

Django

TomerBu

נושאים להיום:

מבוא והכרות

דרישות לפרויקט

התקנות ויצירת פרויקט ראשון

מבנה הפרויקט

views

urls

בקשות ותגובה

path segments

הכרות עם תבניות html

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.

Django Web Framework

מערכת לבניית אתרים

Intro to Django

[EXPAND ALL / COLLAPSE ALL](#)

Object-relational mapper

נכתב Class בפייתון -> המערכת תגדיר טבלה בדטה-בייס
המערכת מגדרה לכם מethodות עברו כל פעולות SQL



URLs and views

views:
פונקציה שmagiba לבקשת
מייפוי כתובות urls
ואיזו פונקציה תגיב להן



Templates

החזרה של דפי HTML מועשרים ע"י פייתון



Forms

עבודה עם טפסים (html)
בצורה מאובטחת (ובקלות)



Authentication

*מערכת לניהול/זיהוי/הרשמה

*האם ניתן לבצע פעולה עברו משתמש מסוים



Admin

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



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/5.2/>

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

התקנה חד פעמית לכל המחשב:

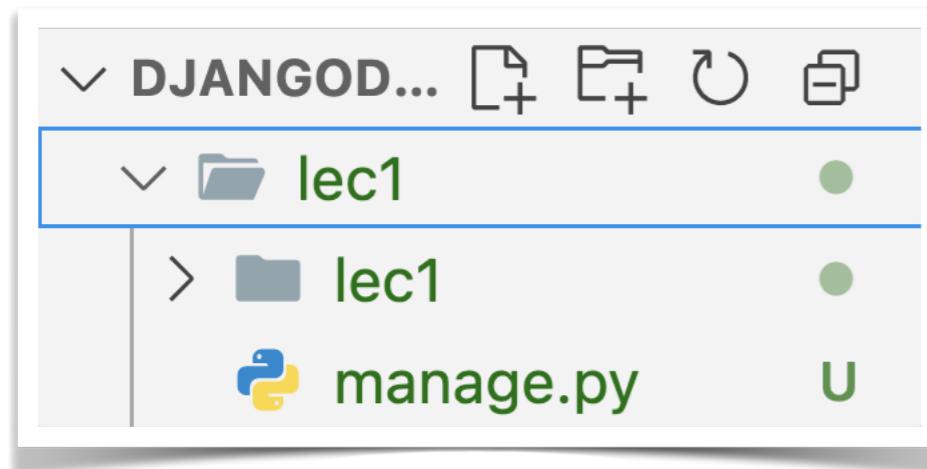
```
pip install django
```

מתוך אוסף כלים שאיתם יוצרים פרויקטים

(התקנה חד פעמית במחשב)

יצירת פרויקט חדש:

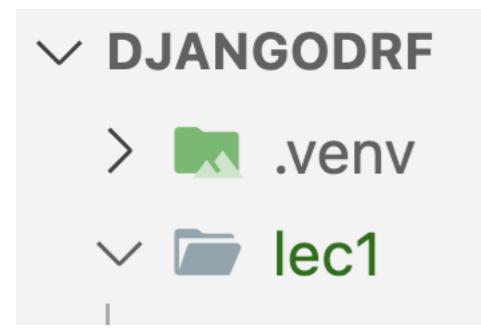
```
django-admin startproject lec1
```



נוצרה תיקיה בשם lec1

פתחת את התיקיה עם VSCODE

```
python -m venv .venv
```



מארק:

```
.\.venv\Scripts\activate
```

```
source ./venv/bin/activate
```

התקנת ספריות לפרויקט ותחילת עבודה:

```
pip install django
```

דרישות לפרויקט:

ראו קובץ מצורף בGITLAB

מבנה הפרויקט:

