

# Django

TomerBu

# **נושאים להיום:**

מבוא והכרות

דרישות לפרויקט

התקנות ויצירת פרויקט ראשון

מבנה הפרויקט

views

urls

בקשות ותגובה

path segments

עבודה עם מסד נתונים

# מבוא

## Meet Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.



### Ridiculously fast.

Django was designed to help developers take applications from concept to completion as quickly as possible.



### Reassuringly secure.

Django takes security seriously and helps developers avoid many common security mistakes.



### Exceedingly scalable.

Some of the busiest sites on the web leverage Django's ability to quickly and flexibly scale.

# מבוא

## Intro to Django

[EXPAND ALL / COLLAPSE ALL](#)

### Object-relational mapper

נכתוב model class והמערכת תיצור עבורנו את כל פעולות crud  
כולל יחסי גומלין



### URLs and views

views:  
פונקציה שמגיבה לבקשת

מייפוי כתובות urls  
ואיזו פונקציה תגיב להן



### Templates

החזרה של דפי HTML מועשרים ע"י פיתון



### Forms

עבודה עם טפסים (html)  
בצורה מאובטחת (ובקלות)



### Authentication

\*מערכת לניהול/זיהוי/הרשמה

\*אם ניתן לבצע פעולה עברו משתמש מסוים



### Admin

מערכת ניהול האתר:  
עריכה צפיה יצירת משתמשים  
יצירה ועריכה של מידע בסיסי



### Internationalization

יכולת לתרגם את האתר לשפות רבות



### Security

כמויות כניסה למשתמש ליום  
כמויות כניסה לדקה  
כמויות כניסה לתוכן מסוים



# דוקומנטציה:

**django**

The web framework for  
perfectionists with deadlines.

OVERVIEW DOWNLOAD DOCUMENTATION NEWS COMMUNITY CODE ISSUES ABOUT

Documentation

Search 5.1 ↗

## Django documentation ¶

Everything you need to know about Django.

### First steps ¶

Are you new to Django or to programming? This is the place to start!

- **From scratch:** [Overview](#) | [Installation](#)
- **Tutorial:** [Part 1: Requests and responses](#) | [Part 2: Models and the admin site](#) | [Part 3: Views and templates](#) | [Part 4: Forms and generic views](#) | [Part 5: Testing](#) | [Part 6: Static files](#) | [Part 7: Customizing the admin site](#) | [Part 8: Adding third-party packages](#)
- **Advanced Tutorials:** [How to write reusable apps](#) | [Writing your first contribution to Django](#)

<https://docs.djangoproject.com/en/5.1>

# תוספים:

sqlite



**SQLite Viewer**

SQLite Viewer for VS Code

Florian Klampfer



Sqlite

דטה-בייס

מהיר מאוד

モootאמ לפיתוח

django



**Django**

⌚ 12ms

Beautiful syntax and scoped snippets for perfectionists with deadlines

Baptiste Darthenay



**Django Template Support**

⌚ 189ms

⌚ formatter, django template support, highlight, format, prettier, auto-co...

junstyle



מג'ע מותקן עם django

# דרישות לפרויקט:

ראו קובץ מצורף בGITLAB

# התקנות:

התקנה של ספריה באופן גלובלי:

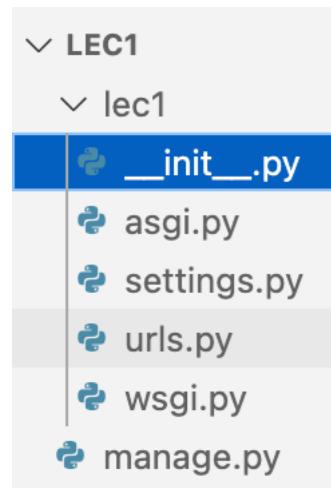
```
pip install django
```

(חדר פימי למחשב)

# יצירת פרויקט חדש:

```
django-admin startproject lec1
```

ישור עבורנו תיקיה בשם lec1



# מבנה הפרויקט:

הקובץ `__init__.py` הוא קובץ שמסמן את התקינה כמודול בפייתון.  
כל הקבצים בתיקיה שייכים למודול.

בעזרת הקובץ זהה אנו יכולים ליצא פונקציונליות ולהריץ קוד כשהמודול מיובא

.....

