



Kontakt:

Uwaga

Wiadomości email sprawdzam raz w tygodniu – po wysłaniu wiadomości proszę przez tydzień jej nie ponawiać. W przypadku, kiedy nie odpowiem proszę o ponowne przesłanie wysłanie oraz wiadomość na MSTeams (prywatną wiadomość)

Email: szymon.guzik@gdansk.merito.pl

LUB

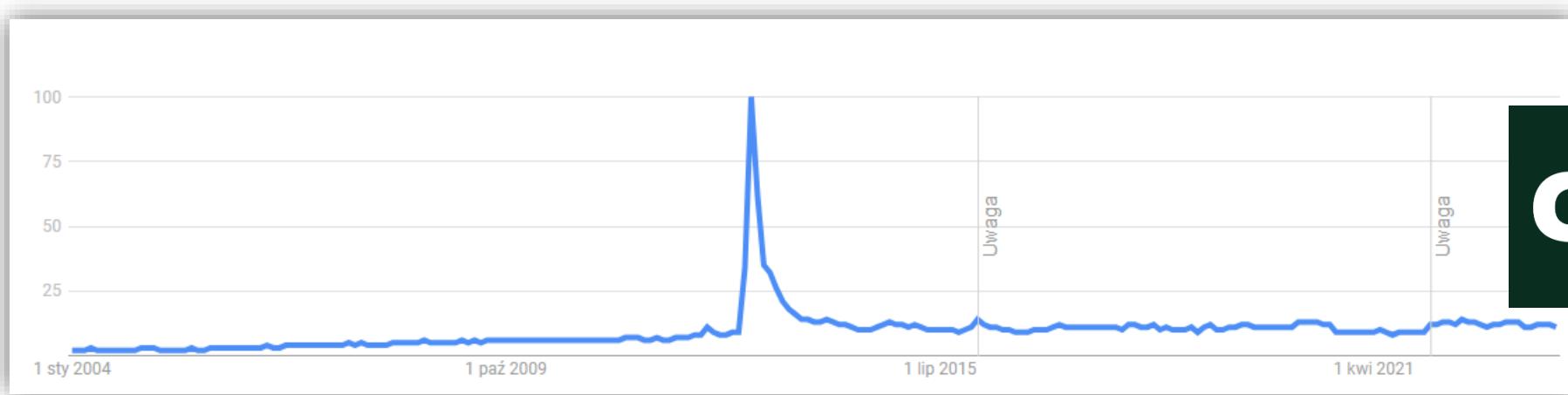
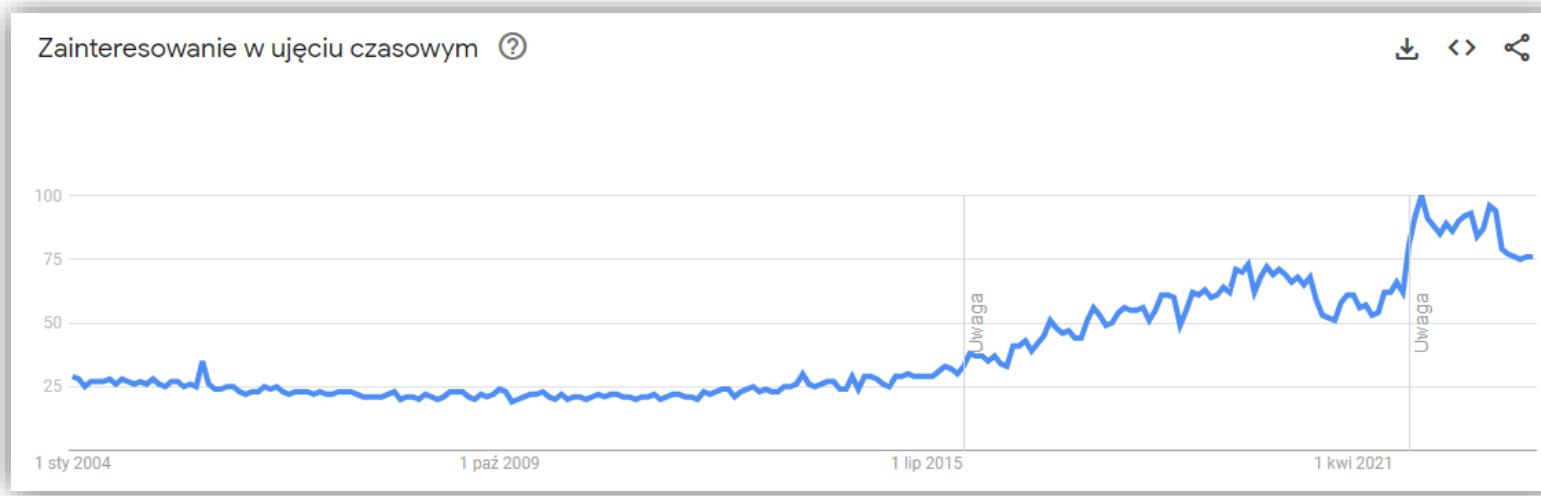
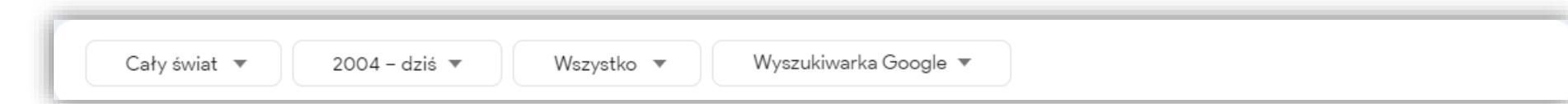


Microsoft
Teams

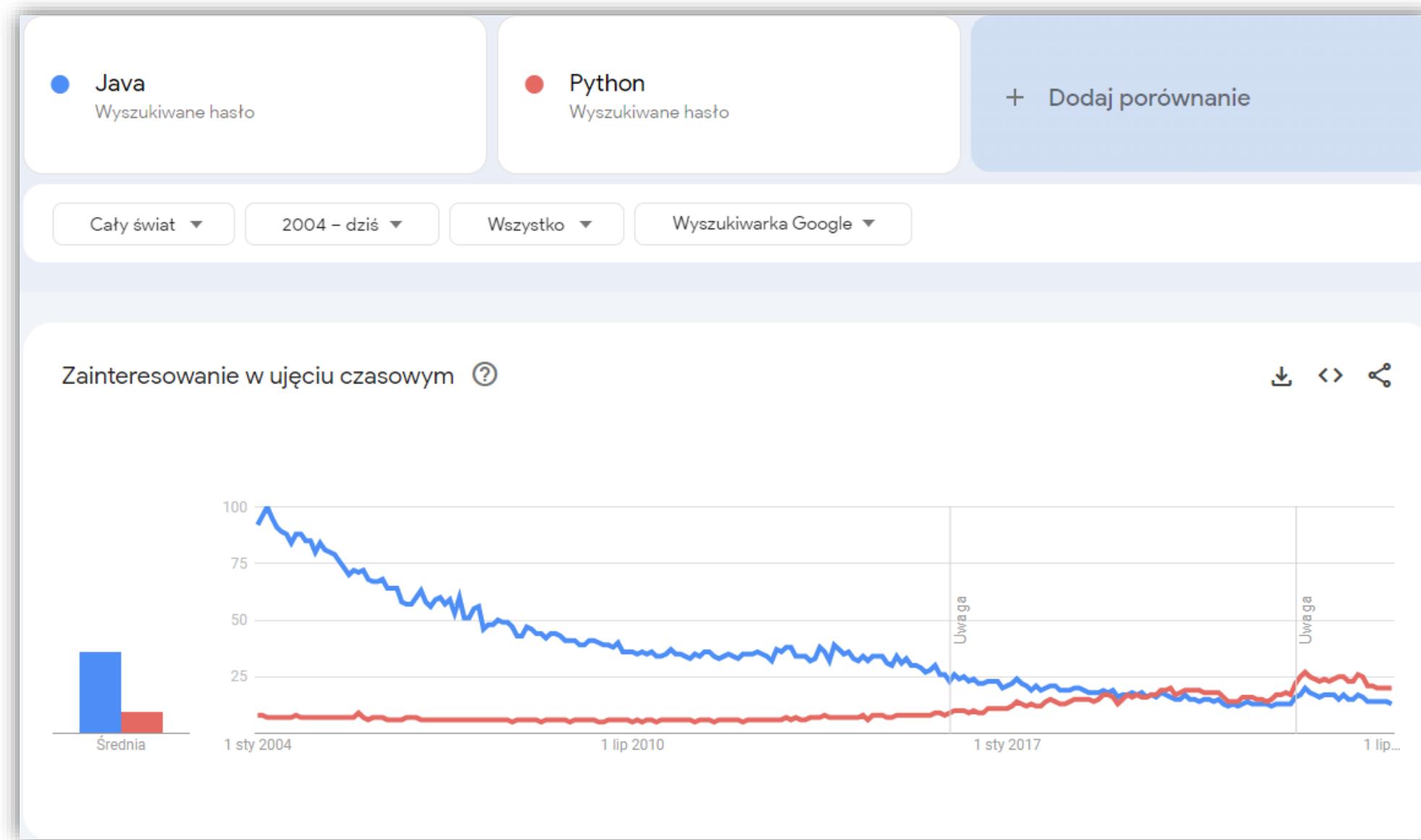
Zasady zaliczenia przedmiotu:

- Zaliczenie laboratorium odbywa się z wykorzystaniem platformy Moodle
- Na każdych laboratoriach wykonywane jest zadanie lub realizacja swojego projektu
- Z każdego ćwiczenia można zdobyć maksymalnie 100 pkt.
- Obecność na laboratoriach jest obowiązkowa
- Dodanie pliku z rozwiązaniem zadania z laboratorium lub przesłanie fragmentu projektu dot. omawianego zagadnienia (2 tygodnie od laboratorium z zadaniem do wykonania)
- Przesłane pliki będą sprawdzane w miarę na bieżąco - po sprawdzeniu od razu otrzymacie pkt za przesłane zadanie
- Ocena:
 - 150 pkt - 3.0
 - 180 pkt - 3.5
 - 210 pkt - 4.0
 - 240 pkt - 4.5
 - 270 pkt - 5.0

Dlaczego Django ?



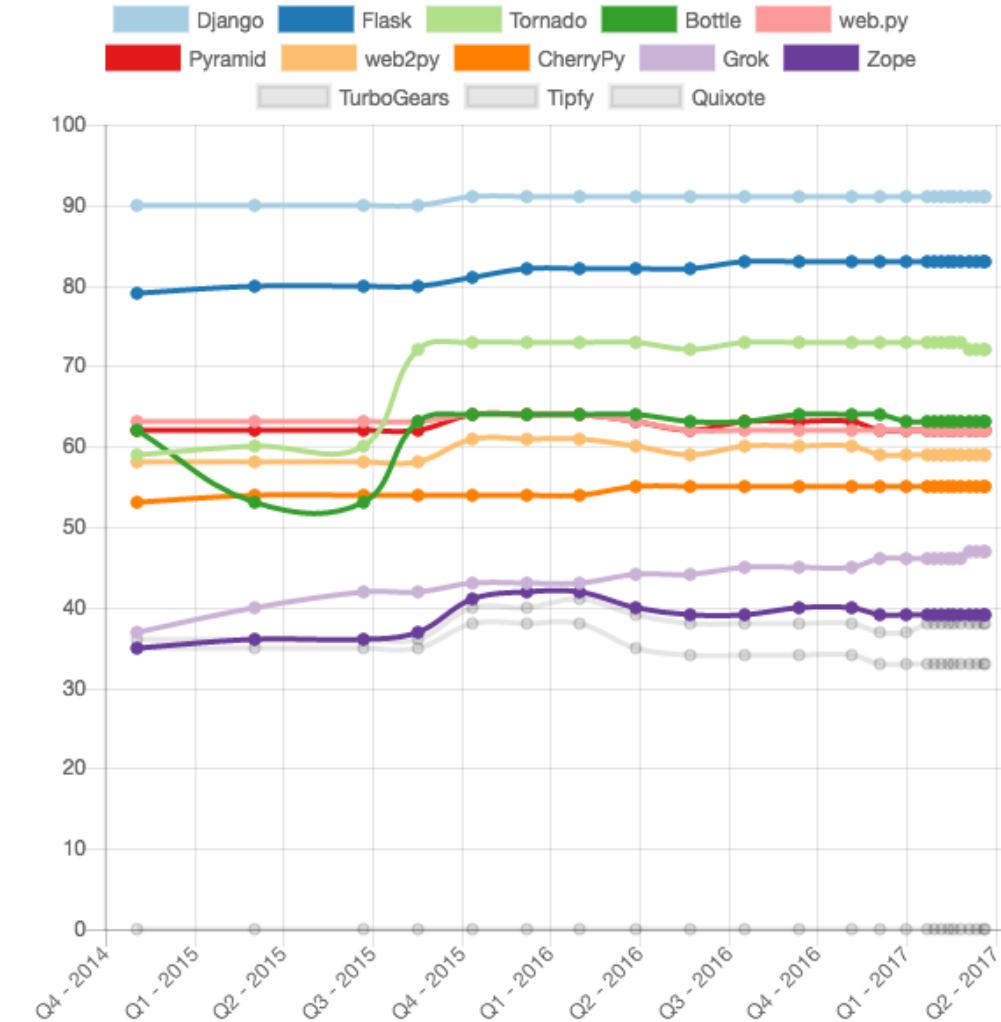
Dlaczego Django ?



Dlaczego Django ?

Python

Framework	Score
Django	91
Flask	83
Tornado	72
Bottle	63
web.py	62
Pyramid	62
web2py	59
CherryPy	55
Grok	47
Zope	39
TurboGears	38
Tipfy	33
Quixote	0



Dlaczego Python ? Dlaczego Django ?

Python stał się jednym z najpopularniejszych języków programowania na świecie i ma wiele zalet, które sprawiają, że warto się go uczyć. Oto kilka powodów, dla których warto nauczyć się Pythona:

- **Łatwość nauce i czytelność:** Python jest znany z prostej składni, która jest bardzo czytelna. Dzięki temu jest doskonałym językiem dla początkujących programistów.
- **Wszechstronność:** Pythona można używać w wielu dziedzinach, od tworzenia stron internetowych (za pomocą Django lub Flask), przez analizę danych (pandas, numpy), uczenie maszynowe (TensorFlow, scikit-learn), aż po automatykę i scripting.
- **Duża społeczność:** Python ma ogólną społeczność, co oznacza, że są dostępne liczne zasoby, takie jak biblioteki, narzędzia oraz fora i grupy wsparcia, które pomogą w rozwiązaniu problemów.
- **Rozwój kariery:** Z powodu rosnącej popularności Pythona, umiejętności programowania w tym języku jest bardzo ceniona przez pracodawców, zwłaszcza w dziedzinach związanych z analizą danych, sztuczną inteligencją czy web developmentem.
- **Biblioteki i frameworki:** Python ma rozbudowany ekosystem bibliotek i narzędzi, które czynią go odpowiednim dla różnorodnych zastosowań. Niezależnie od tego, czy chodzi o analizę danych, tworzenie interfejsów użytkownika czy tworzenie gier, istnieją odpowiednie narzędzia dla Pythona.

Dlaczego Python ? Dlaczego Django ?

Python stał się jednym z najpopularniejszych języków programowania na świecie i ma wiele zalet, które sprawiają, że warto się go uczyć. Oto kilka powodów, dla których warto nauczyć się Pythona:

- **Przenośność:** Python jest wieloplatformowy, co oznacza, że można go uruchomić na różnych systemach operacyjnych.
- **Integracja:** Python może być łatwo zintegrowany z innymi językami, takimi jak C, C++ czy Java, co pozwala na korzystanie z najlepszych aspektów różnych technologii.
- **Rosnący rynek IoT:** Python jest często stosowany w rozwiązaniach Internetu Rzeczy (IoT) dzięki swojej prostocie i wszechstronności.
- **Wsparcie dla uczenia maszynowego i sztucznej inteligencji:** W ostatnich latach Python stał się dominującym językiem w dziedzinie uczenia maszynowego, dzięki bibliotekom takim jak TensorFlow, Keras czy scikit-learn.
- **Otwarte źródło:** Jako język open-source, Python jest bezpłatny do użytku i ma korzyści wynikające z aktywnej i zaangażowanej społeczności

Dlaczego Python ? Dlaczego Django ?

Django to jeden z najbardziej popularnych frameworków do tworzenia aplikacji internetowych opartych na Pythonie. Oto kilka powodów, dla których warto nauczyć się Django:

- **"Baterie w zestawie"**: Jednym z haseł Django jest "framework z bateriami w zestawie". Oznacza to, że Django dostarcza wiele wbudowanych narzędzi i funkcji, które przyśpieszają rozwój aplikacji, takich jak panele administracyjne, uwierzytelnianie użytkowników czy formularze.
- **Skalowalność**: Django zostało zaprojektowane tak, aby można było tworzyć aplikacje skalowalne, od małych stron internetowych po duże platformy obsługujące miliony użytkowników.
- **Bezpieczeństwo**: Bezpieczeństwo jest jednym z priorytetów Django. Framework ten zawiera wiele zabezpieczeń przed powszechnymi atakami, takimi jak wstrzykiwanie SQL, cross-site scripting (XSS) czy fałszywe żądania między stronami (CSRF).
- **ORM (Object-Relational Mapping)**: Django posiada wbudowany system ORM, który pozwala na intuicyjne zarządzanie bazami danych przy użyciu obiektowego języka programowania, co upraszcza proces tworzenia i zarządzania bazami danych.
- **Wsparcie dla wielu baz danych**: Django wspiera wiele systemów baz danych, takich jak PostgreSQL, MySQL, SQLite czy Oracle.

