

Лекция 1. Введение в Django







Вопросы, которые будут рассмотрены в лекции

- Что такое Django и как работает приложение Django
- Что такое локальный сервер и режим отладки, сторонний хост и “боевой” режим
- Основные компоненты Django
- Структура проекта Django
- Создание и запуск первого проекта на Django
- Создание и структура приложения в составе проекта Django
- Главные компоненты приложения Django:
 маршруты (urls.py) и представления (views.py)
- Запуск первого приложения Django



Django – фреймворк для разработки Web-приложений

- Написан на Python и для Python. Это полнокомплектный инструмент, предоставляющий разработчику все необходимое.
- Django является универсальным инструментом для создания любых типов веб-сайтов от систем управления контентом до социальных сетей и новостных порталов. Может работать в любой клиентской среде и доставлять контент в любом формате (HTML, RSS-каналы, JSON, XML и т.д.)
- С использованием Django созданы    
- Django позволяет создавать безопасные, масштабируемые web-приложения, удобные в сопровождении.
- Django используется для обеспечения работы сайтов на стороне сервера и является инструментом для backend-разработчиков.
- Django использует шаблон проектирования (паттерн) MVC (Model-View-Controller), в котором приложение разделяется на три отдельных компонента:
 - модель (описывает структуру данных),
 - представление (отвечает за отображение данных),
 - контроллер (интерпретирует действия пользователя)



Разработчики Django:

- Расселл Кейт-Маги ([англ. Russell Keith-Magee](#))
- Адриан Головатый ([англ. Adrian Holovaty](#)),
- Саймон Виллисон ([англ. Simon Willison](#)),
- Джейкоб Каплан-Мосс ([англ. Jacob Kaplan-Moss](#)),
- Уилсон Майнер ([англ. Wilson Miner](#))

Разработчики заявляли, что не обязуются строго придерживаться какой бы то ни было методологии в развитии проекта, предпочитая делать то, что кажется им наиболее логичным.

Первоначальная Django был разработан как средство для работы новостных ресурсов. Первая версия 0.9 появилась 16 ноября 2005 г.
В апреле 2021г. вышел 3.2 LTS релиз с долгосрочной поддержкой до 2024 г.

Конфигурация сервера

Django проектировался для работы под управлением Apache и с PostgreSQL в качестве базы данных.

С включением поддержки WSGI, Django может работать и на других серверах. Кроме PostgreSQL может использовать СУБД: MySQL, SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle.

В составе Django присутствует собственный веб-сервер для разработки. Сервер автоматически определяет изменения в файлах исходного кода проекта и перезапускается, что ускоряет процесс разработки на Python. Но при этом он работает в однопоточном режиме и пригоден только для процесса разработки и отладки приложения.



Хостинг для Django

Django можно развернуть на PaaS-сервисах RedHat:

- OpenShift, в том числе и бесплатно
- Heroku
- На хостинге PythonAnywhere
- На хостинге Google