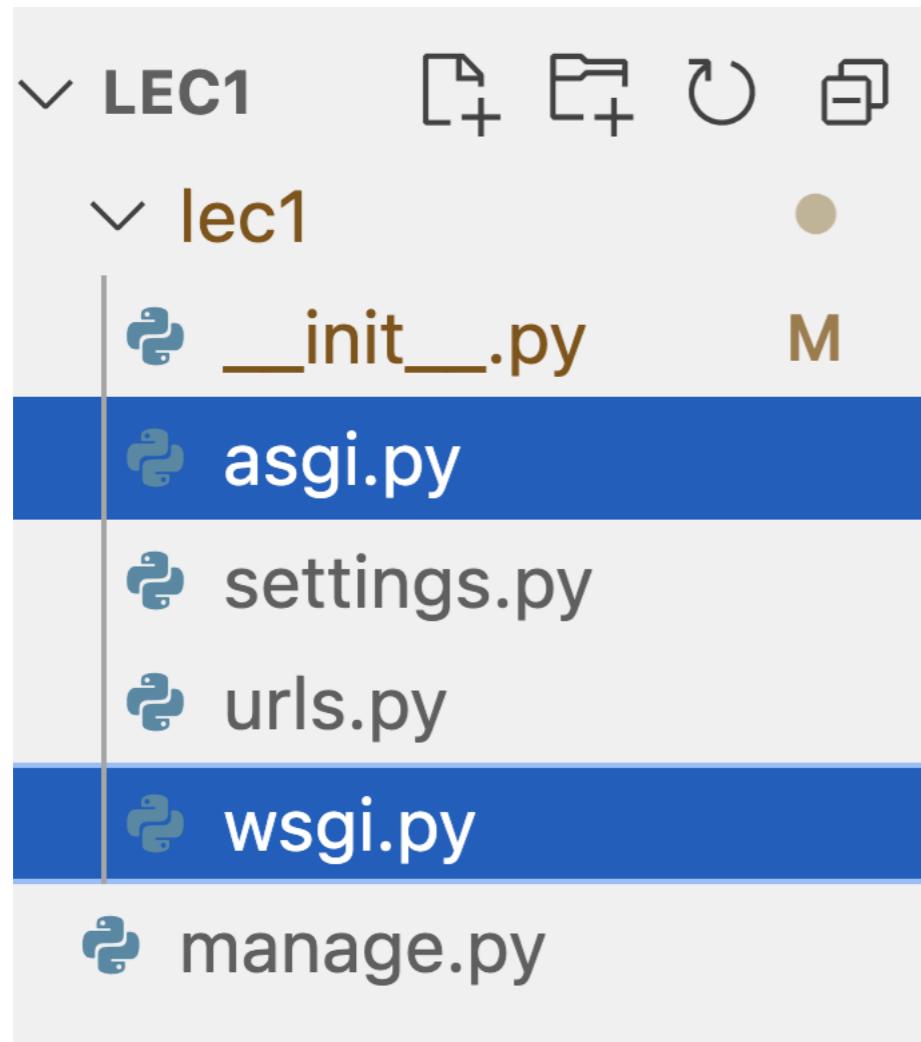
This file indicates that the directory is a Python module.

It can also be used to initialize package-level variables or import specific classes/functions  
to make them available at the package level.

.....

```
# You can add package-level imports here if needed
# from a import b
# than you can use b inside the package
# and import b from lec1 like so: from lec1 import b
```

# מבנה הפרויקט:



asgi.py = async server gateway interface  
wsgi.py = web server gateway interface

קובציים לפרסה של הפרויקט בסביבת production

הם קבצי הגדנות שימושיים לשרתים מסוימים שונים  
להריץ את הפרויקט שלנו.

# מבנה הפרויקט:

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows a project structure with a folder named "LEC1". Inside "LEC1" is a folder "lec1" containing files: \_\_init\_\_.py, asgi.py, settings.py (which is selected), urls.py, wsgi.py, and manage.py.
- settings.py** editor tab:
  - File path: lec1 > settings.py
  - Content:

```
1  """
2  Django settings for lec1 project.
3
4  Generated by 'django-admin startp
5
6  For more information on this file
7  https://docs.djangoproject.com/en
8
9  For the full list of settings and
10 https://docs.djangoproject.com/en
11 """
```

הקובץ settings.py

קובץ הגדרות:

תיקית קבצים  
אבטחה  
סיסמאות

מסד נתונים  
גישה

ספריות בשימוש  
tabनीयत בשימוש  
קובץ urls  
middleware

הגדרות שפה ותאריכים

תיקיה סטטית להגשת קבצים  
טיפוסי בירית מחדל לשדות בדטה-בייס  
ועוד ...

