**Lecture 01. Django первый проект.**

**Установка и настройка**

1. Установить VS Code.

Чтобы установить VS Code, вам нужно сначала скачать установочный файл с официального сайта Visual Studio Code. После этого запустите установочный файл и следуйте инструкциям на экране. После завершения установки вы сможете открыть VS Code и начать использовать его для разработки.

1. Установить интерпретатор Python. Лучше всего установить его из магазина Майкрософт.
2. Установить pip. Для этого можно выполнить в командной строке/консоли следующую команду:

**python -m ensurepip –upgrade**

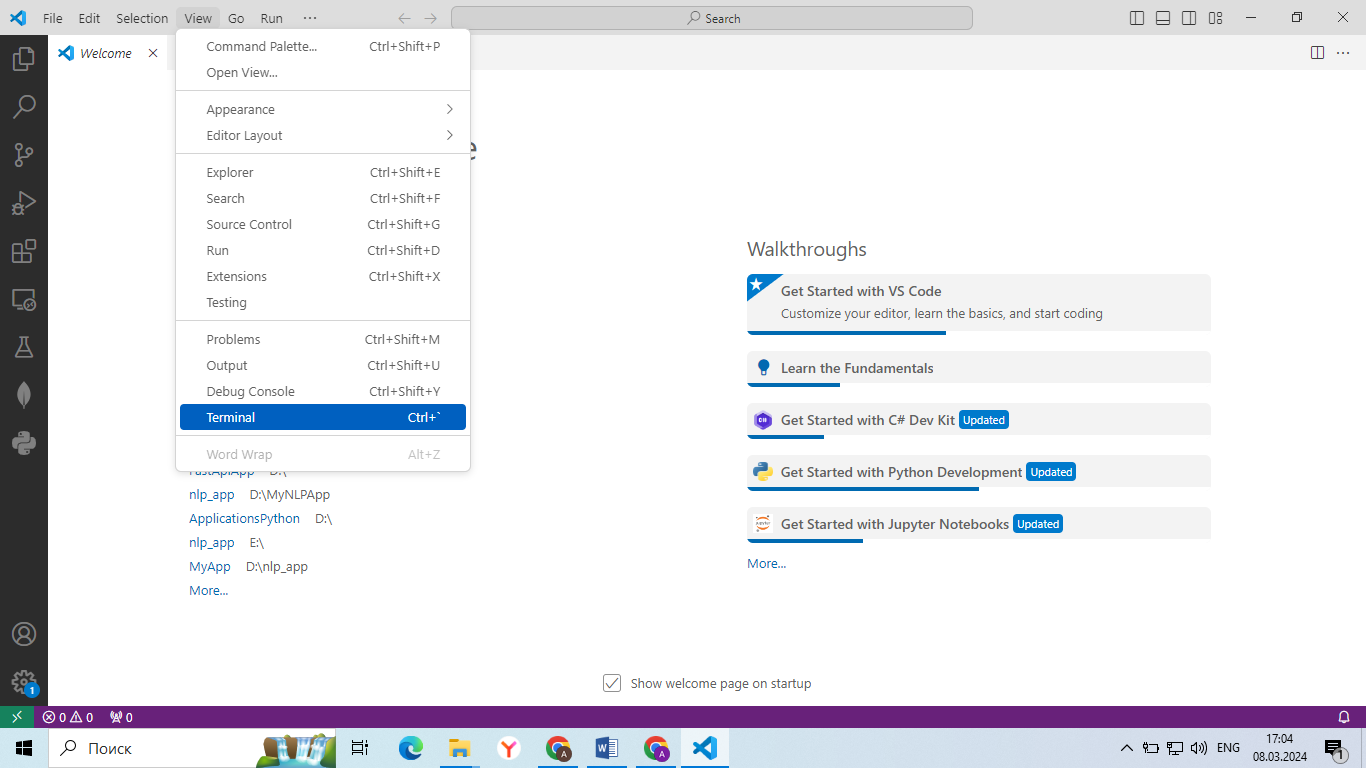
Если pip ранее уже был установлен, то можно его обновить с помощью команды