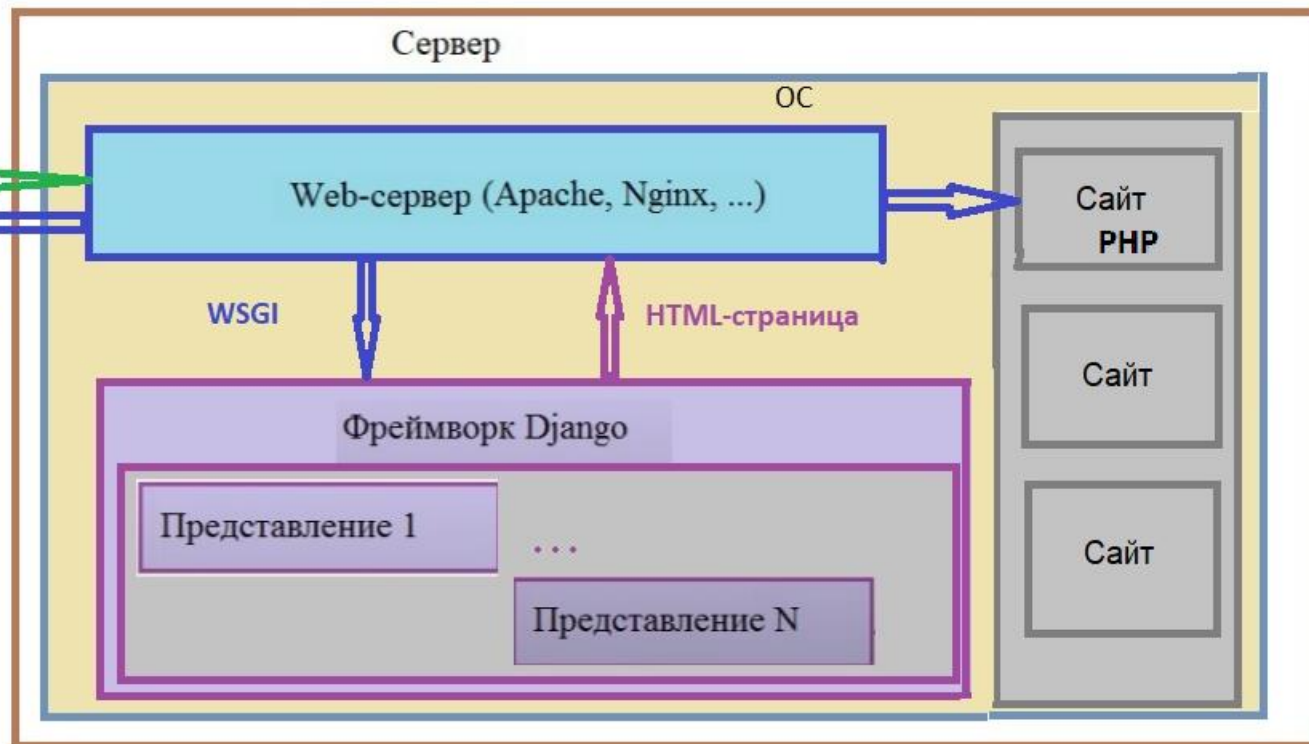
Интернет-пользователь



адрес:
YouTube.com

HTML-страница

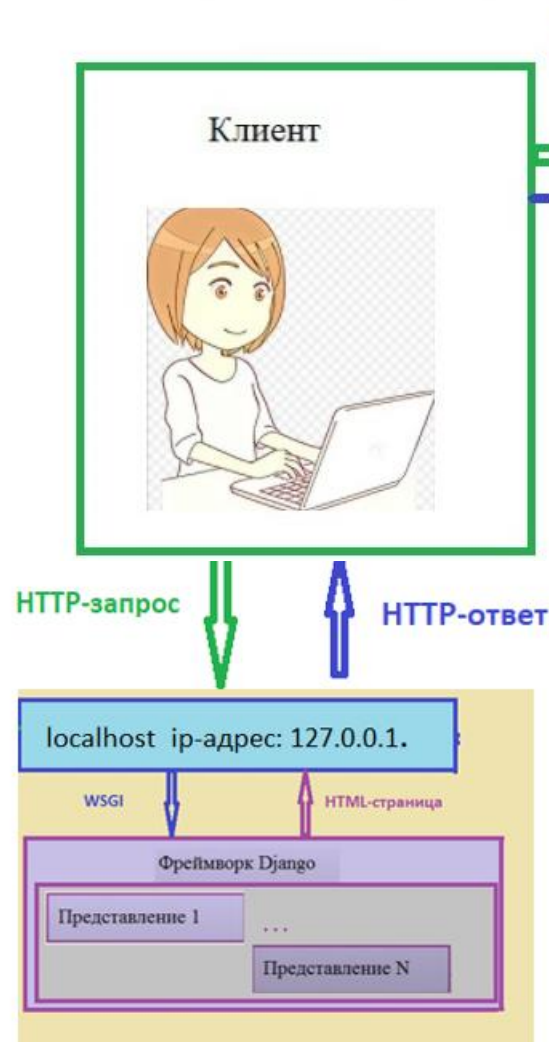
Сетевой компьютер



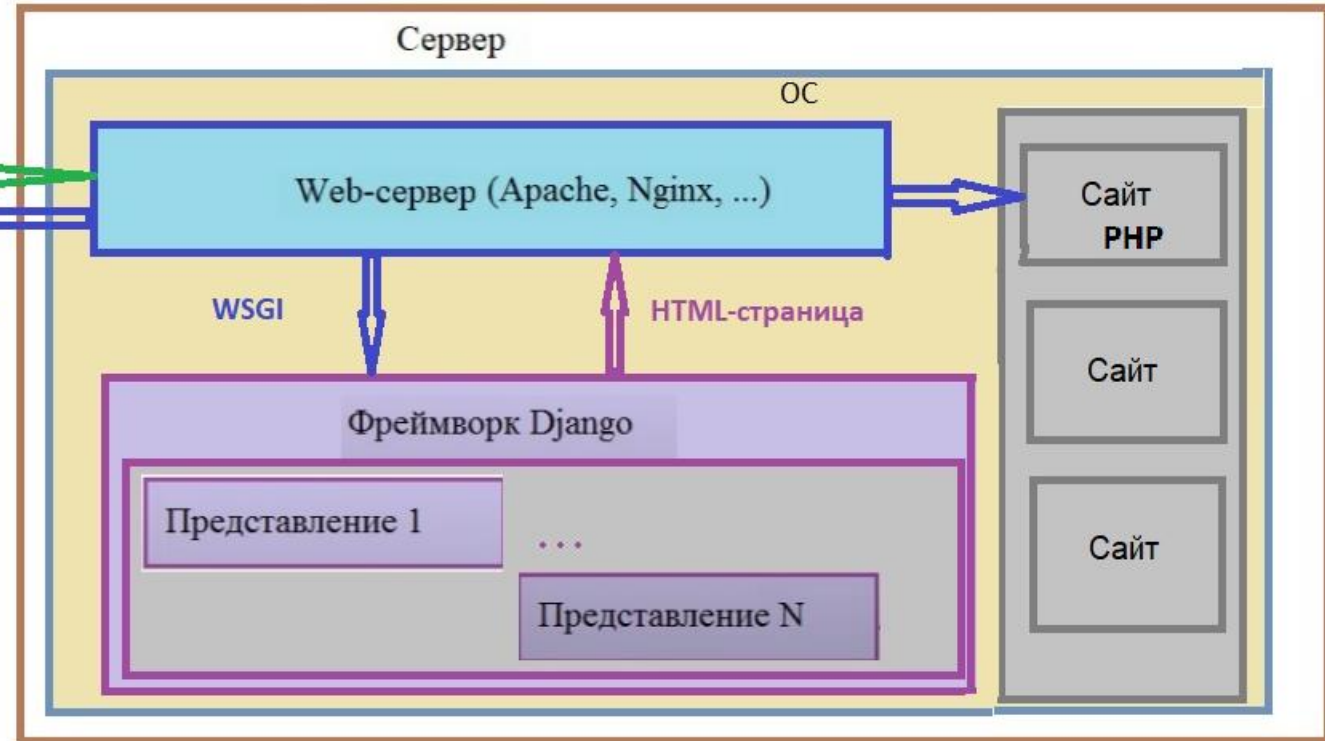
WSGI (*Web Server Gateway Interface*)— это стандарт взаимодействия между Python-программой, выполняющейся на сервере (например, [Apache](#)) и самим веб-сервером



Интернет-пользователь



Сетевой компьютер



WSGI (*Web Server Gateway Interface*)— это стандарт взаимодействия между Python-программой, выполняющейся на сервере (например, [Apache](#)) и самим веб-сервером



Основные компоненты Django

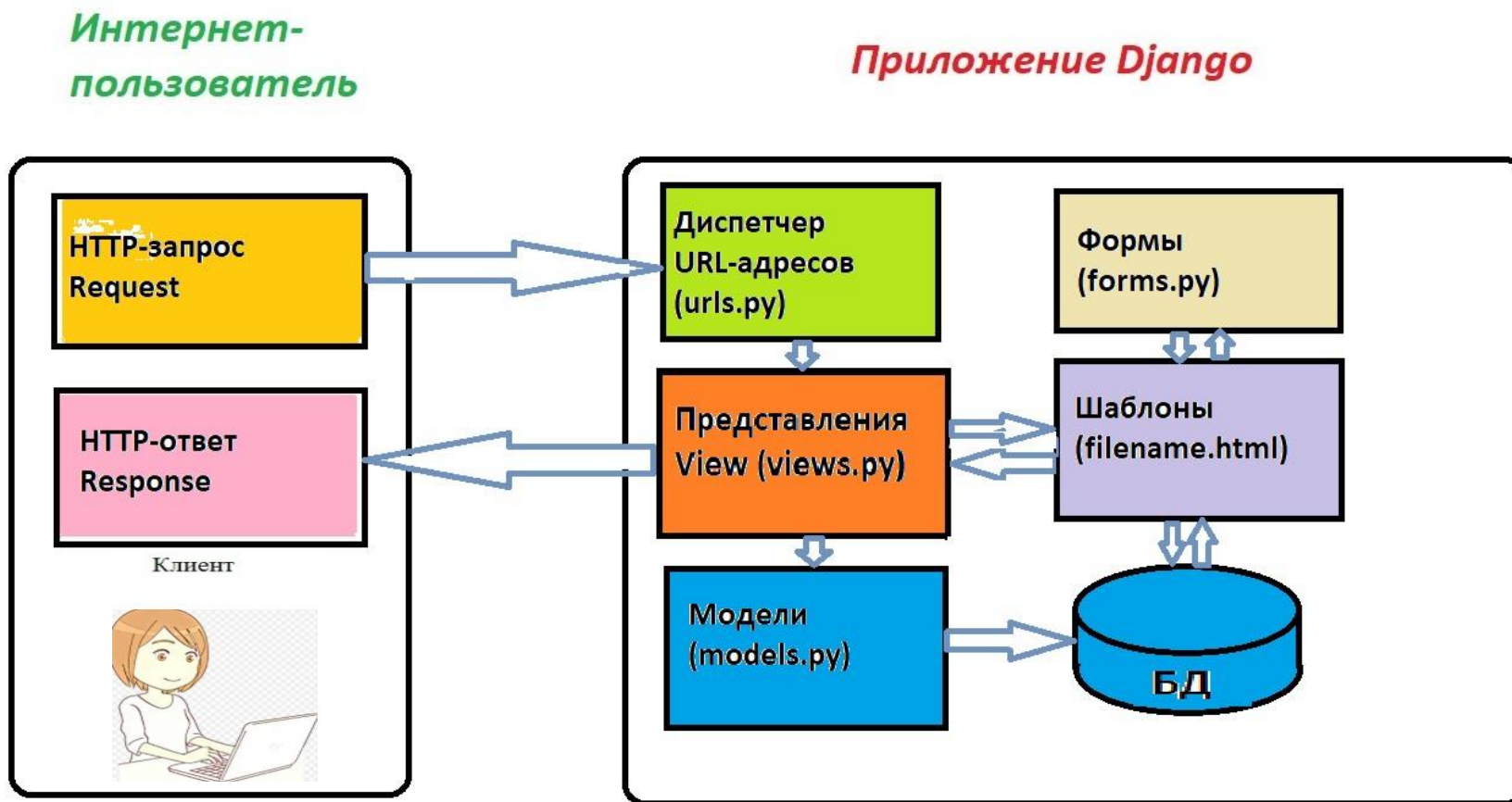


Рис. 1



Диспетчер
URL-адресов
(urls.py)

При получении запроса по адресу URL определяет, какой ресурс должен обрабатывать данный запрос. Переменная `urlpatterns` содержит сопоставления запросов пользователя с функциями их обработки. Функции находятся в файле представлений `views.py`.

