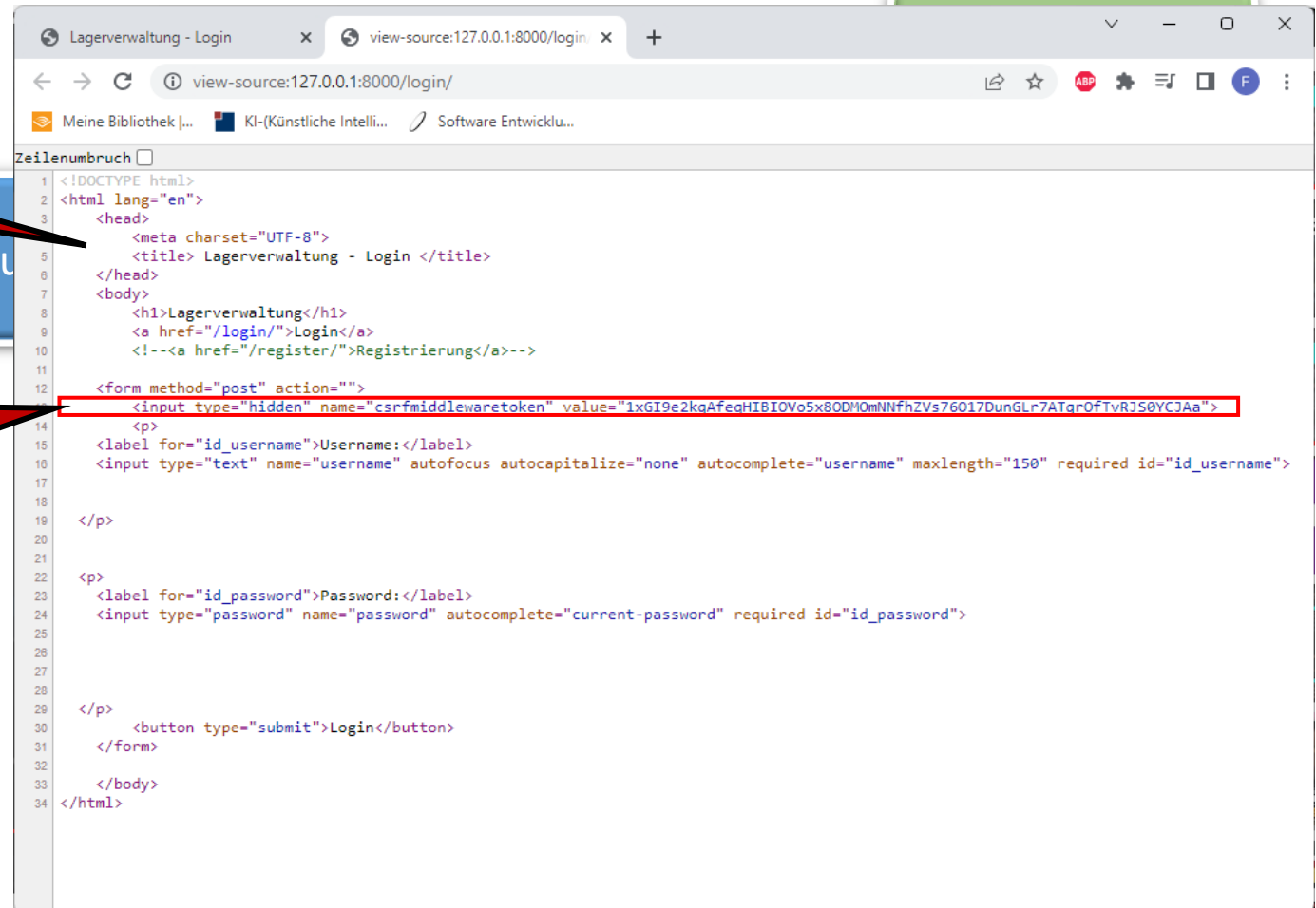
      <button type="submit">Login</button>
    </form>
  </body>
</html>
```

Login und Registrierung

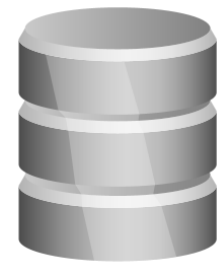


Das wie hier aussieht.

Hier ist der *CSRF-Token* den Django verlangt.



Login und Registrierung



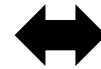
Das wie hier aussieht.



User



Service



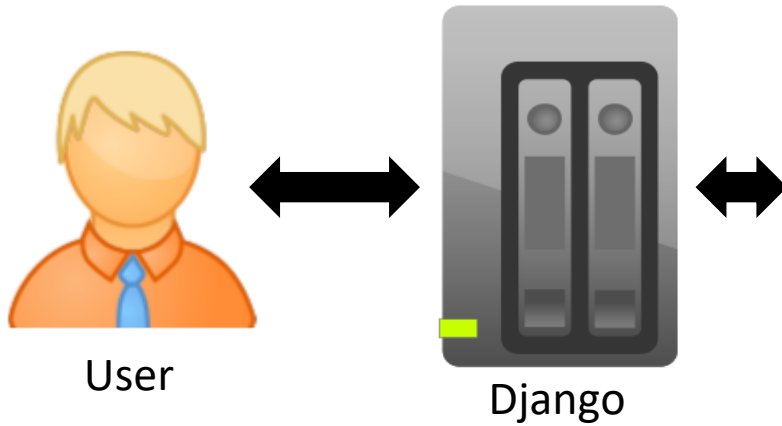
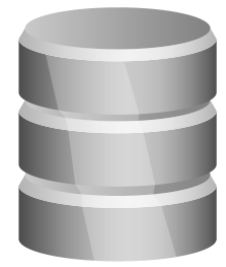
Und hier ist die gerenderte Authentikationsform.

```
Lagerverwaltung - Login x view-source:127.0.0.1:8000/login/ x +
view-source:127.0.0.1:8000/login/
Meine Bibliothek |... KI-(Künstliche Intelli... Software Entwicklu...
Zeilenumbruch
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title> Lagerverwaltung - Login </title>
6   </head>
7   <body>
8     <h1>Lagerverwaltung</h1>
9     <a href="/login/">Login</a>
10    <!--<a href="/register/">Registrierung</a-->
11
12    <form method="post" action="">
13      <input type="hidden" name="csrfmiddlewaretoken" value="1xGI9e2kqAfeqHIBIOVo5x80DMOmNNfhZVs76017DunGLr7ATqrOfTvRJS0YCJAA">
14      <p>
15        <label for="id_username">Username:</label>
16        <input type="text" name="username" autofocus autocapitalize="none" autocomplete="username" maxlength="150" required id="id_username">
17      </p>
18
19      <p>
20        <label for="id_password">Password:</label>
21        <input type="password" name="password" autocomplete="current-password" required id="id_password">
22      </p>
23
24      <button type="submit">Login</button>
25    </form>
26  </body>
27 </html>
```

Lagerverwaltung

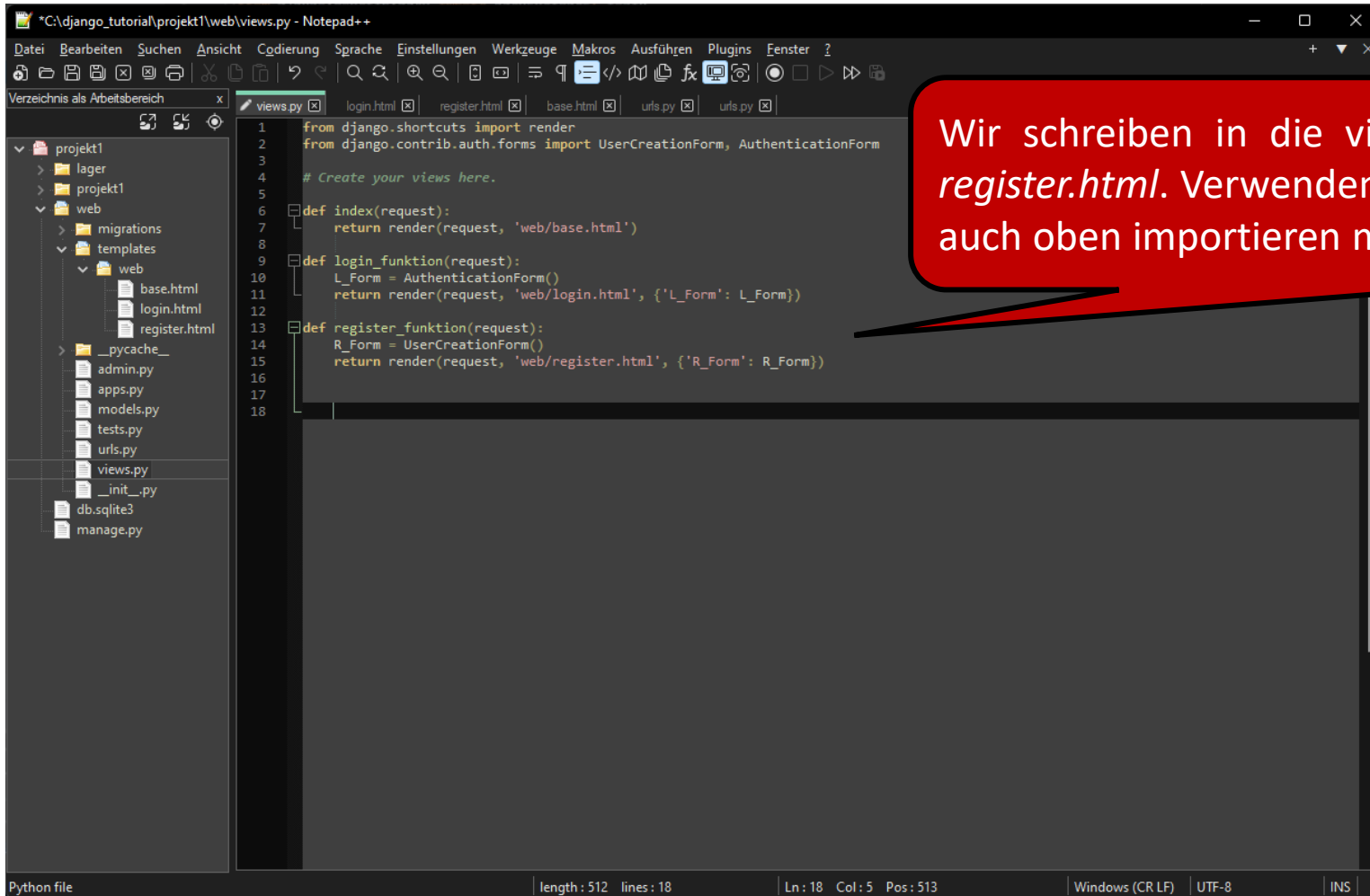
[Login](#)

Login und Registrierung

A screenshot of a web browser window titled 'Lagerverwaltung - Login'. The address bar shows '127.0.0.1:8000/login/'. The page content includes the title 'Lagerverwaltung', a link 'Anmeldung', and a login form with fields for 'Nutzername:' and 'Passwort:', followed by an 'Anmeldung' button. The browser's tab bar shows 'Lagerverwaltung - Login' and the address bar shows '127.0.0.1:8000/login/'. The page content includes the title 'Lagerverwaltung', a link 'Anmeldung', and a login form with fields for 'Nutzername:' and 'Passwort:', followed by an 'Anmeldung' button.

Der User sieht daher folgendes.

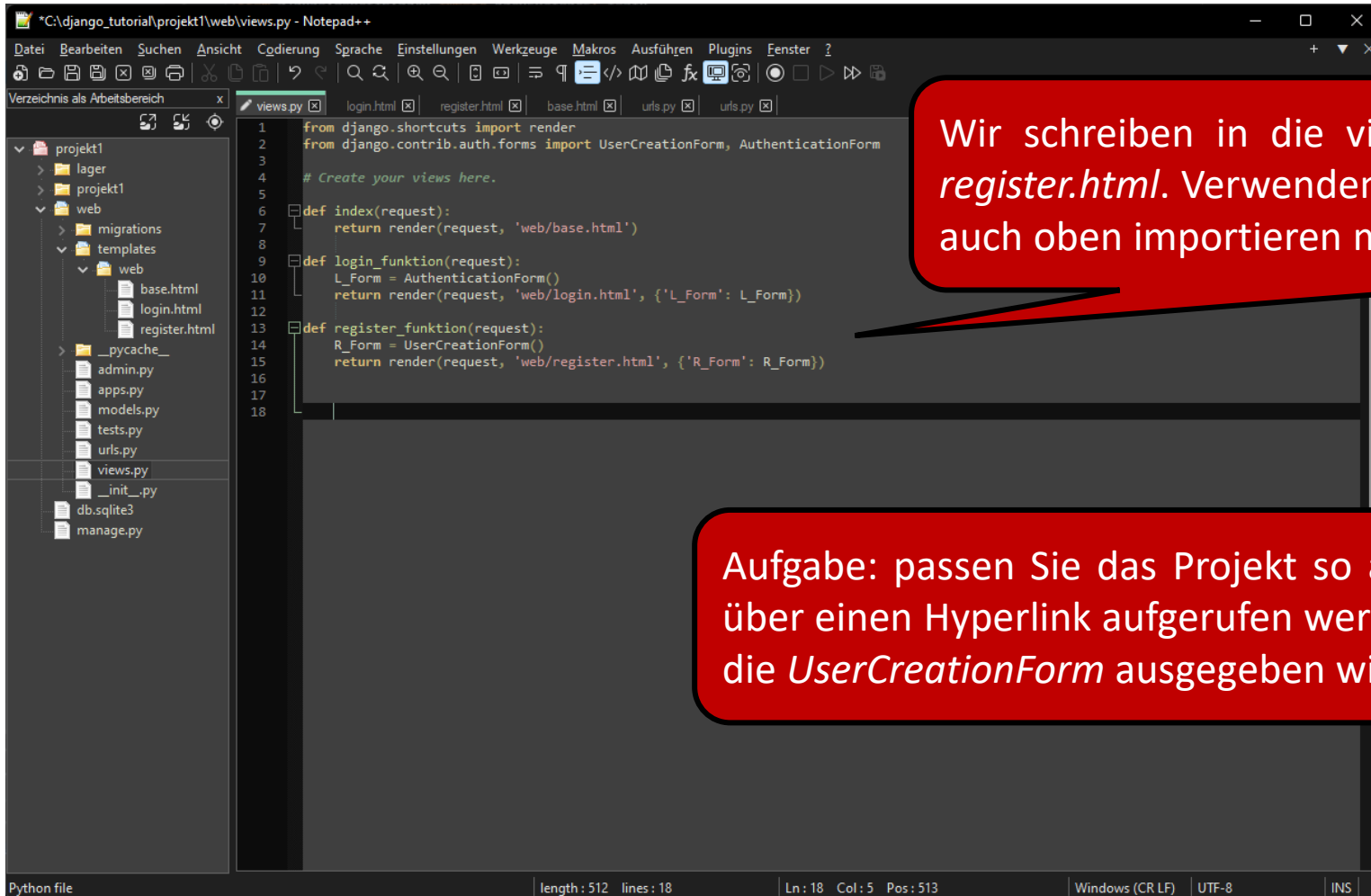
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15     return render(request, 'web/register.html', {'R_Form': R_Form})
16
17
18
```

Wir schreiben in die view.py die Funktion zum *rendern* für die *register.html*. Verwenden hier jedoch die *UserCreationForm*. Die wir auch oben importieren müssen.