Dlaczego Python ? Dlaczego Django ?

Django to jeden z najbardziej popularnych frameworków do tworzenia aplikacji internetowych opartych na Pythonie. Oto kilka powodów, dla których warto nauczyć się Django:

- **Społeczność:** Django ma silną i aktywną społeczność, co oznacza dostęp do licznych zasobów, wtyczek, narzędzi oraz wsparcia od innych deweloperów.
- **Dokumentacja:** Django jest znane z doskonałej dokumentacji, co ułatwia naukę i rozwiązywanie problemów.
- **Modularność:** Django zachęca do modularnego i ponownego użycia kodu poprzez system aplikacji, co ułatwia zarządzanie i rozwijanie skomplikowanych projektów.
- **Wszechstronność:** Django może być używane do tworzenia różnych rodzajów aplikacji internetowych, od blogów i stron firmowych po sklepy internetowe czy platformy społecznościowe.
- **Zawodowe możliwości:** Ze względu na popularność i wszechstronność Django, umiejętność programowania w tym frameworku może otworzyć wiele drzwi w dziedzinie rozwoju aplikacji internetowych.

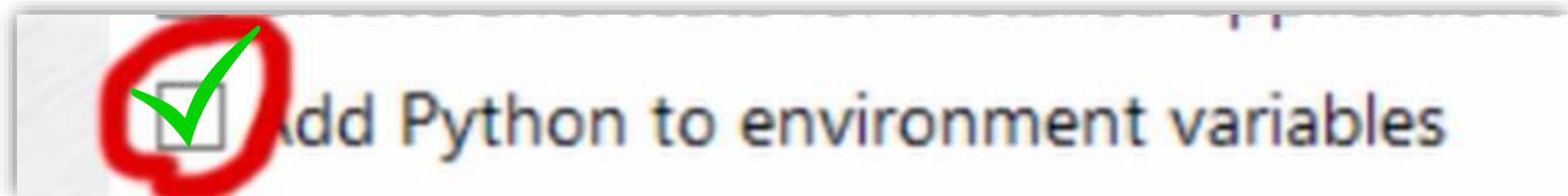
Instalowanie Pythona

<https://www.python.org/>

The screenshot shows the Python.org website's download section. On the left, there's a code snippet demonstrating Python list operations. The main area has a sidebar with links like 'All releases', 'Source code', 'Windows', 'macOS', 'Other Platforms', 'License', and 'Alternative Implementations'. The central part features a large button for 'Python 3.11.5' under the heading 'Download for Windows'. A green circle highlights this button. Below it, a note states 'Note that Python 3.9+ cannot be used on Windows 7 or earlier.' To the right, there's a dark sidebar with text about lists and built-in functions.

Instalowanie Pythona

Terminal - sprawdzanie wersji Pythona



```
Szymon@DESKTOP-3A679F2 MINGW64
$ python --version
Python 3.11.3
```

IDE-zintegrowane środowisko programistyczne (Integrated Development Environment)



IDE-zintegrowane środowisko programistyczne (Integrated Development Environment)

PyCharm Professional

The Python IDE for Professional Developers

[Download](#)[.exe ▾](#)[Free 30-day trial](#)

PyCharm Community Edition

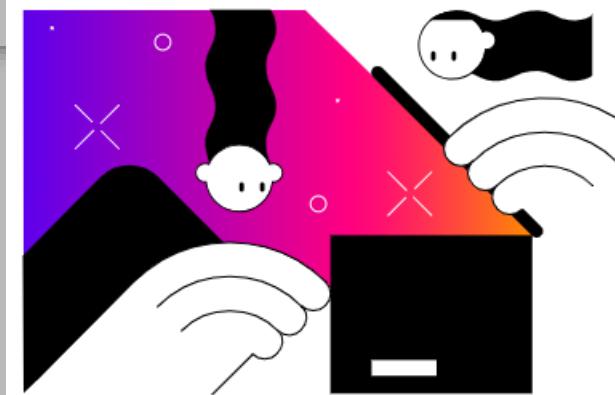
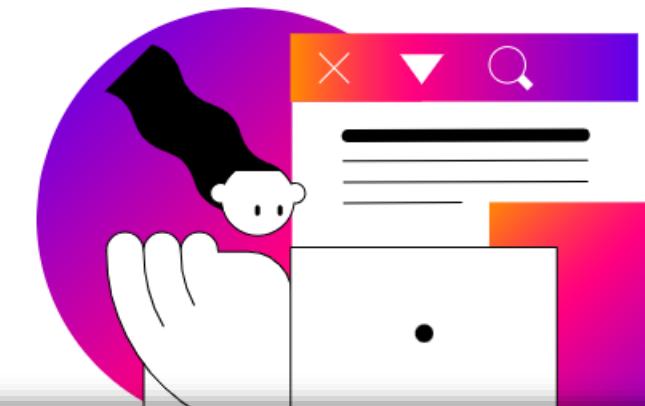
The IDE for Pure Python Development

[Download](#)[.exe ▾](#)[Free, built on open source](#)

IDE-zintegrowane środowisko programistyczne (Integrated Development Environment)

Free Educational Licenses

Learn or teach coding with best-in-class development tools from JetBrains!



Get free access to all developer tools from JetBrains!

[Apply now](#)

IDE-zintegrowane środowisko programistyczne (Integrated Development Environment)

The screenshot shows the Visual Studio Code interface. On the left, there's a dark-themed sidebar with icons for file operations like Open, Save, Find, and Git. Below the sidebar, a large blue button says "Download for Windows" and "Stable Build". Underneath it, another button says "Web, Insiders edition, or other platforms". A note below the buttons states: "By using VS Code, you agree to its license and privacy statement." The main area of the screen is the code editor, which displays a file named "serviceWorker.js". The code is part of a React application and includes imports from "react", "react-dom", and "serviceWorkerRegistration". A specific line of code, `navigator.serviceWorker.ready.then(() => {`, is highlighted with a blue rectangle. To the right of the code editor, there's a terminal window showing the command "node". At the bottom of the screen, there's a status bar with the text "master", "0 0 ▲ 0", "Ln 43, Col 19 Spaces: 2", "UTF-8 LF JavaScript", and some small icons. On the far left edge of the screen, there's a vertical scroll bar.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows

Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.

EXTENSIONS: MARKETPLACE

Python 2019.7.4221 54.9M 4.5
Linting, Debugging (multi-threaded, ...
Microsoft Install

Git 9.8.5 23.1M 5
Supercharge the Git capabilities built...
Eric Amadio Install

C/C++ 0.24.0 23M 3.5
C/C++ IntelliSense, debugging, and ...
Microsoft Install

ESLint 1.9.0 21.9M 4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer Install

Debugger for Ch... 4.11.6 20.6M 4
Debug your JavaScript code in the C...
Microsoft Install

Language Supp... 0.47.0 18.7M 4.5
Java Linting, Intellisense, formatting, ...
Red Hat Install

vscode-icons 8.8.0 17.2M 5
Icons for Visual Studio Code
VSCode Icons Team Install

Vetur 0.21.1 17M 4.5
Vue tooling for VS Code
Pine Wu Install

C# 1.21.0 15.6M 4
C# for Visual Studio Code (powered ...
Microsoft Install

File Edit Selection View Go Debug Terminal Help serviceWorker.js - create-react-app - Visual Studio Code - In...

```
src > JS serviceWorker.js > register > window.addEventListener('load') callback
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
product
productSub
removeSiteSpecificTrackingException
removeWebWideTrackingException
requestMediaKeySystemAccess
sendBeacon
serviceWorker (property) Navigator.serviceWorke...
storage
storeSiteSpecificTrackingException
storeWebWideTrackingException
] userAgent
}
function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      You can now view create-react-app in the browser.
      Local: http://localhost:3000/
      On Your Network: http://10.211.55.3:3000/
      Note that the development build is not optimized.
TERMINAL ... 1: node + ^ x
Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript
```

IDE-zintegrowane środowisko programistyczne (Integrated Development Environment)

Uwaga

Na zajęciach wykorzystywany będzie PyCharm Professional



PyCharm Professional

The Python IDE for Professional Developers

Virtual environment

<https://jaktestowac.pl/lesson/pt1-mk5-s01-l02/> [15.09.2023]

Python Virtual Environment (pot. venv – tego skrótu będziemy używać najczęściej) jest środowiskiem Pythona, które jest **odseparowane i całkowicie niezależne od głównej instalacji Pythona**. Każdy tworzony projekt powinien, zawierać swoje środowisko, dzięki czemu może składać się z unikalnego zestawu pakietów.



Virtual environment

<https://jaktestowac.pl/lesson/pt1-mk5-s01-l02/> [15.09.2023]

Czym jest Python Virtual Environment (venv)?

Python Virtual Environment (potocznie **venv** – tego skrótu będziemy używać najczęściej) jest środowiskiem Pythona, które jest **odseparowane i całkowicie niezależne** od głównej instalacji Pythona. Każdy tworzony przez nas projekt może, a nawet powinien, zawierać swoje środowisko, dzięki czemu może składać się z **unikalnego zestawu pakietów**.

Brzmi zawile? Bardzo upraszczając – **venv** jest **nowym katalogiem** na dysku, w którym znajduje się kopia Pythona. Zaraz po utworzeniu zawiera ona jedynie podstawowe pakiety. Zupełnie tak, jakbyśmy zainstalowali Pythona sami w nowej lokalizacji.

Różnica polega na tym, że **venv** tworzymy za pomocą jednego polecenia i oczywiście potrzebny jest już wcześniej zainstalowany Python. Nie zaśmiecamy sobie w żaden sposób systemu i możemy tworzyć dowolnie dużą liczbę takich środowisk.

W prosty sposób można aktywować **venv** (jak? – zrobimy to już w następnych lekcjach), doinstalować tylko te pakiety, których potrzebujemy, oraz uruchamiać stworzone przez nas skrypty.

Po co stosować venv? Jakie ma zalety?

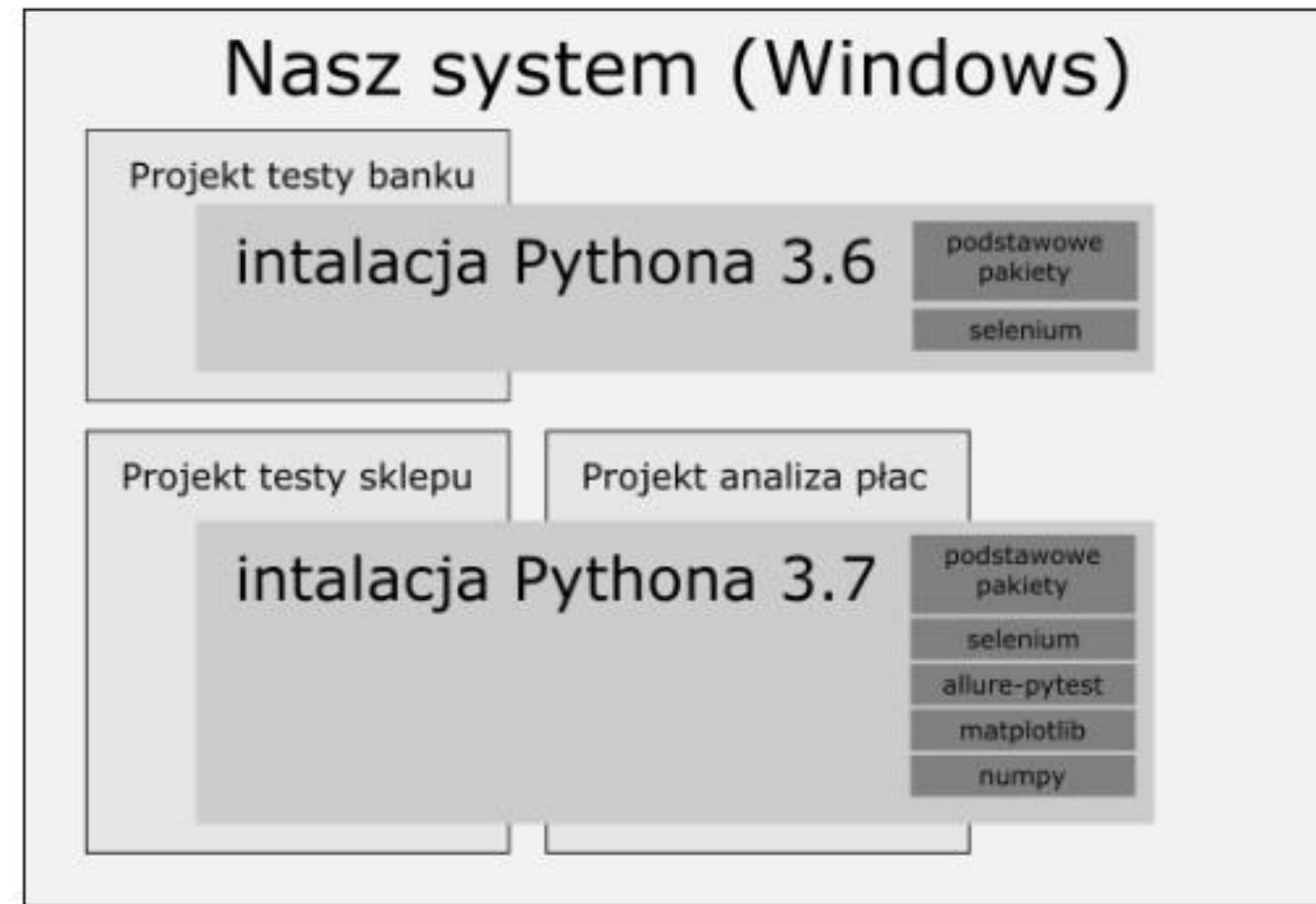
Największą i niewątpliwą zaletą **venv** jest **izolacja środowiska Python** dla danego projektu. Dzięki temu, każde stworzone przez nas środowisko, **może mieć własny zestaw pakietów**, który jest wymagany dla danego projektu.

Często dochodzi do sytuacji, gdy projekty nad którymi pracujemy wymagają różnych wersji pakietów, albo pakietów, które nie są kompatybilne ze sobą. **Venv** pozwala na pełną dowolność w komponowaniu zestawu potrzebnych bibliotek, dzięki czemu możemy mieć jednocześnie środowisko o nazwie **venv1** z pakietami w wersji 1.0 oraz **venv2** z pakietami w wersji 2.0 z możliwością szybkiego i bezproblemowego przełączania się między nimi.

Virtual environment

<https://jaktestowac.pl/lesson/pt1-mk5-s01-l02/> [15.09.2023]

Zobacz sam jak może przykładowo wyglądać stan instalacji Pythona, pakietów i projektów bez zastosowania venv:



Virtual environment

<https://jaktestowac.pl/lesson/pt1-mk5-s01-l02/> [15.09.2023]

I po zastosowaniu `venv`, czy dostrzegasz teraz zalety tego podejścia?:

Nasz system (Windows)

instalacja Pythona 3.6

podstawowe
pakiety

instalacja Pythona 3.7

podstawowe
pakiety

Projekt testy sklepu

`venv`
(Python 3.7)

podstawowe
pakiety

selenium

allure-pytest

Projekt testy banku

`venv`
(Python 3.6)

podstawowe
pakiety

selenium

Projekt analiza płac

`venv`
(Python 3.7)

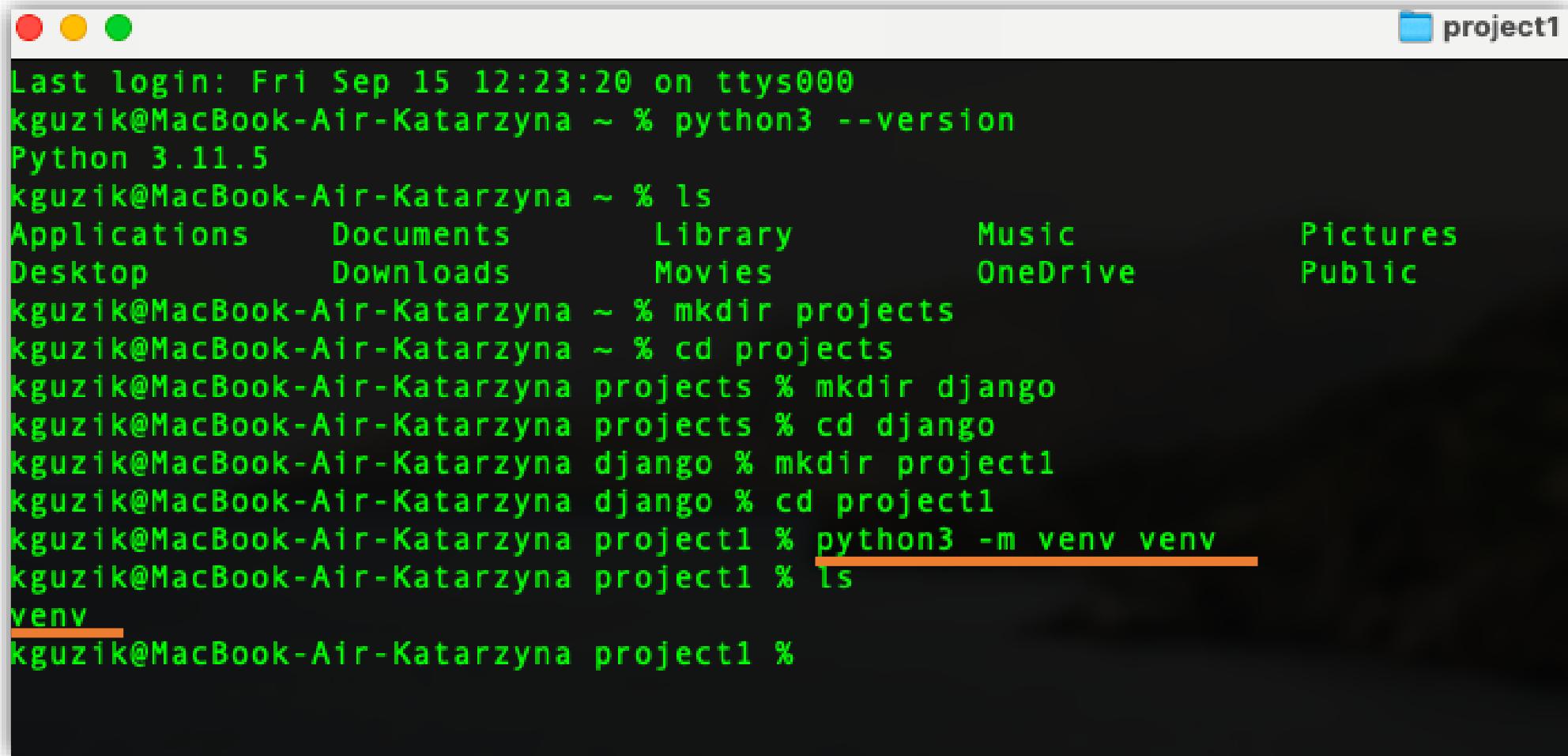
podstawowe
pakiety

matplotlib

numpy

Tworzenie katalogu projektu z wirtualnym środowiskiem (venv)

Tworzenie projektu pod Mac/Unix



The screenshot shows a terminal window on a Mac OS X desktop. The window title is "project1". The terminal content is as follows:

```
Last login: Fri Sep 15 12:23:20 on ttys000
kguzik@MacBook-Air-Katarzyna ~ % python3 --version
Python 3.11.5
kguzik@MacBook-Air-Katarzyna ~ % ls
Applications    Documents    Library    Music    Pictures
Desktop        Downloads    Movies     OneDrive  Public
kguzik@MacBook-Air-Katarzyna ~ % mkdir projects
kguzik@MacBook-Air-Katarzyna ~ % cd projects
kguzik@MacBook-Air-Katarzyna projects % mkdir django
kguzik@MacBook-Air-Katarzyna projects % cd django
kguzik@MacBook-Air-Katarzyna django % mkdir project1
kguzik@MacBook-Air-Katarzyna django % cd project1
kguzik@MacBook-Air-Katarzyna project1 % python3 -m venv venv
kguzik@MacBook-Air-Katarzyna project1 % ls
venv
kguzik@MacBook-Air-Katarzyna project1 %
```

The command `python3 -m venv venv` is highlighted with a red underline.