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

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



Представления
View (views.py)

Содержит функции, которые принимают данные запроса пользователя в виде объекта **request** и генерируют ответ **response**. Ответ отправляется пользователю в виде HTML страницы.

Могут содержать обращения к модели и базе данных и применять шаблоны.

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

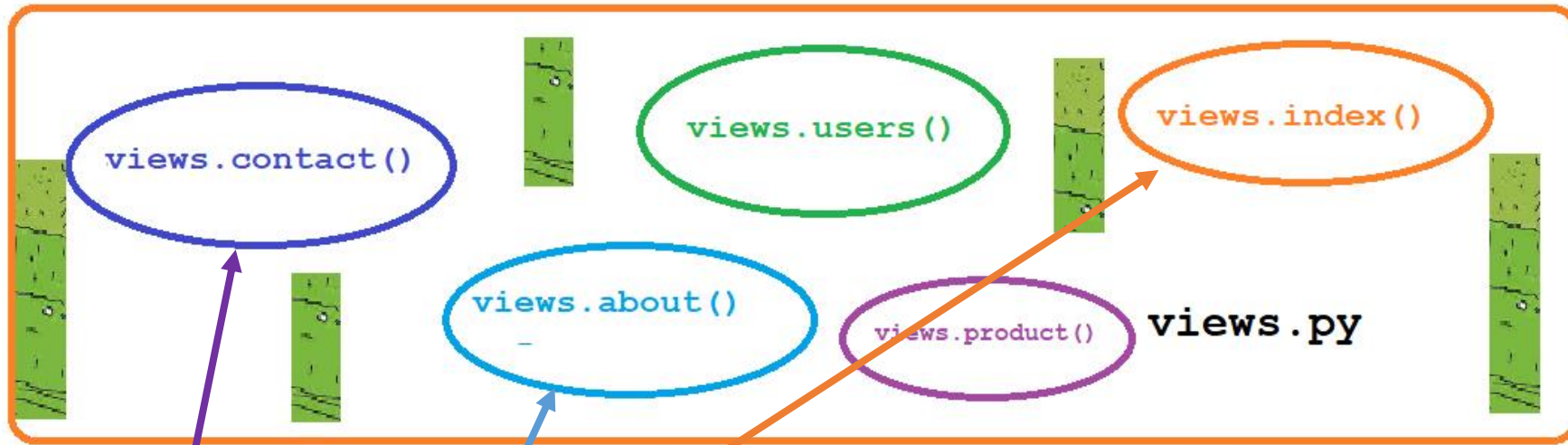
# Create your views here.

def index(request):
    return HttpResponseRedirect("<h2>Главная</h2>")

def about(request):
    return HttpResponseRedirect("<h2>О сайте</h2>")

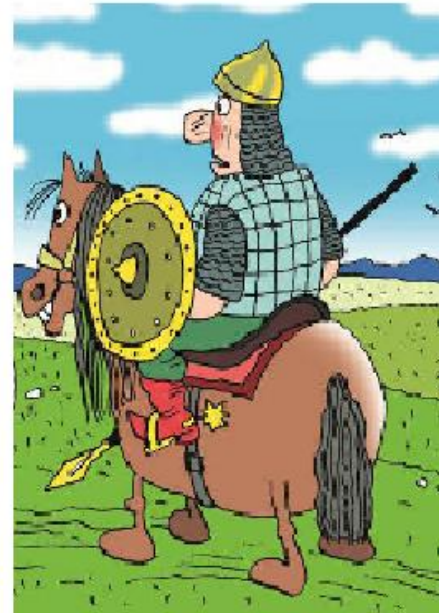
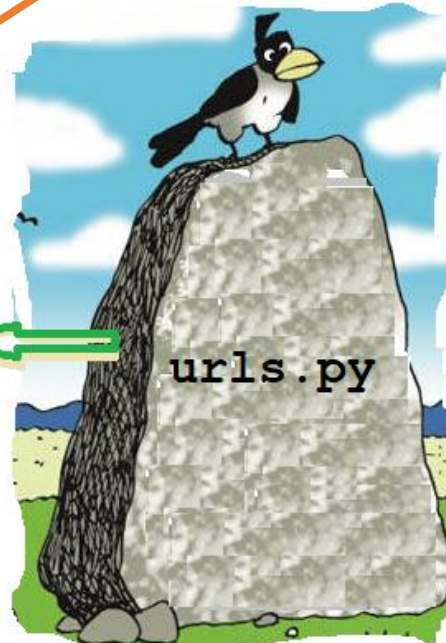
def contact(request):
    return HttpResponseRedirect("<h2>Контакты</h2>")
```





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

`urls.py`



Модели
(models.py)

Модель описывает данные, используемые в приложении. Данные представлены классами. Класс обычно соответствует таблице в базе данных.

```
class Genre(models.Model):
```

```
    name = models.CharField(max_length=20, help_text=" Введите жанр книги", verbose_name="Жанр книги")
```

```
    def __str__(self):
```

```
        return self.name
```

```
class Language(models.Model):
```

```
    name = models.CharField(max_length=20, help_text=" Введите язык книги", verbose_name="Язык книги")
```

```
    def __str__(self):
```

```
        return self.name
```

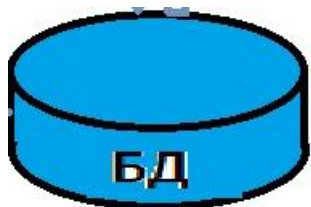


Шаблоны
(filename.html)

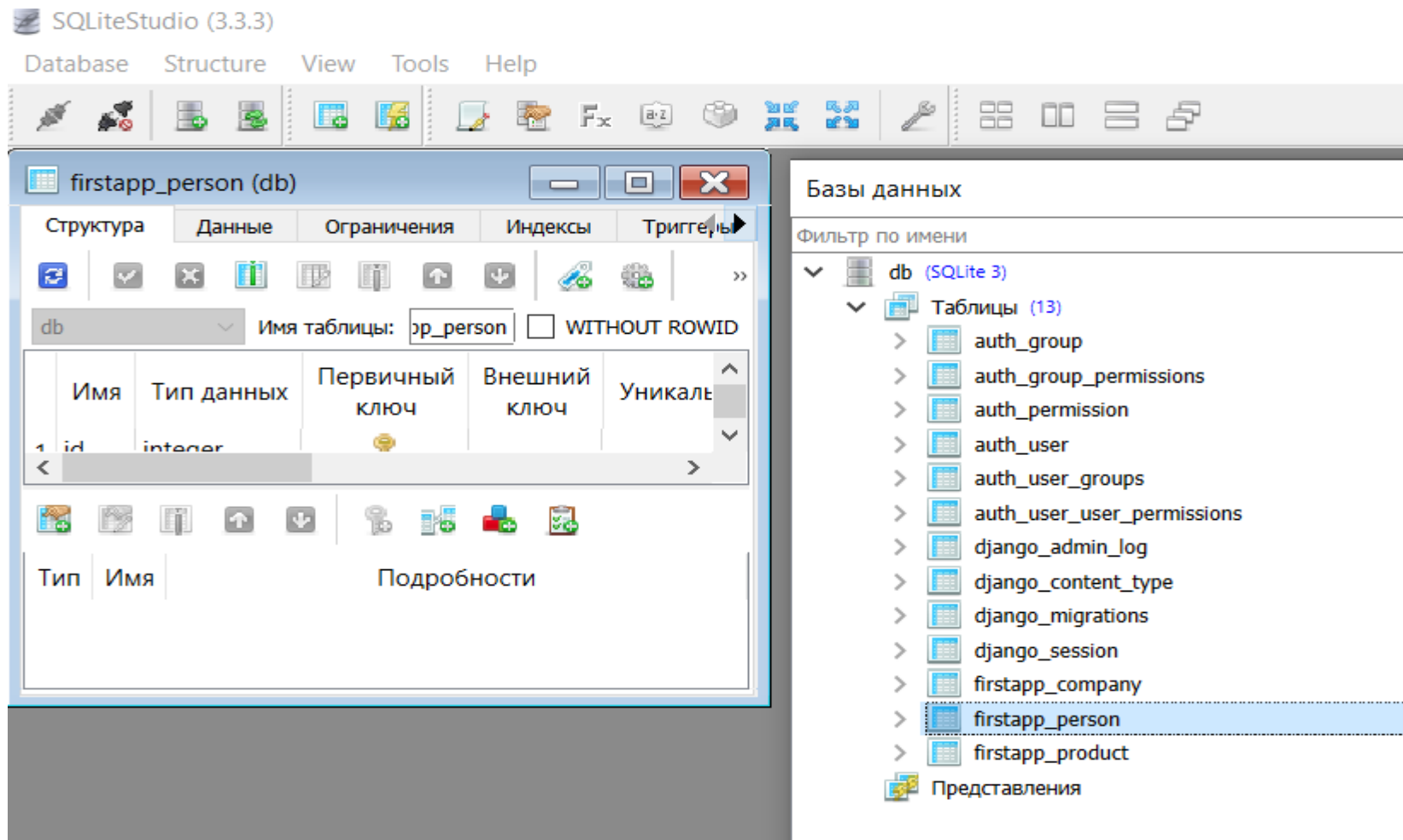
Шаблоны – это HTML страницы, отображающие данные на сайте Django. DB приложения хранятся в папке templates.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hello, django!</title>
</head>
<body>
  <h1>{{header}}</h1>
  <p>{{message}}</p>
</body>
</html>
```





