Django Framework

development backend web apps

For beginners

Урок 1. Основные понятия и установка IDE

Основные понятия Django

Django является Python веб-фреймворком высокого уровня, который помогает делать быструю разработку и чистый, прагматичный дизайн. Другими словами, это полноценный веб-фреймворк “с батарейками” в комплекте (“batteries included”).

Готовый интерфейс для администратора, одна из особенностей Django, чрезвычайно полезен для ввода данных и администрирования (рис. 1). Документация Django была высоко оценена профессиональным сообществом, так как она чрезвычайно хорошо написана для open-source проекта (https://docs.djangoproject.com/en/4.0/).

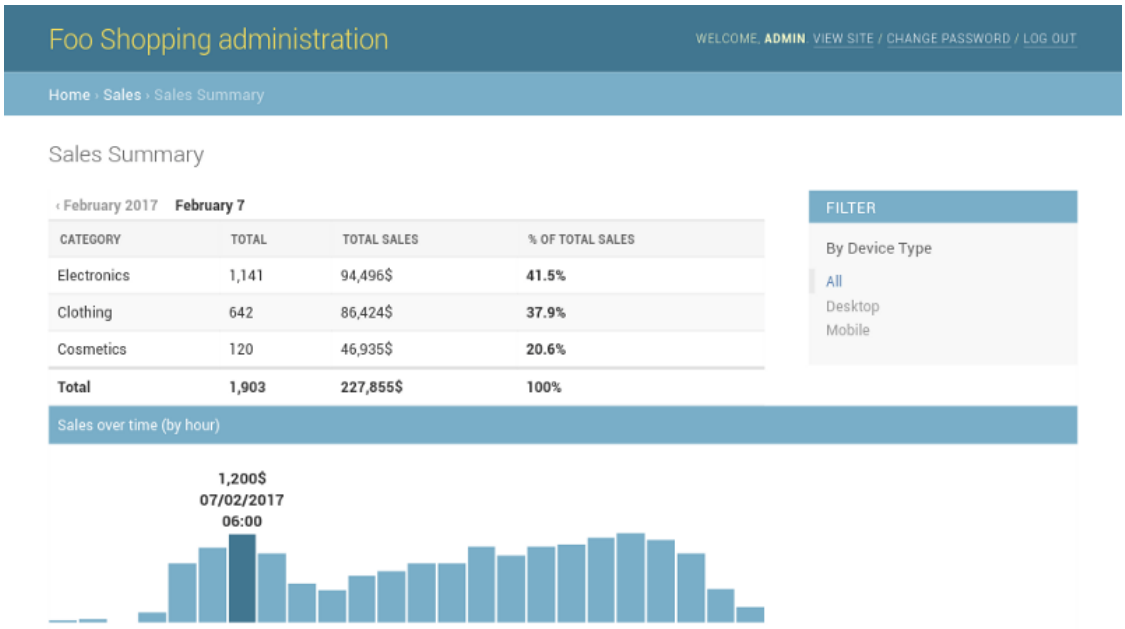


Рис. 1. Интерфейс административной панели Django

Наконец, Django прошёл боевое тестирование на сайтах с высоким трафиком (например, Instagram, YouTube и многие другие). Он с большим вниманием относится к безопасности. Он предоставляет защиту от распространённых атак, таких как межсайтовый скриптинг (XSS), подделка межсайтовых запросов, кликджэкинг.

Django можно использовать для создания любого веб-приложения, но это может быть не лучшим вариантом для каждого из вариантов использования. Например, для создания прототипа простого веб-сервиса с жёсткими ограничениями памяти лучше использовать Flask, в то время как в конечном итоге можно перейти на Django из-за его надёжности и широких возможностей.

Одна из наиболее распространённых архитектур приложений, которую используют в современной web-разработке - это шаблон Model — View — Template (MVT). Он обеспечивает чёткое разделение задач и обязанностей между важными аспектами приложения (рис. 2). Сделаем обзор этих компонентов:

- Модель (Model) отвечает за управление данными и основной бизнес-логики.
- Представление (View) отображает эти данные для пользователя.
- Шаблон (Template) представляет собой результат обработки запроса, сгенерированный, например, в HTML-файл.