Tworzenie katalogu projektu z wirtualnym środowiskiem (venv)

Tworzenie projektu pod Mac/Unix

Uwaga
Trzeba aktywować zmienną środowiskową

```
[kguzik@MacBook-Air-Katarzyna project1 % source venv/bin/activate  
(venv) kguzik@MacBook-Air-Katarzyna project1 %
```



Środowisko wirtualne, w którym projekt aktualnie „pracuje”

Uwaga
Dezaktywacja zmiennej środowiskowej

```
(venv) kguzik@MacBook-Air-Katarzyna project1 % deactivate  
kguzik@MacBook-Air-Katarzyna project1 %
```

Tworzenie katalogu projektu z wirtualnym środowiskiem (venv)

Tworzenie projektu pod Windows

```
Szymon@DESKTOP-3A679F2 MINGW64 /e/projects/Django
$ python -m venv venv
_____
Szymon@DESKTOP-3A679F2 MINGW64 /e/projects/Django
$ ls
venv
_____
Szymon@DESKTOP-3A679F2 MINGW64 /e/projects/Django
$ |
```

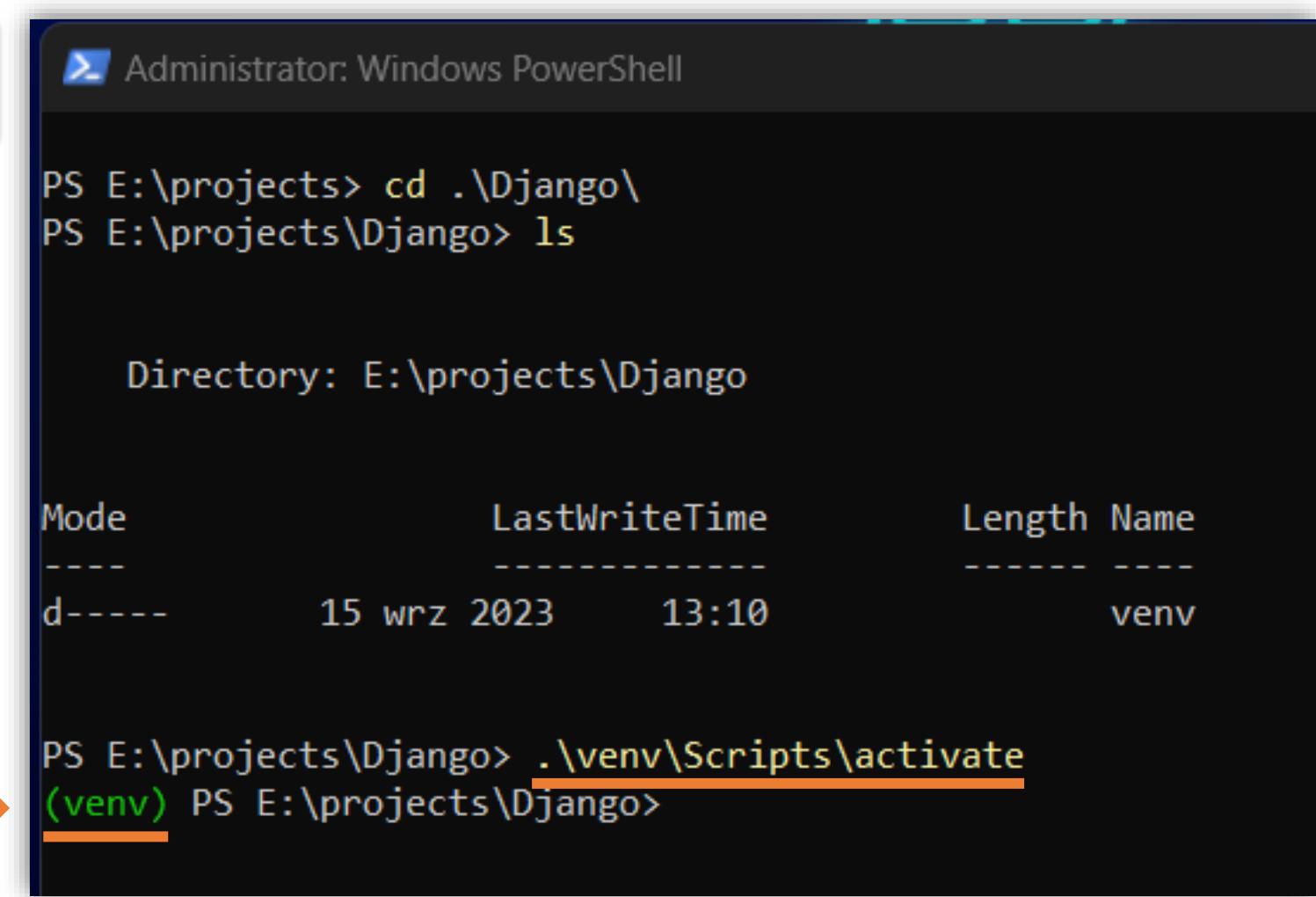
Tworzenie katalogu projektu z wirtualnym środowiskiem (venv)

Tworzenie projektu pod Windows

Uwaga

Trzeba aktywować zmienną środowiskową

Środowisko wirtualne, w którym projekt aktualnie „pracuje”



Administrator: Windows PowerShell

```
PS E:\projects> cd .\Django\  
PS E:\projects\Django> ls  
  
Directory: E:\projects\Django  
  
Mode                LastWriteTime         Length Name  
----                -              -          -  
d-----        15 wrz 2023      13:10   venv  
  
PS E:\projects\Django> .\venv\Scripts\activate  
(venv) PS E:\projects\Django>
```

Tworzenie katalogu projektu z wirtualnym środowiskiem (venv)

Tworzenie projektu pod Windows

Problem Windows

Brak możliwości uruchomienia zaznaczonego polecenia

Rozwiązanie

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine
```

Tworzenie katalogu projektu z wirtualnym środowiskiem (venv)

Tworzenie projektu pod Windows

Uwaga

Dezaktywacja zmiennej środowiskowej

```
(venv) PS E:\projects\ Django> deactivate venv
PS E:\projects\ Django> ■
```

Tworzenie projektu Django

pip install django

```
(venv) PS E:\projects\ Django> pip install django
Collecting django
  Downloading Django-4.2.5-py3-none-any.whl (8.0 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.0/8.0 MB 31.9 MB/s eta 0:00:00
Collecting asgiref<4,>=3.6.0
  Downloading asgiref-3.7.2-py3-none-any.whl (24 kB)
Collecting sqlparse>=0.3.1
  Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━ 41.2/41.2 kB ? eta 0:00:00
Collecting tzdata
  Downloading tzdata-2023.3-py2.py3-none-any.whl (341 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━ 341.8/341.8 kB 20.7 MB/s eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-4.2.5 sqlparse-0.4.4 tzdata-2023.3

[notice] A new release of pip available: 22.3.1 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS E:\projects\ Django>
```

Tworzenie projektu Django

```
django-admin startproject {nazwa} {lokalizacja}
```

```
(venv) PS E:\projects\ Django> django-admin startproject crud_blog .  
(venv) PS E:\projects\ Django> ls
```

Directory: E:\projects\ Django

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	15 wrz 2023	14:06	crud_blog
d----	15 wrz 2023	13:10	venv
-a----	15 wrz 2023	14:06	687 manage.py

Uruchamianie lokalnego serwera

Linux/Mac

```
python3 manage.py runserver
```

Windows

```
python manage.py runserver
```

```
(venv) PS E:\projects\ Django> django-admin startproject crud_blog .  
(venv) PS E:\projects\ Django> ls
```

Mode	LastWriteTime	Length	Name
d----	15 wrz 2023	14:06	
d----	15 wrz 2023	13:10	
-a---	15 wrz 2023	14:06	687 manage.py

Uruchamianie lokalnego serwera

Linux/Mac/Windows

```
(venv) PS E:\projects\ Django> python manage.py runserver
Watching for file changes with StaticReloader
Performing system checks...

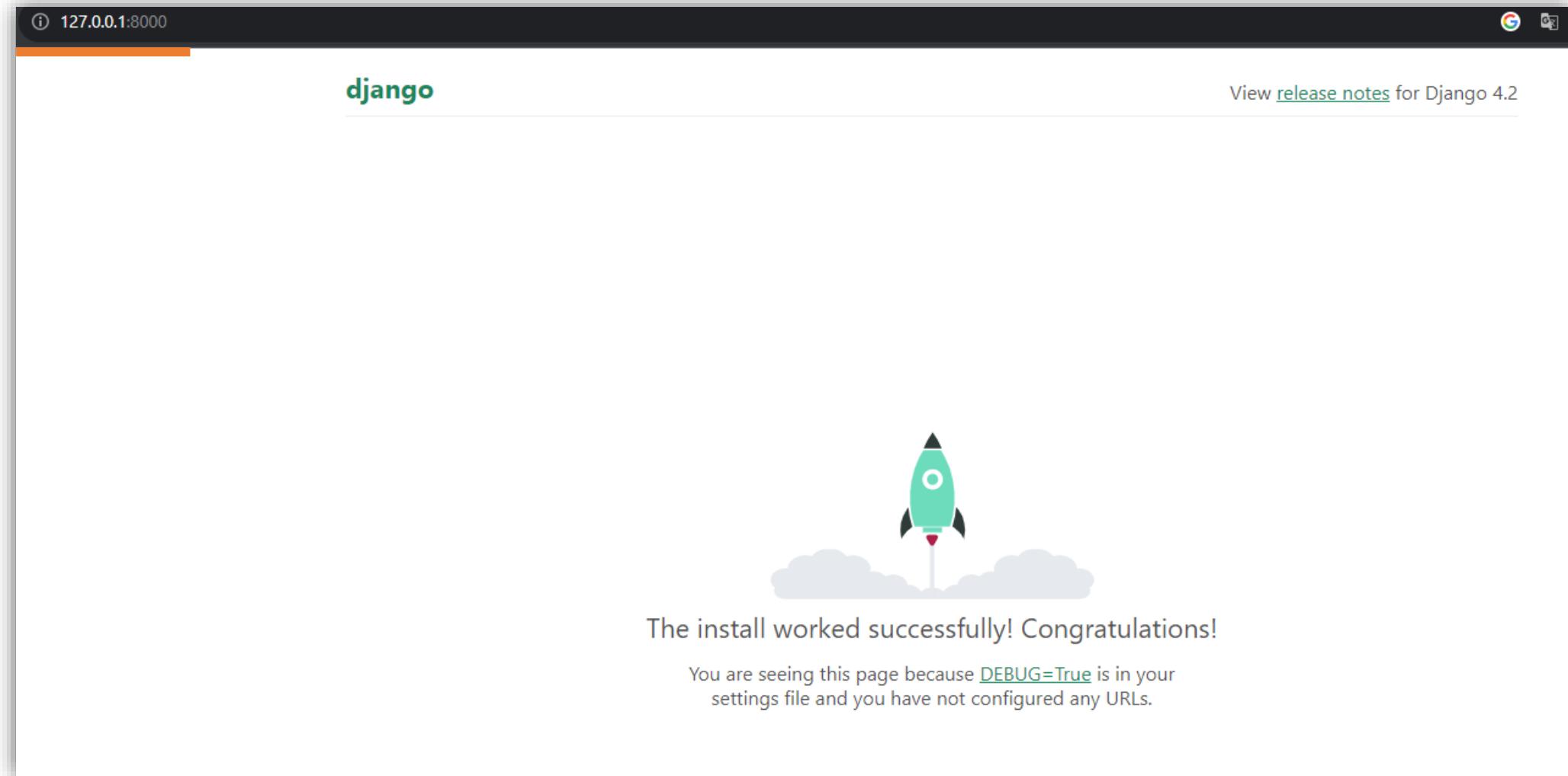
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly unless you run 'python manage.py migrate' to apply them.
September 15, 2023 - 14:19:23
Django version 4.2.5, using settings 'crud_blog.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[15/Sep/2023 14:19:38] "GET / HTTP/1.1" 200 10664
Not Found: /favicon.ico
[15/Sep/2023 14:19:38] "GET /favicon.ico HTTP/1.1" 404 2113
```

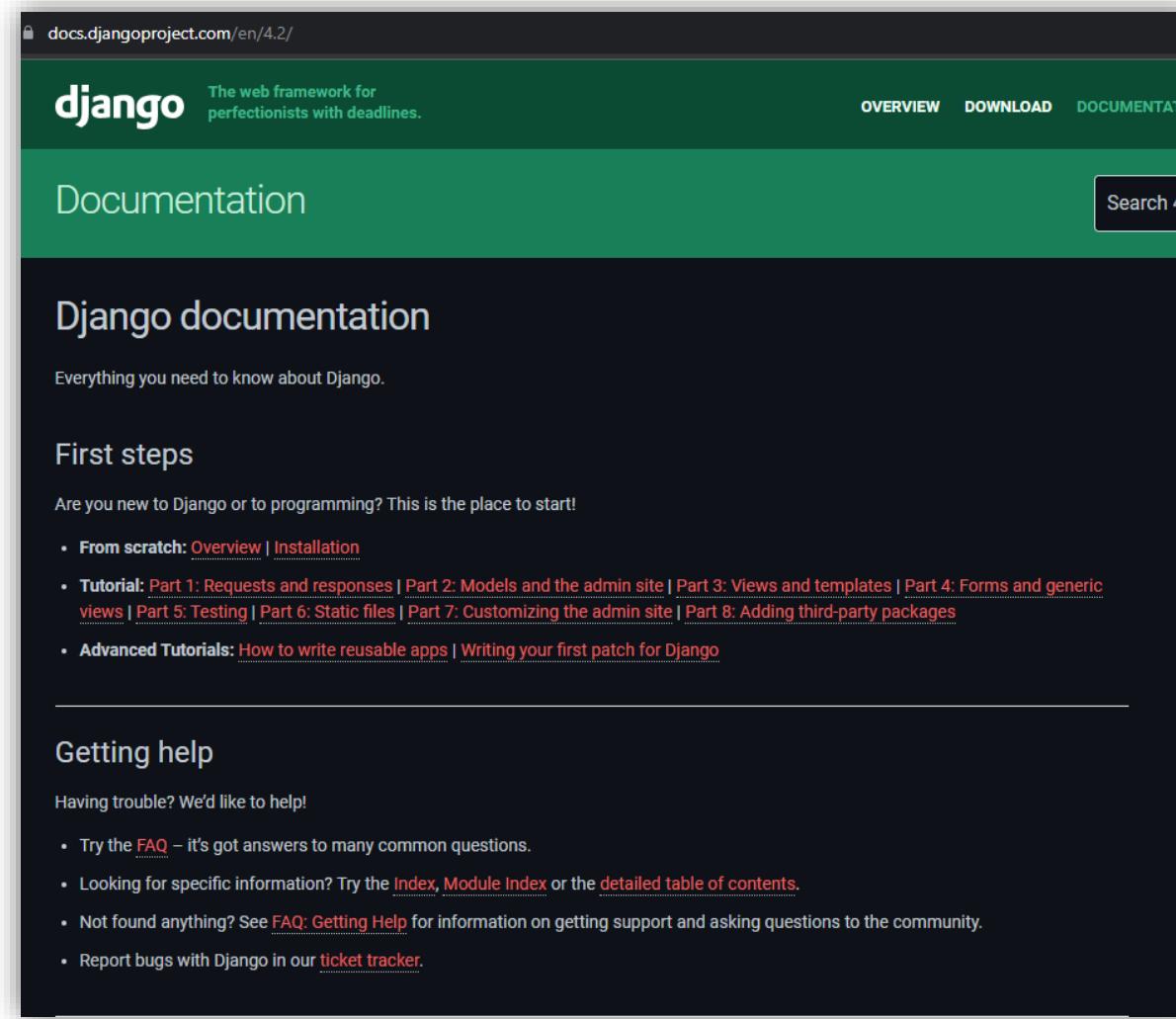
Uruchamianie lokalnego serwera

Linux/Mac/Windows



Dokumentacja Django

<https://docs.djangoproject.com/en/4.2/>



The screenshot shows the Django documentation homepage at <https://docs.djangoproject.com/en/4.2/>. The page has a dark green header with the Django logo and tagline "The web framework for perfectionists with deadlines." The navigation bar includes links for OVERVIEW, DOWNLOAD, and DOCUMENTATION. A search bar on the right is labeled "Search 4.2". The main content area features a large title "Django documentation" and a subtitle "Everything you need to know about Django." Below this, a section titled "First steps" provides links to various tutorials and advanced topics. Another section titled "Getting help" offers resources for troubleshooting and community support.

docs.djangoproject.com/en/4.2/

django The web framework for perfectionists with deadlines.

OVERVIEW DOWNLOAD DOCUMENTATION

Documentation

Search 4.2

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 patch for Django](#)

Getting help

Having trouble? We'd like to help!

- Try the [FAQ](#) – it's got answers to many common questions.
- Looking for specific information? Try the [Index](#), [Module Index](#) or the [detailed table of contents](#).
- Not found anything? See [FAQ: Getting Help](#) for information on getting support and asking questions to the community.
- Report bugs with Django in our [ticket tracker](#).

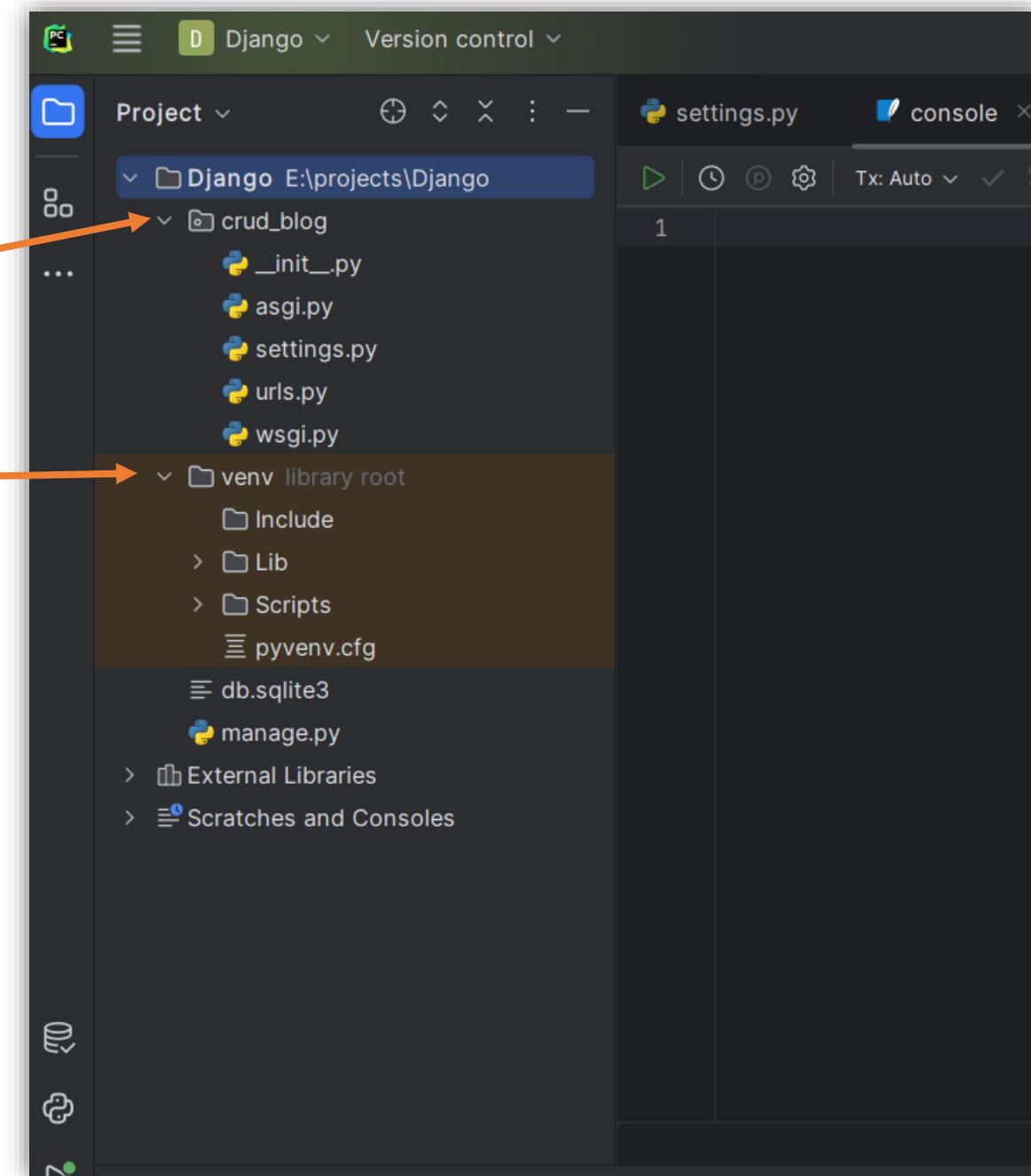
Omówienie plików

Uwaga

Projekt należy otworzyć w IDE

Aplikacja

Zmienna środowiskowa



Tworzenie nowej aplikacji

django-admin startapp {nazwa_aplikacji}