Login und Registrierung



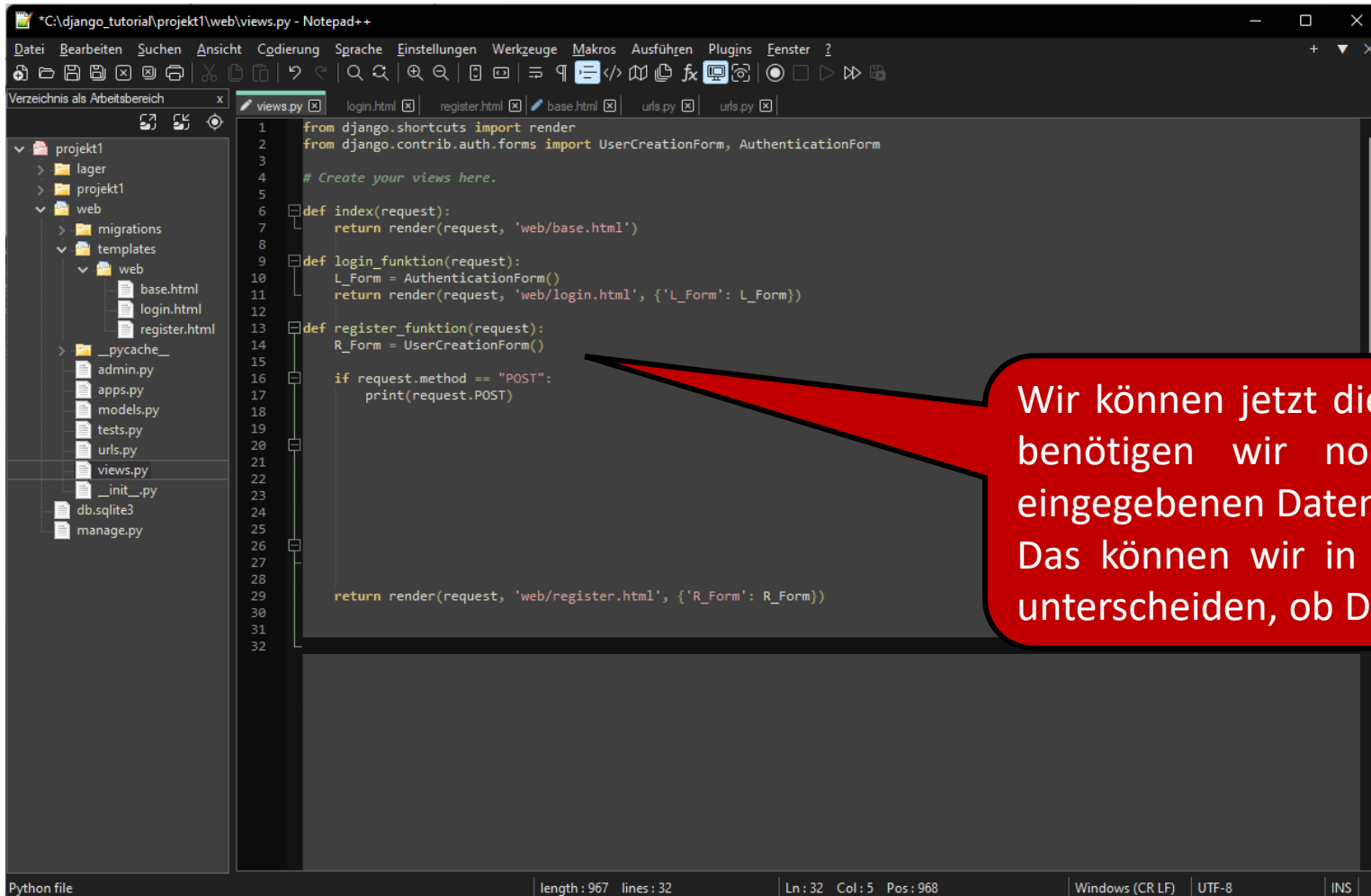
```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15     return render(request, 'web/register.html', {'R_Form': R_Form})
16
17
18
```

Python file | length: 512 | lines: 18 | Ln: 18 Col: 5 Pos: 513 | Windows (CR LF) | UTF-8 | INS

Wir schreiben in die view.py die Funktion zum *rendern* für die *register.html*. Verwenden hier jedoch die *UserCreationForm*. Die wir auch oben importieren müssen.

Aufgabe: passen Sie das Projekt so an, dass auch die *registerform* über einen Hyperlink aufgerufen werden kann und das auf der Seite die *UserCreationForm* ausgegeben wird.

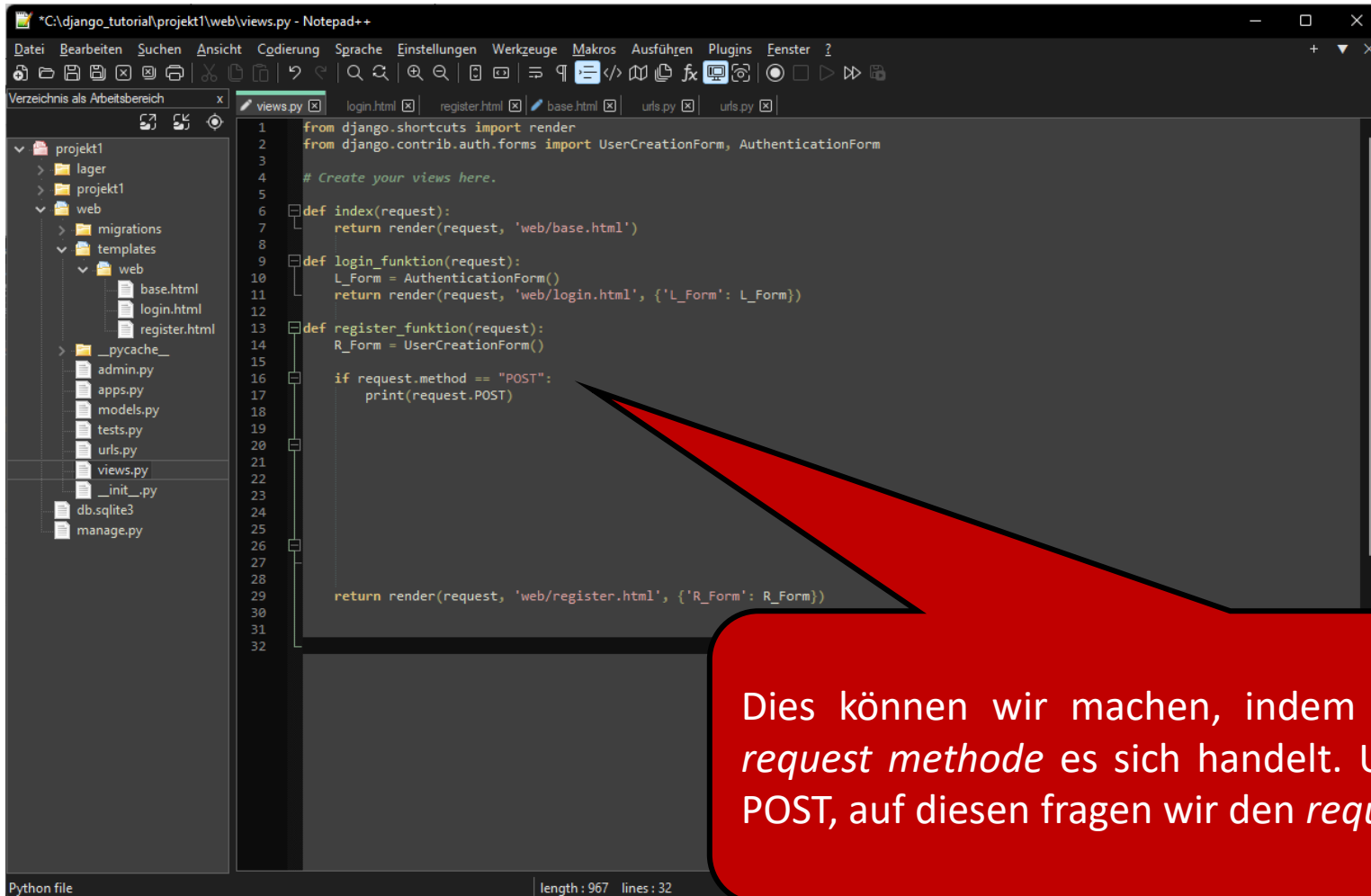
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         print(request.POST)
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Wir können jetzt die Form für die Registrierung ausgeben, jedoch benötigen wir noch die Möglichkeit, dass der Server die eingegebenen Daten entgegennimmt und einen User anlegt. Das können wir in derselben Funktion machen, wir müssen nur unterscheiden, ob Daten übergeben werden oder nicht.

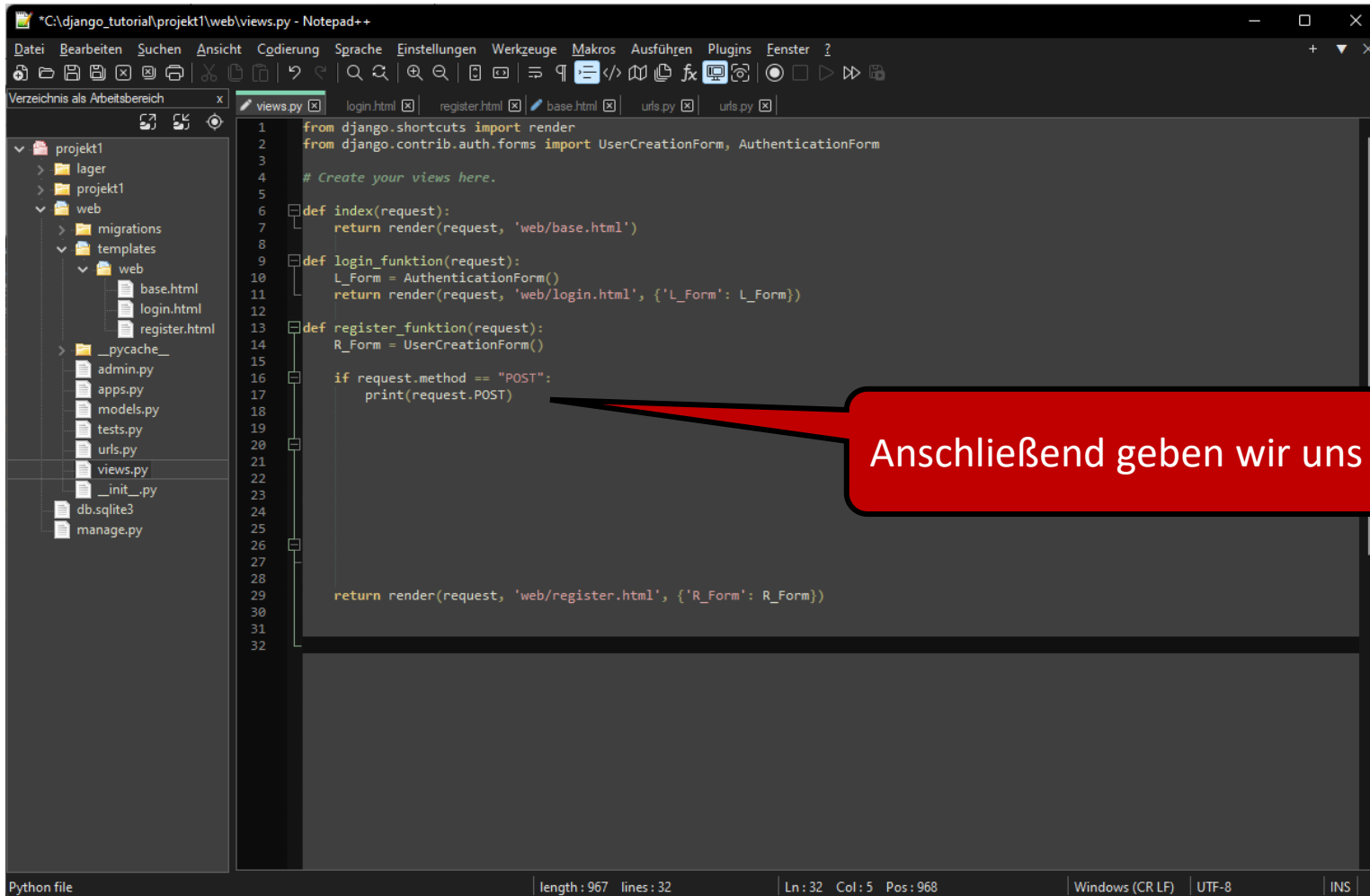
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         print(request.POST)
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Dies können wir machen, indem wir prüfen, um was für eine *request methode* es sich handelt. Unsere *html form* sendet einen POST, auf diesen fragen wir den *request* ab.