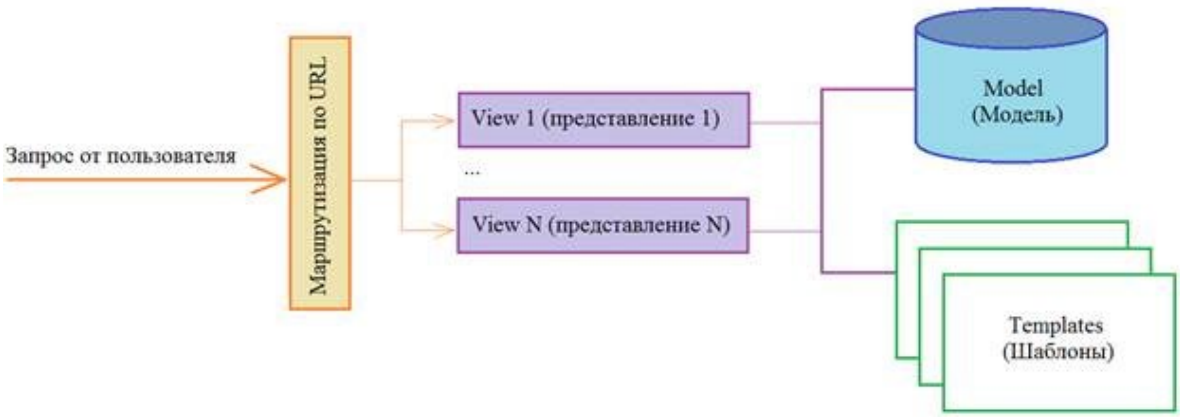


Рис. 2. Представление архитектуры MVT

По итогу, разделение задач на отдельные группы служит следующим целям:

- Код для определенного набора задач гораздо более удобен в обслуживании, поскольку он не предполагает работу с совершенно несвязанными частями приложения. В целом, эта концепция называется разделением интересов и применима во всей разработке ПО.

- Разработка приложений становится более гибкой, поскольку несколько совершенно разных уровней представления и контроллера могут подключаться к одному уровню модели. Это позволяет различным приложениям совместно использовать одну и ту же бизнес-логику и данные, представляя их и взаимодействуя с ними по-разному для различных пользователей.

Model

Модели Django обеспечивают легкий доступ к базовому механизму хранения данных, а также могут инкапсулировать любую основную бизнес-логику, которая всегда должна оставаться в силе, независимо от того, какое приложение её использует.

Модели существуют независимо от остальной части системы и предназначены для использования любым приложением, имеющим к ним доступ. Фактически, методы манипулирования базой данных могут быть использованы даже из интерактивного интерпретатора, без загрузки веб-сервера или какой-либо специфичной для приложения логики.

В следующих уроках будет подробно рассмотрены модели Django, в том числе то, как они определяются и используются, как включить собственную бизнес-логику и многое другое.

Views

Представления, несмотря на то, что они имеют общее название с исходным определением MVC, Django views имеют мало общего с традиционным определением. Вместо этого они объединяют часть ответственности традиционного представления со всеми задачами контроллера. Представление принимает вводимые пользователем данные (простые запросы), ведёт себя в соответствии с логикой взаимодействия приложения и возвращает отображение, подходящее пользователям для доступа к данным, представленным моделями.

Представления обычно определяются как стандартные функции Python, которые вызываются, когда пользователь запрашивает определенный URL-адрес. Даже простой запрос информации считается действием, поэтому представления предназначены для обработки этого наряду с изменениями данных и другими отправками. Представления могут получать доступ к моделям, извлекая и обновляя информацию по мере необходимости для выполнения задачи, запрошенной пользователем.

Поскольку представления просто вызываются как функции, не требуя какой-либо конкретной структуры, их можно указать несколькими способами. Помимо простой функции, представление может принимать форму любого Python callable, включая классы, методы экземпляра, функции-декораторы.