**python -m pip install --upgrade pip**

**Создание и запуска Django приложения**

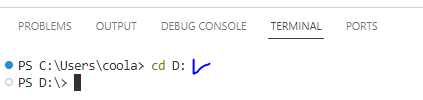
Для создания и запуска Django приложения с использованием виртуальной среды в Visual Studio Code:

**Шаг 1**. Откройте терминал в VS Code.



Перейдите на диск D:

cd D:



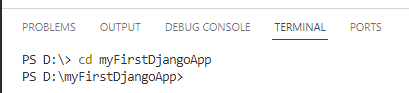
Создайте новую папку с названием myFirstDjangoApp:

mkdir myFirstDjangoApp



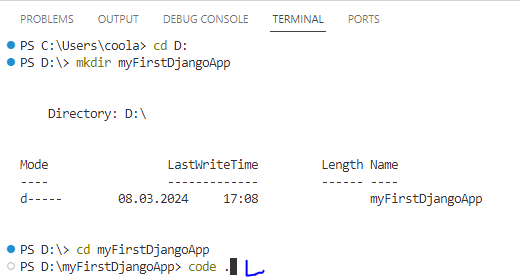
Перейти в созданную папку:

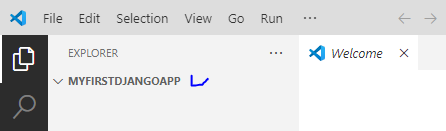
cd myFirstDjangoApp



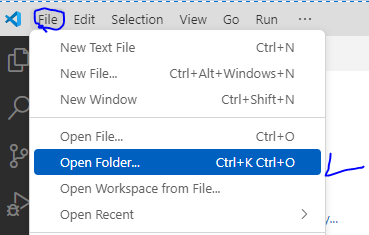
Откройте эту папку в VS Code:

code .





**На заметку***: Если команда code . не работает, тогда из меню "Файл" откройте папку.*



**Шаг 2:** Установка virtualenv и создание виртуальной среды

Убедитесь, что у вас установлен Python и инструмент virtualenv. Если нет, установите их с помощью следующих команд:

pip install virtualenv

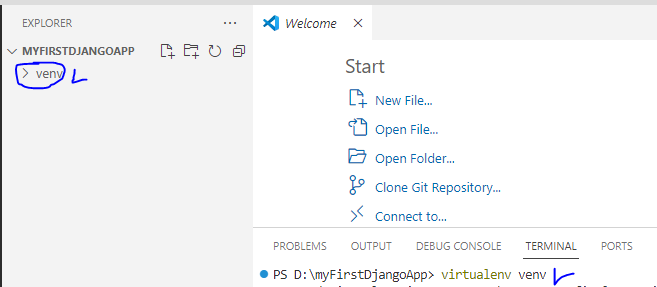




**Шаг 3:** Создание виртуальной среды и активация

Откройте терминал в VS Code, перейдите в папку, где будет расположен ваш проект, и создайте виртуальную среду:

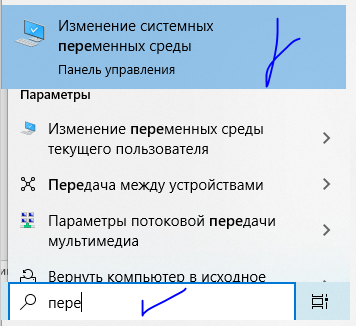
virtualenv venv

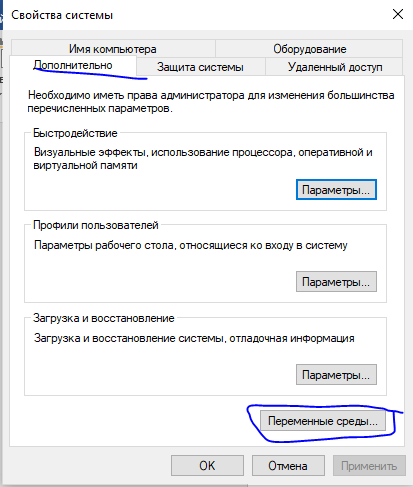


**На заметку:** *Если команда не работает, то добавьте путь к исполняемому файлу virtualenv в переменную среды Windows.*

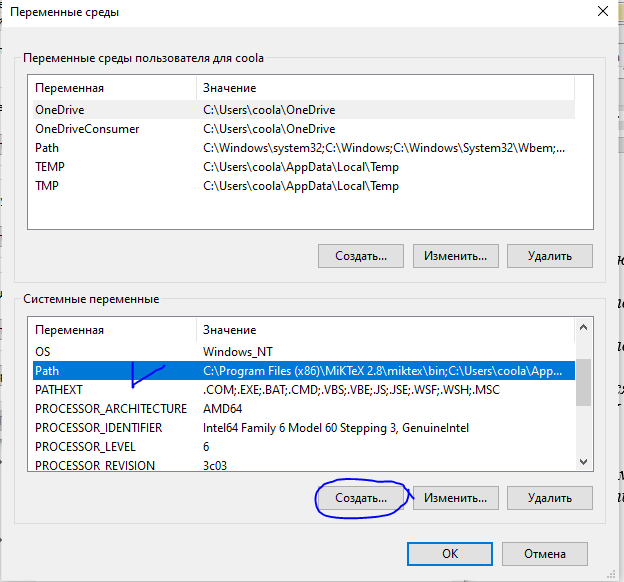
*Чтобы добавить путь к исполняемому файлу virtualenv в переменную среды Windows, выполните следующие шаги:*

* *Нажмите Win + S, введите "переменные среды" и выберите "Изменение переменных среды системы".*





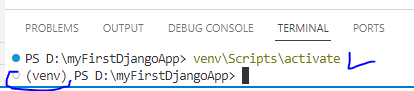
* *В разделе "Переменные среды" найдите переменную PATH в разделе "Системные переменные" и нажмите "Изменить".*



* *Нажмите "Создать" и добавьте путь к папке, где находится исполняемый файл virtualenv (например, "C:\PythonXX\Scripts", где XX - версия Python).*
* *Нажмите "ОК", чтобы закрыть все окна и сохранить изменения.*
* *После этого попробуйте выполнить команду "virtualenv venv" в новом экземпляре терминала или командной строки, и она должна быть доступна.*

Активируйте виртуальную среду:

venv\Scripts\activate



На заметку:Если команда venv\Scripts\activate вызывает ошибку:

| Невозможно загрузить файл D:\Учеба\PythonCourse\django\venv\Scripts\activate.ps1, так как выполнение сценариев отключено в этой системе. Для получения дополнительных сведений см. about\_Execution\_Policies по адресу https:/[go.microsoft.com/fwlink/?LinkID=135170](https://vk.com/away.php?to=http%3A%2F%2Fgo.microsoft.com%2Ffwlink%2F%3FLinkID%3D135170&cc_key=). строка:1 знак:1 + venv\Scripts\activate + ~~~~~~~~~~~~~~~~~~~~~ + CategoryInfo : Ошибка безопасности: (:) [], PSSecurityException + FullyQualifiedErrorId : UnauthorizedAccess |
| --- |

То в терминале выполните следующую команду:

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

**Шаг 4**: Установка Django

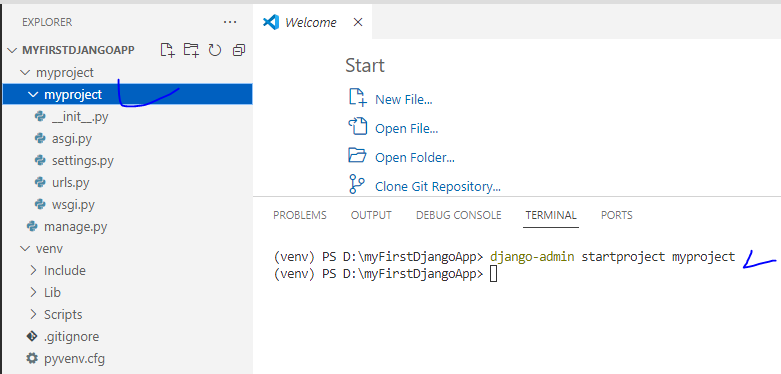
pip install django



**Шаг 5**: Создание проекта Django

Создайте новый проект Django в той же папке:

django-admin startproject myproject



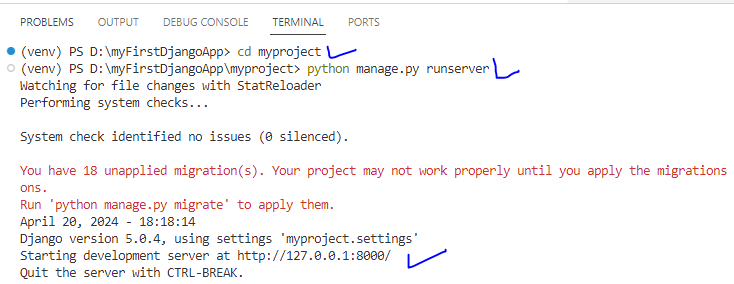
Созданный каталог (проекта Django) будет состоять из следующих элементов:

* **manage.py**: выполняет различные команды проекта, например, создает и запускает приложение
* myproject - собственно папка проекта myproject, которая содержит следующие файлы:
* **\_\_init\_\_.py**: данный файл указывает, что папка, в которой он находится, будет рассматриваться как модуль. Это стандартный файл для программы на языке Python.
* **settings.py**: содержит настройки конфигурации проекта
* **urls.py**: содержит шаблоны URL-адресов, по сути определяет систему маршрутизации проекта
* **wsgi.py**: содержит свойства конфигурации WSGI (Web Server Gateway Inerface). Он используется при развертывании проекта.
* **asgi.py**: название файла представляет сокращение от Asynchronous Server Gateway Interface и расширяет возможности WSGI, добавляя поддержку для взаимодействия между асинхронными веб-серверами и приложениями.

**Шаг 6:** Запуск сервера разработки

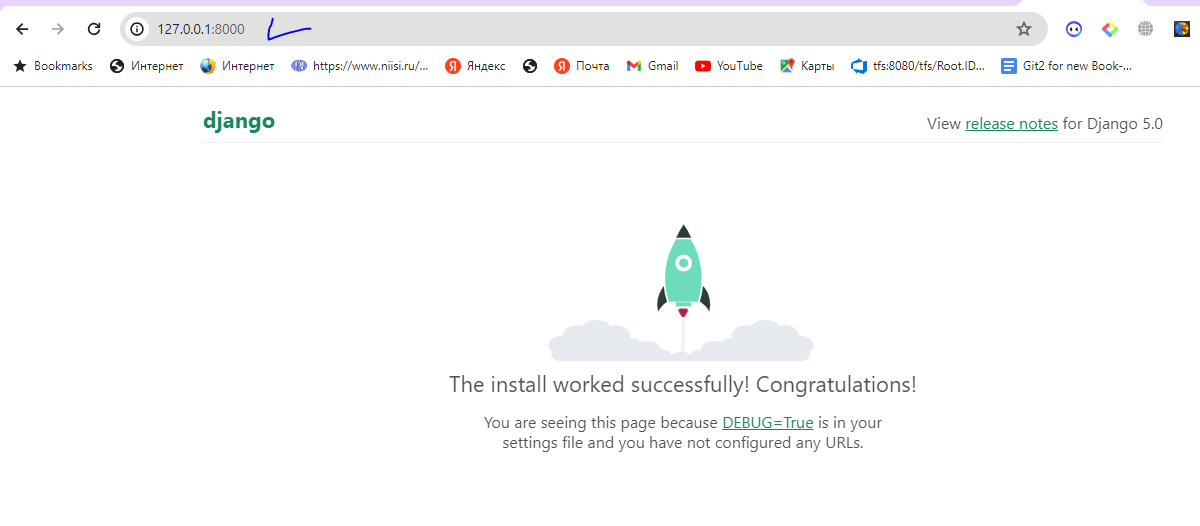
Запустите сервер разработки Django с помощью следующей команды:

python manage.py runserver



**Шаг 7**: Просмотр приложения в браузере

Откройте браузер и перейдите по адресу http://127.0.0.1:8000/, чтобы увидеть ваше новое Django приложение.