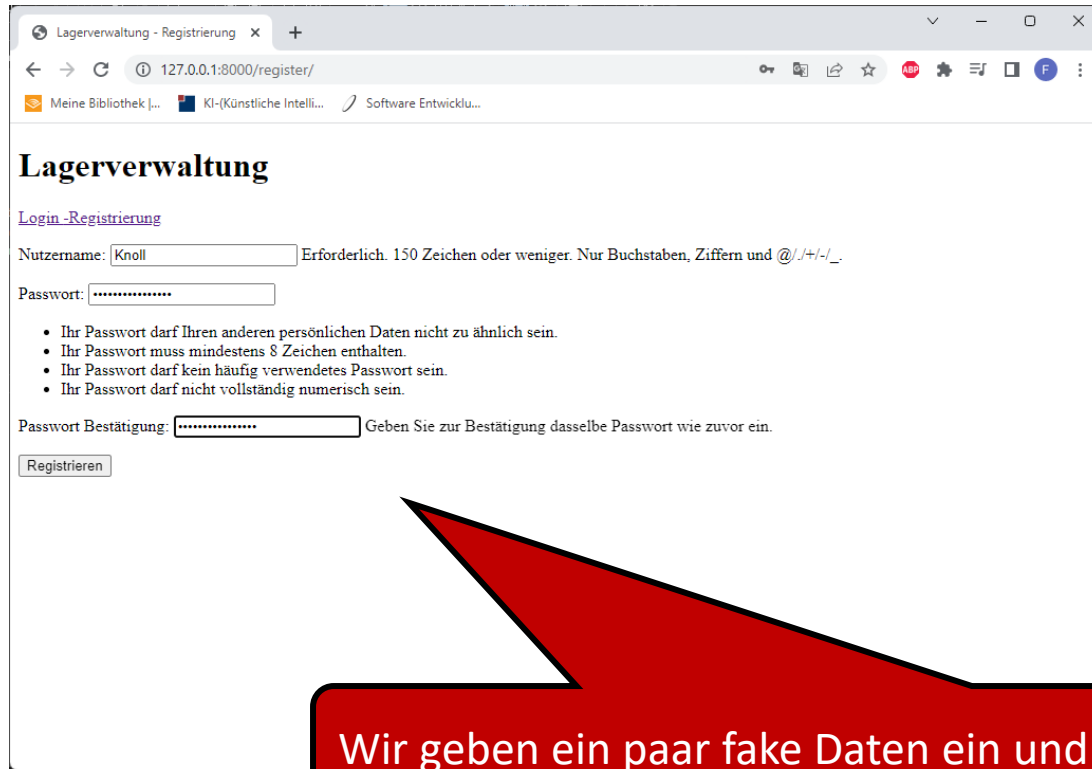
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         print(request.POST)
18
19
20
21
22
23
24
25
26
27
28
29     return render(request, 'web/register.html', {'R_Form': R_Form})
30
31
32
```

Anschließend geben wir uns den übergebenen Post einmal aus.

Login und Registrierung



Lagerverwaltung - Registrierung

127.0.0.1:8000/register/

Meine Bibliothek | KI-(Künstliche Intelli... | Software Entwicklu...

Lagerverwaltung

[Login - Registrierung](#)

Nutzername: Erforderlich. 150 Zeichen oder weniger. Nur Buchstaben, Ziffern und @/./+/-/_.

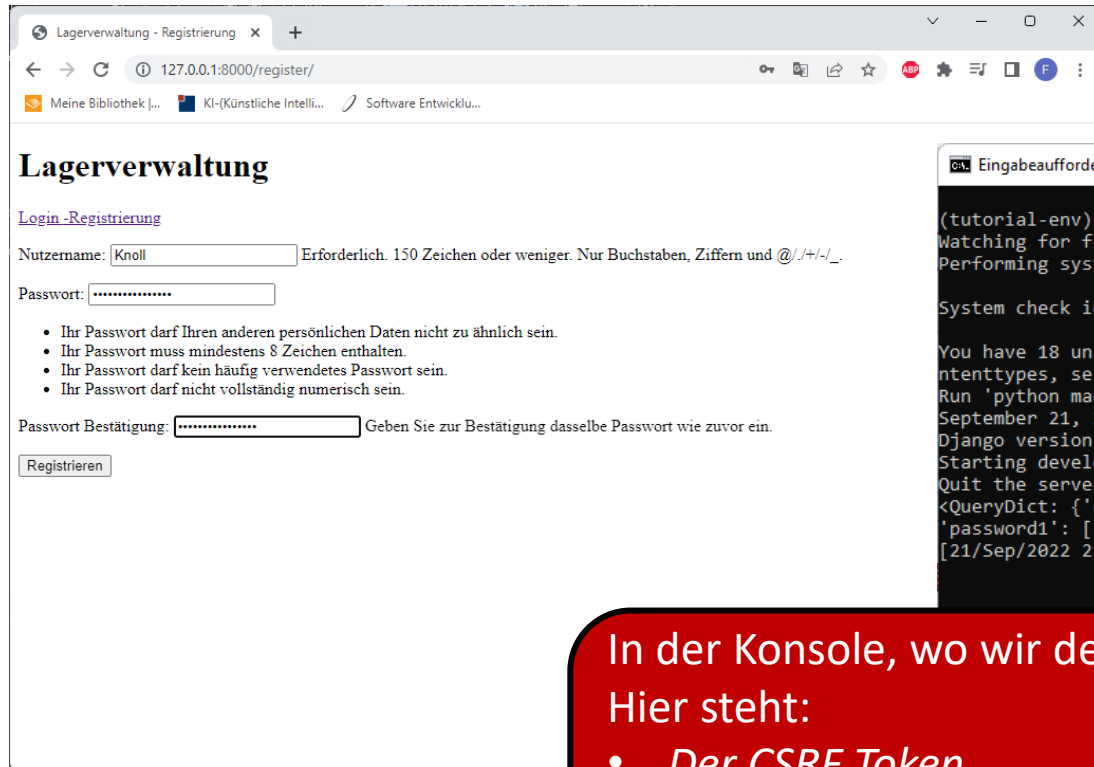
Passwort:

- Ihr Passwort darf Ihren anderen persönlichen Daten nicht zu ähnlich sein.
- Ihr Passwort muss mindestens 8 Zeichen enthalten.
- Ihr Passwort darf kein häufig verwendetes Passwort sein.
- Ihr Passwort darf nicht vollständig numerisch sein.

Passwort Bestätigung: Geben Sie zur Bestätigung dasselbe Passwort wie zuvor ein.

Wir geben ein paar fake Daten ein und klicken auf den Registrieren-Button.

Login und Registrierung



Lagerverwaltung - Registrierung

127.0.0.1:8000/register/

Lagerverwaltung

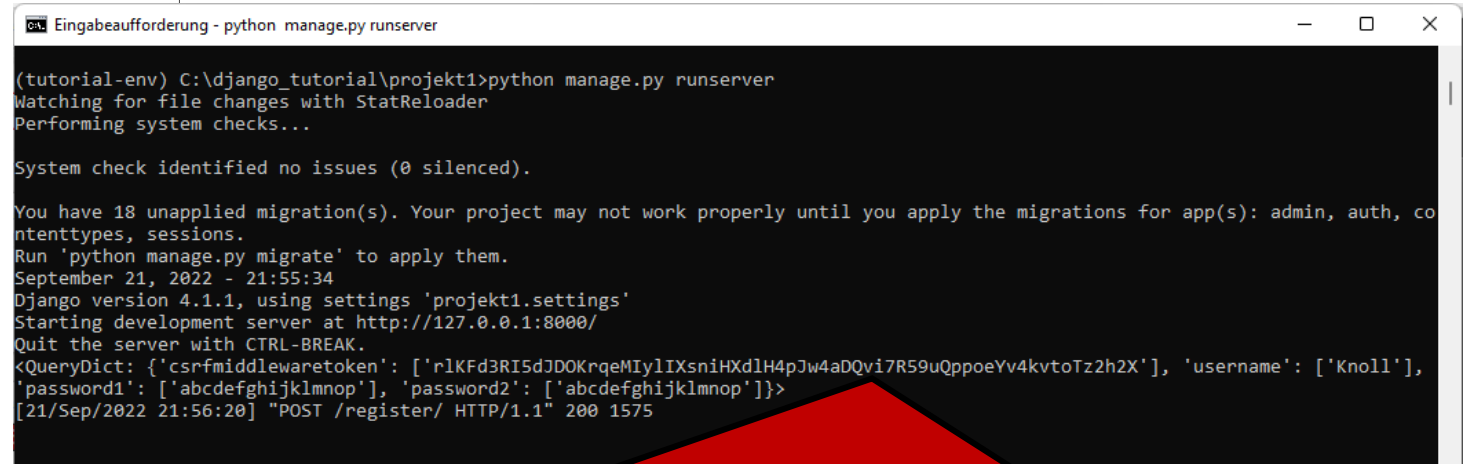
[Login-Registrierung](#)

Nutzername: Erforderlich. 150 Zeichen oder weniger. Nur Buchstaben, Ziffern und @/./+/_/.

Passwort:

- Ihr Passwort darf Ihren anderen persönlichen Daten nicht zu ähnlich sein.
- Ihr Passwort muss mindestens 8 Zeichen enthalten.
- Ihr Passwort darf kein häufig verwendetes Passwort sein.
- Ihr Passwort darf nicht vollständig numerisch sein.

Passwort Bestätigung: Geben Sie zur Bestätigung dasselbe Passwort wie zuvor ein.



```
(tutorial-env) C:\django_tutorial\projekt1>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

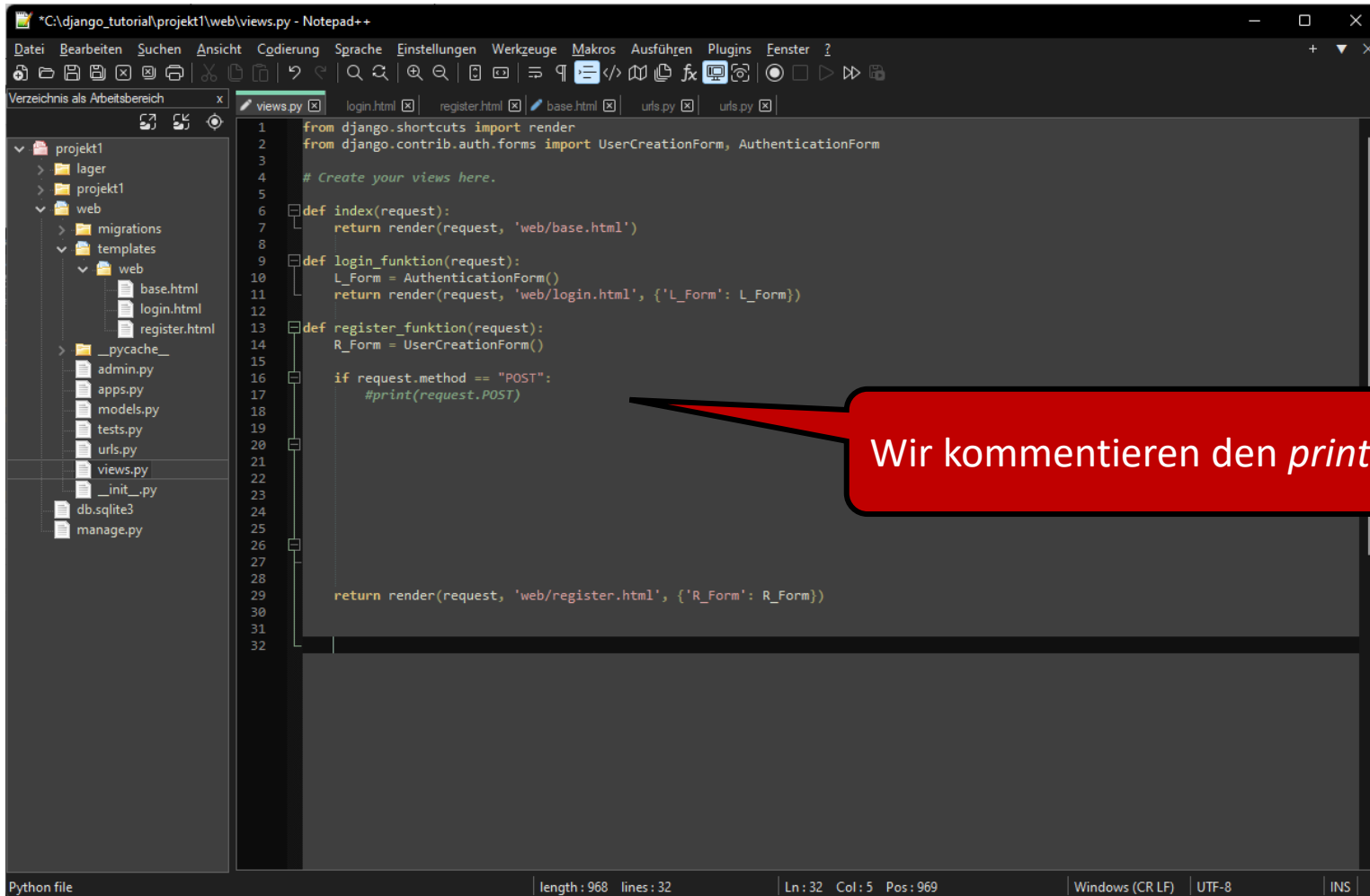
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, co
ntenttypes, sessions.
Run 'python manage.py migrate' to apply them.
September 21, 2022 - 21:55:34
Django version 4.1.1, using settings 'projekt1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
<QueryDict: {'csrfmiddlewaretoken': ['r1KFd3RI5dJDOKrqeMIy1IXsniHXdlH4pJw4aDQvi7R59uQppoeYv4kvtoTz2h2X'], 'username': ['Knoll'],
'password1': ['abcdefghijklmnop'], 'password2': ['abcdefghijklmnop']}>
[21/Sep/2022 21:56:20] "POST /register/ HTTP/1.1" 200 1575
```

In der Konsole, wo wir den Server gestartet haben, sehen wir die *print* Ausgabe. Hier steht:

- *Der CSRF Token*
- *Der Username*
- *Das Password1*
- *Das Password2*

Login und Registrierung



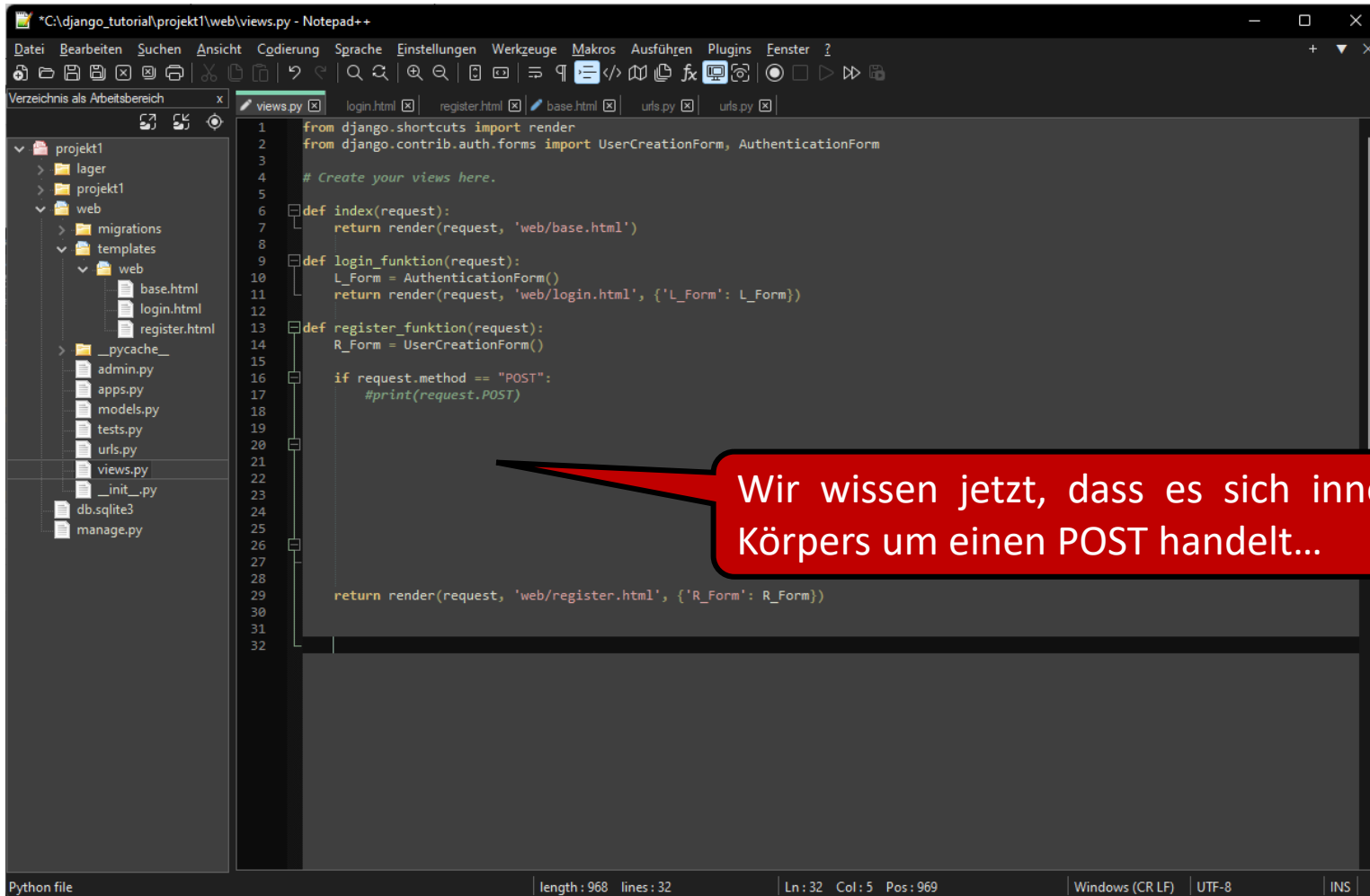
The screenshot shows a Notepad++ window titled "C:\django_tutorial\projekt1\web\views.py - Notepad++". The left sidebar displays a file explorer for the "projekt1" directory, showing subdirectories like "lager", "projekt1", "web", "migrations", "templates", and "web". The main editor area shows the following Python code:

```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19
20
21
22
23
24
25
26
27
28
29     return render(request, 'web/register.html', {'R_Form': R_Form})
30
31
32
```

A red callout bubble with a black border points to the commented-out line `#print(request.POST)` on line 17. The bubble contains the text: "Wir kommentieren den *print* erstmal wieder aus."

The status bar at the bottom indicates: "Python file | length: 968 lines: 32 | Ln: 32 Col: 5 Pos: 969 | Windows (CR LF) | UTF-8 | INS

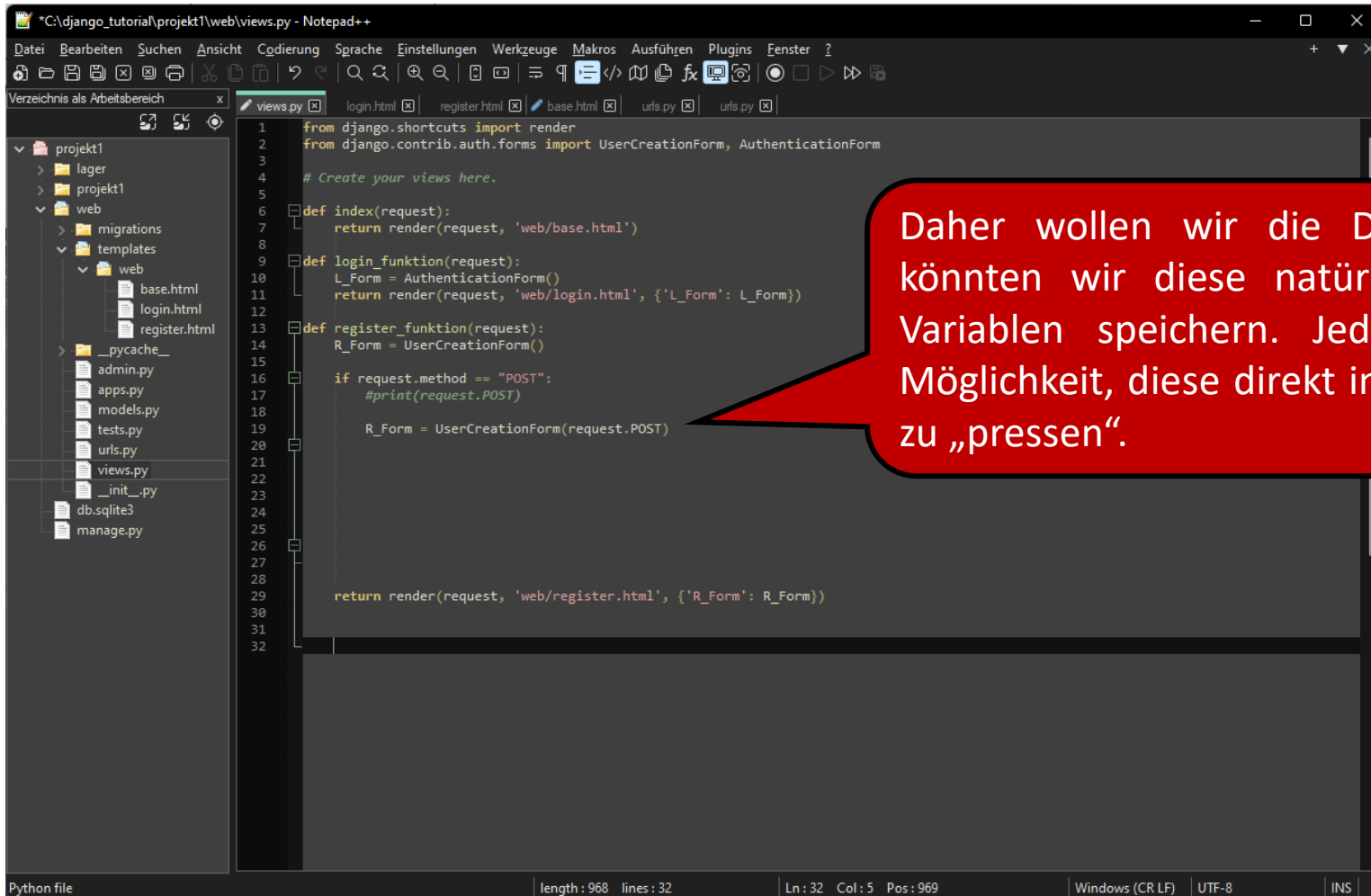
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19
20
21
22
23
24     return render(request, 'web/register.html', {'R_Form': R_Form})
25
26
27
28
29
30
31
32
```

Wir wissen jetzt, dass es sich innerhalb des if Körpers um einen POST handelt...

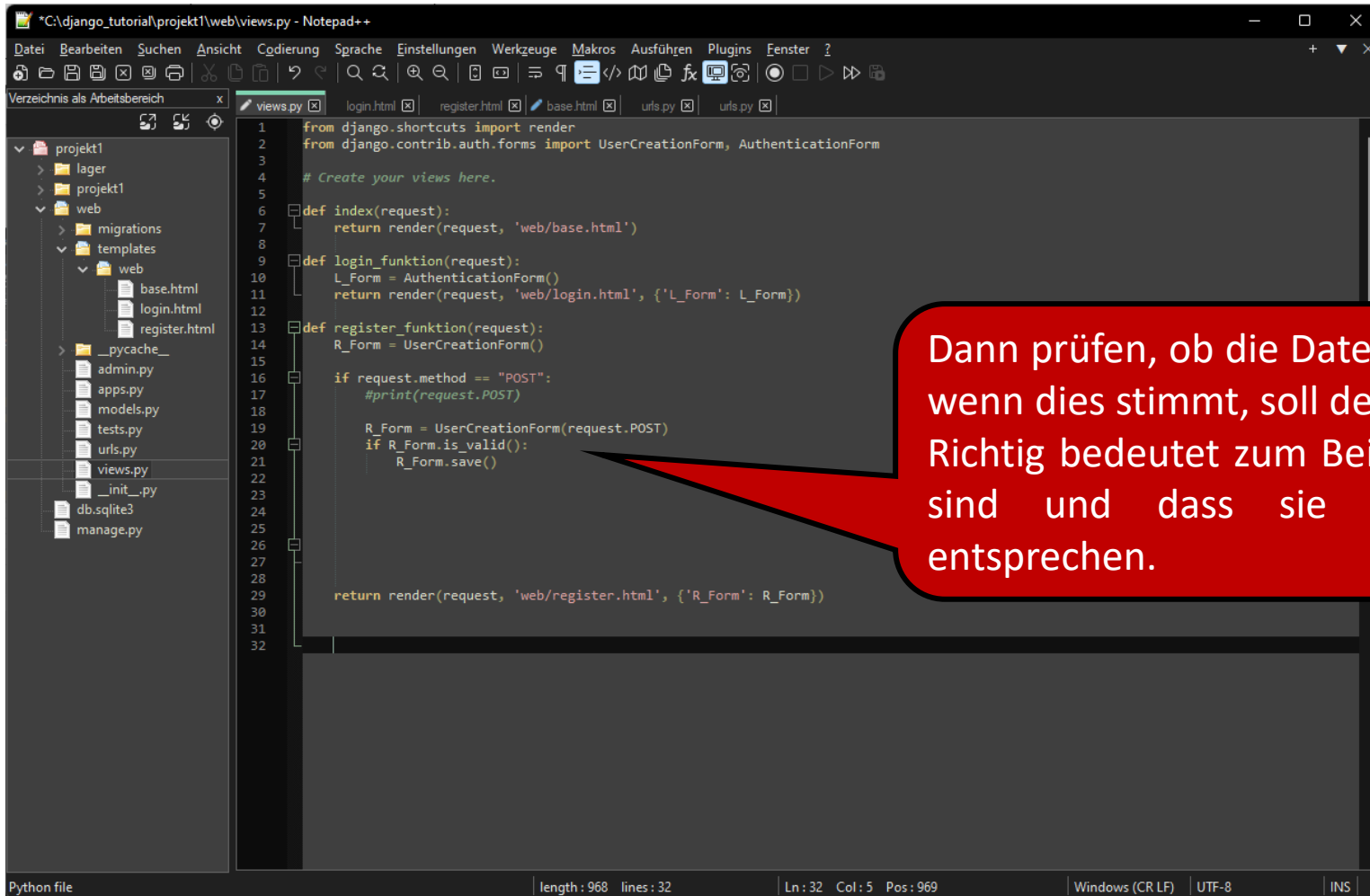
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19         R_Form = UserCreationForm(request.POST)
20
21
22
23
24
25
26
27
28
29 return render(request, 'web/register.html', {'R_Form': R_Form})
30
31
32
```

Daher wollen wir die Daten entgegennehmen. Nun könnten wir diese natürlich alle einzeln in separate Variablen speichern. Jedoch bietet Django uns die Möglichkeit, diese direkt in die bereits verwendete Form zu „pressen“.

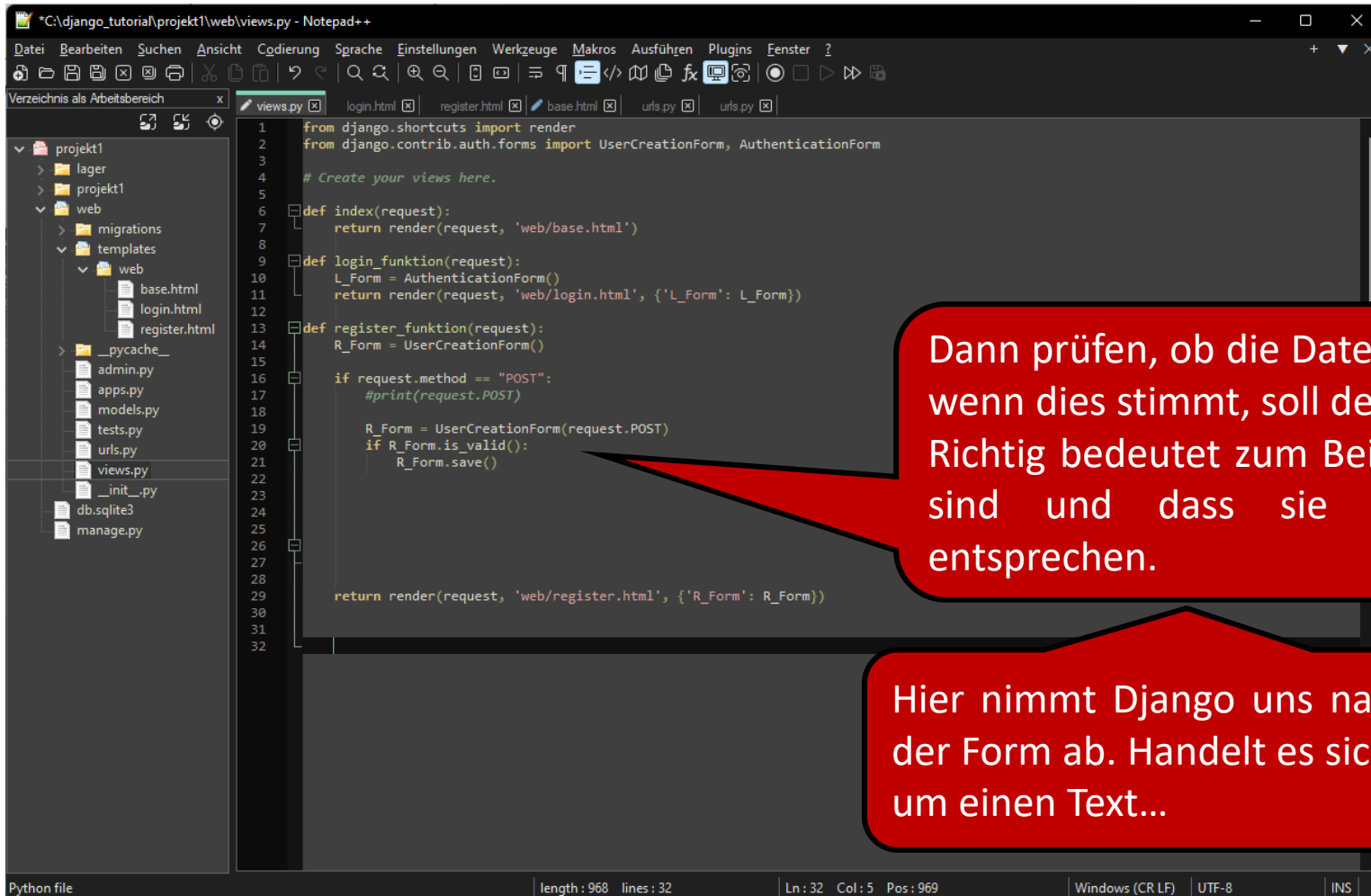
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19         R_Form = UserCreationForm(request.POST)
20         if R_Form.is_valid():
21             R_Form.save()
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Dann prüfen, ob die Daten in der Form „Richtig“ sind und wenn dies stimmt, soll der Eintrag gespeichert werden. Richtig bedeutet zum Beispiel dass die Passwörter gleich sind und dass sie den Sicherheitsanforderungen entsprechen.

Login und Registrierung

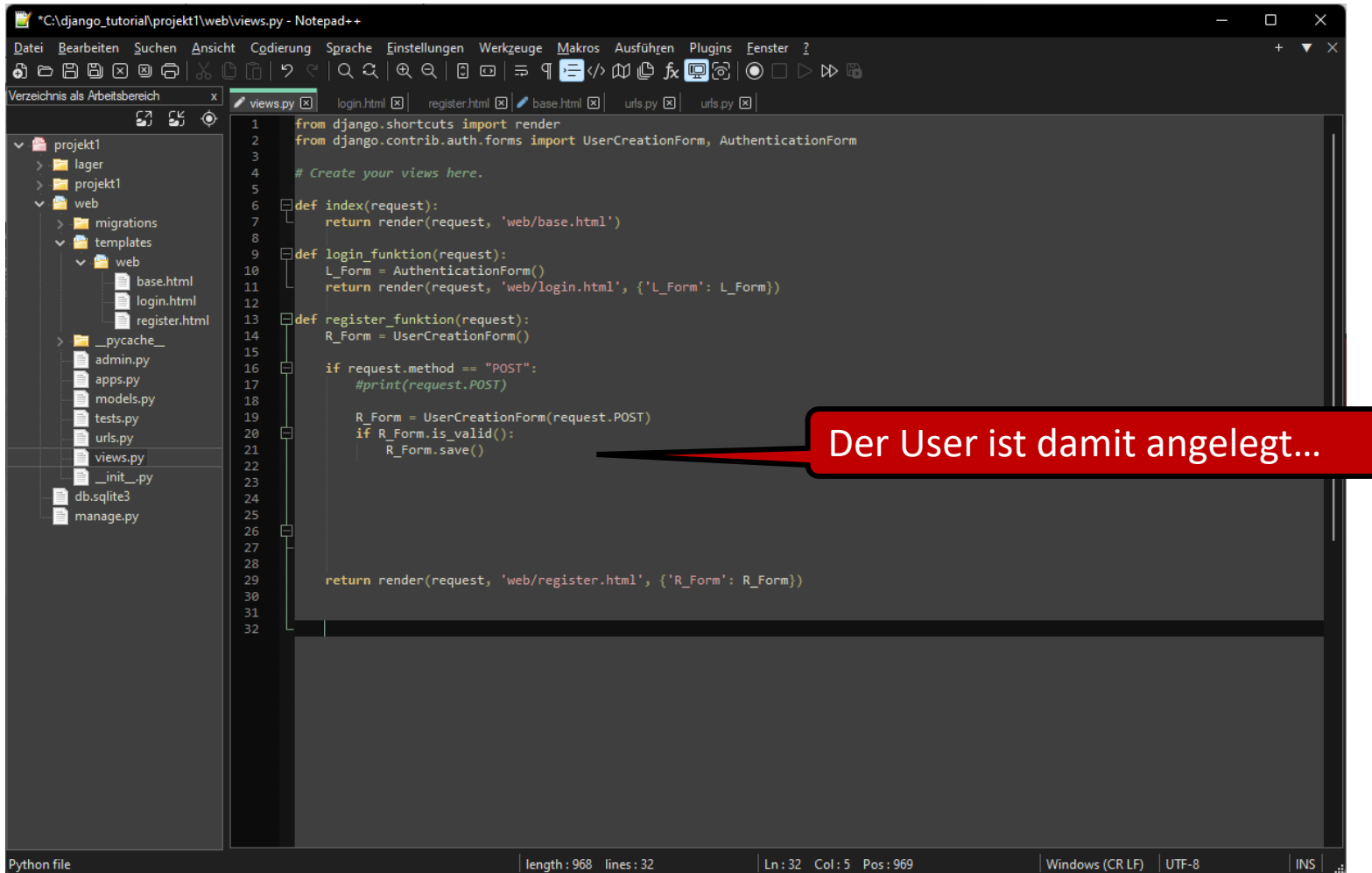