```
(venv) PS E:\projects\ Django> django-admin startapp crud_blog_web
(venv) PS E:\projects\ Django>
```

Aplikacje domyślne oraz „dedykowana”

The screenshot shows a PyCharm interface with a Django project structure on the left and the `settings.py` file content on the right.

Project Tree:

- Django E:\projects\ Django
 - crud_blog
 - `__init__.py`
 - `asgi.py`
 - `settings.py` (highlighted)
 - `urls.py`
 - `wsgi.py`
 - crud_blog_web (highlighted)
 - venv library root
 - db.sqlite3
 - manage.py

settings.py Content:

```
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'crud_blog_web',
41 ]
```

Annotations:

 - An orange arrow points from the highlighted `crud_blog_web` application in the project tree to the corresponding entry in the `INSTALLED_APPS` list.
 - A bracket on the right side of the code block groups the first seven entries under the heading "Aplikacje domyślne".
 - A bracket at the bottom groups the `crud_blog_web` entry under the heading "Aplikacje dedykowana".

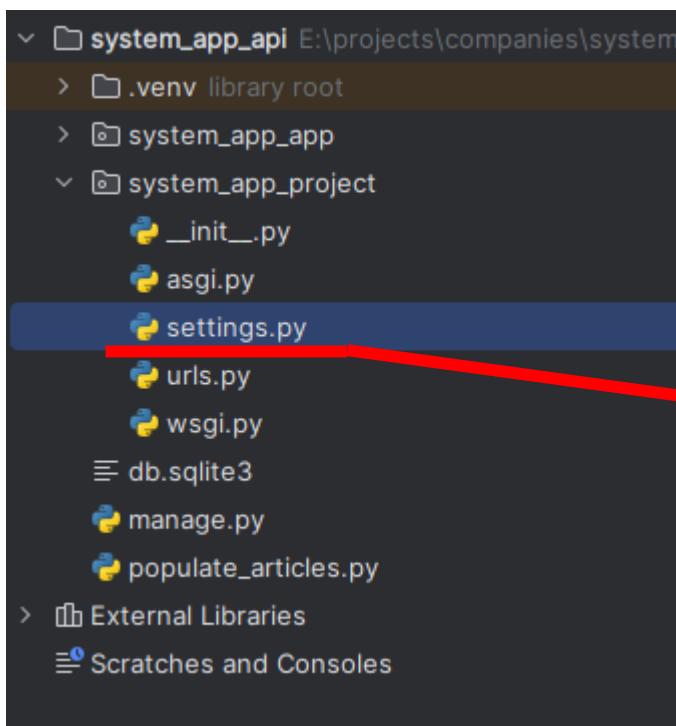
Konfiguracja PostgreSQL-a

```
pip install psycopg2
```

VS

```
pip install psycopg
```

Nowa generacja sterownika Psycopg, obecnie znana jako psycopg (psycopg3). W poleceniu nie ma 3 😊



```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'moja_baza_danych',
        'USER': 'moj_uzycownik',
        'PASSWORD': 'moje_haslo',
        'HOST': 'localhost', # Pusty, jeśli baza danych działa
        'PORT': '',          # Domyslnie 5432
    }
}
```

Migracje

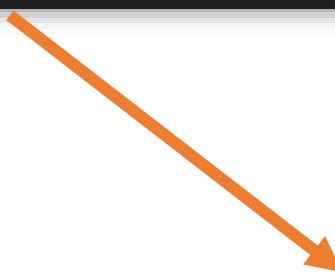
Migracje w Django to mechanizm, który pozwala na wprowadzanie zmian w schemacie bazy danych (tzn. strukturze tabel, indeksach, relacjach itp.) w sposób kontrolowany i wersjonowany. Dzięki temu, kiedy wprowadzasz zmiany w modelach aplikacji Django, nie musisz ręcznie modyfikować bazy danych ani pisać surowych skryptów SQL do dokonywania tych zmian.



Migracje

Po uruchomieniu projektu w terminalu widoczna jest informacja dot. migracji

```
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.  
Run 'python manage.py migrate' to apply them.
```



You have 18 unapplied migration(s).
Run 'python manage.py migrate' to a
September 15, 2023 - 15:33:00

Migracje

Wykonywanie migracji

```
(venv) PS E:\projects\ Django> python manage.py migrate
```

```
Operations to perform:
```

```
  Apply all migrations: admin, auth, contenttypes, sessions
```

```
Running migrations:
```

```
  Applying contenttypes.0001_initial... 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
```

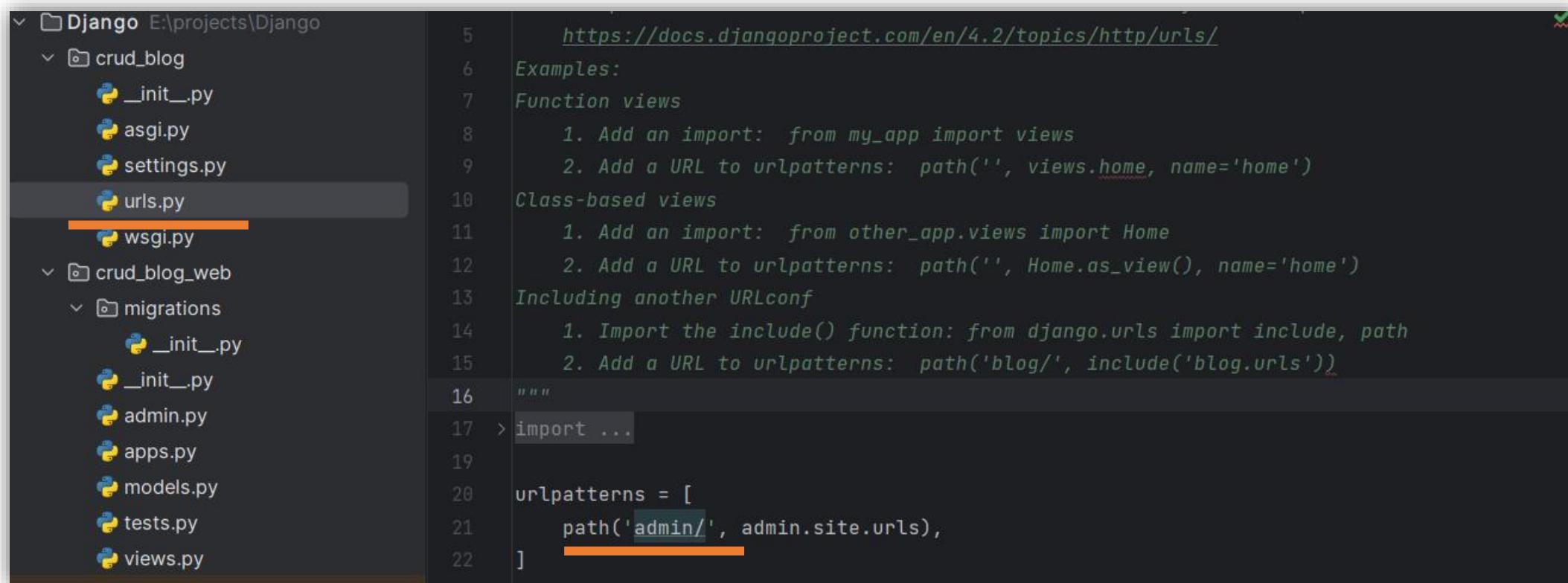
```
  Applying auth.0011_update_proxy_permissions... OK
```

```
  Applying auth.0012_alter_user_first_name_max_length... OK
```

```
  Applying sessions.0001_initial... OK
```



Tworzenie użytkownika superuser oraz logowanie do admina



The screenshot shows a code editor with a file tree on the left and a code editor window on the right.

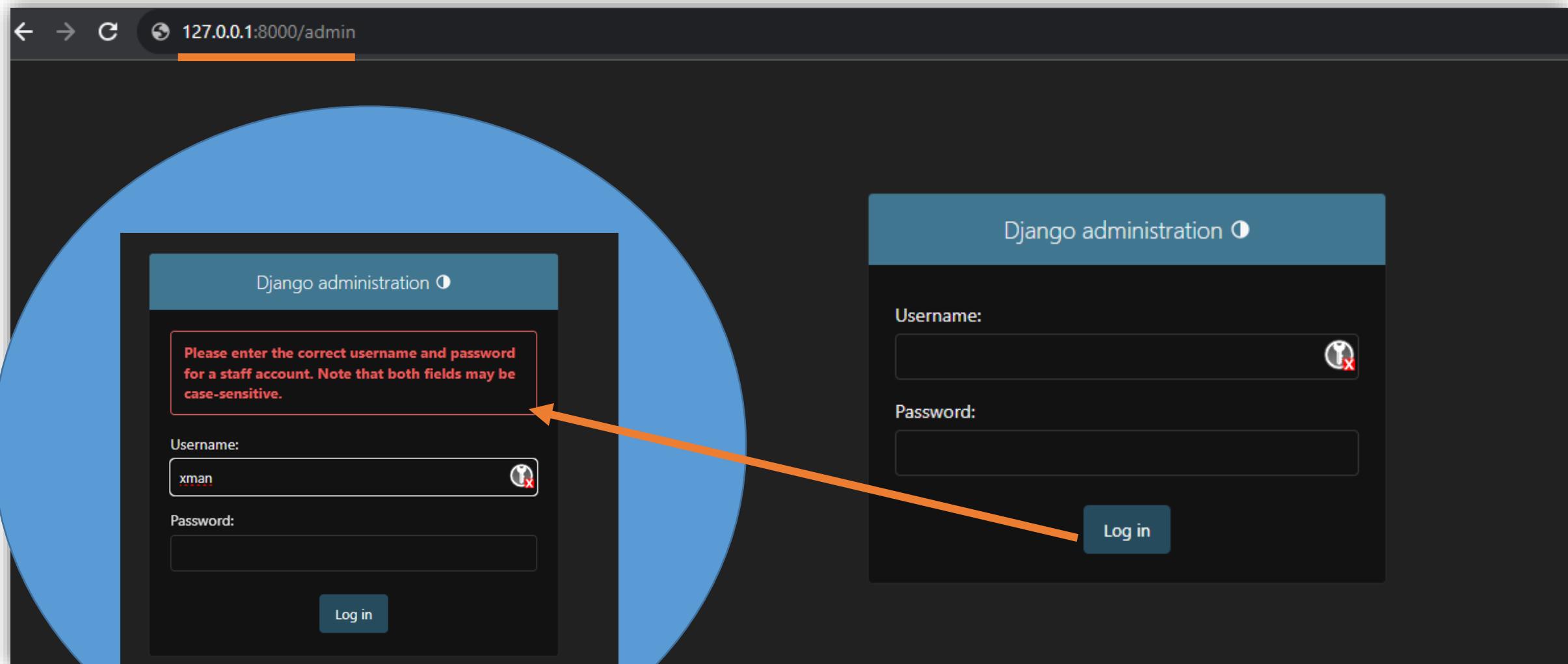
File Tree:

- Django E:\projects\ Django
- crud_blog
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
- crud_blog_web
 - migrations
 - __init__.py
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - views.py

Code Editor Content (urls.py):