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

- EXPLORER** sidebar: Shows a project structure with a folder named "LEC1" containing a subfolder "lec1" which includes files: __init__.py, asgi.py, settings.py (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 """
```

קובץ הגדרות

הקובץ settings.py

קובץ הגדרות:

תיקית קבועים
אבטחה
סיסמאות

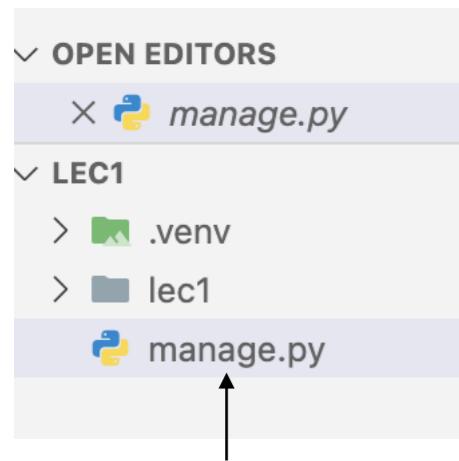
מסד נתונים
גישה

ספריות בשימוש
tabnionts בשימוש
קובץ urls
middleware

הגדרות שפה ותאריכים

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

מבנה הפרויקט:



manage.py

קובץ ראשי - הקובץ שמכיל פונקציית `main` לפרויקט.

היא נקודת הכניסה לתוכנית.

הנקודה שבה המערכת מגדרה שקובץ `settings.py` הוא קובץ ההגדרות שלנו ומריצה את התוכנית.

`python manage.py runserver`