Django может работать со многими распространенными СУБД - PostgreSQL, MySQL, Oracle, но по умолчанию использует СУБД SQLite. Для работы с базой данных удобно использовать приложение SQLiteStudio



Формы
(forms.py)

Формы - это объекты, состоящие из набора разного типа полей на веб-странице для получения данных от пользователя для последующей передачи их на сервер.

```
from django import forms
```

```
class UserForm(forms.Form):
```

```
    name = forms.CharField(widget=forms.TextInput(attrs={"class": "myfield"}))
```

```
    age = forms.IntegerField(widget=forms.NumberInput(attrs={"class": "myfield"}))
```



Создание проекта Django в PyCharm

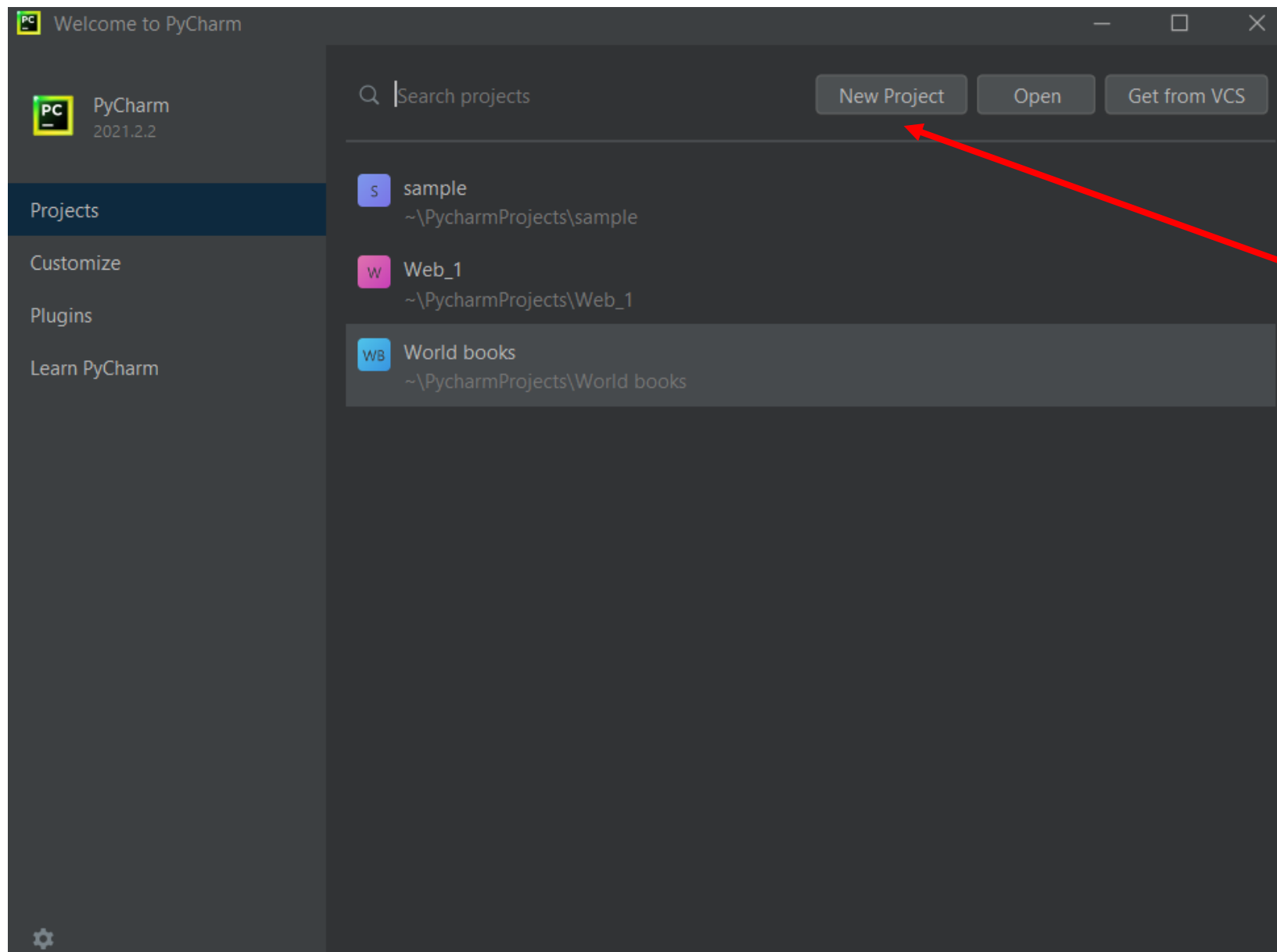
Шаг 1. Создать шаблон нового PyCharm-проекта

В главном меню **PyCharm** выбрать **File/New Project**.

В открывшемся окне в строке **Location** задать имя проекта, например, **Kino**

Нажать **Create**.





Создать проект



PC New Project

Location:

▼ Python Interpreter: New Virtualenv environment

☒ New environment using Virtualenv

Location:

Base interpreter: C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe

☐ Inherit global site-packages

☐ Make available to all projects

☐ Previously configured interpreter

Interpreter: <No interpreter>

☒ Create a main.py welcome script
Create a Python script that provides an entry point to coding in PyCharm.

Create Cancel

Указать имя проекта



Шаг 2. Описать среду разработки. Добавить в проект библиотеку Django

Выбрать в меню **File** опцию **Setting** для настройки параметров проекта.

В открывшемся окне выбрать опцию **Project Interpreter**.

Установить интерпретатор Python и библиотеку Django (в верхней строке поиска набрать **Django** и щелкнуть по кнопке **Install Package**).