```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19         R_Form = UserCreationForm(request.POST)
20         if R_Form.is_valid():
21             R_Form.save()
22
23     return render(request, 'web/register.html', {'R_Form': R_Form})
24
25
26
27
28
29
30
31
32
```

Dann prüfen, ob die Daten in der Form „Richtig“ sind und wenn dies stimmt, soll der Eintrag gespeichert werden. Richtig bedeutet zum Beispiel dass die Passwörter gleich sind und dass sie den Sicherheitsanforderungen entsprechen.

Hier nimmt Django uns natürlich sämtliche Überprüfung der Form ab. Handelt es sich um eine Zahl, handelt es sich um einen Text...

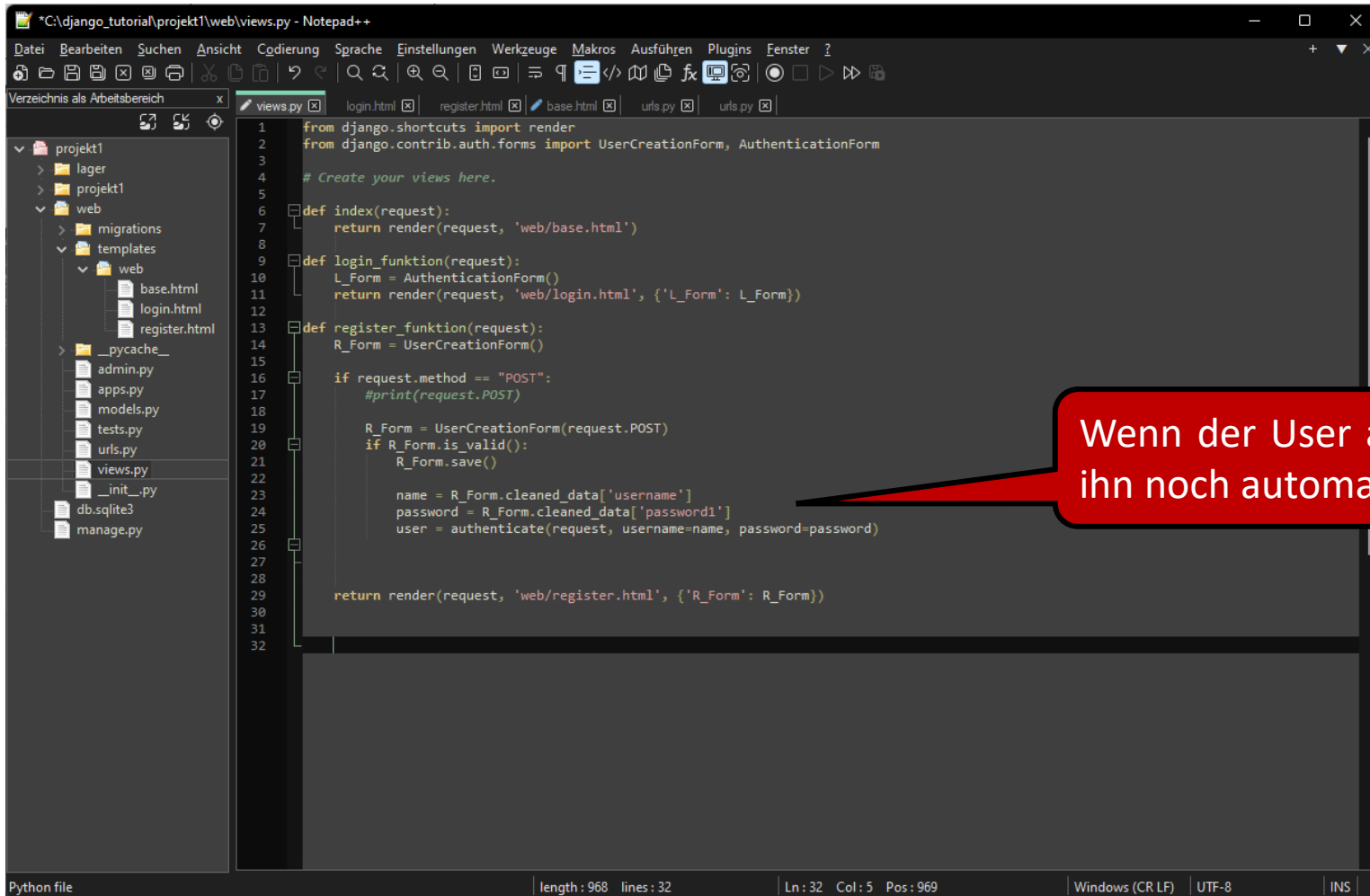
Login und Registrierung



The image shows a Notepad++ window editing a Django project's views.py file. The file path is *C:\django_tutorial\projekt1\web\views.py. The code defines three functions: index, login_funktion, and register_funktion. The register_funktion function handles POST requests and saves a new user if the form is valid. A red callout bubble points to the R_Form.save() line with the text "Der User ist damit angelegt...".

```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19         R_Form = UserCreationForm(request.POST)
20         if R_Form.is_valid():
21             R_Form.save()
22
23
24
25
26
27
28
29 return render(request, 'web/register.html', {'R_Form': R_Form})
30
31
32
```

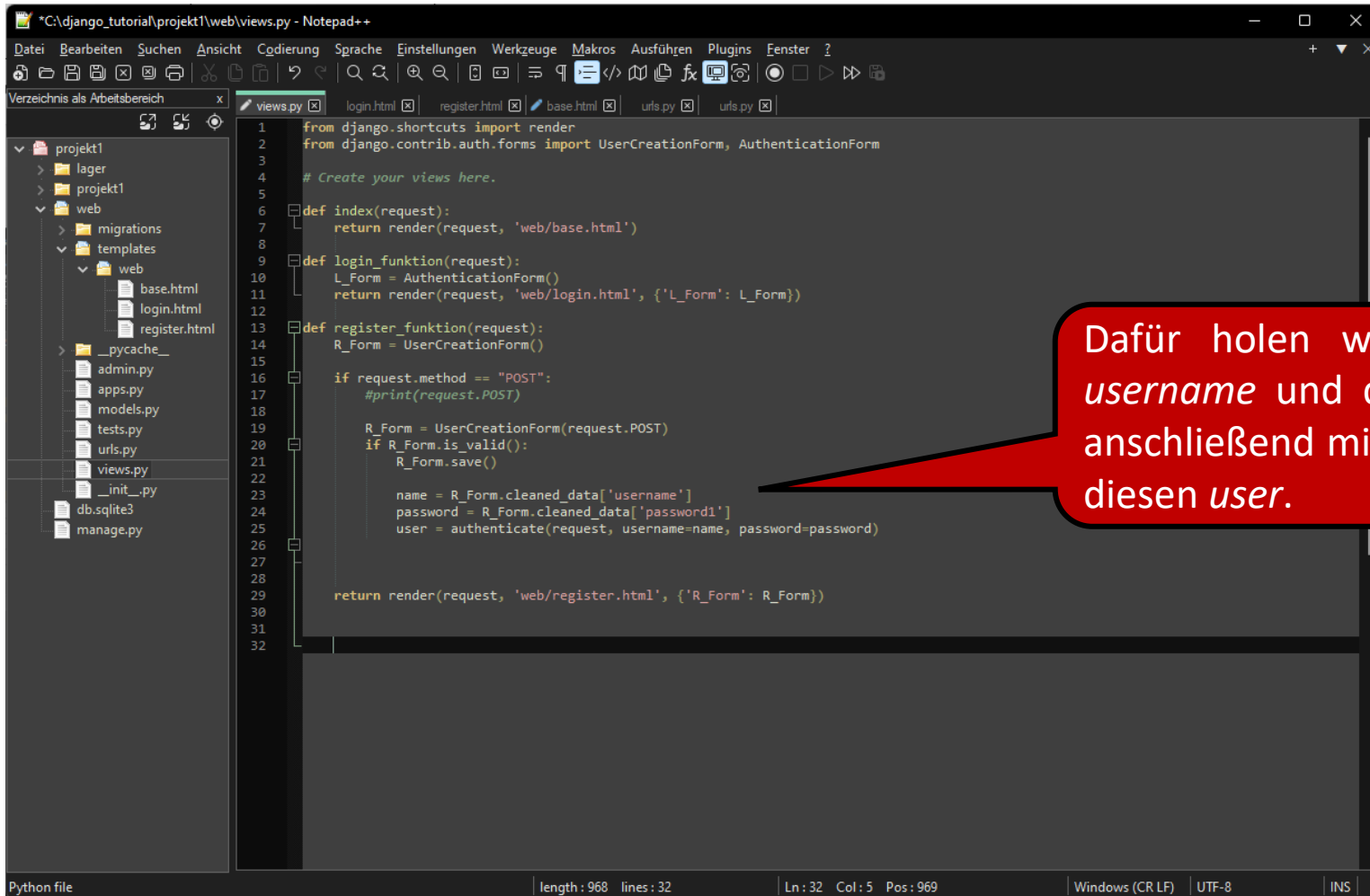
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19         R_Form = UserCreationForm(request.POST)
20         if R_Form.is_valid():
21             R_Form.save()
22
23             name = R_Form.cleaned_data['username']
24             password = R_Form.cleaned_data['password1']
25             user = authenticate(request, username=name, password=password)
26
27
28     return render(request, 'web/register.html', {'R_Form': R_Form})
29
30
31
32
```

Wenn der User angelegt ist, wollen wir ihn noch automatisch einloggen.

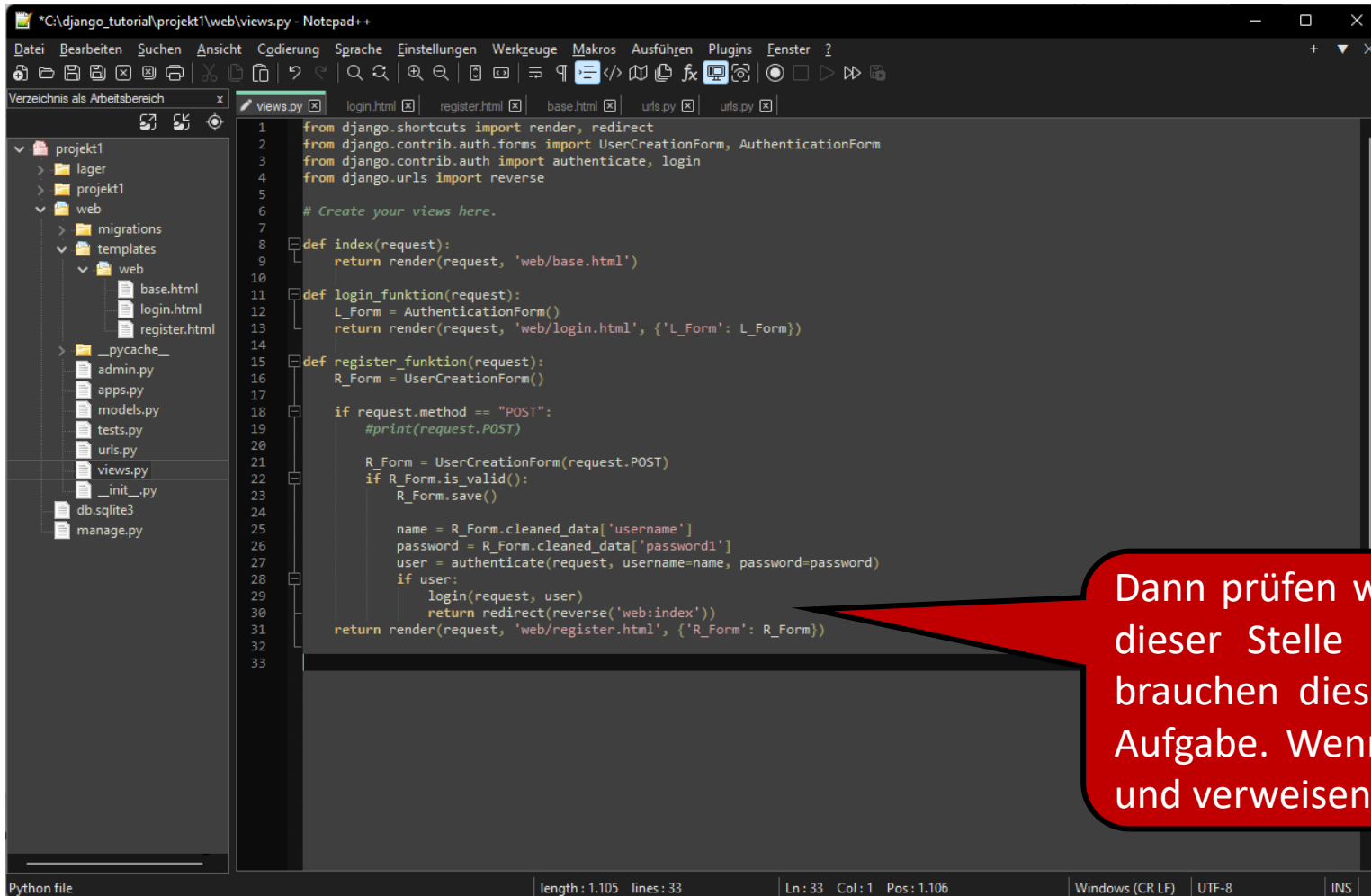
Login und Registrierung



```
1 from django.shortcuts import render
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3
4 # Create your views here.
5
6 def index(request):
7     return render(request, 'web/base.html')
8
9 def login_funktion(request):
10     L_Form = AuthenticationForm()
11     return render(request, 'web/login.html', {'L_Form': L_Form})
12
13 def register_funktion(request):
14     R_Form = UserCreationForm()
15
16     if request.method == "POST":
17         #print(request.POST)
18
19         R_Form = UserCreationForm(request.POST)
20         if R_Form.is_valid():
21             R_Form.save()
22
23             name = R_Form.cleaned_data['username']
24             password = R_Form.cleaned_data['password1']
25             user = authenticate(request, username=name, password=password)
26
27
28
29     return render(request, 'web/register.html', {'R_Form': R_Form})
30
31
32
```

Dafür holen wir uns aus der Form den *username* und das *password* und holen uns anschließend mit der *authentication* Funktion diesen *user*.