כדי להריץ את התוכנית נכתב בטרמינל:

`python manage.py runserver`

התקנות:

התקנה של ספריה באופן גלובלי:

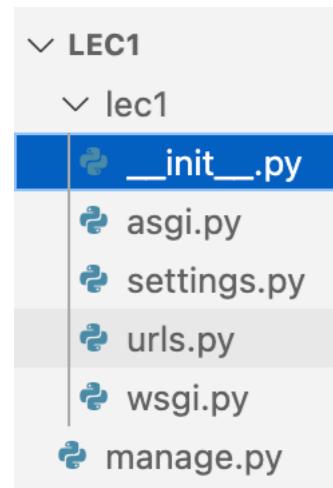
```
pip install django
```

(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

.....

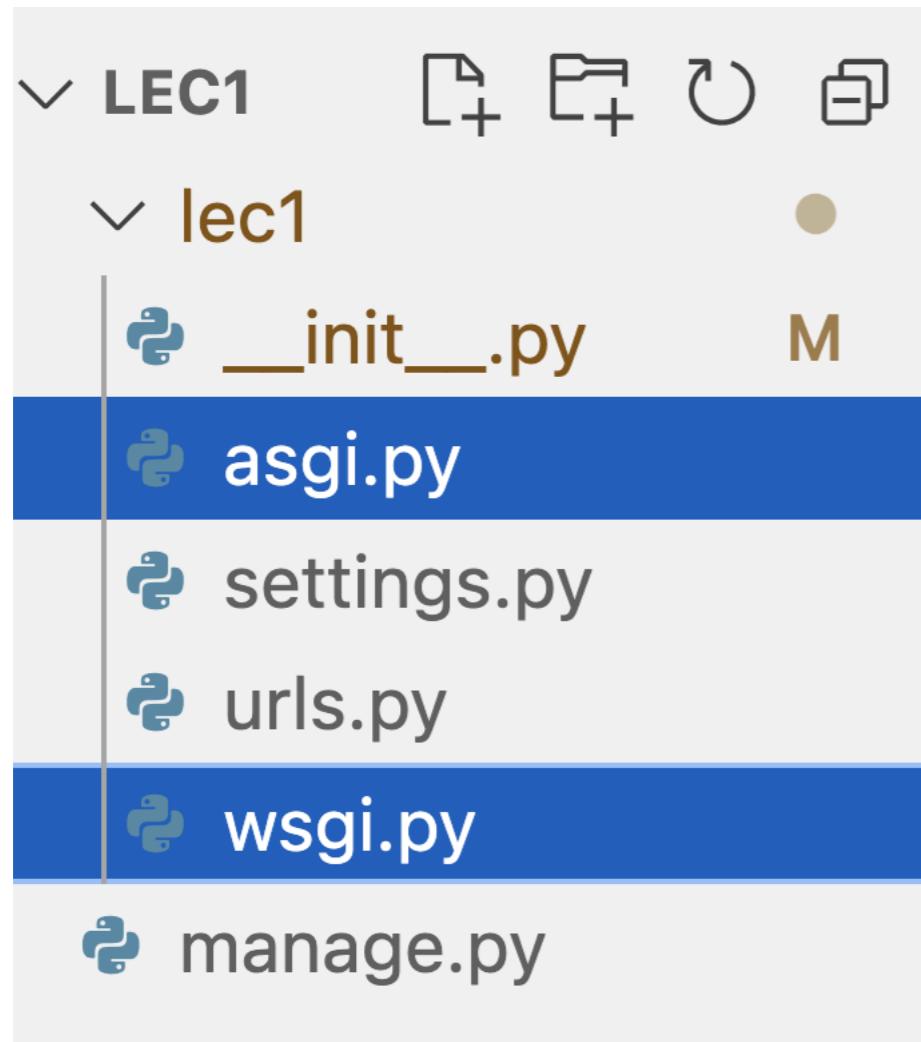
This file indicates that the directory is a Python module.

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

קבצים לפרסה של הפרויקט בסביבת production

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

מבנה הפרויקט:

urls.py

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

def my_fun(request):
    return HttpResponse("Foo!")

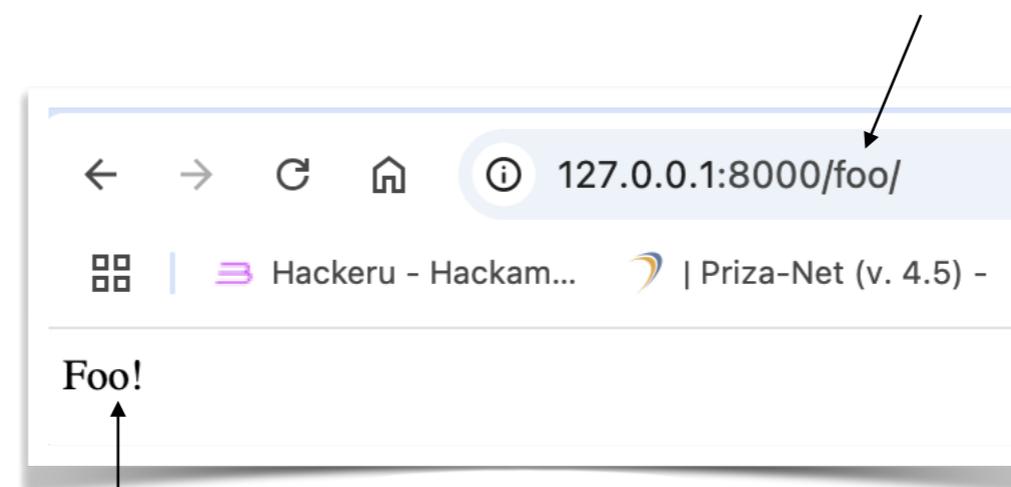
urlpatterns = [
    path('admin/', admin.site.urls),
    path('foo/', my_fun, name='foo')
]
```