Настройка URL

Django предоставляет отдельный слой, чтобы сделать представления доступными для внешнего мира по определённым URL-адресам. Предоставляя регулярное выражение в качестве компонента URL, одно объявление может содержать широкий спектр конкретных URL-адресов в удобочитаемом и удобном для обслуживания виде.

Эта конфигурация определяется отдельно от самих представлений, чтобы позволить настраивать представление по нескольким URL-адресам, возможно, с разными параметрами в каждом месте.

Самое важное из всего этого то, что наличие URL-адресов в качестве отдельной части процесса побуждает разработчиков рассматривать URL-адреса как часть общего дизайна приложения. Поскольку они должны использоваться в закладках, сообщениях в блогах и маркетинговых кампаниях. URL-адреса иногда более заметны, чем приложение. В конце концов, пользователи, которые просматривают веб-страницы, увидят ваш URL-адрес еще до того, как решат посетить ваш сайт.

Template

В то время как views отвечают за представление данных пользователю, задача представления этих данных обычно делегируется шаблонам, которые являются важной частью разработки Django.

Django предоставляет простой язык шаблонов для этой цели, так что разработчикам шаблонов не нужно изучать Python просто для работы с шаблонами. Язык шаблонов не зависит от какого-либо конкретного языка представления. Он в основном использует формат HTML, но может быть использовать для создания любой текстовый формат.

Нужно иметь в виду, что этот механизм шаблонов - это всего лишь один инструмент, который могут использовать представления для отображения пользователю. Многие представления могут использовать HTTP-перенаправления на другие URL-адреса.

DRY – Don’t Repeat Yourself

Если вы не новичок, то хорошо должны знать, как легко писать “шаблонный” код. Вы пишете код один раз для одной цели, затем снова для другой, и снова, и снова. Через некоторое время вы понимаете, сколько кода было продублировано, и, если вам повезёт, у вас есть время и силы, чтобы посмотреть, что общего, и переместить эти фрагменты в общее место, например, обернув повторяющийся код в функцию. Этот процесс - один из основных причин существования фреймворка.

Фреймворки обеспечивают большую часть этого общего кода, пытаясь упростить его, чтобы избежать дублирования кода в будущем. В совокупности это представляет собой обычную практику программирования: не повторяйтесь.

Основная идея данного подхода заключается в том, что необходимо написать что-то только один раз. Это снижает риск случайного введения несоответствия между двумя фрагментами кода, которые должны совпадать.

Установка IDE и Django

В качестве среды разработки для реализации учебного проекта будем использовать IDE PyCharm Community. Она бесплатная и имеет ряд преимуществ:

- Автокомплит и рефакторинг кода
- Удобный графический debugger
- Поддержка последних версий Django (то, что нам и нужно)

Общее требование для установки — наличие на вашем внешнем накопителе свободного места не менее 2,5 Gb.

Для Windows:

Для начала установки необходимо скачать дистрибутив с официального сайта JetBrains — https://www.jetbrains.com/ru-ru/pycharm/download/. После скачивания запустите .exe файл и следуйте всем инструкциям, прописанным в нём.

Для Ubuntu:

Здесь существует 2 способа установки:

- через магазин приложений Ubuntu Software

- через терминал

1 способ:

В магазине приложений Ubuntu Software введите в поисковую строку «PyCharm Community». Далее нажмите на «Install / Установить».

Установка займёт не более 10 минут (зависит от скорости соединения вашего провайдера). После завершения иконка PyCharm появится в меню, и Вы сможете совершить свой первый запуск.

2 способ:

Зайдите в терминал и введите следующую команду для начала установки:

```
codeby@PC:~$ sudo snap install pycharm-community --classic
```

В данному случае установка будет производиться через утилиту, встроенную в Ubuntu, snap — система для развёртывания и управления пакетами.