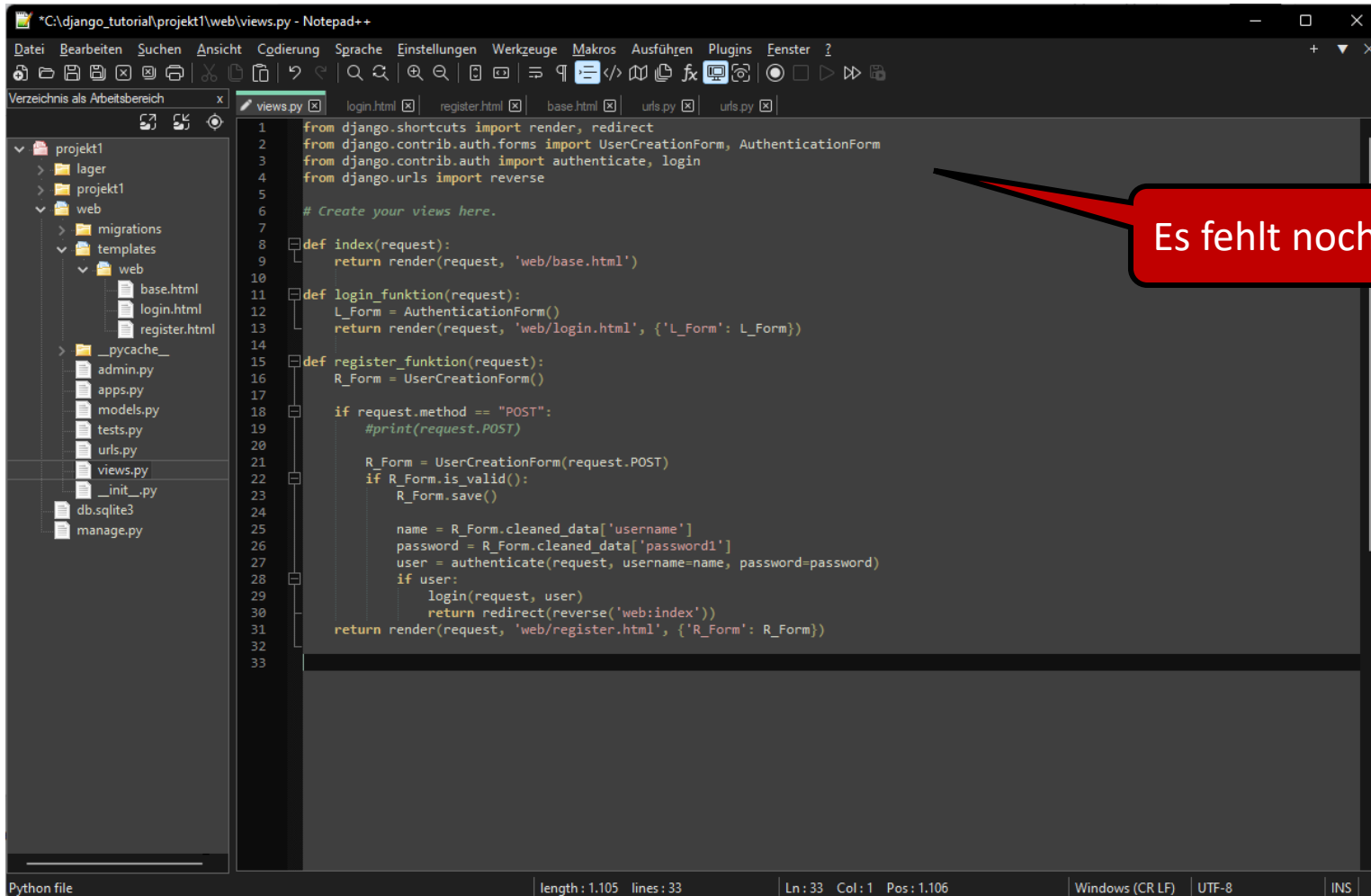
Login und Registrierung



```
1 from django.shortcuts import render, redirect
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3 from django.contrib.auth import authenticate, login
4 from django.urls import reverse
5
6 # Create your views here.
7
8 def index(request):
9     return render(request, 'web/base.html')
10
11 def login_funktion(request):
12     L_Form = AuthenticationForm()
13     return render(request, 'web/login.html', {'L_Form': L_Form})
14
15 def register_funktion(request):
16     R_Form = UserCreationForm()
17
18     if request.method == "POST":
19         #print(request.POST)
20
21         R_Form = UserCreationForm(request.POST)
22         if R_Form.is_valid():
23             R_Form.save()
24
25             name = R_Form.cleaned_data['username']
26             password = R_Form.cleaned_data['password1']
27             user = authenticate(request, username=name, password=password)
28             if user:
29                 login(request, user)
30                 return redirect(reverse('web:index'))
31             return render(request, 'web/register.html', {'R_Form': R_Form})
32
33
```

Dann prüfen wir ob dieser *user* existiert, ist an dieser Stelle eigentlich überflüssig, aber wir brauchen diese Funktion gleich noch für eine Aufgabe. Wenn er existiert, loggen wir ihn ein und verweisen ihn zur base.html.

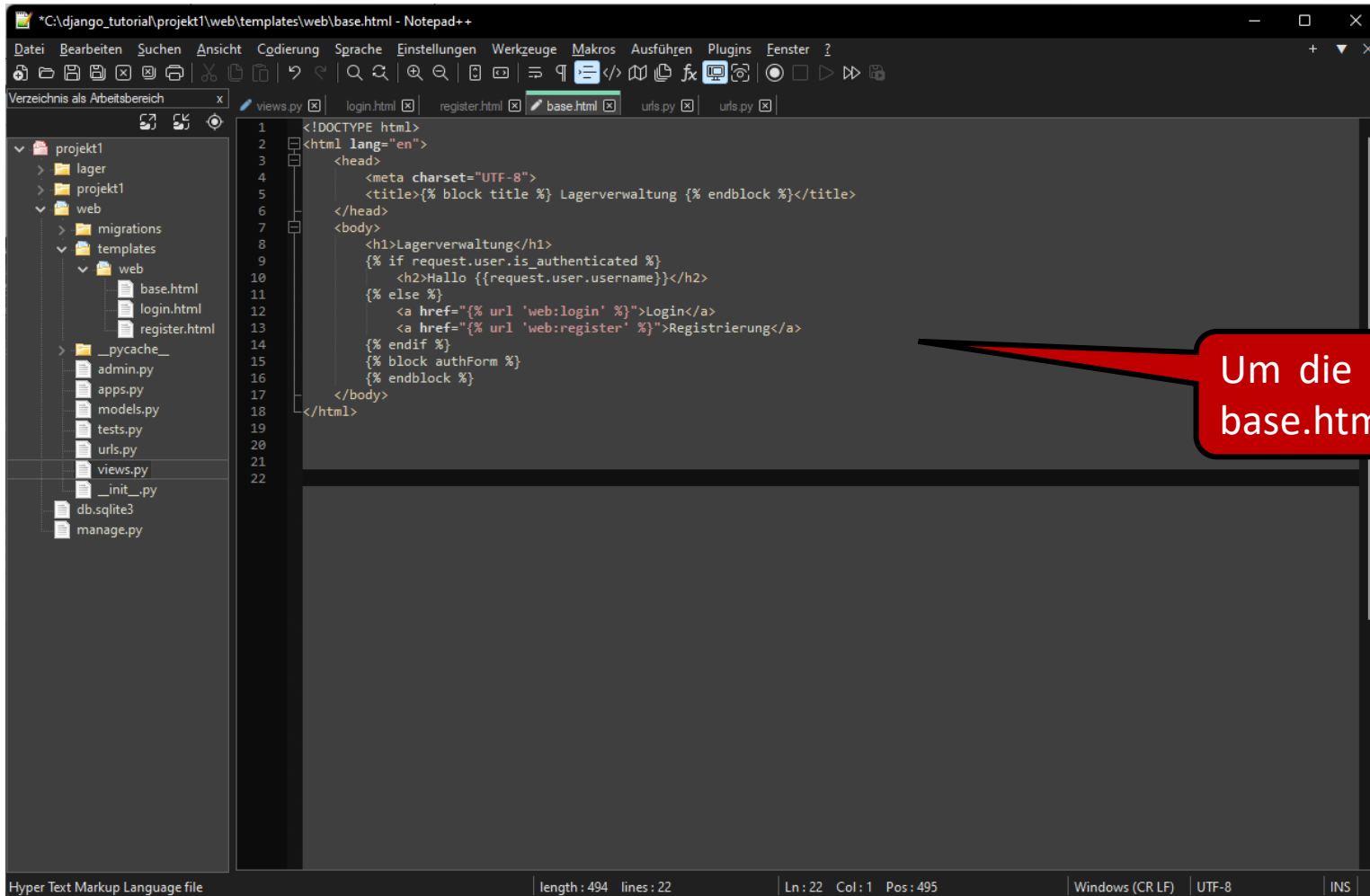
Login und Registrierung



```
1 from django.shortcuts import render, redirect
2 from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
3 from django.contrib.auth import authenticate, login
4 from django.urls import reverse
5
6 # Create your views here.
7
8 def index(request):
9     return render(request, 'web/base.html')
10
11 def login_funktion(request):
12     L_Form = AuthenticationForm()
13     return render(request, 'web/login.html', {'L_Form': L_Form})
14
15 def register_funktion(request):
16     R_Form = UserCreationForm()
17
18     if request.method == "POST":
19         #print(request.POST)
20
21         R_Form = UserCreationForm(request.POST)
22         if R_Form.is_valid():
23             R_Form.save()
24
25             name = R_Form.cleaned_data['username']
26             password = R_Form.cleaned_data['password1']
27             user = authenticate(request, username=name, password=password)
28             if user:
29                 login(request, user)
30                 return redirect(reverse('web:index'))
31     return render(request, 'web/register.html', {'R_Form': R_Form})
32
33
```

Es fehlt noch ein paar *importe* der Funktionen.

Login und Registrierung



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8 <h1>Lagerverwaltung</h1>
9 {% if request.user.is_authenticated %}
10 <h2>Hallo {{request.user.username}}</h2>
11 {% else %}
12 <a href="{% url 'web:login' %}">Login</a>
13 <a href="{% url 'web:register' %}">Registrierung</a>
14 {% endif %}
15 {% block authForm %}
16 {% endblock %}
17 </body>
18 </html>
19
20
21
22
```

Um die Registrierung zu testen, wollen wir die base.html etwas anpassen.

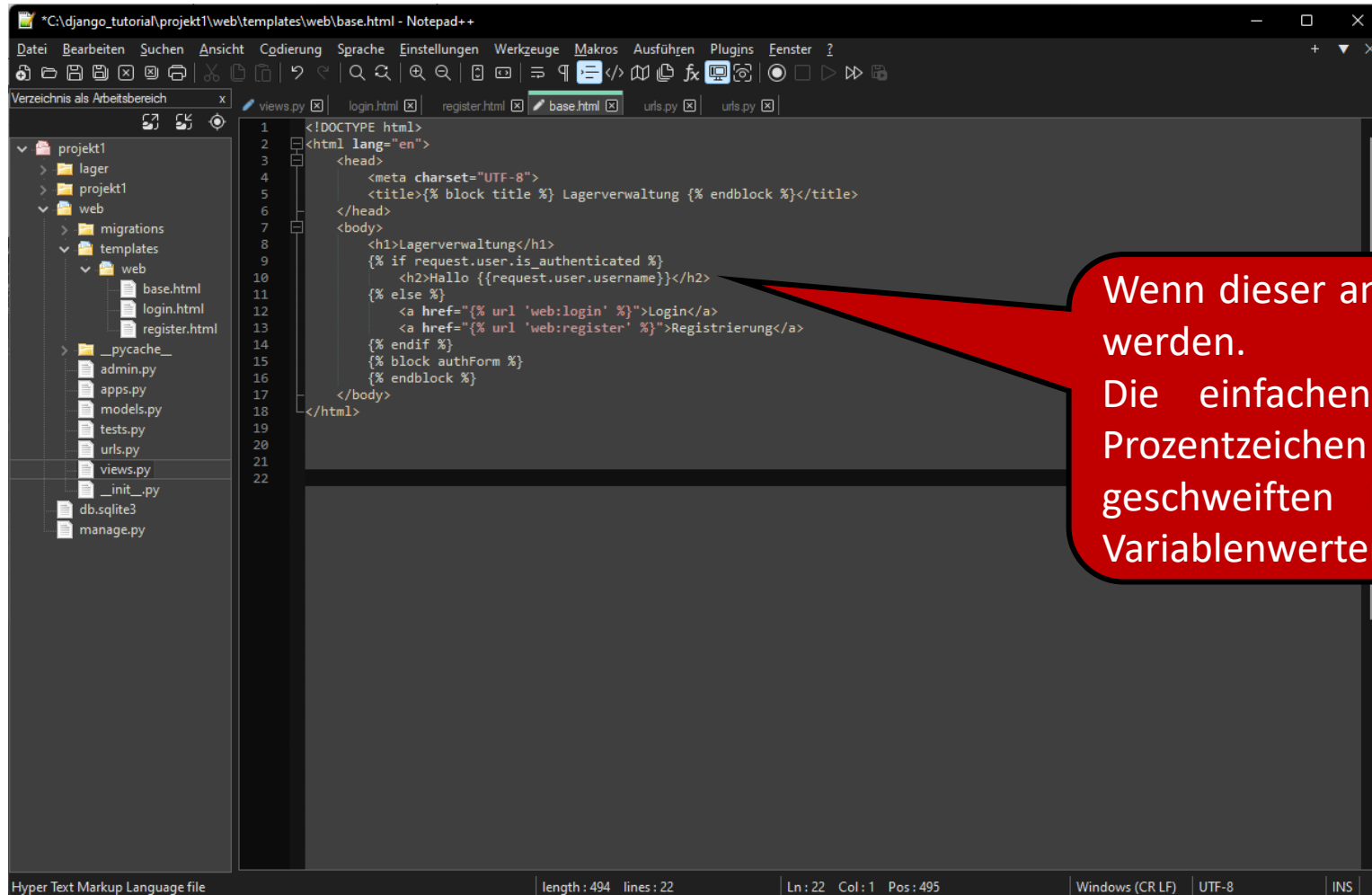
Login und Registrierung

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8   <h1>Lagerverwaltung</h1>
9   {% if request.user.is_authenticated %}
10    <h2>Hallo {{request.user.username}}</h2>
11   {% else %}
12    <a href="{% url 'web:login' %}">Login</a>
13    <a href="{% url 'web:register' %}">Registrierung</a>
14   {% endif %}
15   {% block authForm %}
16   {% endblock %}
17 </body>
18 </html>
19
20
21
22
```

Hyper Text Markup Language file | length : 494 | lines : 22 | Ln : 22 Col : 1 Pos : 495 | Windows (CR LF) | UTF-8 | INS

Wir verwenden einen Django-Tag der uns eine *if*-Verzweigung erlaubt. In dem Fragen wir ab, ob der *user* angemeldet ist.