# מבנה הפרויקט:

url.py

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

מגדירים כתובות ומינ הview שמאזב לכתובות.

# הרכבת הפרויקט בסביבת הפיתוח:

בטרמינל (בתוך התיקייה lec1)  
נקlid את הפקודה הבאה:

**python3 manage.py runserver**

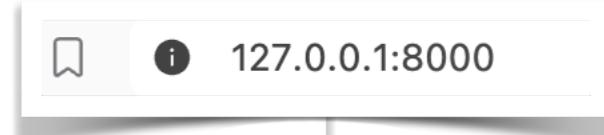
```
└ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
You have 18 unapplied migration(s). Your project may not
Run 'python manage.py migrate' to apply them.
```

```
January 05, 2025 - 09:35:26
Django version 5.1.4, using set
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

כדי לראות את האתר - נקליד את הכתובת בדף:::



The install worked successfully! Congratulations!

[View release notes for Django 5.1](#)

You are seeing this page because DEBUG=True is in your  
settings file and you have not configured any URLs.

**django**

# חלוקת של פרויקט ל-Apps

פרויקט יכול להכיל מספר חלקים שנקראים **Apps** בעולם של Django

זה מאפשר חלוקה של האתר לפי נושאים שונים - מודולציה:

**פרויקט 1**

**Blog App**

**Forum App**

**Analytics App**

**Shop App**

# יצירת app בתוך הפרויקט:

נՐץ בטרמינל:

**python manage.py startapp cards**

1

**lec1/settings.py:**

מוסיף את האפליקציה לקובץ ההגדרות של האפליקציה הראשית

2

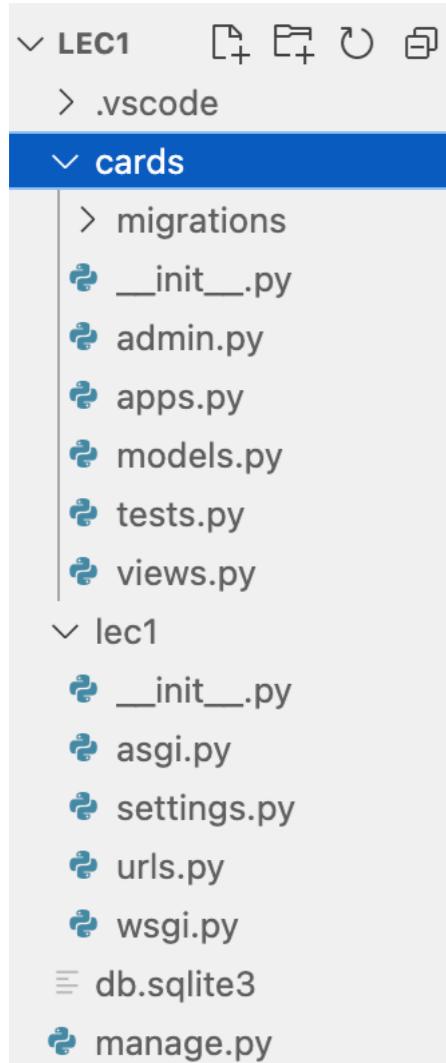
```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'cards'  
]
```

# חלוקת של פרויקט ל-Apps

3

לאחר שהרכנו את הפקודה ליצור App בשם cards  
ונוצרה עבורנו תיקייה בשם cards

בתיקיה יש קבצים חדשים שנכיר בהמשך כגון models.py, views.py, migrations ותיקיה叫做 models.py, views.py, migrations ותיקיה叫做 cards.py, admin.py, apps.py, models.py, tests.py, views.py שאותם נכיר בהמשך.



# חלוקת של פרויקט ל-Apps

לסיכום -

ניתן לחלק את הפרויקט למספר אפליקציות לטובה מודולציה.

ליצירת App:

נקlid בטרמינל את הפקודה הבאה כדי ליצור app חדש בשם myapp

**python3 manage.py startapp myapp**

מוסיף את שם הקומ שיצרנו בקובץ ההגדרות

תחת

INSTALLED\_APPS

# Urls - Views

המשתמש מזין כתובות מסוימת בדף ומקבל תגובה מתאימה.

URL

View

דוגמאות:

<https://www.lec1.com/>

דף ראשי

<https://www.lec1.com/blog>

בלוג

<https://www.lec1.com/blog/post1>

כתב מסויימת בבלוג

# views:

lec1/cards/views.py

קובץ views: קובץ פונקציות שמחזירות תגובה ללקוח:

```
from django.shortcuts import render
from django.http import HttpResponseRedirect

# views are functions that take a web request and return a web response

def index(request):
    return HttpResponseRedirect("Hello, world. You're at the cards index.")
```

# urls:

lec1/lec1/urls.py

הגדרת נתיבים:

```
from django.contrib import admin
from django.urls import path
from cards.views import index

urlpatterns = [
    path('admin/', admin.site.urls),
    path('cards/', index),
]
```

python manage.py runserver

localhost:8000

localhost:8000/cards

# תרגיל:

views.py x

cards &gt; views.py

```
1 from django.http import HttpResponse  
2  
3 def index(request):  
4     return HttpResponse("Cards index.")  
5  
6 # your code here...  
7
```

צrho 2 פונקציות חדשות בקובץ views

בדומה לפונקציה index

הfonקציות יחזירו מחרוזת בהתאם לשם הפונקציה.

יש לקרוא לפונקציה הראשונה foo

יש לקרוא לפונקציה השנייה bar

urls.py x

lec1 &gt; urls.py

```
1 from django.contrib import admin  
2 from django.urls import path  
3 from django.urls import include  
4 from cards.views import index, foo, bar  
5  
6  
7 urlpatterns = [  
8     path('admin/', admin.site.urls),  
9     path('cards/', index),  
10    # your code here...  
11]
```

עדכנו את הקובץ urls.py בapp הראשי של הפרויקט

# פתרונות:

lec1/cards/views.py

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Cards index.")

def foo(request):
    return HttpResponse("Cards foo.")

def bar(request):
    return HttpResponse("Cards bar.")
```

lec1/lec1/urls.py

```
from django.contrib import admin
from django.urls import path
from django.urls import include
from cards.views import index, foo, bar

urlpatterns = [
    path('admin/', admin.site.urls),
    path('cards/', index),
    path('cards/foo/', foo),
    path('cards/bar/', bar),
]
```

צרו פרויקט חדש לתרגול:  
בשם djangopractice

(פתחו את התוכנה בVSCode)

צרו אפליקציה בתוך הפרויקט  
בשם students

הוסיפו את האפליקציה לקובץ settings.py

צרו 2 נתיבים עם תגובה:  
/students  
/teachers

# יצירת קובץ urls פרטני לכל מודול:

```
from django.urls import path
from students import views

urlpatterns = [
    path('add/', views.add_name),
    path('get/', views.get_names),
]
```

בכל מודול יוצרים קובץ urls  
ומגדירים בו את כל מה שקשרו למודול:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('students/', include('students.urls')),
]
```

פעם אחת ביצירת מודול:  
מקשרים את קובץ urls שלו:

כל הכתובות שמתחילה ב  
/students  
וופנו לקובץ urls של students

היא פונקציה ב-Django שמאפשרת לכלול קבצי **urls.py** נוספים בתוך קובץ URL ראשי,  
וכך ליצור מבנה מודולרי וברור של נתיבי הפרויקט.

כל נתיב שמתחילה במילה students יופנה לקובץ students/urls.py להמשך טיפול.

עכשו הפרויקט שלנו מסודר ולמדנו מהם Views ומהם urls איך להשתמש בהם.

## בקשות ותגובהות:

```
from django.http import HttpResponseRedirect, HttpRequest, HttpResponseBadRequest,  
HttpResponseRedirect  
  
from django.views.decorators.csrf import csrf_exempt  
  
names = ['Alice', 'Bob', 'Charlie']
```

```
def get_names(request:HttpRequest):  
    return HttpResponseRedirect("", ".join(names))
```

בקשת :GET

בקשת :POST

```
@csrf_exempt # ביטול ההגנה באופן זמני  
def add_name(request:HttpRequest):  
    if request.method == 'POST':  
        name = request.POST.get('name')  
        if name:  
            names.append(name)  
        else:  
            return HttpResponseRedirect('Name is required!')  
  