פונקציה נקראת View

מגדירים כתובות ומיצי הview שmagiv לכתובות.

כתובות

פונקציה



מבנה הפרויקט:

urls.py

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

def my_fun(request):
    return HttpResponse("Foo!")

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

לא נכתב פה את הפונקציות וגם את הנתיבים

נרצה לחלק את התוכנה למודולים

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 Lec1

Blog App

Forum App

Analytics App

Shop App

Cards App

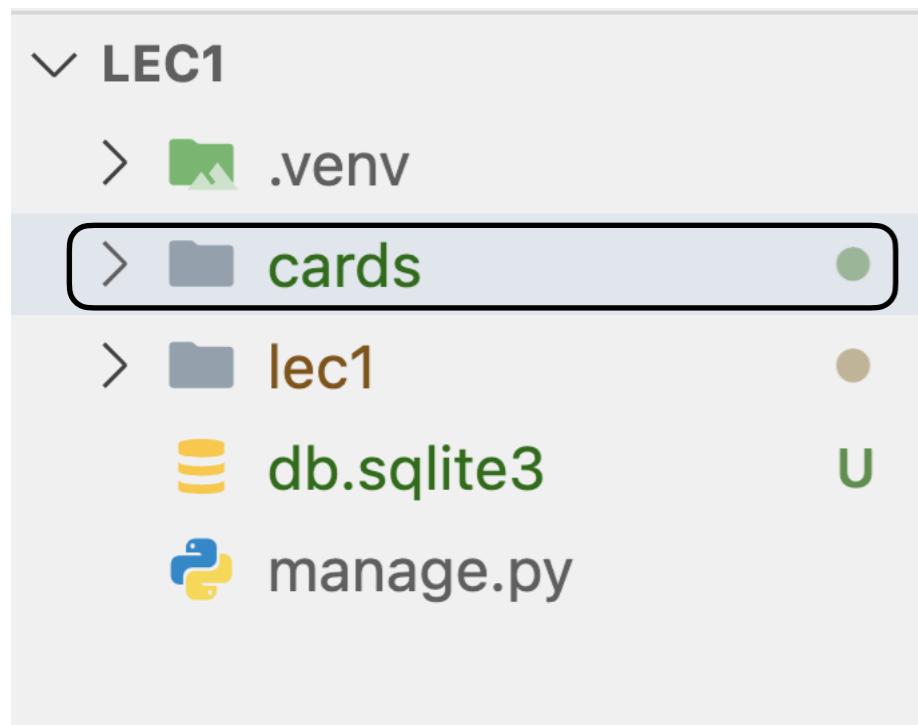
Users App

About App

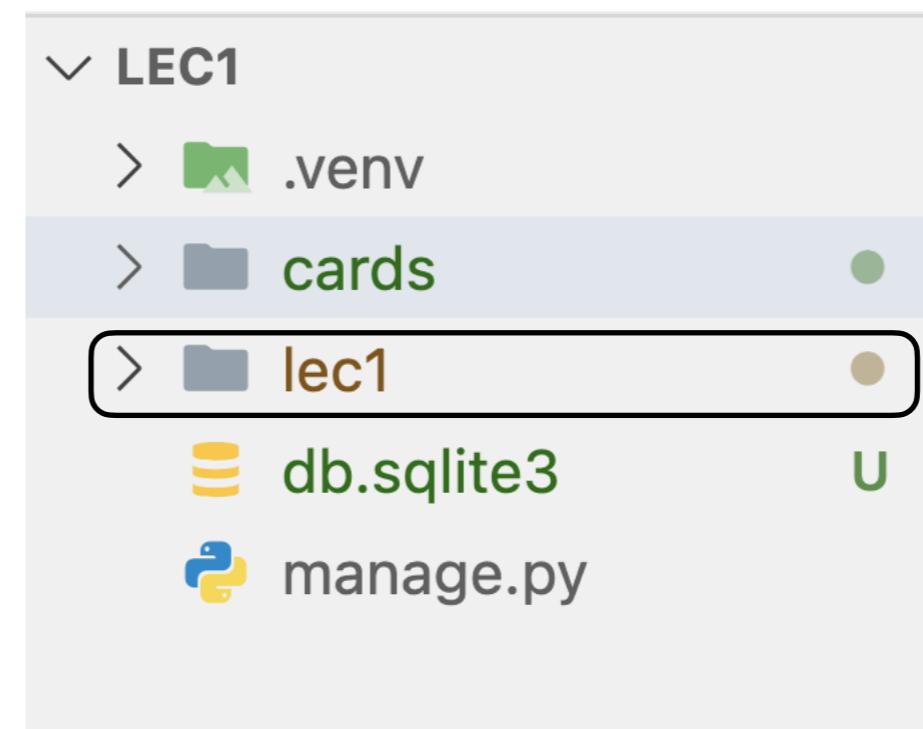
יצירת app בתוך הפרויקט:

app יצרנו
או מודול בפרויקט

לפי נושא - במקרה שלנו כרטיסיות



המודול הראשי של הפרויקט



יצירת app בתוך הפרויקט:

נՐץ בטרמינל:

python manage.py startapp cards

1

lec1/settings.py:

מוסיף את האפליקציה לקובץ ההגדרות של האפליקציה הראשית

2

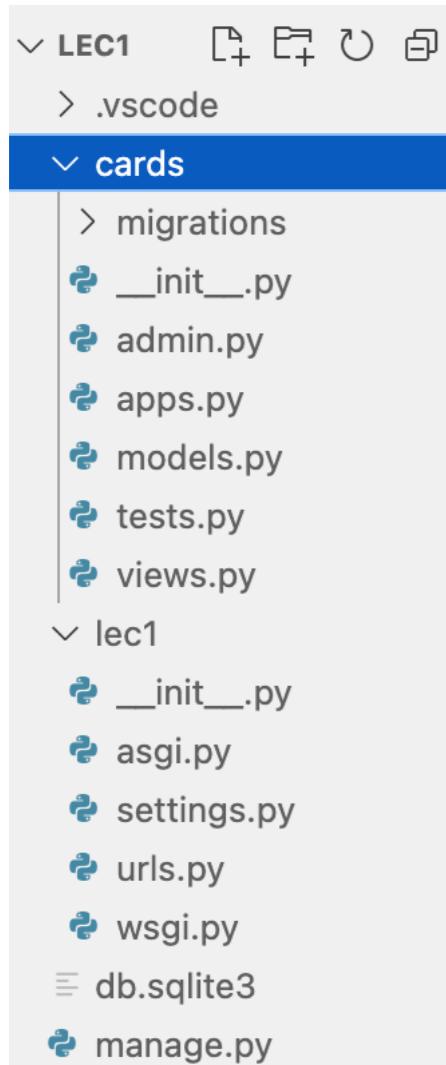
```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'cards'  
]
```

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` ו-`migrations` והתיקיה `models.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>