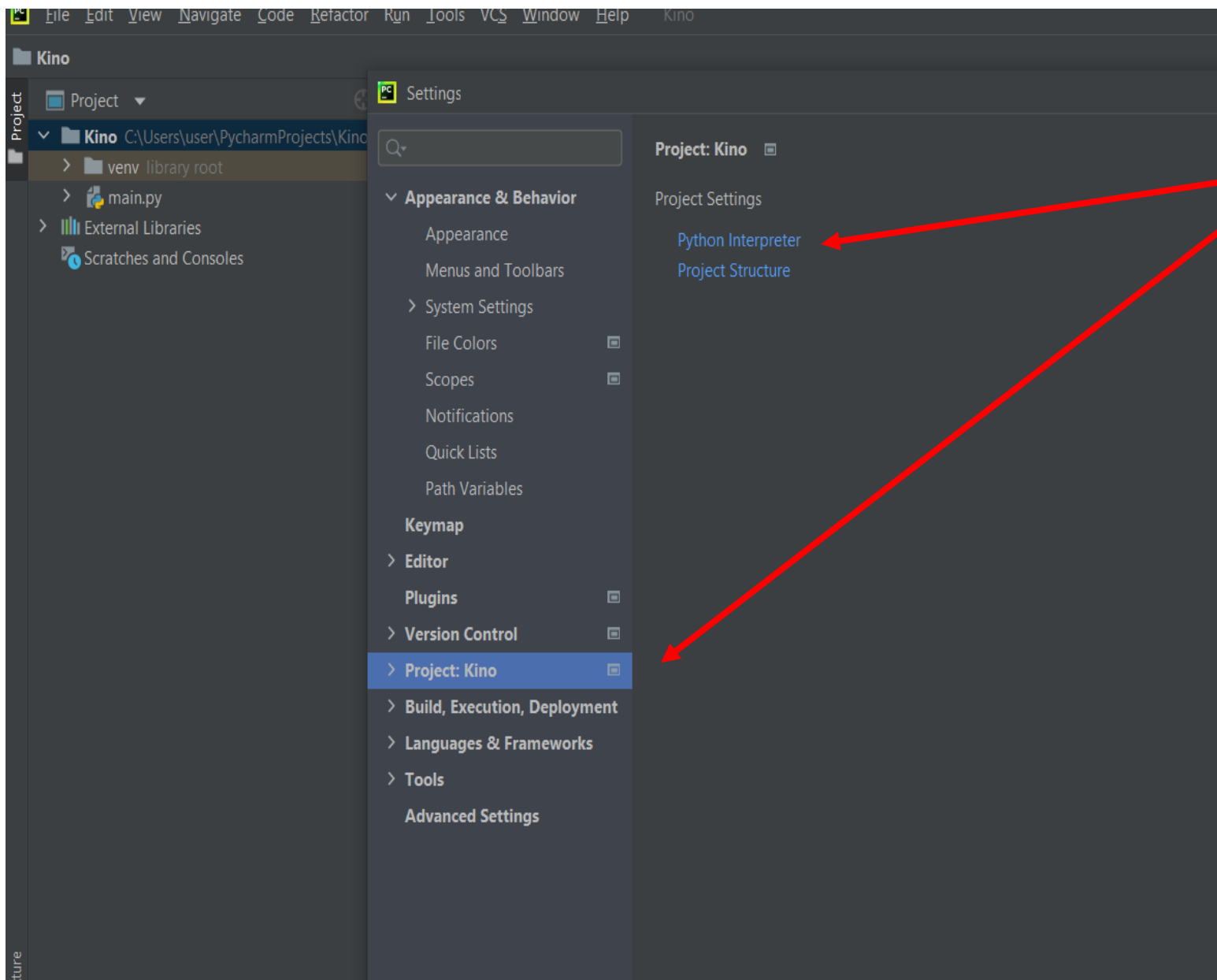
Библиотека Django будет добавлена в проект.

В папке виртуальной среды проекта Kino в папке Scripts появятся два файла - для выполнения команд управления проектом Django (например, команды startproject)

django-admin.exe

django-admin.py





Открываем меню File/Settings,
в окне выбираем позицию
Project Kino и далее
Python Interpreter



File Edit View Navigate Code Refactor Run Tools VCS Window Help Kino

Kino

Project

- Project
- Kino C:\Users\user\PycharmProjects\Kino
 - venv library root
 - main.py
- External Libraries
- Scratches and Consoles

Settings

- Appearance & Behavior
 - Appearance
 - Menus and Toolbars
 - System Settings
 - File Colors
 - Scopes
 - Notifications
 - Quick Lists
 - Path Variables
- Keymap
- Editor
- Plugins
- Version Control
- Project: Kino
 - Python Interpreter**
 - Project Structure
- Build, Execution, Deployment
- Languages & Frameworks
- Tools
- Advanced Settings