```
5      https://docs.djangoproject.com/en/4.2/topics/http/urls/
6  Examples:
7  Function views
8  1. Add an import: from my_app import views
9  2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11 1. Add an import: from other_app.views import Home
12 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14 1. Import the include() function: from django.urls import include, path
15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 > import ...
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22 ]
```

Tworzenie użytkownika superuser oraz logowanie do admina



Tworzenie użytkownika superuser oraz logowanie do admina

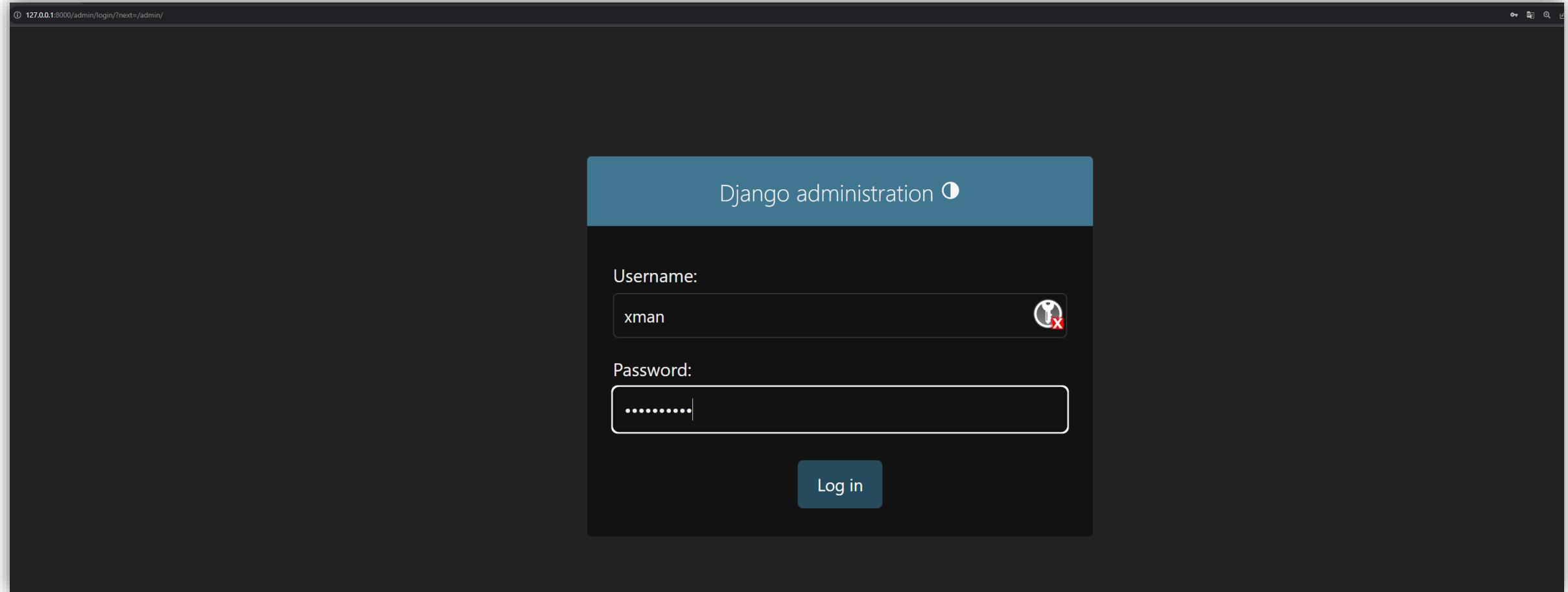
```
(venv) PS E:\projects\ Django> python manage.py createsuperuser
Username (leave blank to use 'szymon'): xman
Email address: szymon.guzik@gdansk.merito.pl
Password:
Password (again):
Superuser created successfully.
```

Username: xman

Email: szymon.guzik@gdansk.merito.pl

Password: qwerty2023

Tworzenie użytkownika superuser oraz logowanie do admina



Tworzenie użytkownika superuser oraz logowanie do admina

The screenshot shows the Django administration interface at the URL `127.0.0.1:8000/admin/`. The top navigation bar includes links for `LOG OUT`, `CHANGE PASSWORD`, and `VIEW SITE`. The main content area is titled "Site administration". On the left, there is a sidebar with "AUTHENTICATION AND AUTHORIZATION" heading. It contains two sections: "Groups" and "Users". Under "Groups", there are "Add" and "Change" buttons. Under "Users", there are also "Add" and "Change" buttons. To the right of the sidebar, there are two panels: "Recent actions" which is empty, and "My actions" which also says "None available".

Modele, migracje, rejestracja

W frameworku Django, modele są kluczowym elementem, który pozwala na **definiowanie struktury bazy danych oraz zachowania związane z danymi**. Modele w Django to w zasadzie klasy Pythona, które definiują i reprezentują tabele w bazie danych. Modele dostarczają również metody i narzędzia do tworzenia, odpytywania, modyfikowania i usuwania rekordów w bazie danych



Modele, migracje, rejestracja

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a Django project structure:

- Django E:\projects\ Django
- crud_blog
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
- crud_blog_web
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py

The code editor shows the contents of the `models.py` file in the `migrations` directory of the `crud_blog` app. The file contains the following Python code:

```
1 from django.db import models
2
3 # Create your models here.
4 class Article(models.Model):
5     title = models.CharField(max_length=64)
6
7
```

Modele, migracje, rejestracja

```
(venv) PS E:\projects\ Django> python .\manage.py makemigrations
Migrations for 'crud_blog_web':
crud_blog_web\migrations\0001_initial.py
- Create model Article
```

Modele, migracje, rejestracja

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a Django project structure:

- Django E:\projects\ Django
- crud_blog
- __init__.py
- asgi.py
- settings.py
- urls.py
- wsgi.py
- crud_blog_web
- migrations
- 0001_initial.py
- __init__.py
- __init__.py
- admin.py
- apps.py
- models.py
- tests.py
- views.py
- venv library root
- db.sqlite3
- manage.py

The code editor shows the content of `0001_initial.py`:

```
# Generated by Django 4.2.5 on 2023-09-15 15:25
from django.db import migrations, models

class Migration(migrations.Migration):
    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Article',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False)),
                ('title', models.CharField(max_length=64)),
            ],
        ),
    ]
}
```

The file `0001_initial.py` is highlighted with a yellow background. Lines 7 and 8 are highlighted with orange, and the entire class definition is highlighted with a larger orange area.

Modele, migracje, rejestracja

Uwaga

W celu wykonania migracji należy ponownie użyć komendy

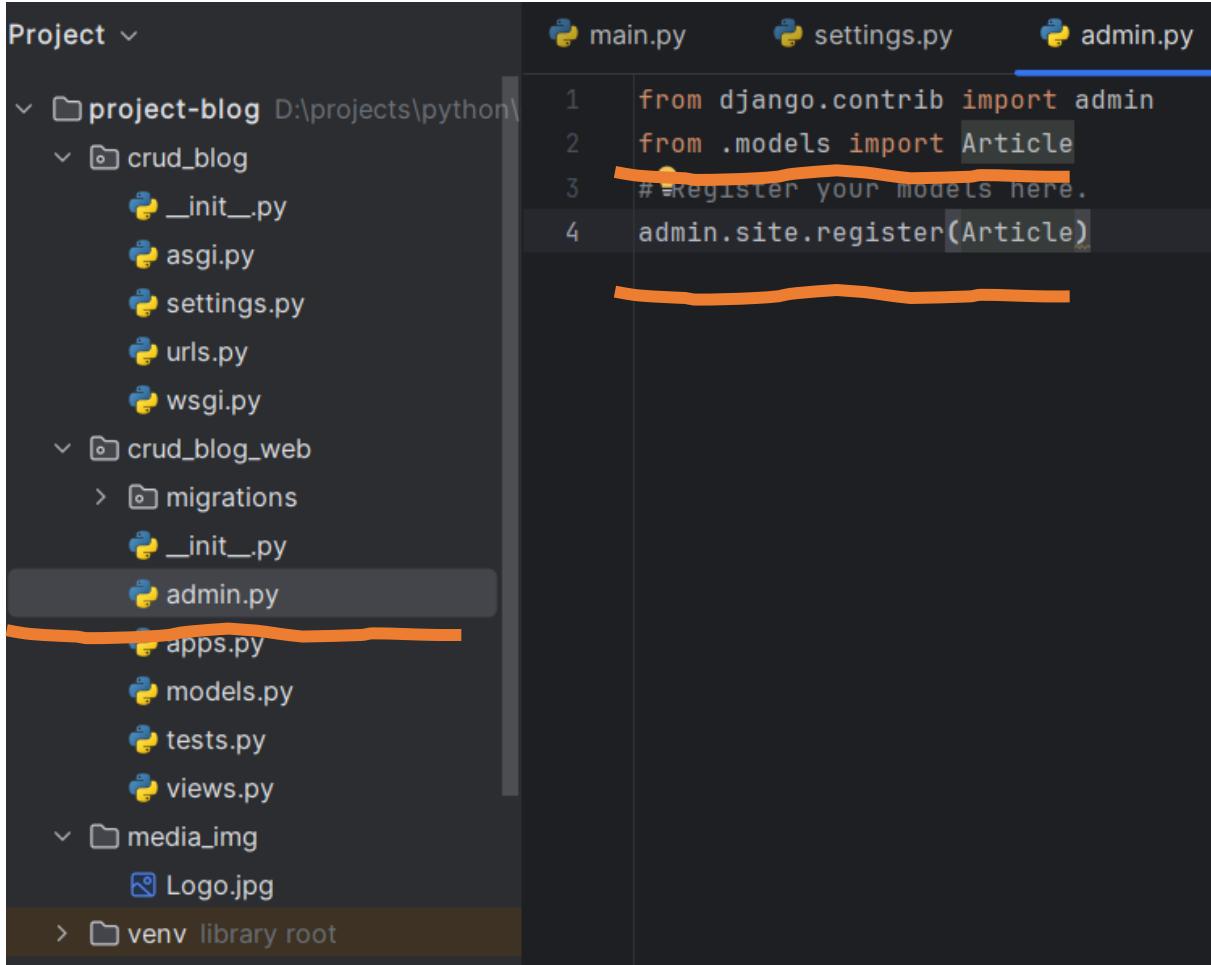
```
(venv) PS E:\projects\ Django> python manage.py migrate
```

```
Migrations for 'crud_blog_web':  
  crud_blog_web\migrations\0001_initial.py  
    - Create model Article  
(venv) PS E:\projects\ Django> python manage.py migrate  
Operations to perform:  
  Apply all migrations: admin, auth, contenttypes, crud_blog_web, sessions  
Running migrations:  
  Applying crud_blog_web.0001_initial... OK
```

Modele, migracje, rejestracja

Uwaga

Rejestracja modelu pozwala między innymi na zarządzanie tabelami z poziomu panelu admina



```
Project ▾
  ✓ project-blog D:\projects\python\project-blog
    ✓ crud_blog
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
    ✓ crud_blog_web
      > migrations
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        views.py
    ✓ media_img
      Logo.jpg
    > venv library root

main.py      settings.py      admin.py >
1  from django.contrib import admin
2  from .models import Article
3  # Register your models here.
4  admin.site.register(Article)
```

Modele, migracje, rejestracja

Uwaga

Rejestracja modelu pozwala między innymi na zarządzanie tabelami z poziomu panelu admina

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

CRUD_BLOG_WEB

Articles + Add

Change user

xman

Username: xman

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: algorithm: pbkdf2_sha256 iterations: 600000 salt: anU4Lk***** hash: +DrcaW*****

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

Model fields

<https://docs.djangoproject.com/en/4.2/ref/models/fields/>

django The web framework for perfectionists with deadlines.

OVERVIEW DOWNLOAD DOCUMENTATION NEWS COMMUNITY CODE ISSUES ABOUT ▾ DONATE ⓘ

Documentation

Model field reference

This document contains all the API references of [Field](#) including the [field options](#) and [field types](#) Django offers.

See also
If the built-in fields don't do the trick, you can try [django-localflavor \(documentation\)](#), which contains assorted pieces of code that are useful for particular countries and cultures.
Also, you can easily [write your own custom model fields](#).

Note
Technically, these models are defined in [django.db.models.fields](#), but for convenience they're imported into [django.db.models](#); the standard convention is to use `from django.db import models` and refer to fields as `models.<Foo>Field`.

Field options

The following arguments are available to all field types. All are optional.

Search 4.2 documentation (Ctrl + K) ⚡

Support Django!

 Martin De Wulf donated to the Django Software Foundation to support Django development. [Donate today!](#)

Contents

- [Model field reference](#)
 - [Field options](#)
 - [null](#)
 - [blank](#)
 - [choices](#)
 - [Enumeration types](#)
 - [db_column](#)
 - [db_comment](#)
 - [db_index](#)

Model fields

Aktualizacja modelu

```
from django.db import models

# Create your models here.

2 usages

class Article(models.Model):
    # pole 150 znaków
    # pole nie może być puste
    # Mogę stworzyć inny artykuł o tym samym tytule
    title = models.CharField(max_length=150, blank=False, unique=False)
    # Jeżeli nikt nie poda roku powstania artykułu to pole zostanie uszupełnione rokiem 2023
    year = models.PositiveSmallIntegerField(default=2023)
```

```
(venv) PS E:\projects\Django> python .\manage.py makemigrations
```

```
(venv) PS E:\projects\Django> python manage.py migrate
```

Model fields

Aktualizacja modelu

```
from django.db import models

# Create your models here.

2 usages

class Article(models.Model):
    # pole 150 znaków
    # pole nie może być puste
    # Mogę stworzyć inny artykuł o tym samym tytule
    title = models.CharField(max_length=150, blank=False, unique=False)
    # Jeżeli nikt nie poda roku powstania artykułu to pole zostanie uszupełnione rokiem 2023
    year = models.PositiveSmallIntegerField(default=2023)
```

```
(venv) PS E:\projects\Django> python .\manage.py makemigrations
```

```
(venv) PS E:\projects\Django> python manage.py migrate
```

Model fields

The screenshot shows the Django administration interface for adding a new article. The top navigation bar includes back, forward, and search icons, along with the URL 127.0.0.1:8000/admin/crud_blog_web/article/add/. The main title is "Django administration" with a subtitle "Home > Crud_Blog_Web > Articles > Add article". On the left, a sidebar lists "AUTHENTICATION AND AUTHORIZATION" (Groups, Users) and "CRUD_BLOG_WEB" (Articles). The "Articles" section has a "+ Add" button. The right panel is titled "Add article" and contains two form fields: "Title:" (empty input field) and "Year:" (input field containing "2023"). At the bottom are three buttons: "SAVE", "Save and add another", and "Save and continue editing".

← → ⌂ ⓘ 127.0.0.1:8000/admin/crud_blog_web/article/add/

Django administration

Home > Crud_Blog_Web > Articles > Add article

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

CRUD_BLOG_WEB

Articles + Add

Add article

Title:

Year: 2023

SAVE Save and add another Save and continue editing

Rzutowanie pobieranych danych

```
1  from django.db import models
2
3
4  # Create your models here.
5  # 2 usages
6
7  class Article(models.Model):
8      # pole 150 znaków
9      # pole nie może być puste
10     # Mogę stworzyć inny artykuł o tym samym tytule
11     title = models.CharField(max_length=150, blank=False, unique=False)
12     # Jeżeli nikt nie poda roku powstania artykułu to pole zostanie uzupełnione rokiem 2023
13     year = models.PositiveSmallIntegerField(default=2023)
14
15
16     def __str__(self):
17         return self.title
```

Rzutowanie pobieranych danych

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

CRUD_BLOG_WEB

CRUD_BLOG_WEB	
Articles	+ Add

Select article to change

Action: ----- Go 0 of 1 selected

ARTICLE

Article object (1)

1 article

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

CRUD_BLOG_WEB

CRUD_BLOG_WEB	
Articles	+ Add

Select article to change

Action: ----- Go 0 of 1 selected

ARTICLE

Uniwersytet Merito

1 article

Od tego momentu samodzielnie sprawdzacie wyniki kodowanie na projekcie ☺

Kod będzie realizowany na różnych projektach i aplikacjach

Należy go analogicznie dostosowywać
Dzięki temu unikniemy bezmyślnego
przepisywania



Zmiana nazwy aplikacji w panelu admina

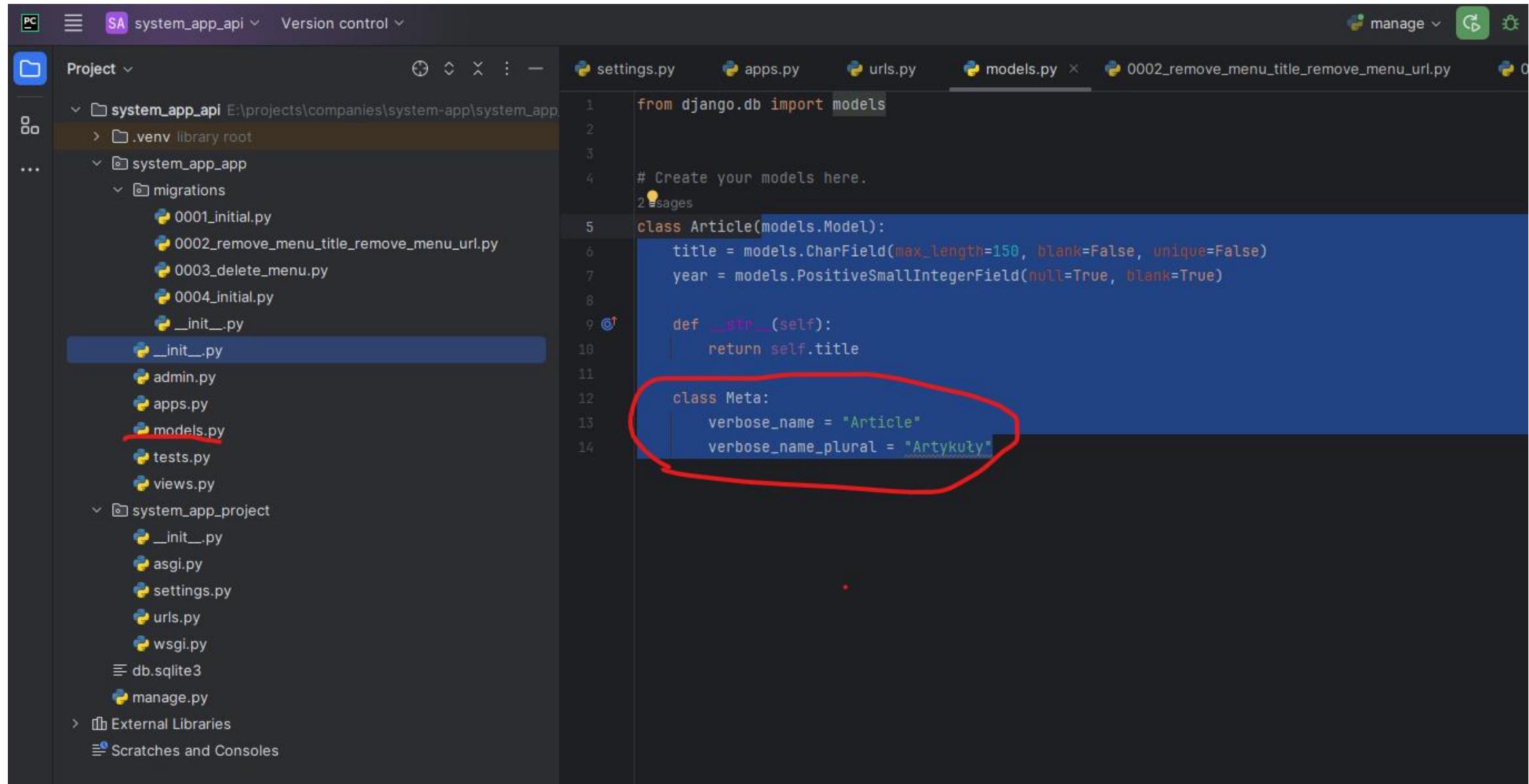
The screenshot shows a PyCharm IDE interface with the following details:

- Project View:** Shows the project structure under "system_app_api". The "apps.py" file in the "system_app_app/migrations" directory is selected.
- Code Editor:** Displays the content of the "apps.py" file. The code defines a AppConfig class named "SystemApp AppConfig". The "verbose_name" attribute is set to "System App Panel".
- Annotations:** A red underline highlights the "verbose_name" line, and a red bracket groups the entire line.
- Status Bar:** Shows "SystemApp AppConfig" at the bottom.