Login und Registrierung

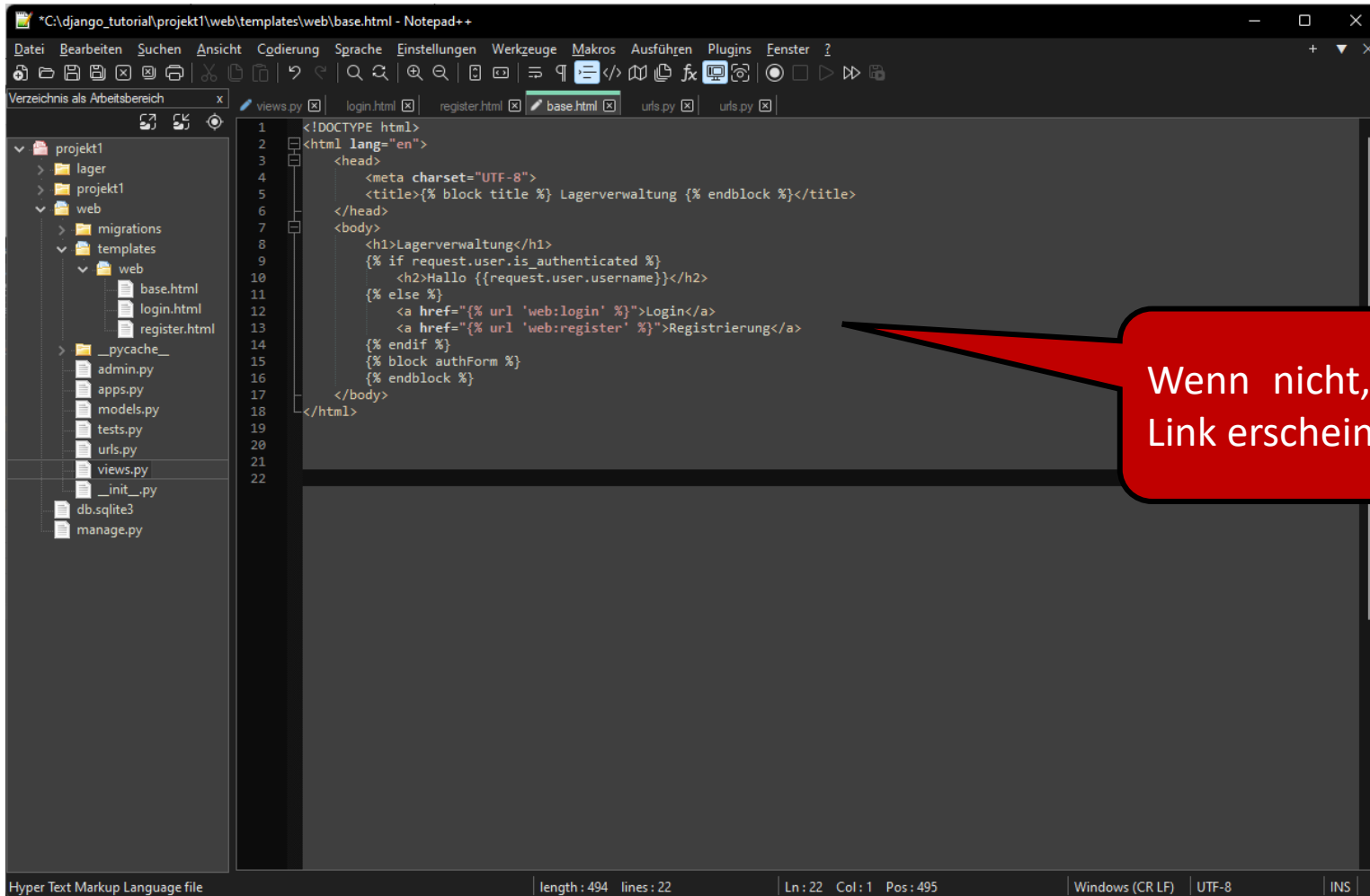


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8 <h1>Lagerverwaltung</h1>
9 {% if request.user.is_authenticated %}
10 <h2>Hallo {{request.user.username}}</h2>
11 {% else %}
12 <a href="{% url 'web:login' %}">Login</a>
13 <a href="{% url 'web:register' %}">Registrierung</a>
14 {% endif %}
15 {% block authForm %}
16 {% endblock %}
17 </body>
18 </html>
19
20
21
22
```

Wenn dieser angemeldet ist, soll sein Name ausgegeben werden.

Die einfachen geschweiften Klammern mit einem Prozentzeichen {% %} sind für Befehle und die doppelten geschweiften Klammern werden zum Rendern von Variablenwerten verwendet.

Login und Registrierung

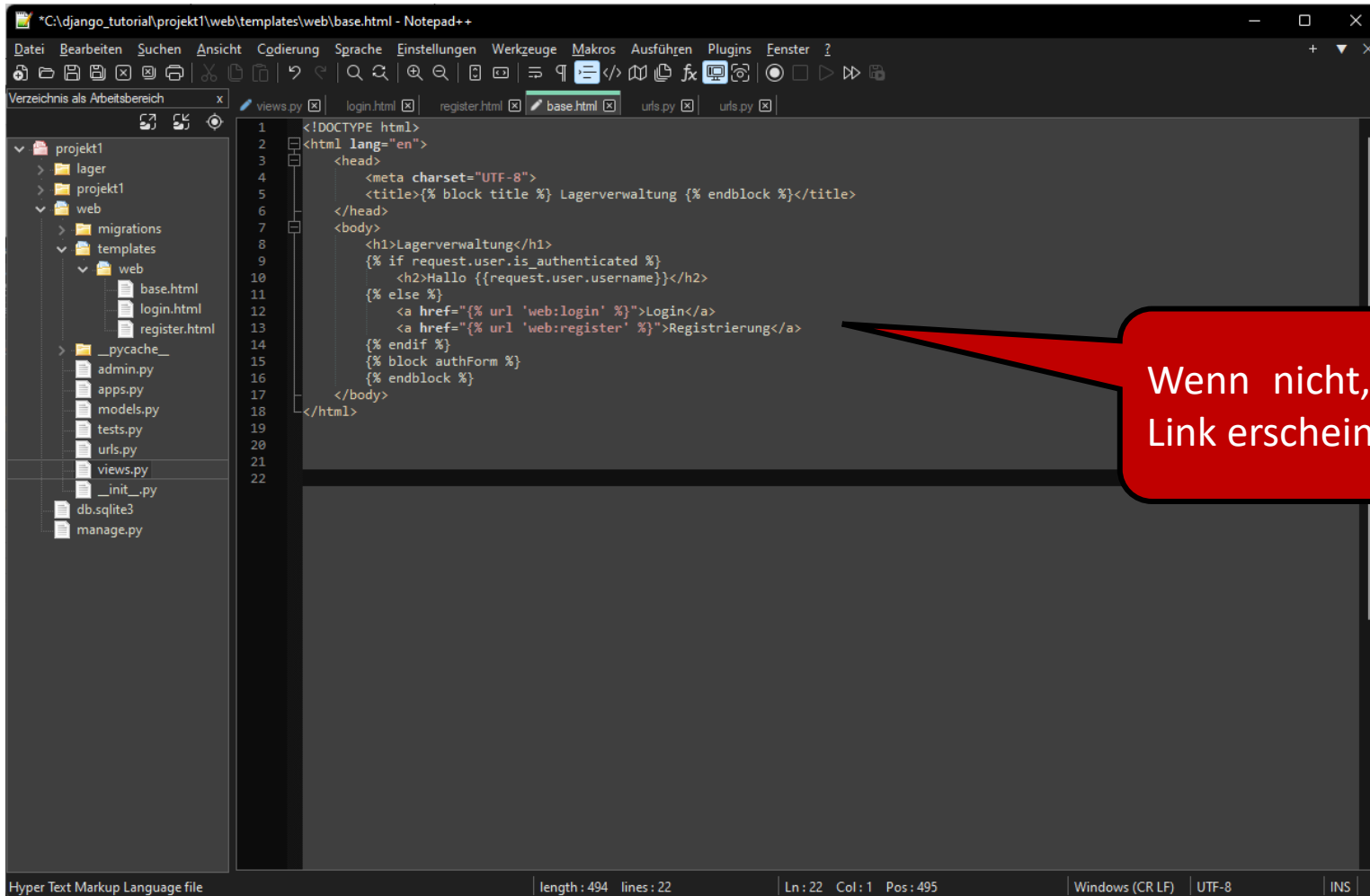


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8 <h1>Lagerverwaltung</h1>
9 {% if request.user.is_authenticated %}
10 <h2>Hallo {{request.user.username}}</h2>
11 {% else %}
12 <a href="{% url 'web:login' %}">Login</a>
13 <a href="{% url 'web:register' %}">Registrierung</a>
14 {% endif %}
15 {% block authForm %}
16 {% endblock %}
17 </body>
18 </html>
19
20
21
22
```

Hyper Text Markup Language file | length : 494 lines : 22 | Ln : 22 Col : 1 Pos : 495 | Windows (CR LF) | UTF-8 | INS

Wenn nicht, soll der Login und Registrierung Link erscheinen.

Login und Registrierung



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>{% block title %} Lagerverwaltung {% endblock %}</title>
6 </head>
7 <body>
8 <h1>Lagerverwaltung</h1>
9 {% if request.user.is_authenticated %}
10 <h2>Hallo {{request.user.username}}</h2>
11 {% else %}
12 <a href="{% url 'web:login' %}">Login</a>
13 <a href="{% url 'web:register' %}">Registrierung</a>
14 {% endif %}
15 {% block authForm %}
16 {% endblock %}
17 </body>
18 </html>
19
20
21
22
```

Hyper Text Markup Language file | length : 494 | lines : 22 | Ln : 22 Col : 1 Pos : 495 | Windows (CR LF) | UTF-8 | INS

Wenn nicht, soll der Login und Registrierung Link erscheinen.

Cool oder?

Login und Registrierung

```
Eingabeaufforderung

(tutorial-env) C:\django_tutorial\projekt1>python manage.py makemigrations
No changes detected

(tutorial-env) C:\django_tutorial\projekt1>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

(tutorial-env) C:\django_tutorial\projekt1>
```

Wo speichern wir jetzt den User, wir brauchen noch eine Datenbank.

Dies kann Django für uns übernehmen, ohne dass wir auch nur eine Zeile SQL programmieren müssen.

Wir stoppen erstmal den Server (strg+c)

Login und Registrierung

```
Eingabeaufforderung

(tutorial-env) C:\django_tutorial\projekt1>python manage.py makemigrations
No changes detected

(tutorial-env) C:\django_tutorial\projekt1>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

(tutorial-env) C:\django_tutorial\projekt1>
```

Als Erstes führen wir den folgenden Befehl aus:
python manage.py makemigrations
Dieser erstellt nun eine „Liste“ von allen Änderungen, die wir im Code vorgenommen haben und der Datenbank angehören. Da wir selber noch keine Modelle geschrieben haben, steht hier, nicht verwunderlich *No changes detected*.

Login und Registrierung

```
Eingabeaufforderung

(tutorial-env) C:\django_tutorial\projekt1>python manage.py makemigrations
No changes detected

(tutorial-env) C:\django_tutorial\projekt1>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

(tutorial-env) C:\django_tutorial\projekt1>
```

Anschließend führen wir den Befehl *migrate* aus:
python manage.py migrate
Dieser Befehl verändert jetzt die Datenbank. Zwar haben mit *makemigrations* keine Änderungen gefunden, jedoch existiert noch keine Datenbank für das Projekt und Django legt nun erstmal alle Standard Tabellen an, wie den User...

Login und Registrierung

```
Eingabeaufforderung - python manage.py runserver

No changes detected

(tutorial-env) C:\django_tutorial\projekt1>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

(tutorial-env) C:\django_tutorial\projekt1>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 21, 2022 - 22:51:07
Django version 4.1.1, using settings 'projekt1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Dann starten wir den Server wieder und testen unsere Registrierung.

Login und Registrierung

Lagerverwaltung - Registrierung

127.0.0.1:8000/register/

Meine Bibliothek |... KI-(Künstliche Intelli... Software Entwicklu...

Lagerverwaltung

[Login-Registrierung](#)

Nutzername: Erforderlich. 150 Zeichen oder weniger. Nur Buchstaben, Ziffern und @/./+/-/_.

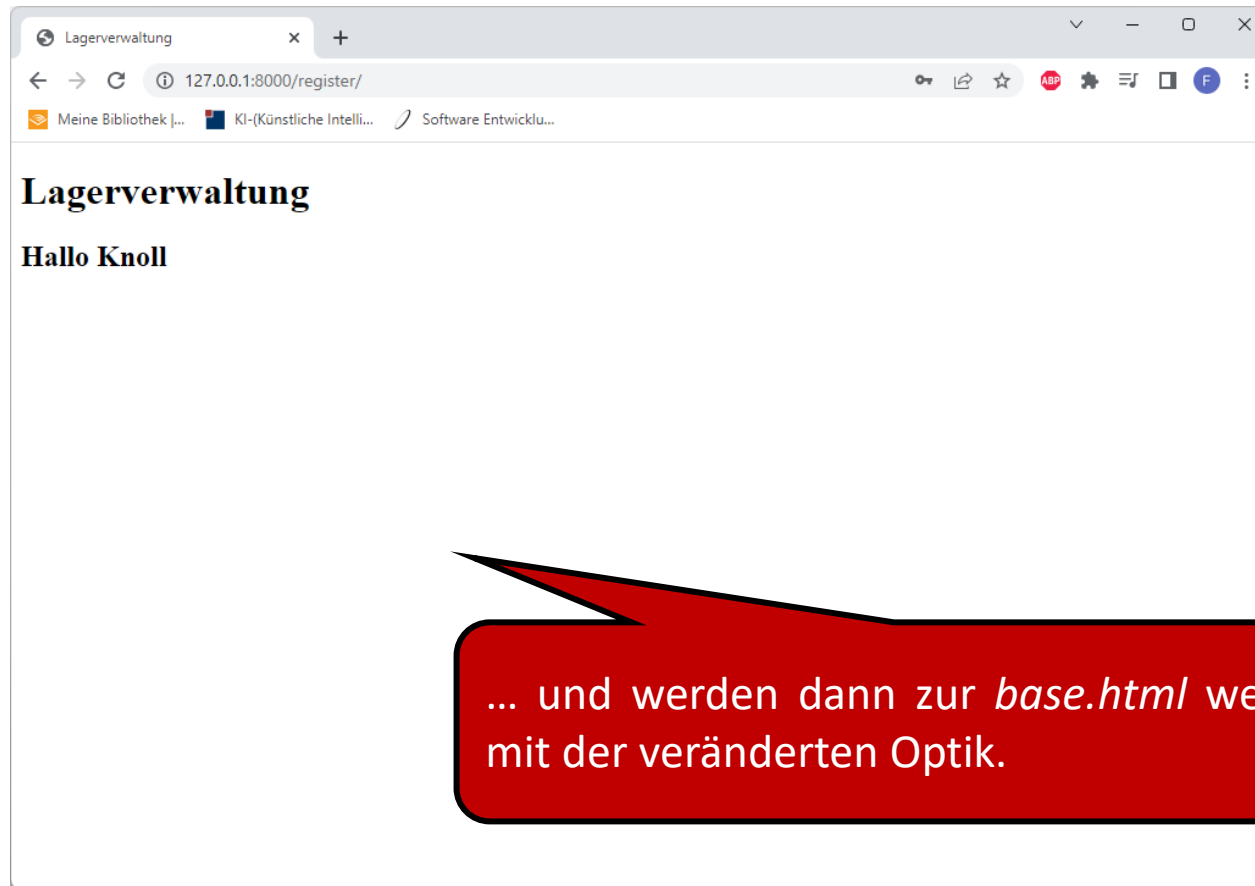
Passwort:

- Ihr Passwort darf Ihren anderen persönlichen Daten nicht zu ähnlich sein.
- Ihr Passwort muss mindestens 8 Zeichen enthalten.
- Ihr Passwort darf kein häufig verwendetes Passwort sein.
- Ihr Passwort darf nicht vollständig numerisch sein.