        return HttpResponseRedirect('Name added successfully!')  
    else:  
        return HttpResponseRedirect("POST Only with name please")
```

# CSRF - Cross-Site Request Forgery

מתקפה ידועה שיש להמנע ממנה בעבודה עם טפסים  
**איך זה עובד:**

לקוח נכנס לדף של w3schools

האתר כולל טופס לשילוח בקשה POST  
לבנק לאומי

במחשב של הלקוח יש עוגיות של בנק לאומי והן נשלחות עם כל בקשה ליזיהו הלקוח.

## התగוננות מפני בקשות כאלה:

בכל שליחה של טופס יש לשלוח מזהה ייחודי שקיבלתם מהשרת שיצר את הטופס

האתר של בנק לאומי:

כל טופס יוכל מזהה ייחודי זמני

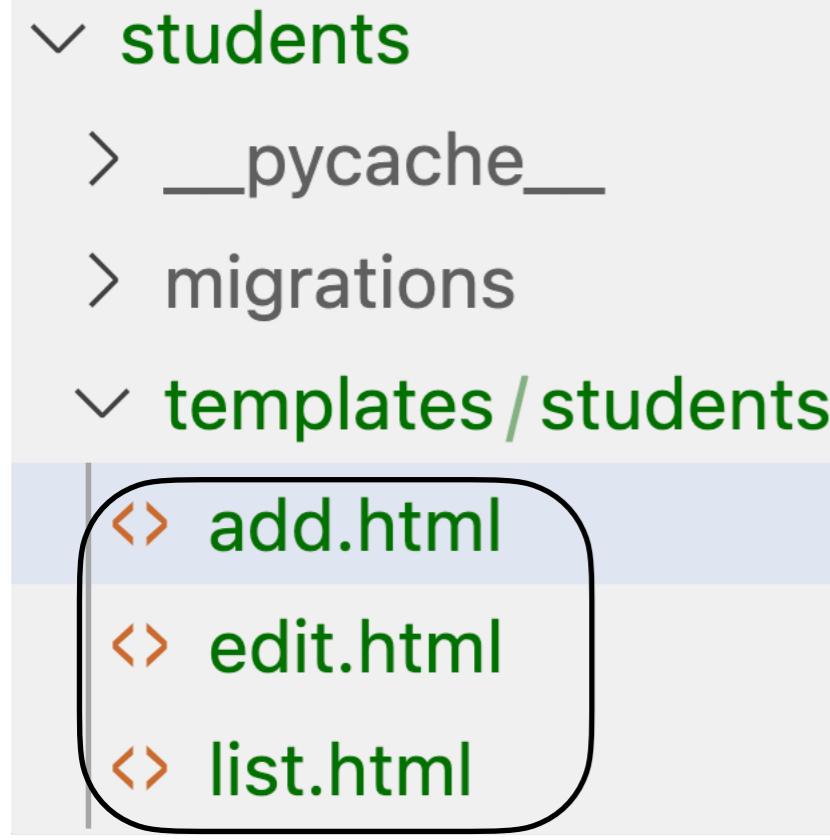
והלקוח יוכל לשלוח את הטופס רק עם המזהה הייחודי זהה בנוסף לפרטי המשתמש.

אתר מתחזה לא יוכל לזייף את החומר token הזמן  
כי לא ניתן token כזה ללקוח שלנו



# עבודה עם :Templates

## הזרת קבצי html:



1 ייצרת תיקיה templates  
שם התקינה חייב להיות templates

2 ייצרת תיקיה students  
בתוך templates  
שם התקינה זהה לשם האפליקציה

3 ניצור קבצי html

Django מכילה מנגנוןים להגנה מפני מתקפות CSRF  
כפי שנראה בהדגמה



## עובדת עם :Templates

### החזרת קבצי :html

### פונקציות Views

```
def get_html(request:HttpRequest):
    return render(request, 'students/list.html')

def add_html(request:HttpRequest):
    return render(request, 'students/add.html')

def edit_html(request:HttpRequest):
    return render(request, 'students/edit.html')
```