```
from django.apps import AppConfig

class SystemApp AppConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'system_app_app'
    verbose_name = 'System App Panel'
```

Zmiana nazwy tabeli po stronie panelu django



The screenshot shows a PyCharm interface with the following details:

- Project Tree:** On the left, the project structure is shown under "system_app_api". It includes ".venv", "library root", "system_app_app" (with "migrations" containing "0001_initial.py", "0002_remove_menu_title_remove_menu_url.py", "0003_delete_menu.py", "0004_initial.py", and "models.py"), and "system_app_project" (with "admin.py", "apps.py", "models.py", "tests.py", and "views.py").
- Code Editor:** The "models.py" file is open in the editor. The code defines a "Article" model with fields "title" and "year", and a "__str__" method. A "Meta" class is defined at the bottom with "verbose_name" set to "Article" and "verbose_name_plural" set to "Artykuły".
- Annotations:** A red oval highlights the "Meta" class definition.

```
from django.db import models

# Create your models here.

class Article(models.Model):
    title = models.CharField(max_length=150, blank=False, unique=True)
    year = models.PositiveSmallIntegerField(null=True, blank=True)

    def __str__(self):
        return self.title

class Meta:
    verbose_name = "Article"
    verbose_name_plural = "Artykuły"
```

Masowe tworzenie danych testowych

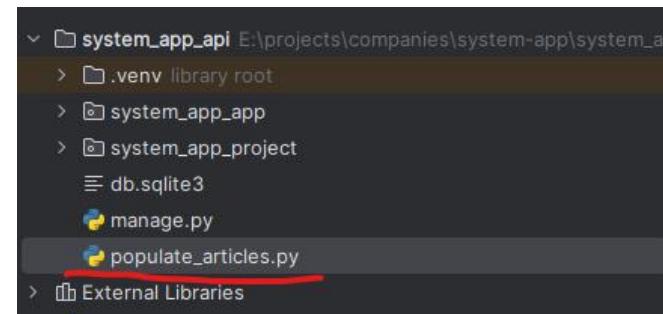
1

Instalowanie Faker

```
pip install Faker
```

Stworzenie pliku
oraz dodanie kodu
do iteracji
`insert`
danych

2



```
1 import os
2 import django
3 from faker import Faker
4 import random
5
6 # Konfiguracja środowiska Django
7 os.environ.setdefault(key='DJANGO_SETTINGS_MODULE', value='system_app_project.settings')
8 django.setup()
9
10 from system_app_app.models import Article # Zmień 'your_app' na nazwę Twojej aplikacji
11
12
13 def populate(N=100): # N określa liczbę rekordów do stworzenia
14     fake = Faker()
15
16     for _ in range(N):
17         title = fake.sentence(nb_words=6)
18         content = fake.text(max_nb_chars=2000)
19         year = random.randint(a: 1990, b: 2021)
20
21         # Tworzenie nowego rekordu Article
22         article = Article(title=title, content=content, year=year)
23         article.save()
24
25
26 if __name__ == '__main__':
27     print("Populating the databases...Please Wait")
28     populate(100) # Wpisz liczbę artykułów, które chcesz wygenerować
29     print('Populating Complete')
30
```

Wywołanie pliku

```
(.venv) PS E:\projects\companies\system-app\system_app_api> python populate_articles.py
Populating the databases...Please Wait
Populating Complete
(.venv) PS E:\projects\companies\system-app\system_app_api>
```

Pole ImageField

Instalacja biblioteki Pillow

```
pip install Pillow
```

Dodanie pola do modelu

```
class Article(models.Model):
    title = models.CharField(max_length=150, blank=False, unique=False)
    content = models.TextField(blank=False, default='')
    year = models.PositiveSmallIntegerField(null=True, blank=True)
    image = models.ImageField(upload_to='media', blank=True, null=True)
```

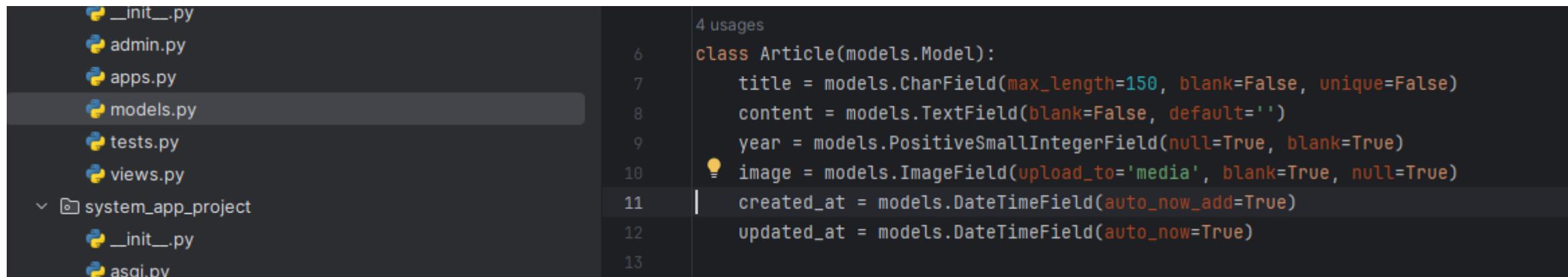
Wykonanie poleceń

```
python manage.py makemigrations
```

```
python manage.py migrate
```

Pola typu timestamps

Założenie - nie mamy wpisów w tabeli (PUSTA TABELA)



```
__init__.py
admin.py
apps.py
models.py
tests.py
views.py
system_app_project
__init__.py
asgi.py

4 usages
6 class Article(models.Model):
7     title = models.CharField(max_length=150, blank=False, unique=False)
8     content = models.TextField(blank=False, default='')
9     year = models.PositiveSmallIntegerField(null=True, blank=True)
10    image = models.ImageField(upload_to='media', blank=True, null=True)
11    created_at = models.DateTimeField(auto_now_add=True)
12    updated_at = models.DateTimeField(auto_now=True)
```

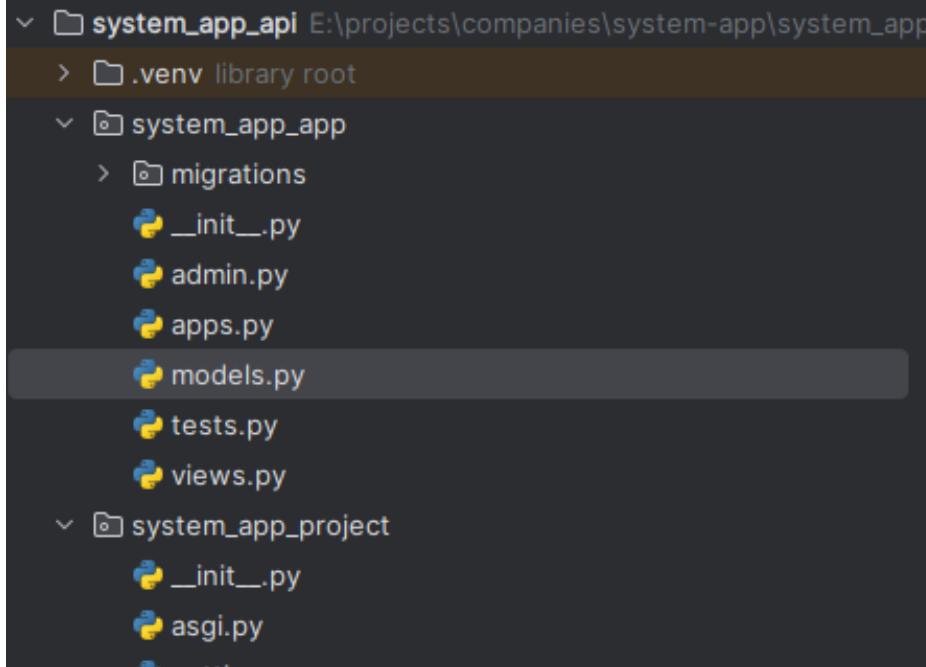
Wykonanie poleceń

python manage.py makemigrations

python manage.py migrate

Pola typu timestamps

Założenie – mamy wpisów w tabeli (NIE PUSTA TABELA) – PART 1



```
1 from django.db import models
2 from django.utils import timezone
3
4 # Create your models here.
5 usages
6 class Article(models.Model):
7     title = models.CharField(max_length=150, blank=False, unique=False)
8     content = models.TextField(blank=False, default='')
9     year = models.PositiveSmallIntegerField(null=True, blank=True)
10    image = models.ImageField(upload_to='media', blank=True, null=True)
11    created_at = models.DateTimeField(default=timezone.now)
12    updated_at = models.DateTimeField(default=timezone.now)
```

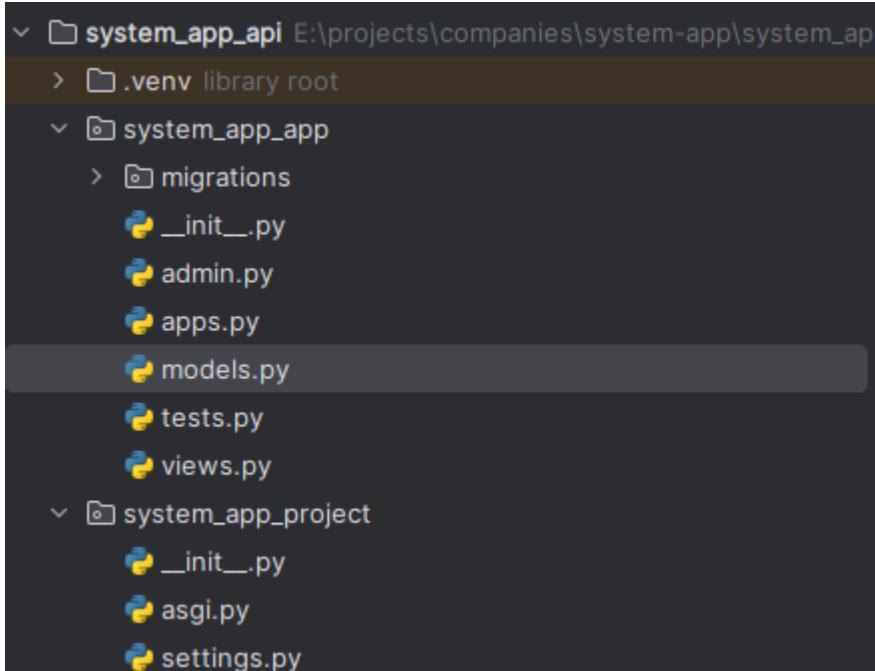
Wykonanie polecień

python manage.py makemigrations

python manage.py migrate

Pola typu timestamps

Założenie – mamy wpisów w tabeli (NIE PUSTA TABELA) – PART 2



```
1  from django.db import models
2  from django.utils import timezone
3
4
5  # Create your models here.
6  # 4 usages
7  class Article(models.Model):
8      title = models.CharField(max_length=150, blank=False, unique=False)
9      content = models.TextField(blank=False, default=' ')
10     year = models.PositiveSmallIntegerField(null=True, blank=True)
11     image = models.ImageField(upload_to='media', blank=True, null=True)
12     # created_at = models.DateTimeField(default=timezone.now)
13     # updated_at = models.DateTimeField(default=timezone.now)
14     created_at = models.DateTimeField(auto_now_add=True)
15     updated_at = models.DateTimeField(auto_now=True)
```

Wykonanie poleceń

python manage.py makemigrations

python manage.py migrate

Zdefiniowanie metody reprezentacji obiektu oraz jej wywołanie

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure:

- system_app_app:
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - views.py
- system_app_project
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
- db.sqlite3
- manage.py
- populate_articles.py
- External Libraries
- Scratches and Consoles

The code editor shows a Python file with the following content:

```
 3
 4
 5     # Create your models here.
 6
 7     4 usages
 8
 9     class Article(models.Model):
10
11         title = models.CharField(max_length=150, blank=False, unique=False)
12         content = models.TextField(blank=False, default='')
13         year = models.PositiveSmallIntegerField(null=True, blank=True)
14         image = models.ImageField(upload_to='media', blank=True, null=True)
15
16     @① def __str__(self):
17         |     return self.title_with_year()
18
19     1 usage
20
21     def title_with_year(self):
22         |     return "{} ({})".format(*args: self.title, self.year)
```

The code editor highlights the `__str__` method definition and its call in the `title_with_year` method with red bars.

Rejestrowanie aplikacji za pomocą dekoratora

The image shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure:

- system_app_api (highlighted in yellow)
 - .venv (library root)
 - system_app_app
 - migrations
 - __init__.py
 - admin.py (highlighted in red)
 - apps.py
 - models.py
 - tests.py
 - views.py
 - system_app_project
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - db.sqlite3
 - manage.py

The code editor shows the contents of admin.py:

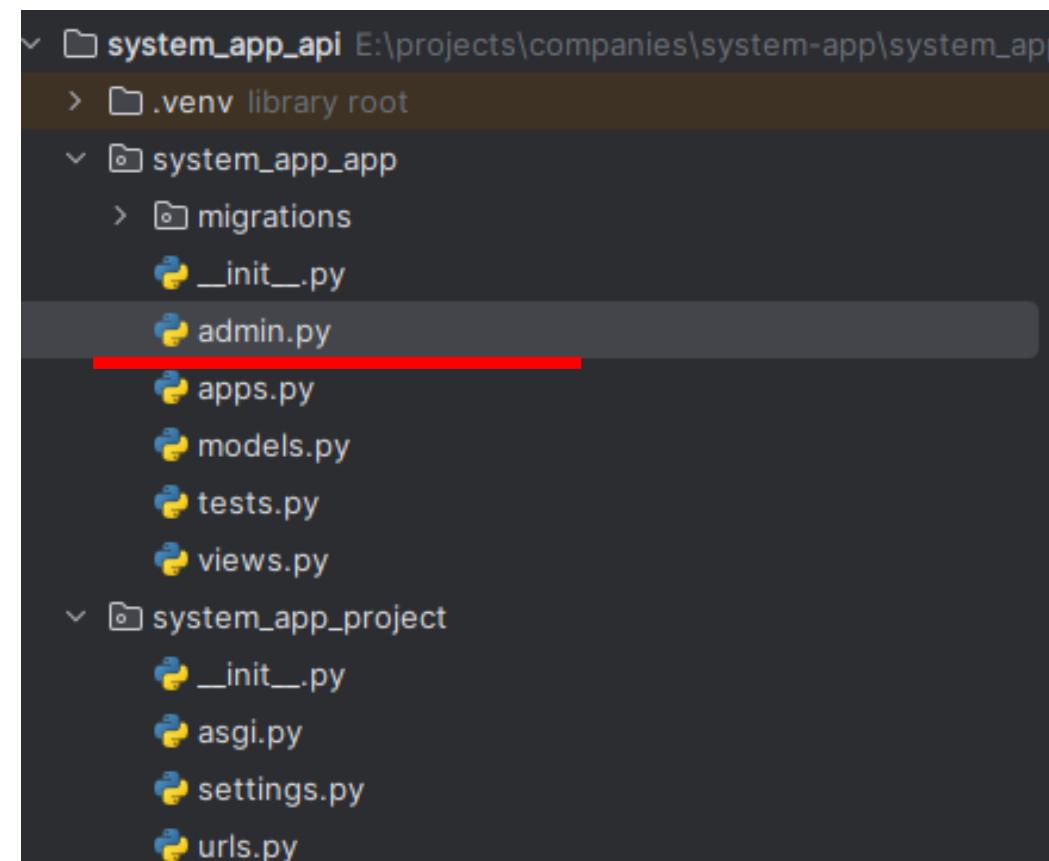
```
from django.contrib import admin
from .models import Article

# Register your models here.
# admin.site.register(Article)

@admin.register(Article)
class ArticleAdmin(admin.ModelAdmin):
    # reszta kodu
    pass
```

Two red curly braces on the right side of the code editor group the entire registration code (lines 5-12) and the class definition (lines 9-12) under the heading "reszta kodu".

Wyświetlanie pól formularza za pomocą deklaracji pola fields



```
1  from django.contrib import admin
2  from .models import Article
3
4  # Register your models here.
5  # admin.site.register(Article)
6
7
8  @admin.register(Article)
9  class ArticleAdmin(admin.ModelAdmin):
10     fields = ['title', 'year', 'content', 'image']
11
12 |
```

Wyświetlanie pól listy za pomocą pola list_display

The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure:

- system_app_api (E:\projects\companies\system-app\system_app)
 - .venv library root
 - system_app_app
 - migrations
 - __init__.py
 - admin.py (highlighted with a red rectangle)
 - apps.py
 - models.py
 - tests.py
 - views.py
 - system_app_project
 - __init__.py
 - asgi.py

The code editor shows the contents of the admin.py file:

```
1  from django.contrib import admin
2  from .models import Article
3
4  # Register your models here.
5  # admin.site.register(Article)
6
7
8  @admin.register(Article)
9  class ArticleAdmin(admin.ModelAdmin):
10     # Pola w formularzu
11     fields = ['title', 'year', 'content', 'image']
12     # Pola na liście
13     list_display = ['title', 'year']
14
```

Line 13, which contains the definition of the list_display attribute, is highlighted with a red rectangle.

Wyświetlanie filtra na liście za pomocą pola list_filter

The image shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure with folders like 'system_app_api', '.venv', 'system_app_app' (which contains 'migrations', '__init__.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', and 'views.py'), and 'system_app_project' (which contains '__init__.py', 'asgi.py', and 'settings.py'). The 'admin.py' file is selected in the file explorer and is shown in the code editor.