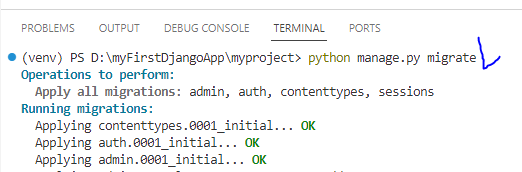
Теперь у вас есть базовое Django приложение, созданное и запущенное с использованием виртуальной среды в Visual Studio Code.

**Шаг 8**: Миграция

Из шага видно, что наш сервер разработки Django успешно запущен и работает на адресе http://127.0.0.1:8000/. Однако, вам рекомендуется применить все накопленные миграции для приложений Django, чтобы ваше приложение работало правильно.

Для применения накопленных миграций выполните следующую команду в терминале, находясь в директории вашего проекта:

python manage.py migrate



Эта команда применит все накопленные миграции для приложений Django, таких как admin, auth, contenttypes, sessions, и ваше приложение будет готово к использованию без проблем.

**Поздравляю вас, если вы дошли до этой точки. Тем, кто достиг этой точки, я ценю и люблю.**

**Создание первого приложения**

Веб-приложение или проект Django состоит из отдельных приложений. Вместе они образуют полноценное веб-приложение. Каждое приложение представляет какую-то определенную функциональность или группу функциональностей. Один проект может включать множество приложений. Это позволяет выделить группу задач в отдельный модуль и разрабатывать их относительно независимо от других. Кроме того, мы можем переносить приложение из одного проекта в другой независимо от другой функциональности проекта.

При создании проекта он уже содержит несколько приложений по умолчанию.

* **django.contrib.admin**
* **django.contrib.auth**
* **django.contrib.contenttypes**
* **django.contrib.sessions**
* **django.contrib.messages**
* **django.contrib.staticfiles**

Список всех приложений можно найти в проекте в файле **settings.py** в переменной **INSTALLED\_APPS**:

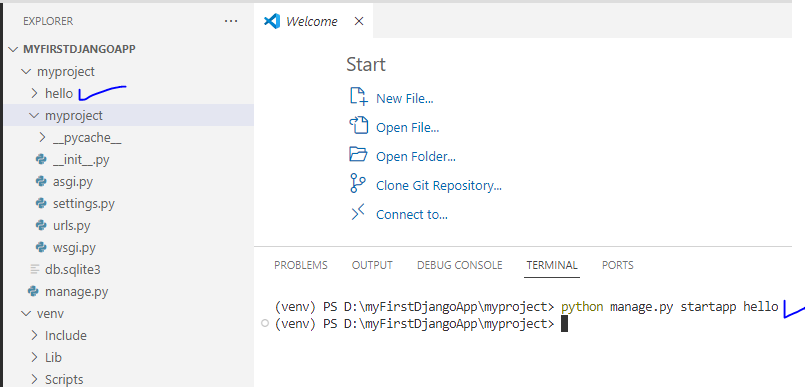
| INSTALLED\_APPS = [      'django.contrib.admin',      'django.contrib.auth',      'django.contrib.contenttypes',      'django.contrib.sessions',      'django.contrib.messages',      'django.contrib.staticfiles',  ] |
| --- |

И, конечно, мы можем создавать свои приложения, которые реализуют определенный функционал. Для создания приложения в проекте применяется команда

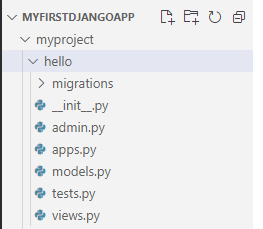
python manage.py startapp название\_приложения

Так, создадим в проекте, созданном в прошлой теме или новом, первое приложение, которое будет называться hello. Для этого выполним в командной строке/терминале следующую команду:

python manage.py startapp hello



В результате в проекте Django будет создано приложение hello. И после выполнения этой команды мы увидим в проекте новую папку, которое будет хранить все файлы созданного приложения:

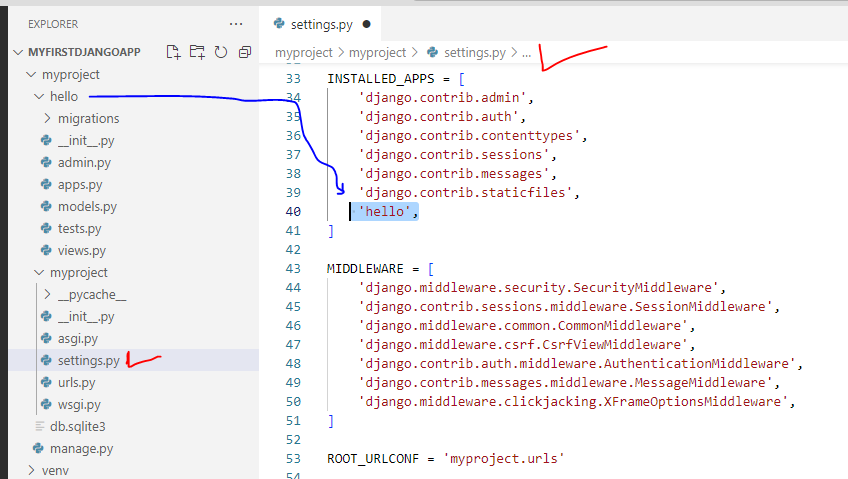
Hello World в Django

Рассмотрим вкратце его структуру:

* папка **migrations**: предназначена для хранения миграций - скриптов, которые позволяют синхронизировать структуру базы данных с определением моделей
* **\_\_init\_\_.py**: указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве пакета
* **admin.py**: предназначен для административных функций, в частности, здесь призводится регистрация моделей, которые используются в интерфейсе администратора
* **apps.py**: определяет конфигурацию приложения
* **models.py**: хранит определение моделей, которые описывают используемые в приложении данные
* **tests.py**: хранит тесты приложения
* **views.py**: определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ

Но пока приложение никак не задействуется. Его надо зарегистрировать в проекте Django. Для этого откроем файл **settings.py** и добавим в конец массива INSTALLED\_APPS наше приложение:

| INSTALLED\_APPS = [      'django.contrib.admin',      'django.contrib.auth',      'django.contrib.contenttypes',      'django.contrib.sessions',      'django.contrib.messages',      'django.contrib.staticfiles',      'hello',  ] |
| --- |



В проекте может быть несколько приложений, и каждое из них надо добавлять таким вот образом.

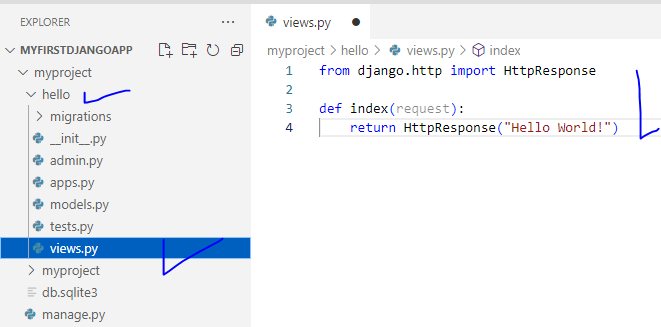
Теперь определим какие-нибудь простейшие действия, которые будет выполнять данное приложение, например, отправлять в ответ пользователю строку "Hello World".

Для этого перейдем в проекте приложения **hello** к файлу **views.py**, который по умолчанию должен выглядеть следующим образом:

| from django.shortcuts import render  # Create your views here. |
| --- |

Изменим код следующим образом:

| from django.http import HttpResponse    def index(request):      return HttpResponse("Hello World!") |
| --- |



В данном случае мы импортируем класс **HttpResponse** из стандартного пакета *django.http*. Затем определяется функция index(), которая в качестве параметра получает объект запроса request. Класс HttpResponse предназначен для создания ответа, который отправляется пользователю. И с помощью выражения return HttpResponse("Hello World!") мы отправляем пользователю строку " Hello World!"