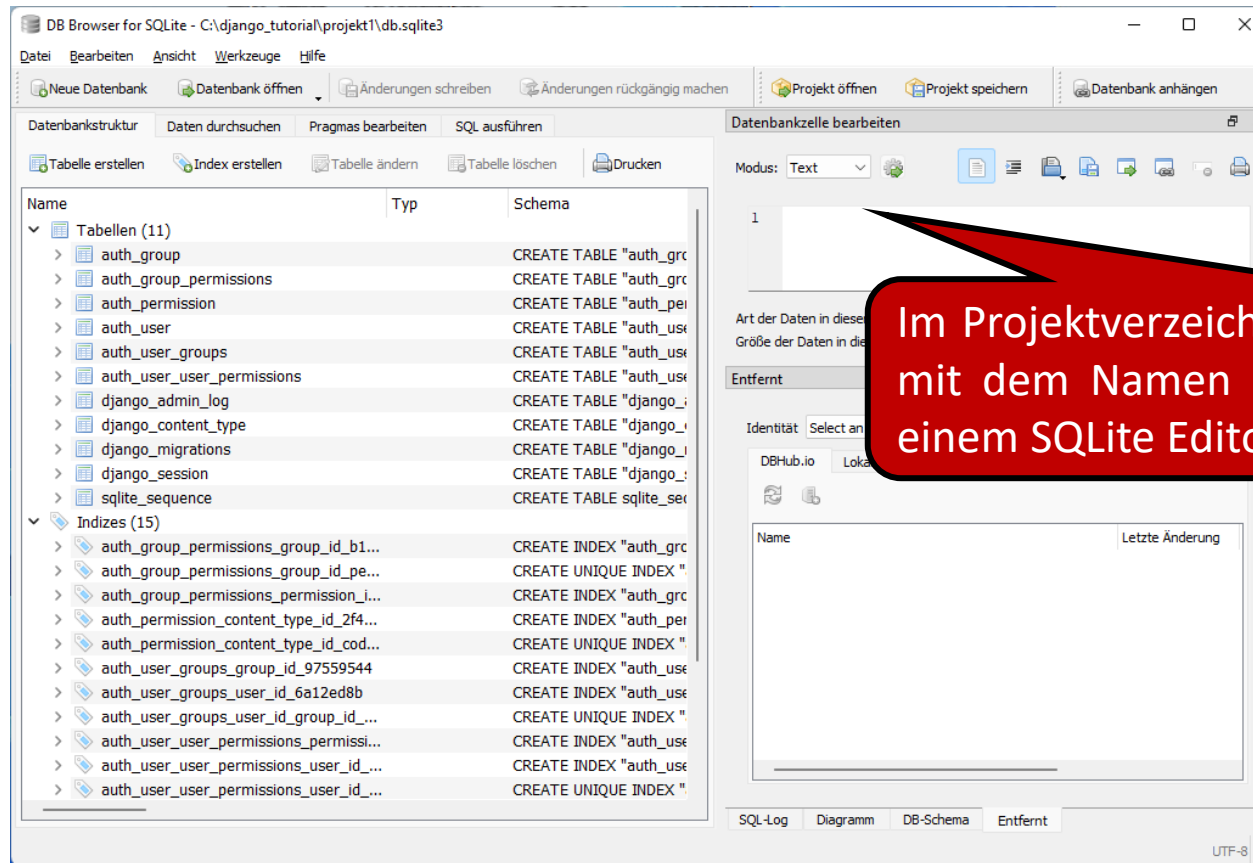
Passwort Bestätigung: Geben Sie zur Bestätigung dasselbe Passwort wie zuvor ein.

Wenn wir es schaffen den Passwortschutz zu überwinden, legen wir einen User an...

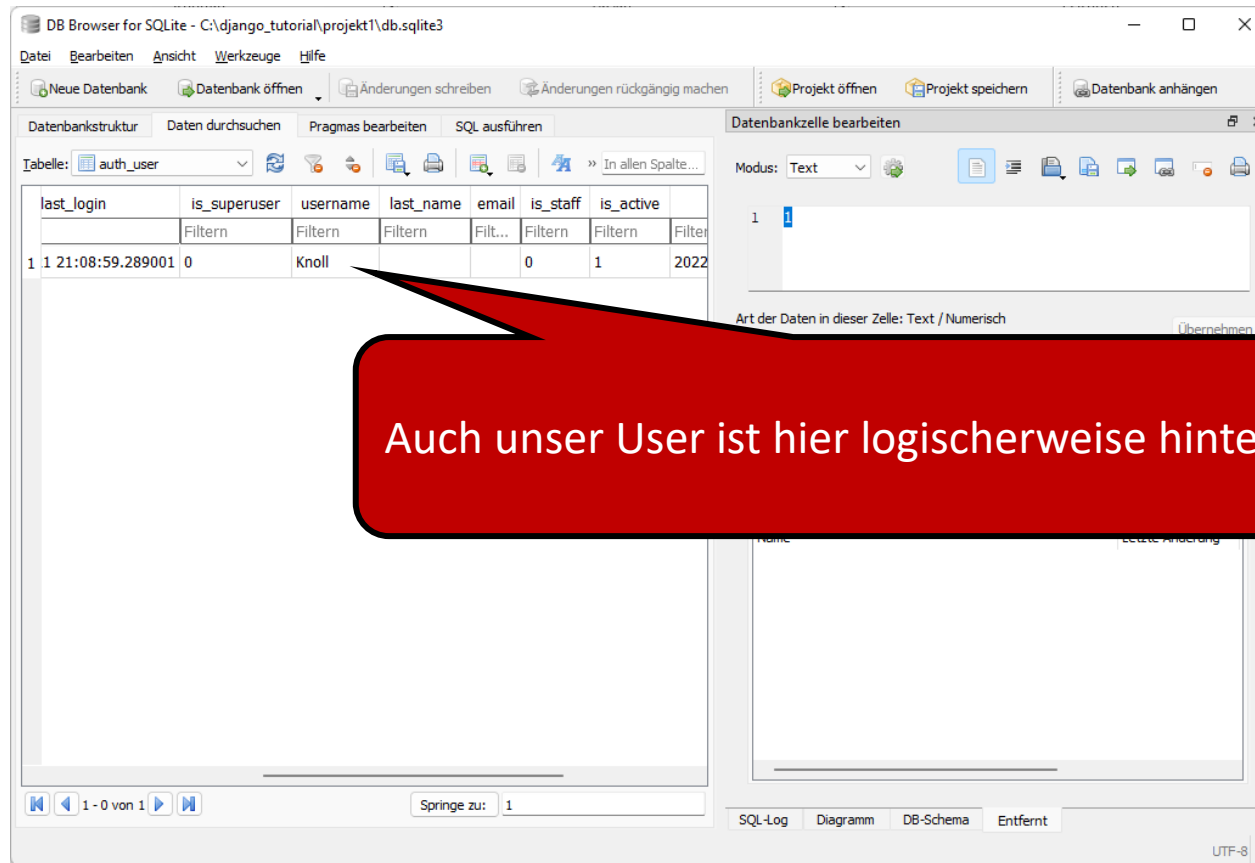
Login und Registrierung



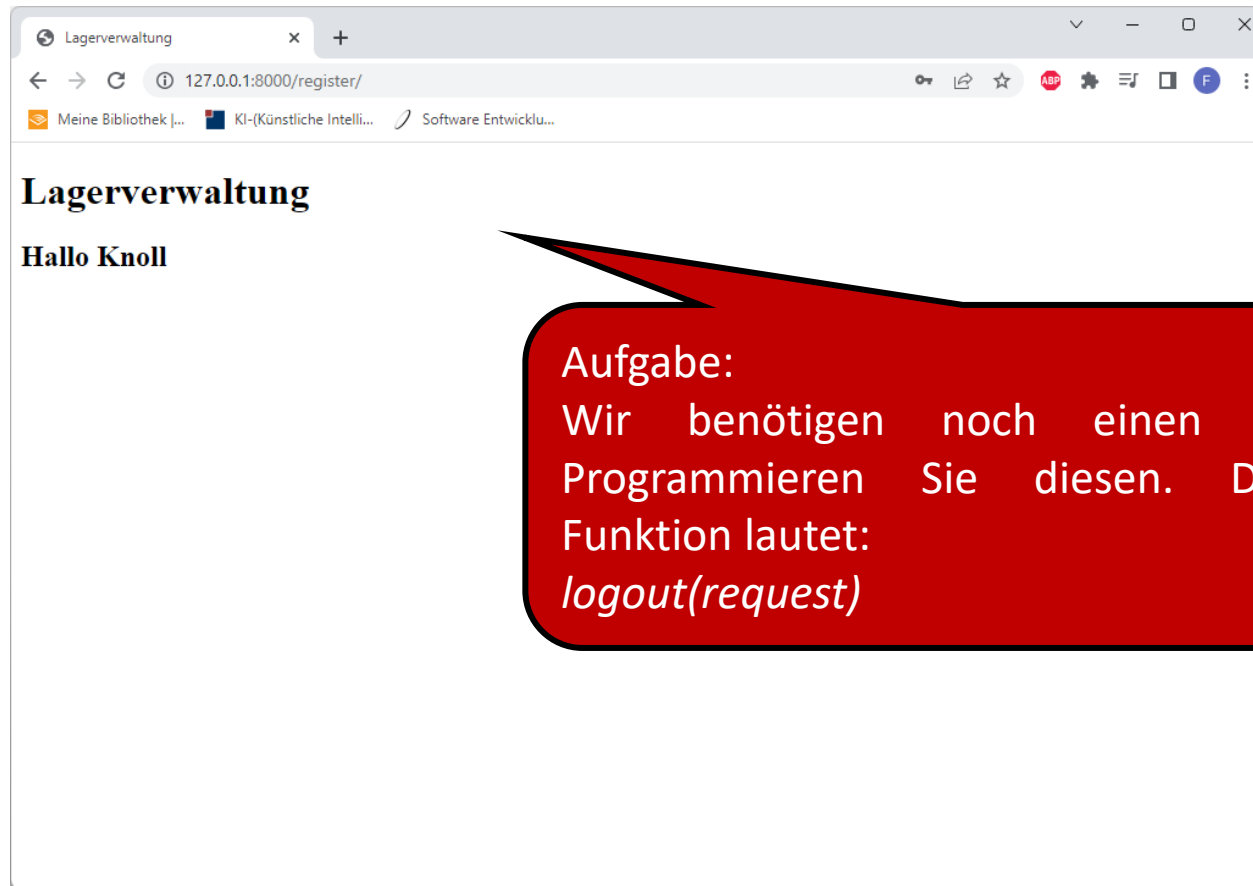
Login und Registrierung



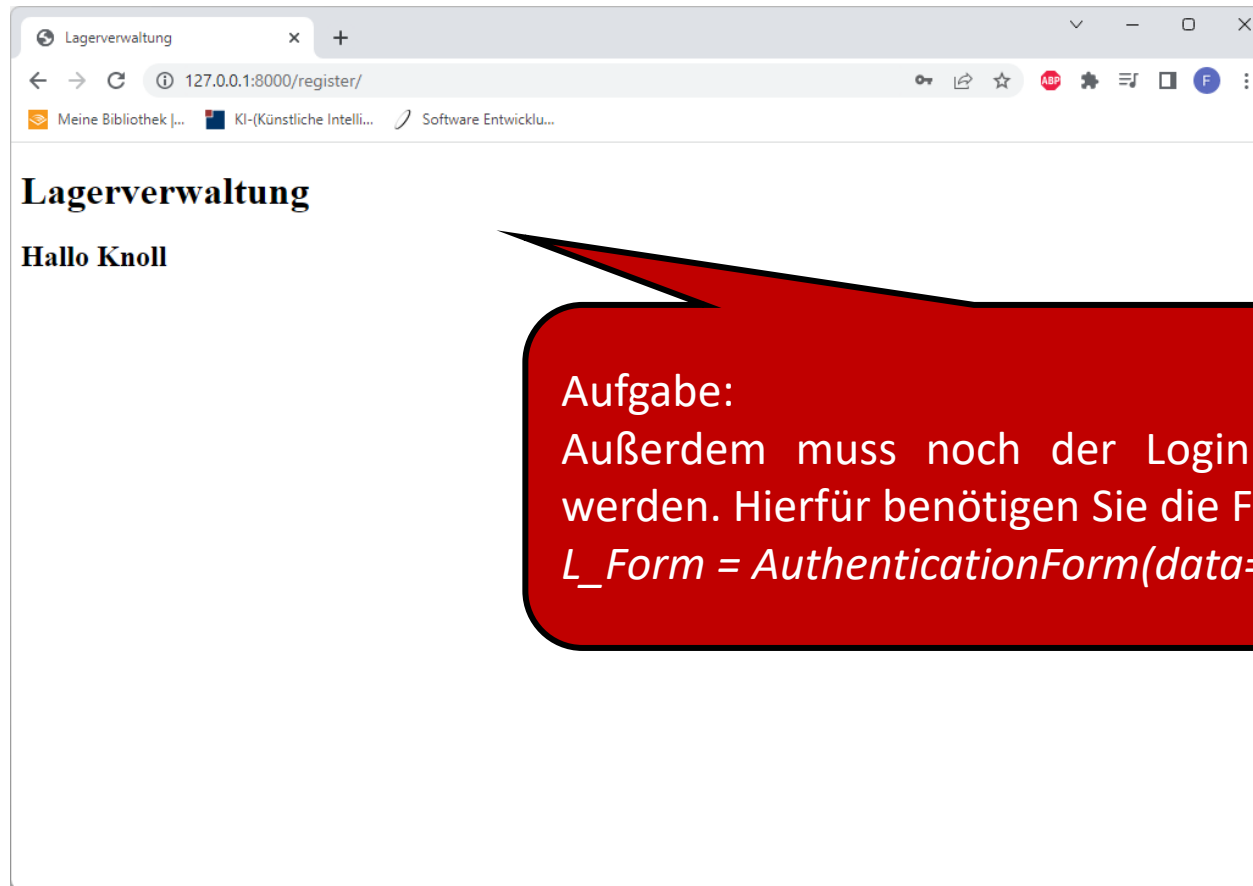
Login und Registrierung



Login und Registrierung



Login und Registrierung



Aufgabe:

Außerdem muss noch der Login vervollständigt werden. Hierfür benötigen Sie die Funktion:

`L_Form = AuthenticationForm(data=request.POST)`