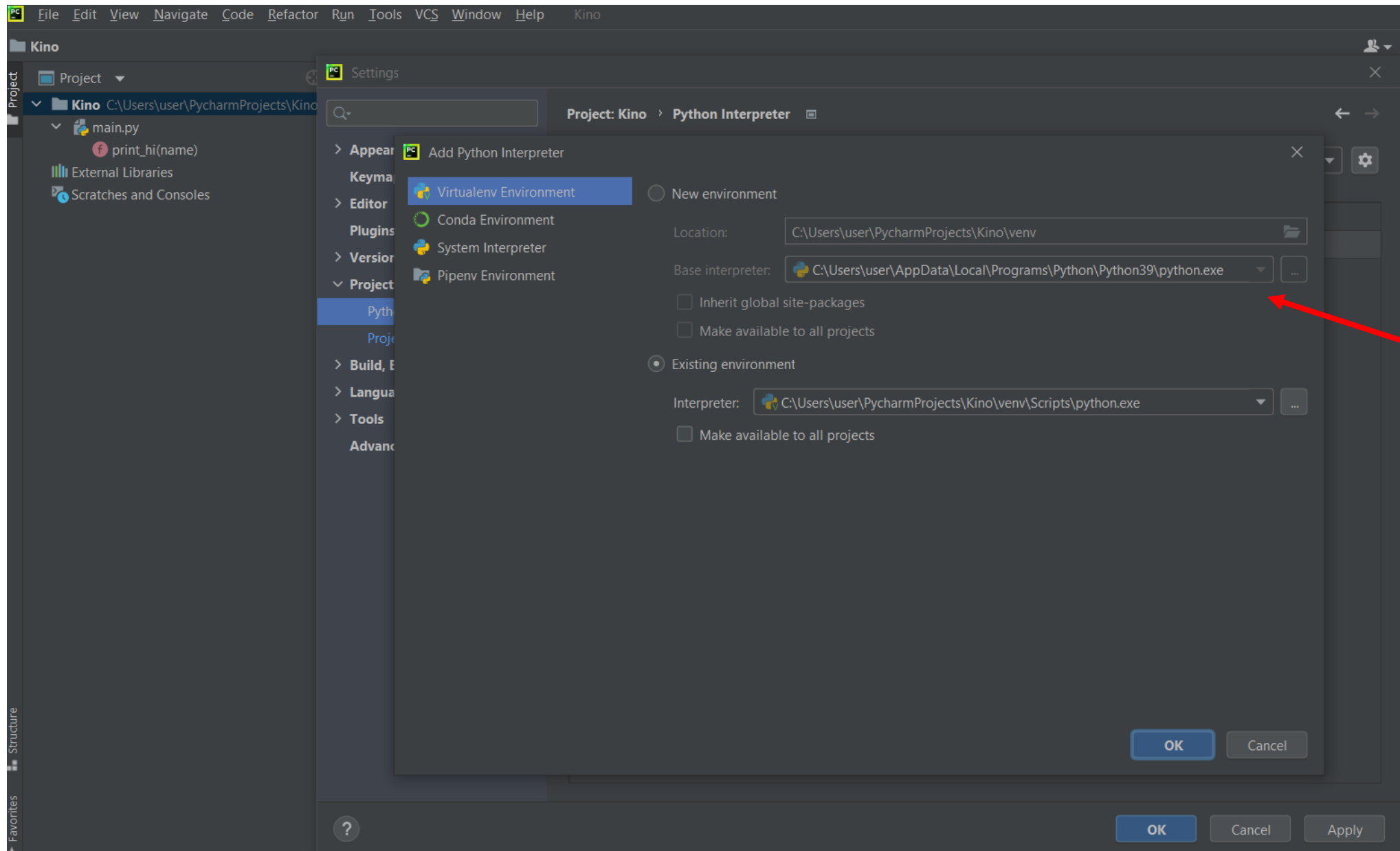
Project: Kino > Python Interpreter

Python Interpreter: <No interpreter>

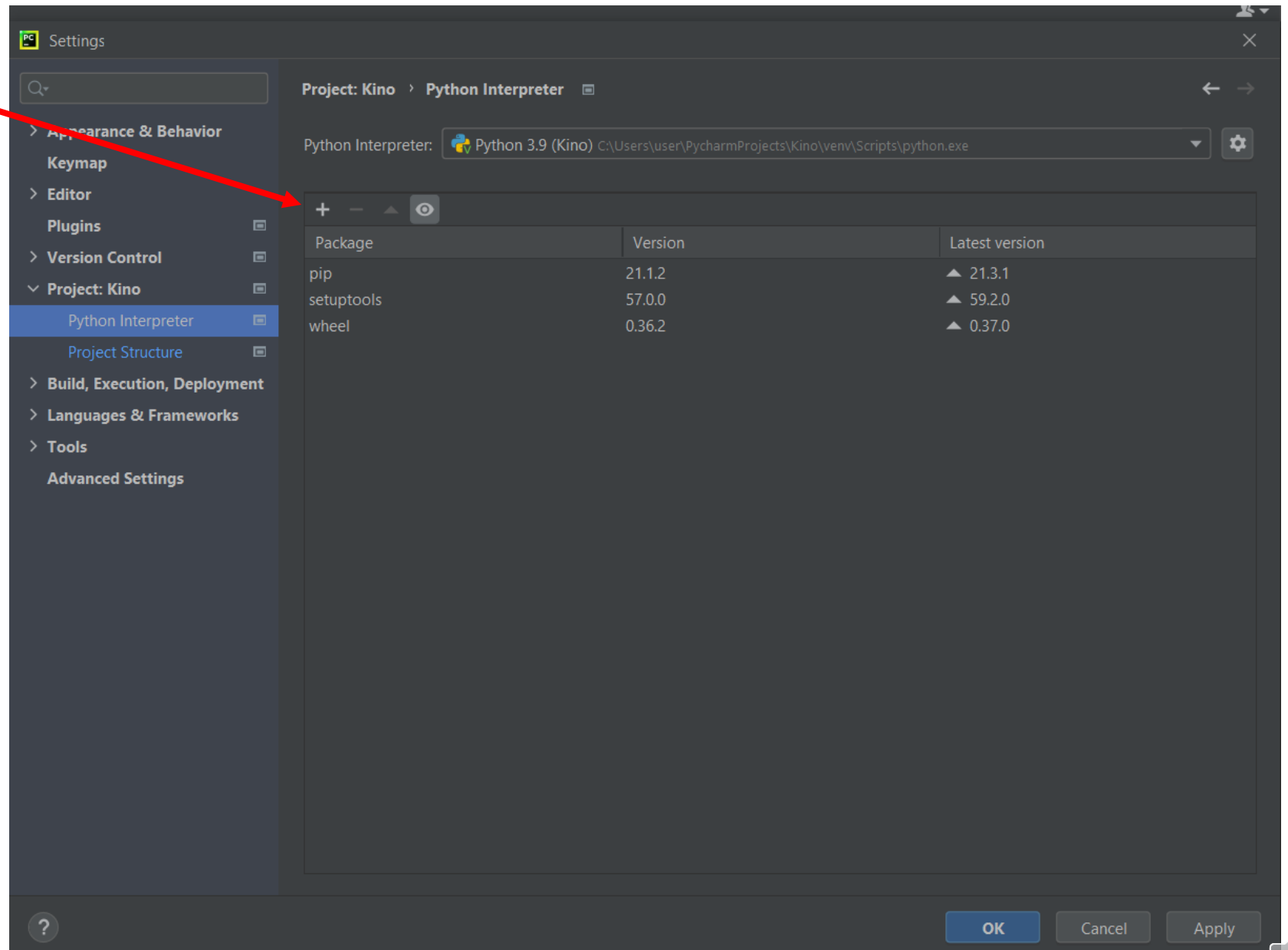
Reset

Package	Version	Latest version
Nothing to show		

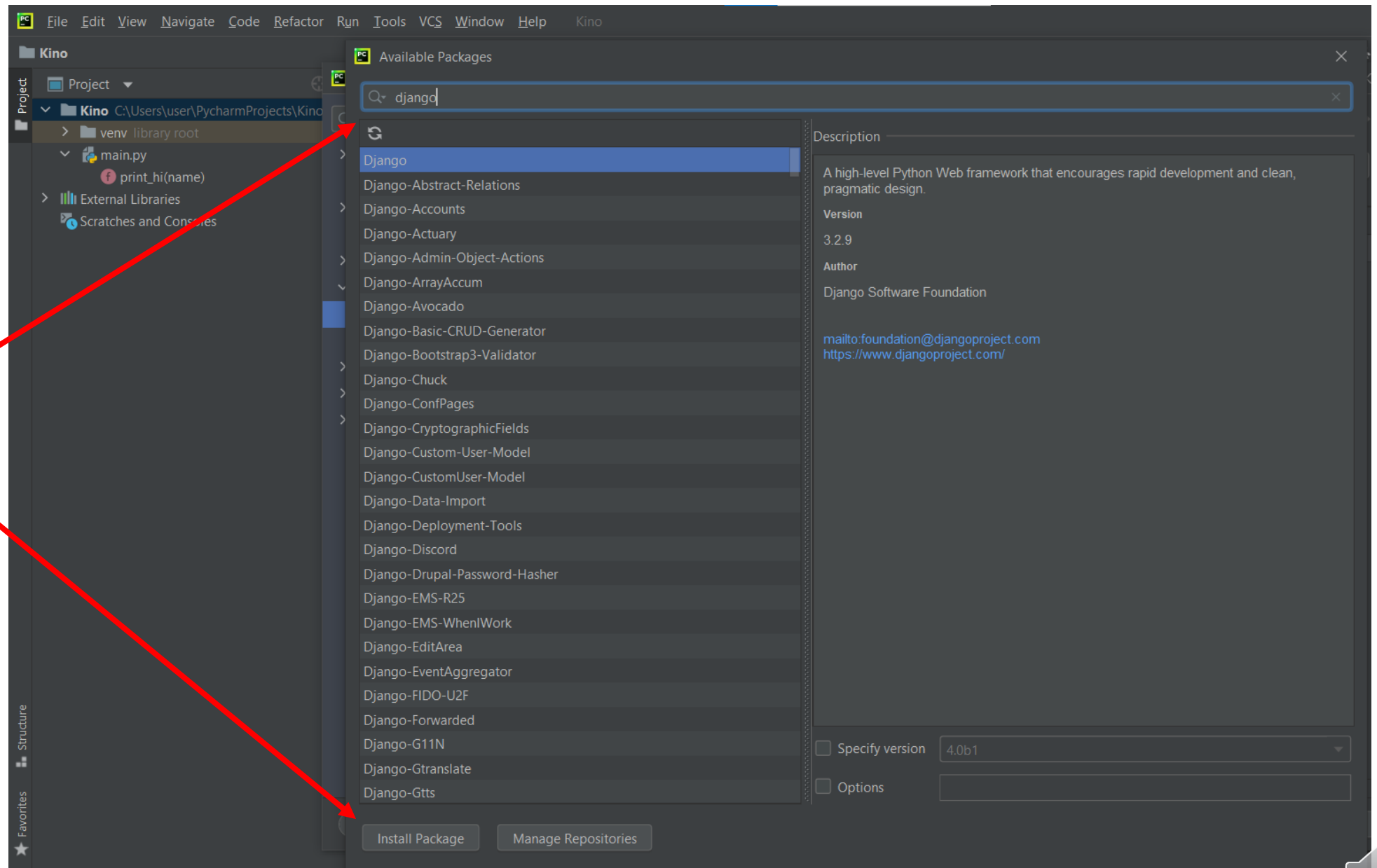




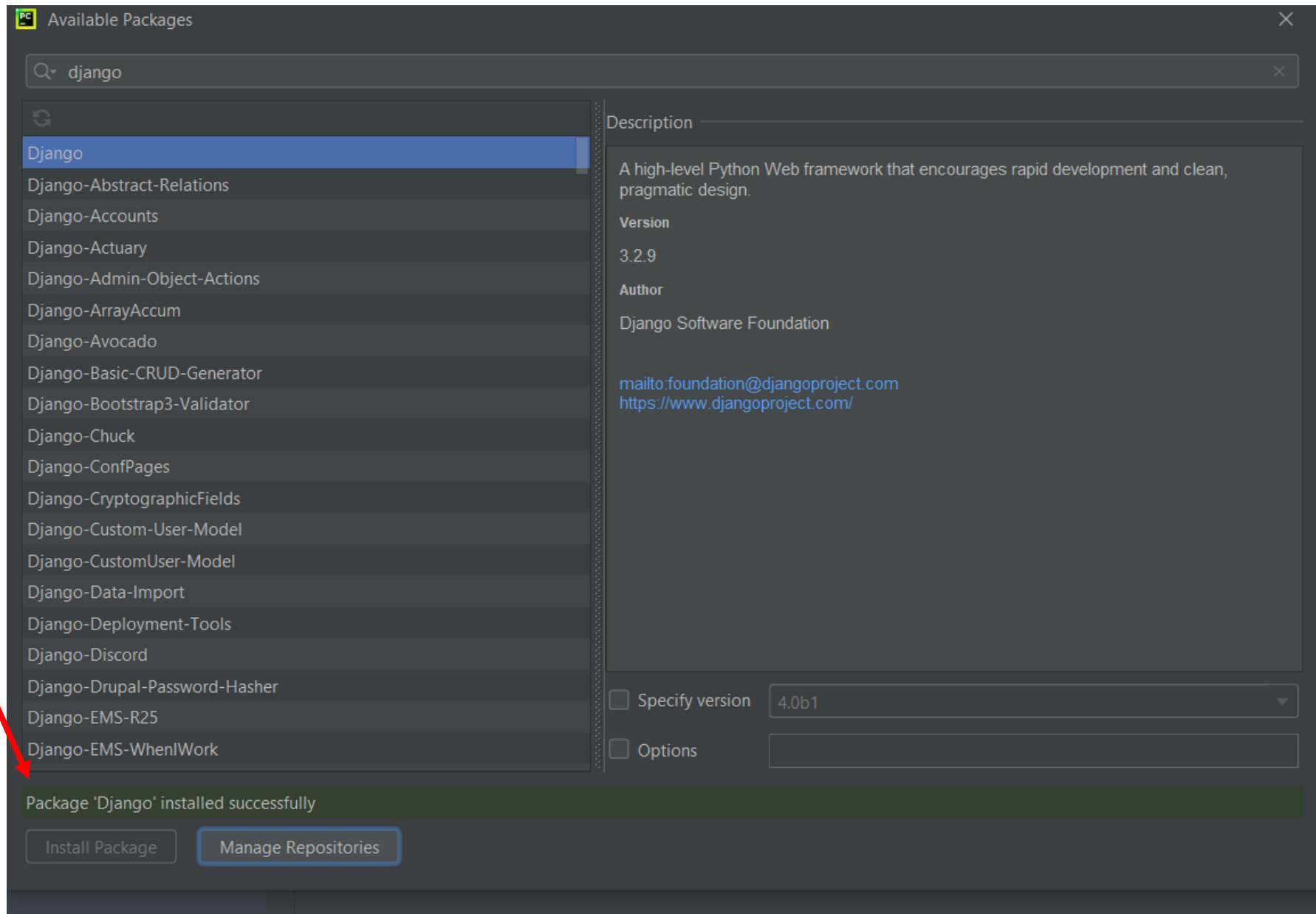
Подключаем Django



В окне набираем
Django и нажимаем
Install Package



Django успешно
установлен