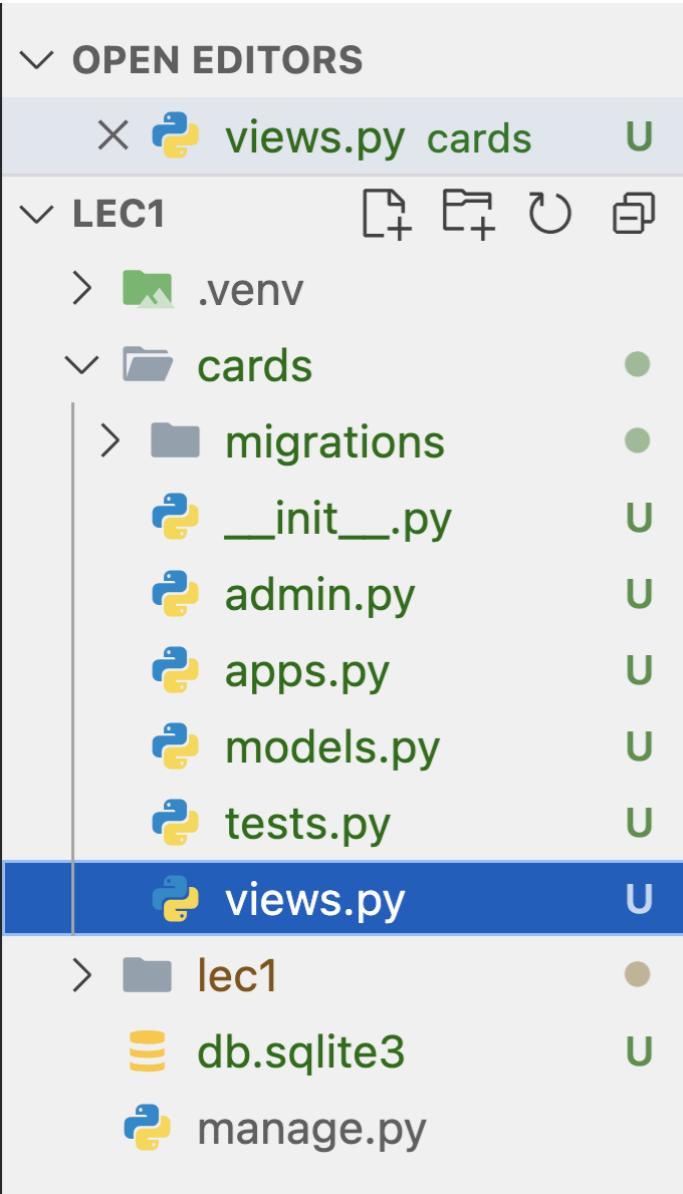
כתב מסויימת בבלוג

Urls - Views

 views.py U X

נוסיך View חדש

1

cards >  views.py

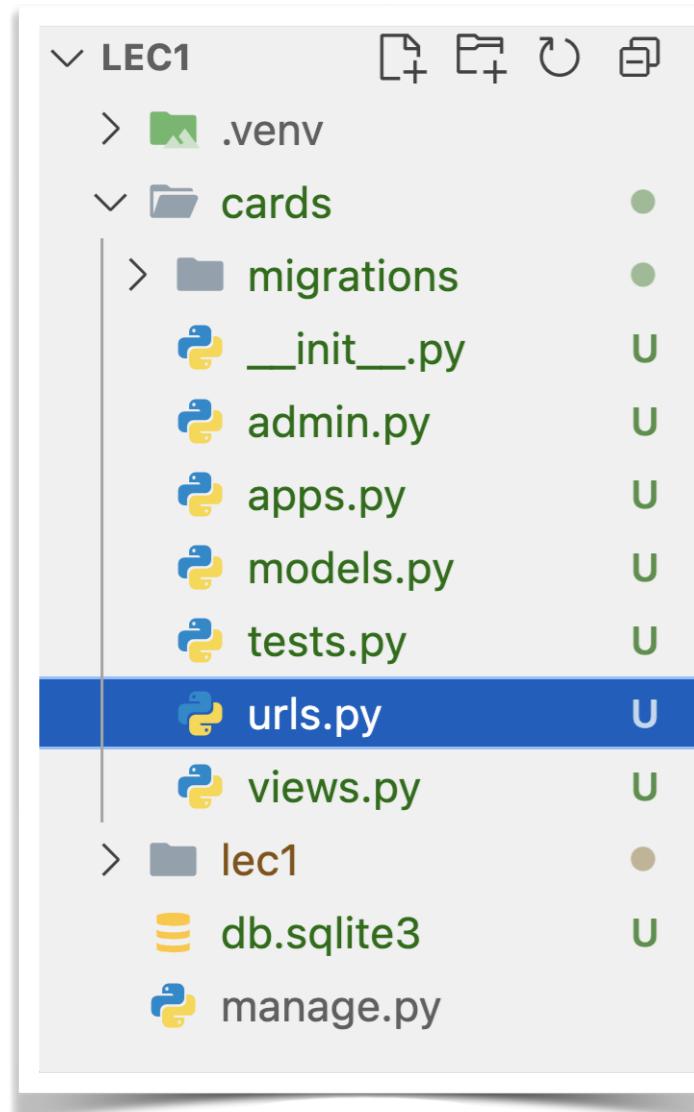
```
from django.shortcuts import render
from django.http import HttpResponseRedirect
# Create your views here.
```

```
def hi(request):
    return HttpResponseRedirect('Hola')
```

Urls - Views

ניצור קובץ urls.py בתחום cards

2



```
from django.urls import path
from cards.views import hi

urlpatterns = [
    path("hi/", hi, name="hi")
]
```

Urls - Views

3

נירוך את הקובץ urls.py בתיק lec1 (הפרויקט הראשי)

```
from django.contrib import admin
from django.urls import path, include
from cards.urls import urlpatterns as cards_urls

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

ນבker בכתובת הבאה כדי לבדוק:

<http://127.0.0.1:8000/cards/hi/>

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:

1

views.py x

cards > views.py

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

צורך פונקציות חדשות בקובץ views

בדומה לפונקציה hi

הפונקציות יחזירו מחרוזת בהתאם לשם הפונקציה.

יש לקרוא לפונקציה הראשונה foo

יש לקרוא לפונקציה השנייה bar

urls.py u x

cards > urls.py > ...