```
from django.urls import path
from students import views
```

```
urlpatterns = [
    path('add/', views.add_html),
    path('get/', views.get_html),
    path('edit/', views.edit_html),
]
```

### ניתובים Urls

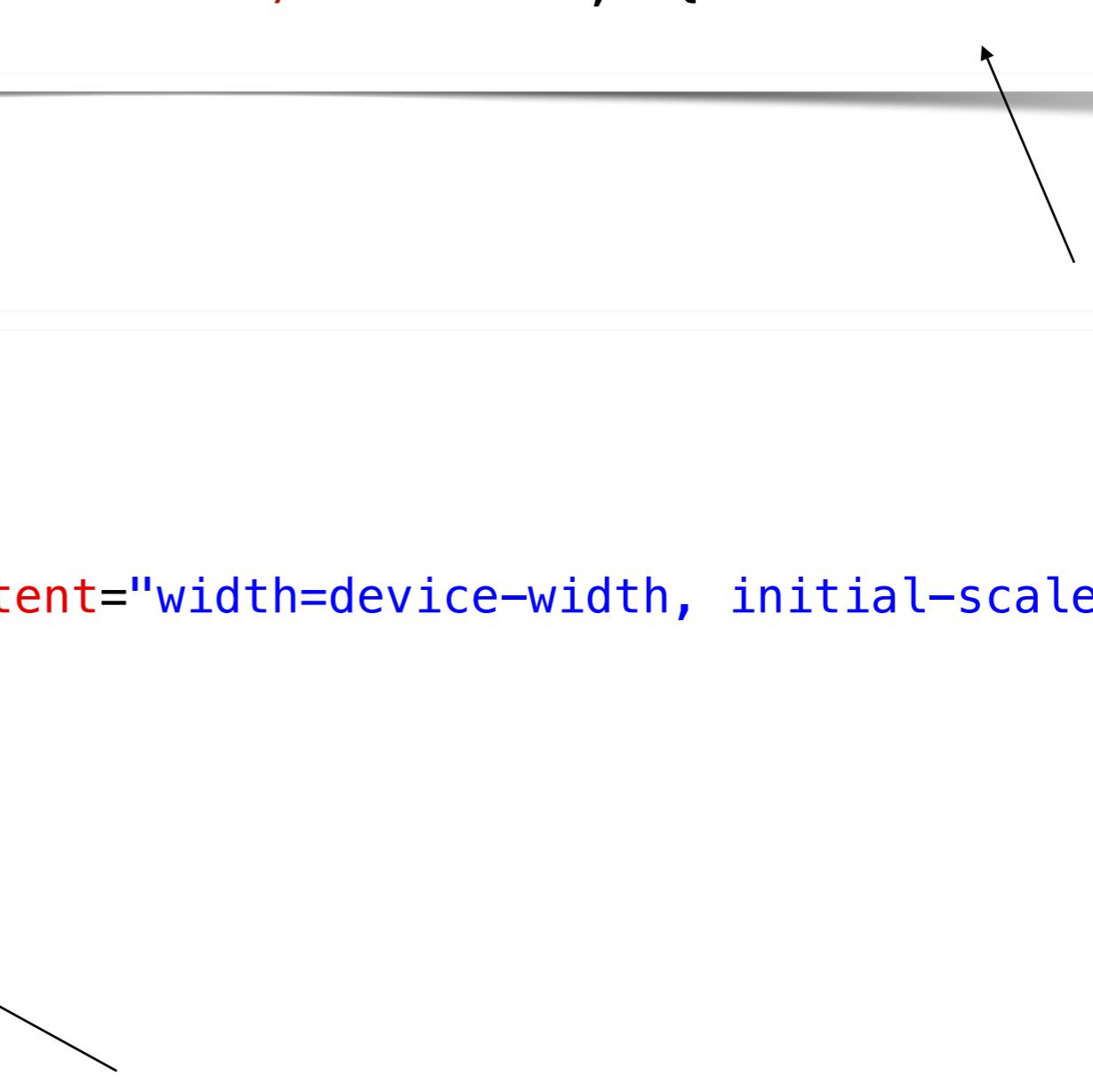
## שליחת data להצגה ב-template

```
names = ['Alice', 'Bob', 'Charlie']
```

```
def get_html(request:HttpRequest):
    return render(request, 'students/list.html', {'names': names})
```

lec1/students/templates/students/list.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1>List of Students</h1>
    {% for name in names %}
      <p>{{ name }}</p>
    {% endfor %}
  </body>
</html>
```



# טופס עם בקשה POST

lec1/students/templates/students/add.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1>Add Student</h1>