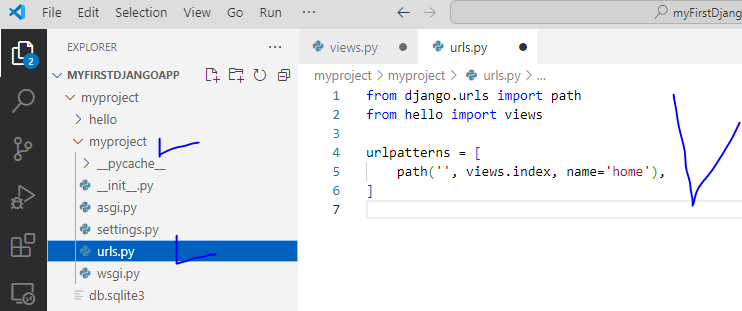
Теперь также в основном проекте Django откроем файл **urls.py**, который позволяет сопоставить маршруты с представлениями, которые будут обрабатывать запрос по этим маршрутам. По умолчанию этот файл выглядит следующим образом:

| from django.contrib import admin  from django.urls import path    urlpatterns = [      path('admin/', admin.site.urls),  ] |
| --- |

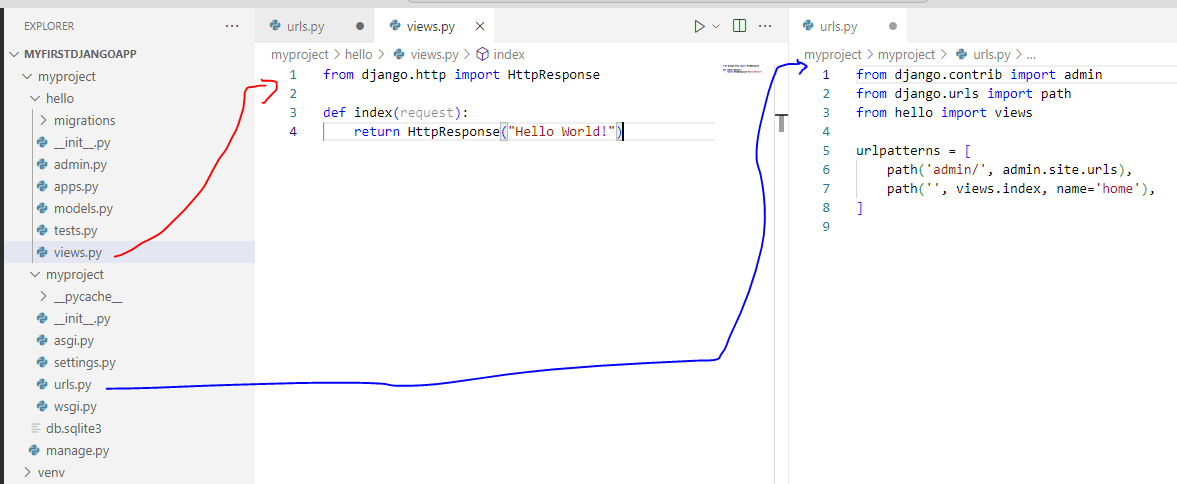
Первой строкой из модуля *django.contrib* импортируется класс AdminSite, который предоставляет возможности работы с интерфейсом администратора. Второй строкой из модуля *django.urls* импортируется функция **path**. Эта функция задает сопоставление определенного маршрута с функцией обработки. Так, в данном случае маршрут "admin/" будет обрабатываться методом admin.site.urls.

Но выше мы определили функцию index в файле **views.py**, который возвращает пользователю строку "Hello World!". Поэтому изменим файл **urls.py** следующим образом:

| from django.urls import path  from hello import views    urlpatterns = [      path('', views.index, name='home'),  ] |
| --- |



Чтобы использовать функцию views.index вначале импортируем модуль views. Затем определяем сопоставление маршрута ' ' и функции views.index и также дополнительно имя для маршрута: name='home'. По сути маршрут ' ' будет сопоставляться с запросом к корню приложения.



Теперь снова запустим приложение командой

python manage.py runserver

И вновь перейдем в браузере по адресу *http://127.0.0.1:8000/*, и браузер нам отобразит строку "Hello World!":