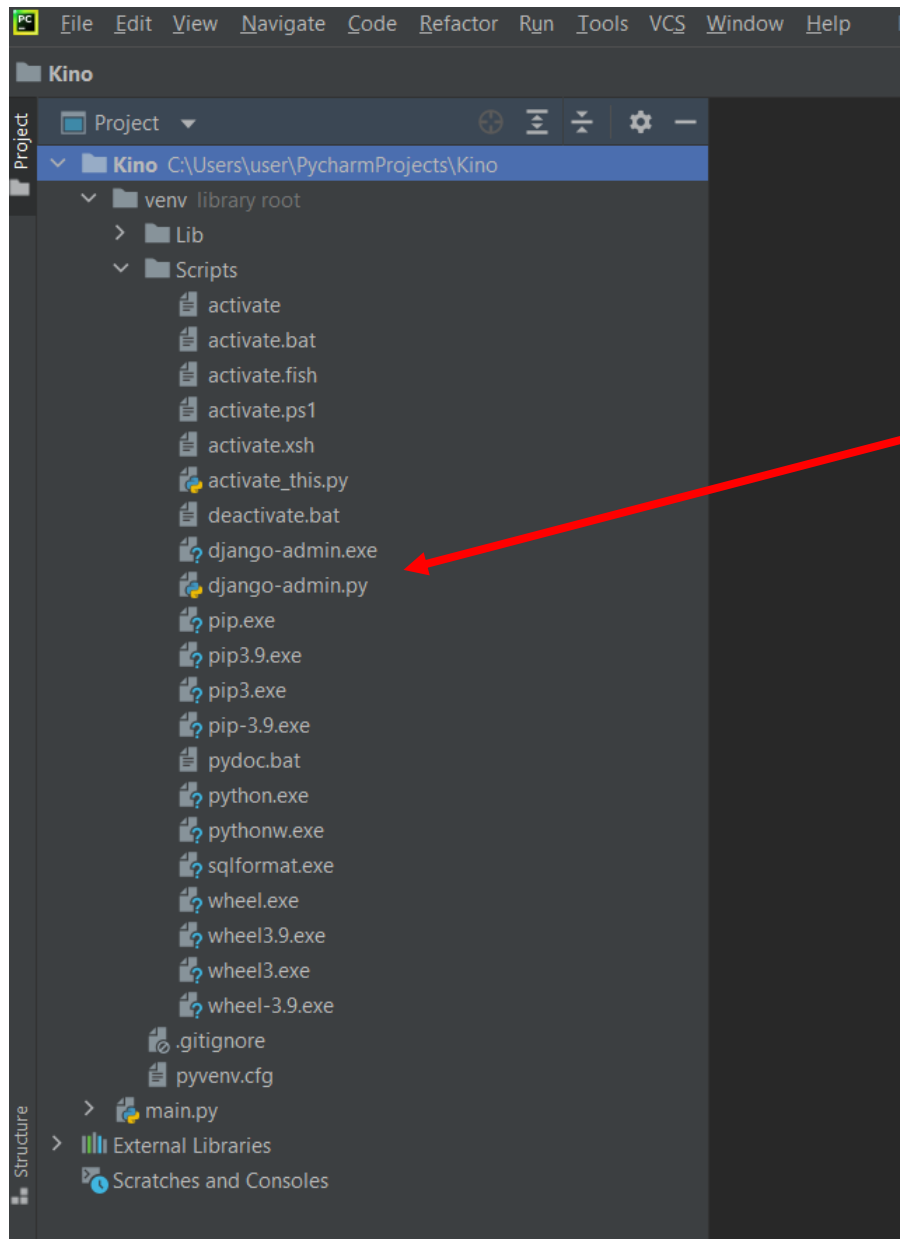
Settings

Project: Kino > Python Interpreter

Python Interpreter: Python 3.9 (Kino) C:\Users\user\PycharmProjects\Kino\venv\Scripts\python.exe

Package	Version	Latest version
Django	3.2.9	▲ 4.0b1
asgiref	3.4.1	3.4.1
pip	21.1.2	▲ 21.3.1
pytz	2021.3	2021.3
setuptools	57.0.0	▲ 59.2.0
sqlparse	0.4.2	0.4.2
wheel	0.36.2	▲ 0.37.0





Откроем папку Script каталога виртуальной среды venv.