    <form method="post">
      {% csrf_token %}
      <label for="name">name</label>
      <input type="text" name="name" id="name" />
      <button type="submit">Add</button>
    </form>
  </body>
</html>
```

הtag הננו מפנוי מתקפת CSRF

# טופס עם בקשה POST

הMETHOD נקראת פעם אחת בבקשת GET כשהלכמה מנוט בדף כתובות:  
/students/add

הדף מציג HTML עם הטופס

הMETHOD מופעלת גם בבקשת POST כשהלכמה לוחץ על כפתור submit  
או בשולחים בקשה POST עם Postman

```
def add_html(request:HttpRequest):
    if request.method == 'POST':
        name = request.POST.get('name')

        if name:
            names.append(name)
            return HttpResponseRedirect('/students/get')
    else:
        return HttpResponseBadRequest('Name is required!')

    return render(request, 'students/add.html')
```

# Path Segments:

```
from django.urls import path
from students import views

urlpatterns = [
    path('add/', views.add_html),
    path('get/', views.get_html),
    path('edit/<int:id>', views.edit_html),
]
```

הגדרת url דינامي:

1

```
def edit_html(request:HttpRequest, id:int):
    return render(request, 'students/edit.html')
```

הוספה פרמטר בview

2

lec1/students/views.py

## Path Segments:

```
def edit_html(request:HttpRequest, id:int):
    if id in range(len(names)) and request.method == 'GET':
        name = names[id]
        return render(request, 'students/edit.html', {'name': name, 'id': id})

    if id in range(len(names)) and request.method == 'POST':
        name = request.POST.get('name')
        names[id] = name
        return HttpResponseRedirect('/students/get')

    else:
        return HttpResponseBadRequest('Invalid ID')
```

lec1/students/templates/students/edit.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>
        Edit Student
    </h1>

    <form method="post">
        {% csrf_token %}
        <label for="name">Name</label>
        <input type="text" name="name" id="name" value="{{name}}>
        <input type="hidden" name="id" value="{{id}}>
        <button type="submit">Save</button>
    </form>
</body>
</html>
```

## עבודה עם דטה-בייס:

lec1/students/models.py

```
from django.db import models  
  
class Student(models.Model):  
    name = models.CharField(max_length=40)
```

1

סורק את כל המודלים באפליקציה:  
אם יש מודל חדש - יוצר קוד sql ליצירת הטבלאות

```
python manage.py makemigrations students
```

2

מפעיל את כל הקוד באנטס מול הדטה-בייס

```
python manage.py migrate
```

3

## עבודה עם דטה-בייס:

```
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
Applying students.0001_initial... OK
```

טבלאות של django  
למשك ניהול  
ולניהול משתמשים

המיגרציה של יצרת סטודנטים

## יצירת משתמש למשק ניהול:

```
python manage.py createsuperuser
```

## עובדת עם דטה-בייס:

צפיה בתוכן של מסד הנתונים ב-`VSCode`

The screenshot shows the VSCode interface with the 'db.sqlite3' database selected in the sidebar. The main area displays the contents of the 'auth\_user' table. The table has columns: id, password, last\_login, and is\_superuser. There are two rows: one with id 1 and another with id 2. The password column contains hashed values. The 'auth\_user' table is highlighted in blue.

	1	pbkdf2_sha256\$870000\$gGINhQGmWAqcgD0iY74d...	NULL	1		
	2					

צפיה בתוכן של מסד הנתונים במשמעות הנהוּל:

<http://localhost:8000/admin>

# הוספתطالب למסך ניהול

lec1/students/admin.py

```
from django.contrib import admin
from students.models import Student

# Register your models here.
admin.site.register([Student])
```

The screenshot shows the Django administration interface. The top navigation bar says "Django administration". Below it, the breadcrumb navigation shows "Home > Students > Students > Add student". On the left, there's a sidebar with "AUTHENTICATION AND AUTHORIZATION" section containing "Groups" and "Users" with "+ Add" buttons. The "STUDENTS" section contains "Students" with a yellow background and a circled "+ Add" button. The main content area has a green success message: "The student "Student object (2)" was added successfully. You may add another student below." It then shows a form titled "Add student" with a "Name:" field containing "Charlie". At the bottom are three buttons: "SAVE", "Save and add another", and "Save and continue editing".