```
 1  from django.contrib import admin
 2  from .models import Article
 3
 4  # Register your models here.
 5  # admin.site.register(Article)
 6
 7
 8  @admin.register(Article)
 9  class ArticleAdmin(admin.ModelAdmin):
10      # Pola w formularzu
11      @† fields = ['title', 'year', 'content', 'image']
12      # Pola na liście
13      @† list_display = ['title', 'year']
14      # Pole filtra
15      @† list_filter = ['year']
```

Wyświetlanie wyszukiwarki na liście za pomocą pola search_fields

system_app_app

migrations

__init__.py

admin.py

apps.py

models.py

tests.py

views.py

system_app_project

__init__.py

asgi.py

settings.py

urls.py

wsgi.py

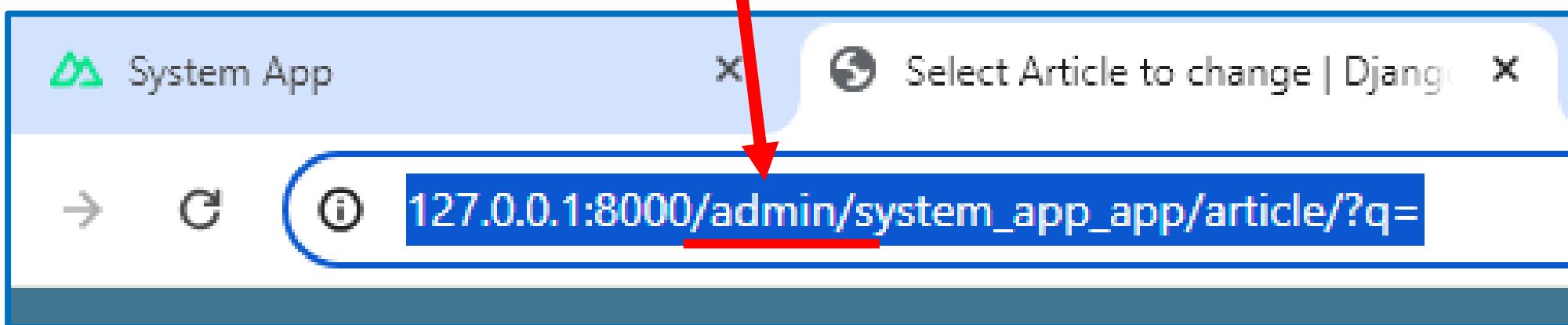
= db.sqlite3

```
3
4 # Register your models here.
5 # admin.site.register(Article)
6
7
8 @admin.register(Article)
9 class ArticleAdmin(admin.ModelAdmin):
10     # Pola w formularzu
11     fields = ['title', 'year', 'content', 'image']
12
13     # Pola na liście
14     list_display = ['title', 'year']
15
16     # Pole filtra
17     list_filter = ['year']
18
19     # Pole wyszukiwarki
20     search_fields = ['title', 'content']
```

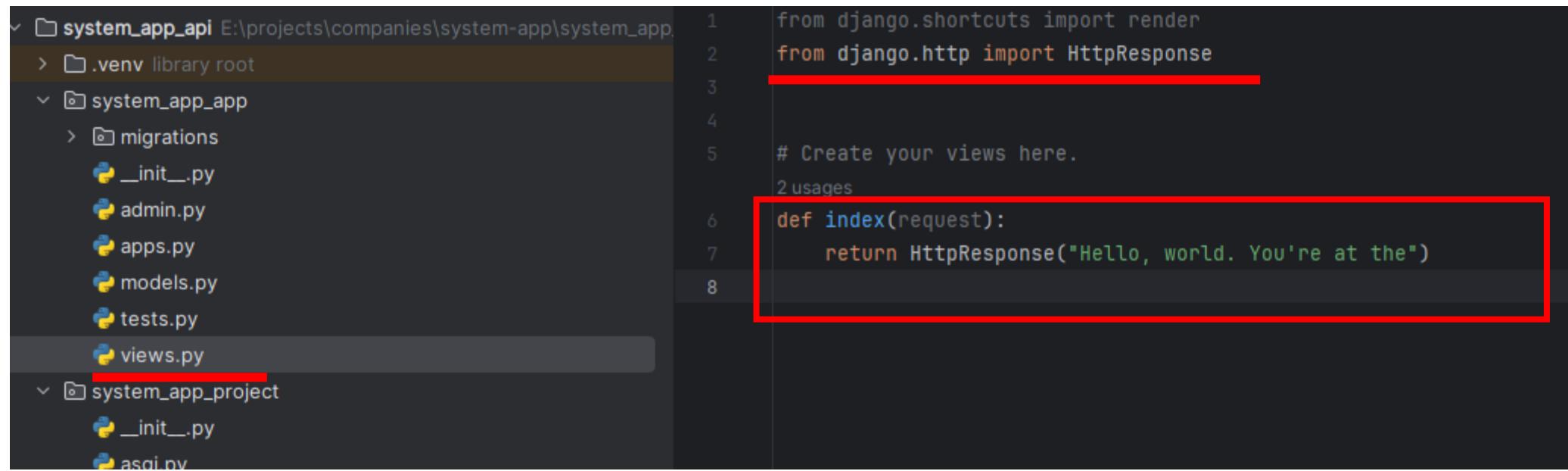
Tworzenie routingu (urls)

```
system_app_project
  __init__.py
  asgi.py
  settings.py
  urls.py (selected)
  wsgi.py
  db.sqlite3
  manage.py
  populate_articles.py
> External Libraries
> Scratches and Consoles
```

```
4     The urlpatterns list routes URLs to views. For more information please see:
5         https://docs.djangoproject.com/en/5.0/topics/http/urls/
6     Examples:
7         Function views
8             1. Add an import: from my_app import views
9                 2. Add a URL to urlpatterns: path('', views.home, name='home')
10            Class-based views
11                1. Add an import: from other_app.views import Home
12                    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13            Including another URLconf
14                1. Import the include() function: from django.urls import include, path
15                    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16
17    > import ...
18
19
20    urlpatterns = [
21        path('admin/', admin.site.urls),
22    ]
```



Tworzenie routingu (urls) – widoki (views.py) w aplikacji



The image shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure:

- system_app_api (containing .venv, system_app_app, and system_app_project)
- .venv (library root)
- system_app_app (containing migrations, __init__.py, admin.py, apps.py, models.py, tests.py, and views.py)
- system_app_project (containing __init__.py and asgi.py)

The code editor shows the contents of views.py:

```
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3
4 # Create your views here.
5
6 def index(request):
7     return HttpResponseRedirect("Hello, world. You're at the")
8
```

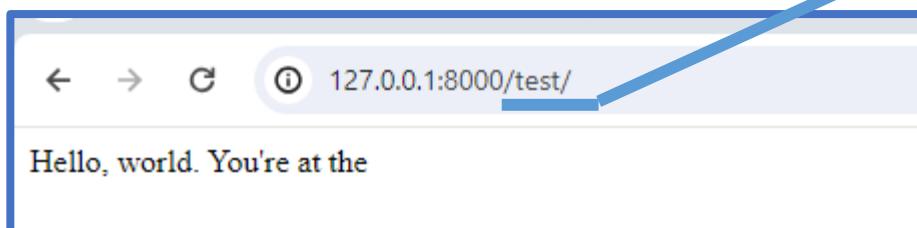
A red rectangle highlights the entire code block in the code editor.

Tworzenie routingu (urls) w projekcie

The screenshot shows a file explorer on the left and a code editor on the right. In the file explorer, a folder named 'system_app_app' is highlighted with a red box. Inside it, a file named 'urls.py' is also highlighted with a red box. The code editor displays a Python script for configuring URLs:

```
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13
14     Including another URLconf
15
16     1. Import the include() function: from django.urls import include
17     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
18
19     """
20     from django.contrib import admin
21     from django.urls import path
22     from system_app_app.views import index
23
24     urlpatterns = [
25         path('admin/', admin.site.urls),
26         path('test/', index),
27     ]
```

A red arrow points from the 'system_app_app' folder in the file explorer to the 'from system_app_app.views import index' line in the code editor. A blue arrow points from the 'test/' URL entry in the code editor to the '127.0.0.1:8000/test/' address bar in the browser window below.



Tworzenie routingu (urls) z podziałem na aplikacje

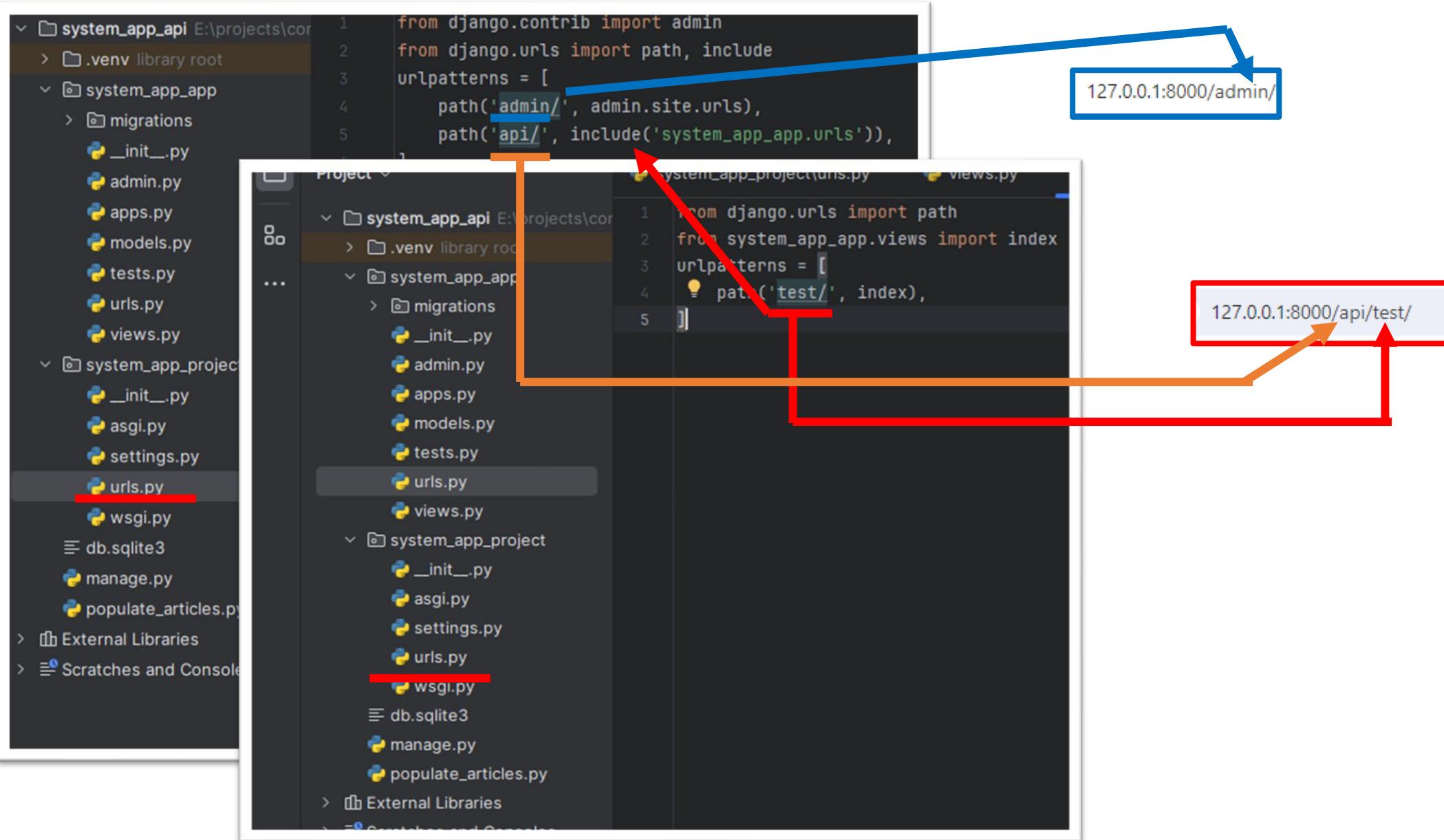
The screenshot shows a code editor with two panes. The left pane displays the file `system_app_api/urls.py` containing:from django.contrib import admin
from django.urls import path, include
urlpatterns = [
 path('admin/', admin.site.urls),
 path('api/', include('system_app_app.urls')),
]The right pane shows the project structure:Project
└── system_app_api
 ├── .venv
 └── system_app_app
 ├── migrations
 ├── __init__.py
 ├── admin.py
 ├── apps.py
 ├── models.py
 ├── tests.py
 ├── urls.py
 └── views.pyA red arrow points from the text "Ten plik istnieje domyślnie" to the `urls.py` file in the app directory.

Ten plik istnieje domyślnie

The screenshot shows a code editor with two panes. The left pane displays the file `system_app_project/urls.py` containing:from django.urls import path
from system_app_app.views import index
urlpatterns = [
 path('test/', index),
]The right pane shows the project structure:Project
└── system_app_api
 ├── .venv
 └── system_app_app
 ├── migrations
 ├── __init__.py
 ├── admin.py
 ├── apps.py
 ├── models.py
 ├── tests.py
 ├── urls.py
 └── views.pyA red arrow points from the text "Ten plik trzeba utworzyć samemu" to the `urls.py` file in the app directory.

Ten plik trzeba utworzyć samemu

Tworzenie routingu (urls) z podziałem na aplikacje



Tworzenie szablonów (templates)-part 1

The screenshot shows a file explorer on the left and a code editor on the right.

File Explorer:

- system_app_api (highlighted)
- .venv (library root)
- system_app_app
- system_app_project
 - __init__.py
 - asgi.py
 - settings.py (highlighted)
 - urls.py
 - wsgi.py
- templates (highlighted)
- db.sqlite3
- manage.py
- populate

Code Editor (settings.py):