Увидим, что в папке Script появились два файла

django-admin.exe

django-admin.py

Эти файлы нужны для выполнения команд управления проектом, например, команды startproject).



Создание первого проекта на Django

Шаг 3. Создание проекта Django

Перейти в окно терминала PyCharm (Terminal в строке в нижней части окна).

В окне терминала набрать и выполнить команду:

django-admin startproject film

```
PS C:\Users\user\PycharmProjects\Kino> django-admin startproject film
PS C:\Users\user\PycharmProjects\Kino> 
```

В папке PyCharm-проекта **Kino** будет создана папка Django-проекта **film**.



Структура проекта на Django

Kino - папка корневого каталога, контейнер проекта, будет содержать все приложения и файлы управления проектом

Kino /film - внешняя папка web-проекта **film**

manage.py - файл для управления из командной строки

db.sqlite3- база данных SQLite. Создается по умолчанию для хранения данных проекта

Kino /film /film - внутренняя папка web-проекта **film**

film__init__.py – файл для указания и регистрации пакетов

film/asgi.py – файл точки входа для ASGI-совместимых вебсерверов

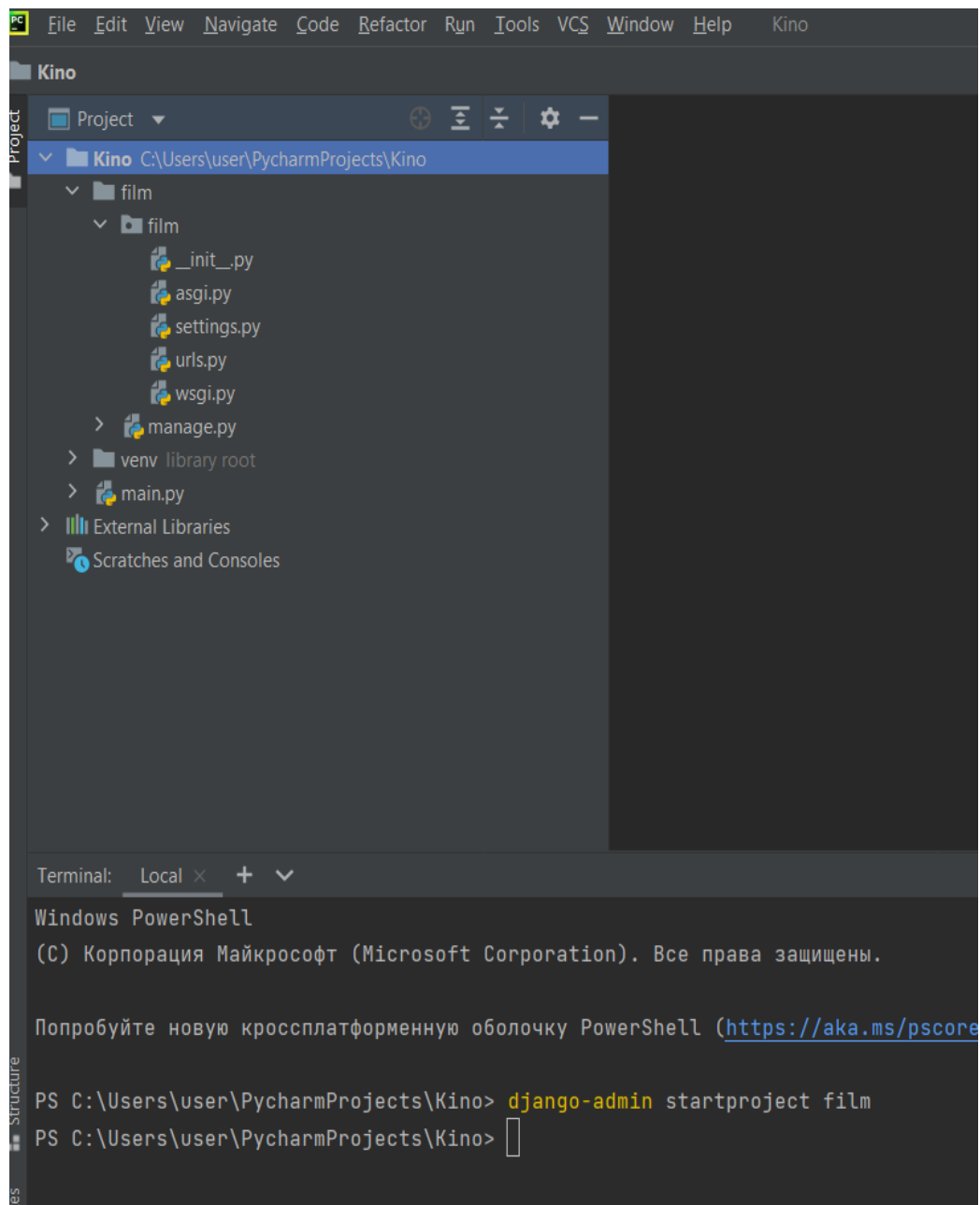
WSGI ([англ. Web Server Gateway Interface](#)) — стандарт взаимодействия между [Python](#)-программой, выполняющейся на стороне сервера, и самим [веб-сервером](#) **ASGI** – духовный наследник WSGI, добавляющий поддержку асинхронный приложений.

film/setting.py – файл установок и конфигурации текущего проекта Django.

film/urls.py – таблица контента Django-проекта (URL-декларации)

film/wsgi.py - файл точки входа для WSGI-совместимых вебсерверов





Структура проекта Kino

Kino - папка корневого каталога, контейнер проекта, будет содержать все приложения и файлы управления проектом

Kino /film - внешняя папка web-проекта **film**

manage.py - файл для управления из командной строки

db.sqlite3 - база данных SQLite. Создается по умолчанию для хранения данных проекта

Kino /film /film - внутренняя папка web-проекта **film**

film__init__.py – файл для указания и регистрации пакетов

film/asgi.py – файл точки входа для ASGI-совместимых вебсерверов

WSGI ([англ. Web Server Gateway Interface](#)) — стандарт взаимодействия между [Python](#)-программой, выполняющейся на стороне сервера, и самим [веб-сервером](#) [ASGI](#) – духовный наследник WSGI, добавляющий поддержку асинхронных приложений.

film/setting.py – файл установок и конфигурации текущего проекта Django.

film/urls.py – таблица контента Django-проекта (URL-декларации)

film/wsgi.py - файл точки входа для WSGI-совместимых вебсерверов



Шаг 4. Запуск Django-проекта на исполнение

В окне Терминала PyCharm выполнить команды:

cd film (переход в папку **film**)

python manage.py runserver (запуск на компьютере отладочного веб-сервера с адресом <http://127.0.0.1:8000>)

```
PS C:\Users\user\PycharmProjects\Kino> cd film
PS C:\Users\user\PycharmProjects\Kino\film> python manage.py runserver
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

по щелчку на ссылке веб-сервера откроется веб-браузер и в него будет загружена страница поздравления с успешной установкой Django.

Здесь адрес локального сервера (хоста) 127.0.0.1

8000 – адрес порта.





Установка прошла успешно! Поздравляю!

Эта страница отображается, так как `DEBUG=True` находится в файле параметров и URL-адреса не настроены.



Документация Django
Темы, ссылки и инструкции



**Учебник: приложение для
опроса**
Начало работы с Django



Сообщество Джанго
Подключайтесь, получайте помощь
или вносите свой вклад