## View להציג סטודנטים מהדטה-בייס:

views.py

```
from students.models import Student

def view_list_db(request:HttpRequest):
    students = Student.objects.all()
    return render(request, 'students/list_db.html', {'students': students})
```

urls.py

```
urlpatterns = [
    path('add/', views.add_html),
    path('get/', views.get_html),
    path('edit/<int:id>', views.edit_html),
    path('get_db/', views.view_list_db),
]
```

## View להציג סטודנטים מהדטה-בייס:

lec1/students/templates/students/list\_db.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Students from database</h1>

    {% for student in students %}
        <p>{{ student.name }}</p>
    {% endfor %}

</body>
</html>
```

## דברים שכדאי לזכור:

`pip install django`

`django-admin startproject lec1`

`python manage.py startapp students`

`INSTALLED_APPS+=['students']`

`views.py` מוסיפים פונקציות ל`request/response` של האפליקציה

יוצרים קובץ `urls.py` באפליקציה החדשה ומגדירים שם

בקובץ `urls.py` של האפליקציה הראשית: עושים `include`

`python manage.py runserver`

עורכים את קובץ `models.py` מוסיפים מודול

`python3 manage.py makemigrations students`

`python3 manage.py migrate`

`python3 manage.py createsuperuser`

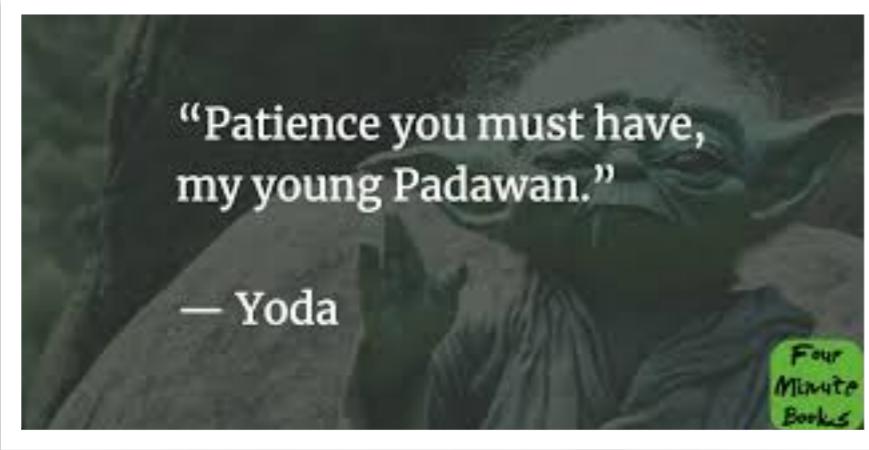
עורכים את הקובץ `admin` להוספת מודלים שלנו למשק ניהול

גישה למשק ניהול:  
`localhost:8000/admin`

# שיעור בית:

- 1) צרו פרויקט חדש בערת הפקודה `python3 django-admin startproject lec1_practice`
- 2) צרו 2 אפליקציות בתחום הפרויקט  
יש לקרוא לאפליקציה הראשונה `quotes`  
ולאפליקציה השנייה לקרוא `forum`
- (לא לשכוח לחבר את האפליקציות לפרויקט הראשי בקובץ `INSTALLED_APPS` בקובץ `py.settings` של האפליקציה הראשית)
- 3) באפליקציה `quotes` יש להגדיר בקובץ `views.py`  
את הפעולות הבאות:
  - תחזר טקסט מתאים `quotes`
  - תחזר טקסט מתאים `about`
- 4) באפליקציה `forum` יש להגדיר בקובץ `views.py`  
את הפעולות הבאות:
  - תחזר טקסט מתאים `topics`
  - תחזר טקסט מתאים `users`
- 5) יש להגדיר קובץ `urls.py` עבור כל אחת מהאפליקציות
- 6) יש לחבר את הקובץ `urls.py` לאפליקציה הראשית  
ע"י עריכה של הקובץ `py.urls` של האפליקציה הראשית.  
(זכרו לכל אפליקציה יש `urls.py` משלה).

# שיעור בית:



7) באפליקציה quotes יש ליצור רשימה של quotes ולהציג את הרשימה ללקוח בתגובה לבקשת GET /quotes/list/

8) באפליקציה quotes יש ליצור view בשם add\_quote שיאפשר להשתמש להוספה quote לרשימה quotes

9) באפליקציה quotes יש להציג את הדסוט quotes בHTML

10) באפליקציה quotes יש לאפשר הוספת quote בעזרת טופס HTML

ריעונות לquotes:

<https://dummyjson.com/quotes/random/10>