```
51 ROOT_URLCONF = 'system_app_project.urls'  
52  
53 TEMPLATES = [  
54     {  
55         'BACKEND': 'django.template.backends.  
56         'DIRS': ["templates"], ←  
57         'APP_DIRS': True,  
58         'OPTIONS': {  
59             'context_processors': [  
60                 'django.template.context_processors.debug',  
61                 'django.template.context_processors.request',  
62                 'django.contrib.auth.context_processors.auth',  
63                 'django.contrib.messages.context_processors.messages',  
64             ],  
65             },  
66         },  
67     ],  
68  
69 WSGI_APPLICATION = 'system_app_project.wsgi.application'  
70  
71 # Database
```

Annotations:

- A blue oval points to the "templates" folder in the file explorer with the text: "Należy utworzyć katalog templates lub o innej nazwie".
- A red arrow points from the "templates" folder in the file explorer to the "DIRS" key in the code editor.
- A red dotted line highlights the "settings.py" file in the file explorer.
- A blue oval points to the "templates" entry in the "DIRS" list of the code editor with the text: "Należy wpisać nazwę stworzonego katalogu w pliku settings.py projektu".

Tworzenie szablonów (templates)-part 2

Należy stworzyć plik twoja_nazwa.html

The screenshot shows a code editor with a dark theme. On the left is a file tree:

- system_app_api (selected)
- .venv (library root)
- system_app_app
- system_app_project
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
- templates (selected)
- articles.html
- db.sqlite3
- manage.py
- populate_articles.py
- External Libraries
- Scratches and Consoles

The main pane displays an HTML file with the following content:

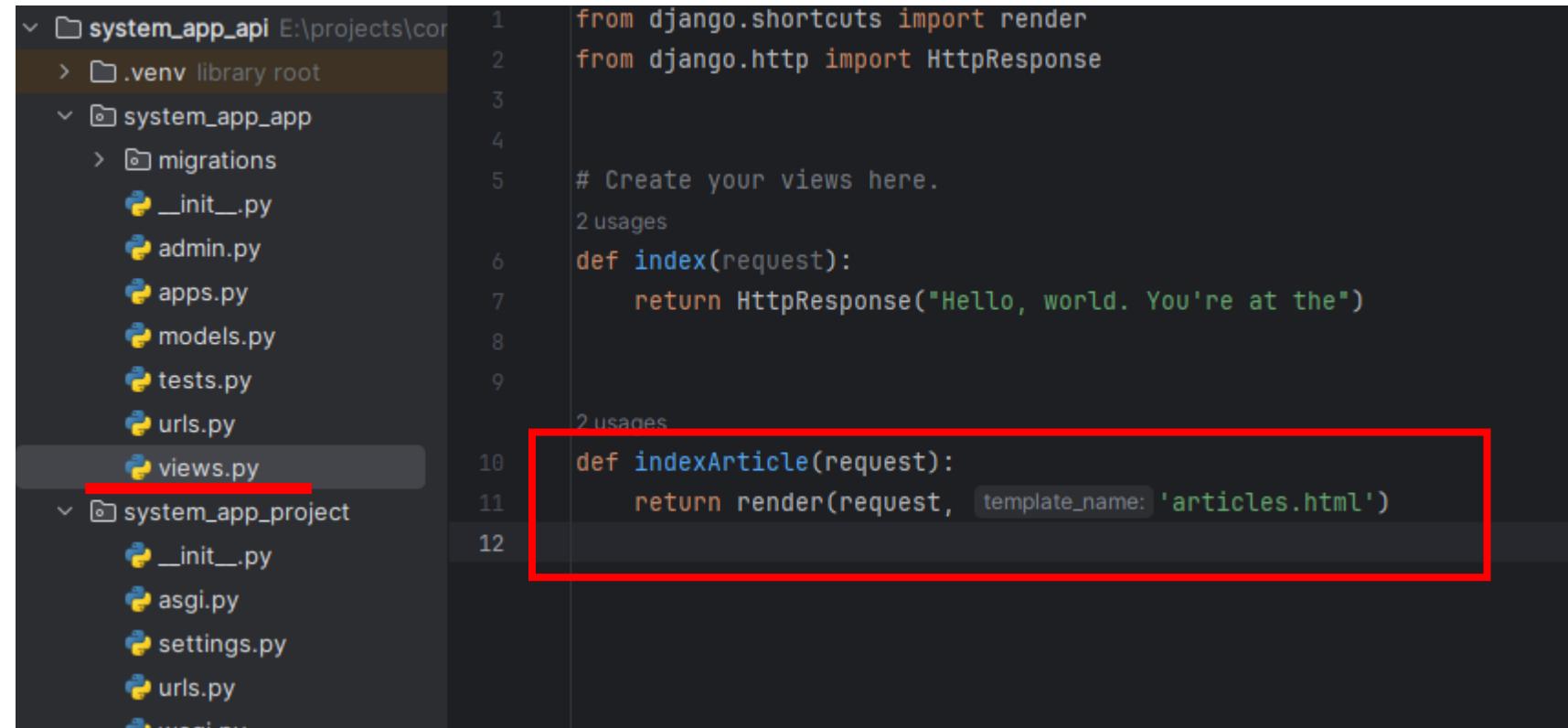
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>To jest mój szablon</h1>
</body>
</html>
```

A red arrow points from the 'articles.html' entry in the file tree to the highlighted file in the main pane.

W templates został
ręcznie stworzony plik
articles.html oraz
wypełniony w powyższą
treść

Tworzenie szablonów (templates)-part 3

Stworzenie metody i zwrócenie szablonu w pliku views.py aplikacji



The screenshot shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure:

- system_app_api (selected)
- .venv library root
- system_app_app
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py (highlighted)
- system_app_project
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py

The code editor contains Python code:

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

# Create your views here.
2 usages
def index(request):
    return HttpResponse("Hello, world. You're at the")

2 usages
def indexArticle(request):
    return render(request, template_name: 'articles.html')
```

A red box highlights the new method `indexArticle`.

Tworzenie szablonów (templates)-part 4

Dodanie adresu do pliku z routingiem aplikacji a nie projektu (bo jest zrobiony podział)

The screenshot shows a code editor with a dark theme. On the left is a file tree for a Django project:

- system_app_api (selected)

 - .venv library root
 - system_app_app
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py (selected)
 - views.py
 - system_app_project
 - __init__.py

The right pane shows the contents of urls.py:

```
1 from django.urls import path
2 from system_app_app.views import index, indexArticle
3 urlpatterns = [
4     path('test/', index),
5     path('html/', indexArticle),
6 ]
```

A red horizontal bar highlights the entire line of code: "path('html/', indexArticle),". A yellow lightbulb icon is visible next to the fifth line.

Tworzenie szablonów (templates)-part 5

Przekazywanie obiektu do szablonu

{"title": title}

The image shows a code editor with a file tree on the left and a code editor window on the right.

File Tree:

- system_app_api E:\projects\cor> .venv library root
- system_app_app> migrations
- __init__.py
- admin.py
- apps.py
- models.py
- tests.py
- urls.py
- views.py
- system_app_project
- __init__.py
- asgi.py
- settings.py
- urls.py
- wsgi.py
- templates
- articles.html

Code Editor (views.py):

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

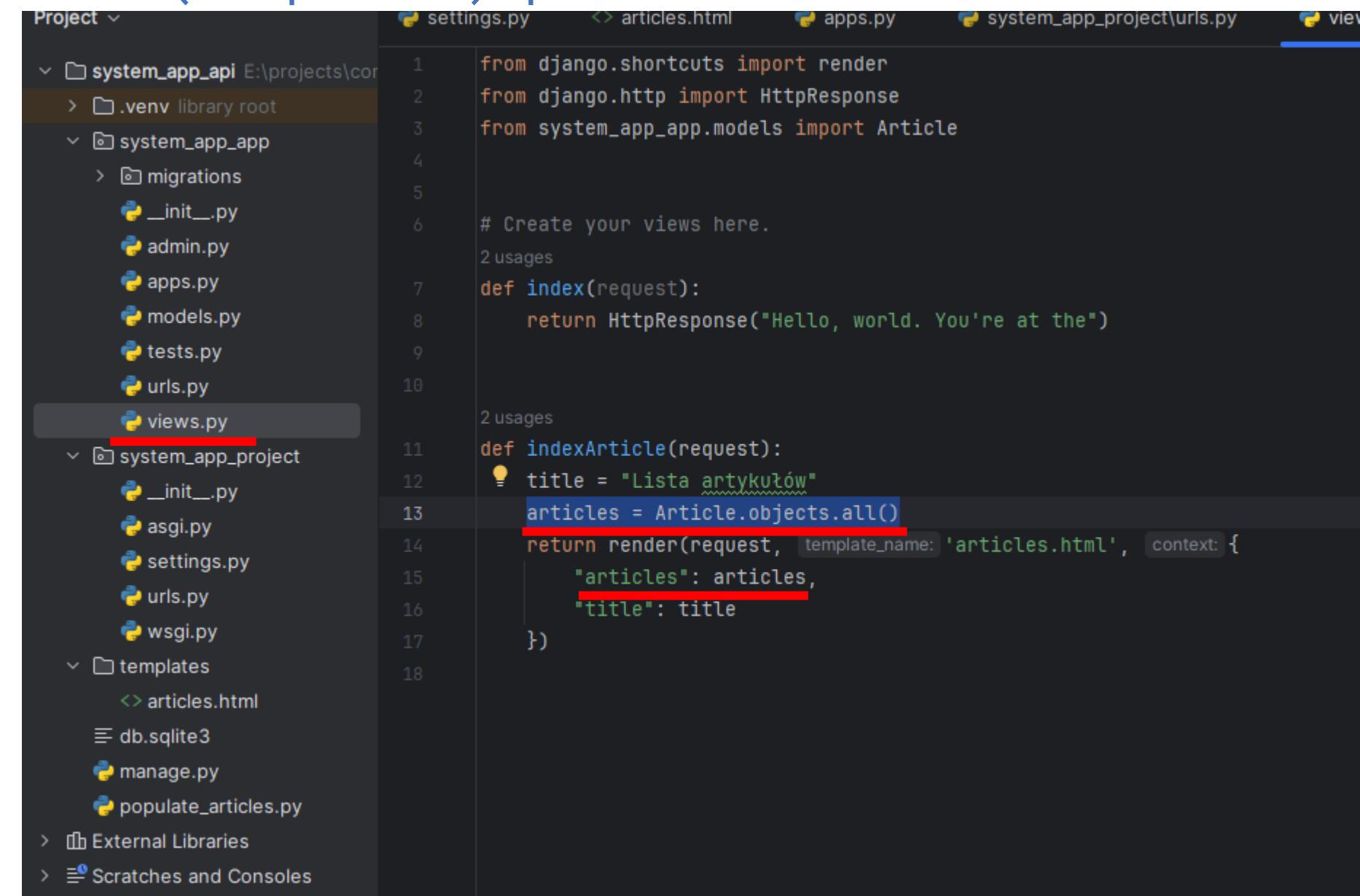
# Create your views here.

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

2 usages
def indexArticle(request):
    title = "Lista artykułów"
    return render(request, template_name: 'articles.html', context: {
        "title": title
    })
```

Tworzenie szablonów (templates)-part 6

Pobieranie z bazy danych
listy artykułów
(na potrzeby szablonu
- zapytania również
będą omawiana
...
parę slajdów dalej ☺)



The screenshot shows a code editor with a dark theme. On the left is a file tree for a Django project named 'system_app_api'. The 'views.py' file under 'system_app_app' is selected and highlighted with a red box. The code in 'views.py' is as follows:

```
from django.shortcuts import render
from django.http import HttpResponse
from system_app_app.models import Article

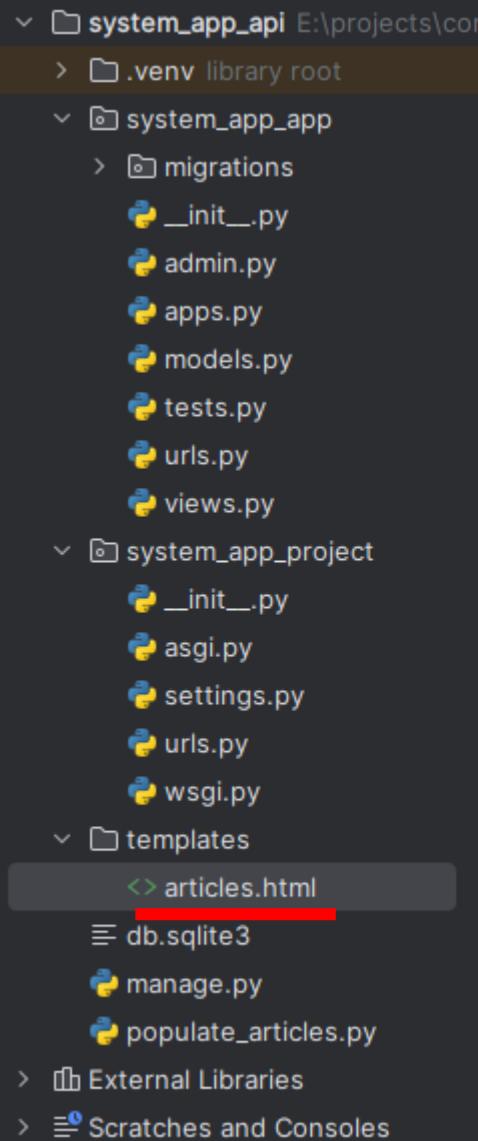
# Create your views here.

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

def indexArticle(request):
    title = "Lista artykułów"
    articles = Article.objects.all()
    return render(request, template_name: 'articles.html', context: {
        "articles": articles,
        "title": title
    })
```

The 'articles.html' template file is shown in the file tree under 'templates'.

Tworzenie szablonów (templates)-part 7



Przekazywanie danych do szablonu wraz z iteracją
(pętla for)

Tworzenie szablonów (templates)-part 8

← → ⌂ ① 127.0.0.1:8000/api/html/

Lista artykułów

Law sister think hundred factor indicate sea. (2005)

Particularly something certain certain already property. (1991)

Ready method never image skill. (2021)

Material share degree grow interview type likely. (2017)

```
from django.utils import timezone

# Create your models here.

class Article(models.Model):
    objects = None
    title = models.CharField(max_length=150, blank=False, unique=False)
    content = models.TextField(blank=False, default='')
    year = models.PositiveSmallIntegerField(null=True, blank=True)
    image = models.ImageField(upload_to='media', blank=True, null=True)
    # created_at = models.DateTimeField(default=timezone.now)
    # updated_at = models.DateTimeField(default=timezone.now)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.title_with_year()

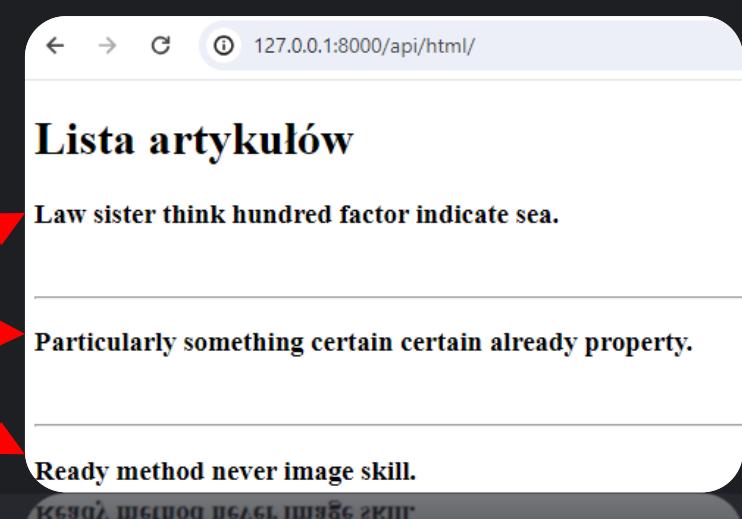
    1 usage
    def title_with_year(self):
        return "{} ({})".format(*args: self.title, self.year)

    class Meta:
        verbose_name = "Article"
        verbose_name_plural = "Artykuły"
```

Tworzenie szablonów (templates)-part 9

- system_app_api E:\projects\cor>
 - .venv library root
- system_app_app>
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
- system_app_project>
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
- templates>
 - articles.html
 - db.sqlite3
 - manage.py
 - populate_articles.py
- External Libraries
- Scratches and Consoles

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport"
6       content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
7     <meta http-equiv="X-UA-Compatible" content="ie=edge">
8     <title>System App - {{ title }}</title>
9   </head>
10  <body>
11    <h1>{{ title }}</h1>
12
13    {% for article in articles %}
14      <h3>{{ article.title }}</h3>
15      <br>
16      <hr/>
17    {% endfor %}
18  </body>
19 </html>
```



Przekazywanie danych do szablonu wraz z iteracją
(pętla for)

Zwracanie danych w formacie JSON

The screenshot shows a code editor with a file tree on the left and a code editor window on the right. The file tree shows a project structure:

- system_app_api (selected)
- .venv library root
- system_app_app
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py (selected)
- system_app_project

The code editor window displays Python code:

```
from django.core import serializers
from django.shortcuts import render
from django.http import JsonResponse
from system_app_app.models import Article

def indexArticleJson(request):
    articles = Article.objects.all()
    data = serializers.serialize('json', articles)
    return JsonResponse(data, content_type="application/json")
```

A red box highlights the entire function body from line 7 to line 11.

Zwracanie danych w formacie JSON – dodanie do routingu

The screenshot shows a file browser on the left with a project structure:

- system_app_api (selected)
- .venv library root
- system_app_app
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py

In the code editor, the `urls.py` file is open, showing the following code:

```
from django.urls import path
from system_app_app.views import index, indexArticle, indexArticleJson

urlpatterns = [
    path('test/', index),
    path('html/', indexArticle),
    path('articles/', indexArticleJson),
```

A red arrow points from the text "Route api jest prefixem articles!" to the URL pattern `path('articles/', indexArticleJson),`.

To the right, a browser window shows the result of the API call at `127.0.0.1:8000/api/articles/`:

```
// 20240129013425
// http://127.0.0.1:8000/api/articles/
[
  {
    "model": "system_app_app.article",
    "pk": 167,
    "fields": {
      "title": "Law sister think hundred factor indic",
      "content": "Market unit arrive. Always issue pr
      build Democrat hundred carry. Remember money move peo
      then ability. Until particular they charge. Effort pi
      consumer. Interesting way military speech. Town style
```

Route api jest
prefixem
articles!

Praca samodzielna studenta na laboratorium oraz w wolnym czasie

Proszę odszukać narzędzie pozwalające na połączenie się z bazą danych typu SQLite lub skonfigurować połączenie w wybranym IDE

Proszę zapoznać się z dokumentacją Django (zadanie do domu)

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

Zadanie 1. – czas wykonania 1h

Należy stworzyć projekt w Django w wybranej tematyce.

Jeżeli nie masz pomysłu na temat to Twoim zadaniem jest stworzenie portalu (do wyboru)

- Sprzedaż nieruchomości
- Salon samochodowy
- Portal z wycieczkami (podobne do booking.com)
- Portal gastronomiczny (podobne do pyszne.pl)
- Portal podobny do allegro

Proszę przemyśleć jakie dane (modele, tabele, kolumny) będą potrzebne do realizacji takiego zadania.

Następnie należy stworzyć modele, wykonać migracje, zarejestrować aplikacje oraz przetestować obsługę po stronie panelu admina. W tym zadaniu nie ma znaczenia na obecną chwilę tworzenie powiązań pomiędzy tabelami w bazie danych.

Zadanie należy spakować w całości i umieścić na platformie moodle.

Oceniana jest jakość wykonania zadania.

Za to zadanie można uzyskać maksymalnie 100 pkt.

Pomagam w zadaniu o ile nie wybiega to z tematyką obecnych laboratoriów