```
1 from django.urls import path
2 from cards.views import hi
3
4 urlpatterns = [
5     path("hi/", hi, name="hi")
6 ]
7
```

עדכנו את הקובץ urls.py בapp שיצרתם

http://127.0.0.1:8000/cards/foo/

http://127.0.0.1:8000/cards/bar/

http://127.0.0.1:8000/cards/hi/

2

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 עם

1

`django-admin startproject lec1practice`

2

צרו VENV לפि ההנחיות בגיטהאב

3

בתוך התיקיה יש להתקין את הספרייה `django`

4

צרו אפליקציה בתוך הפרויקט
בשם `students`

5

הוסיפו את האפליקציה ל`installed_apps` בקובץ `settings.py`

6

צרו 2 נתיבים עם תగובה:

`/students/all`

`/students/me`

7

כל נתיב יחזיר טקסט כמו שעשינו בתרגילים הקודמים

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 HttpResponse, HttpResponseRedirect, JsonResponse
from django.views.decorators.csrf import csrf_exempt

names = ['Alice', 'Bob', 'Charlie']  
בקשת :GET
```

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

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

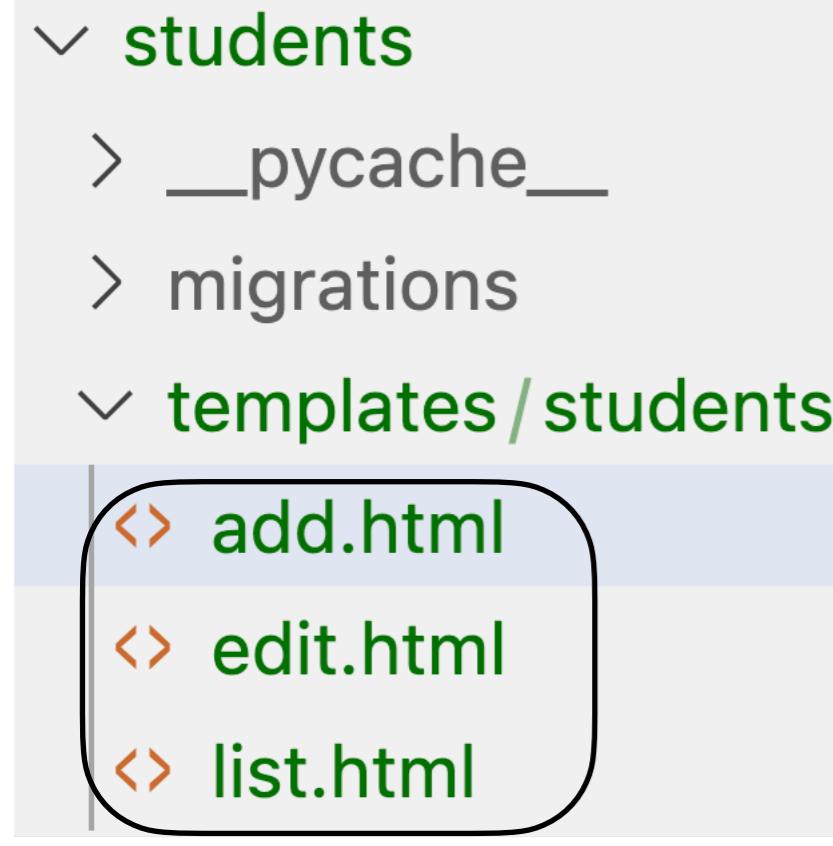
והלקוח יוכל לשלוח את הטופס רק עם המזהה הייחודי זהה בנוסף לפרטי המשתמש.

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



עבודה עם :Templates

הזרת קבועים :html



1 ייצרת תיקיה
templates
שם התקינה חייב להיות.templates

2 ייצרת תיקיה
students
templates
בתוך
שם התקינה זהה לשם האפליקציה

3 ניצור קבועי 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>
```

The diagram illustrates the flow of data from the Python code to the template. An arrow points from the line 'names = ['Alice', 'Bob', 'Charlie']' in the Python code to the line '{% for name in names %}' in the template. Another arrow points from the line 'return render(request, 'students/list.html', {'names': names})' in the Python code to the line '{{ name }}' in the template, indicating that the 'names' variable is passed to the template and used within the loop.

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 לווח 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>
```