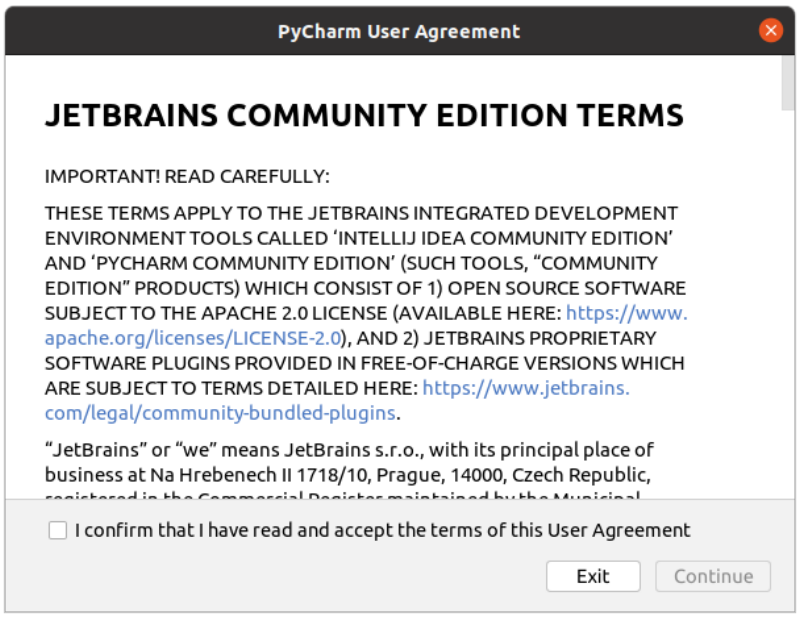
Перед установкой терминал попросит Вас ввести пароль для root-пользователя.

После завершения Вы увидите подобную запись о успешной установке:

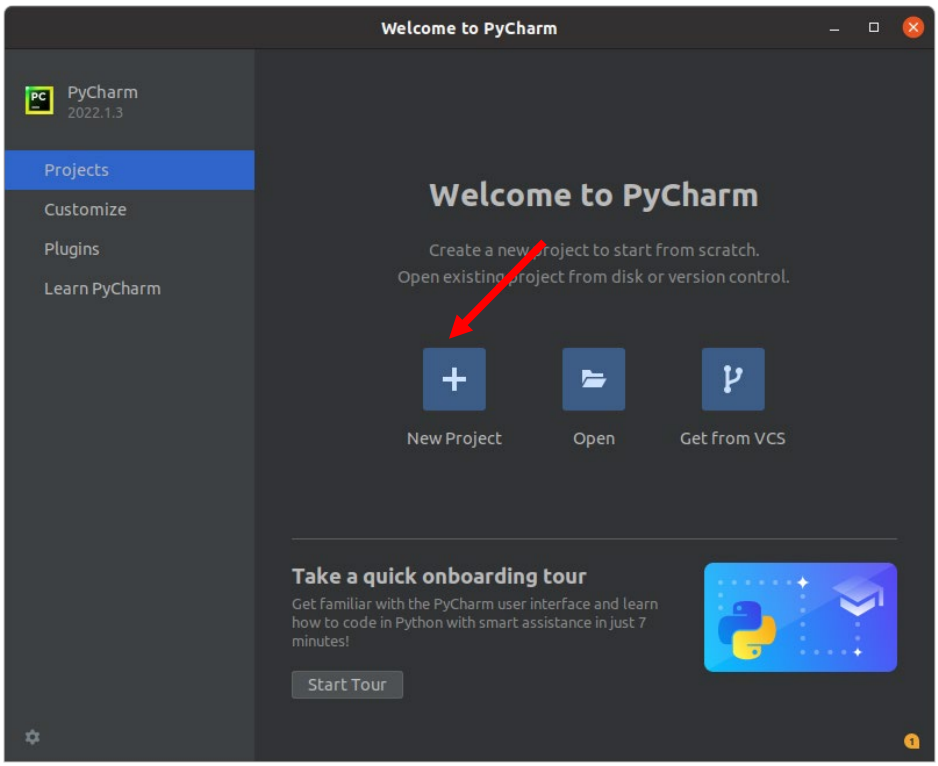
```
pycharm-community 2022.1.3 from jetbrains✓ installed
```

Первый запуск

При первом запуске IDE попросит Вас ознакомиться с лицензией и поставить галочку с её согласием.



Далее откроется меню, предназначенное для управления проектами. Так как у нас пока ещё нет проекта, то пришло время его создать.

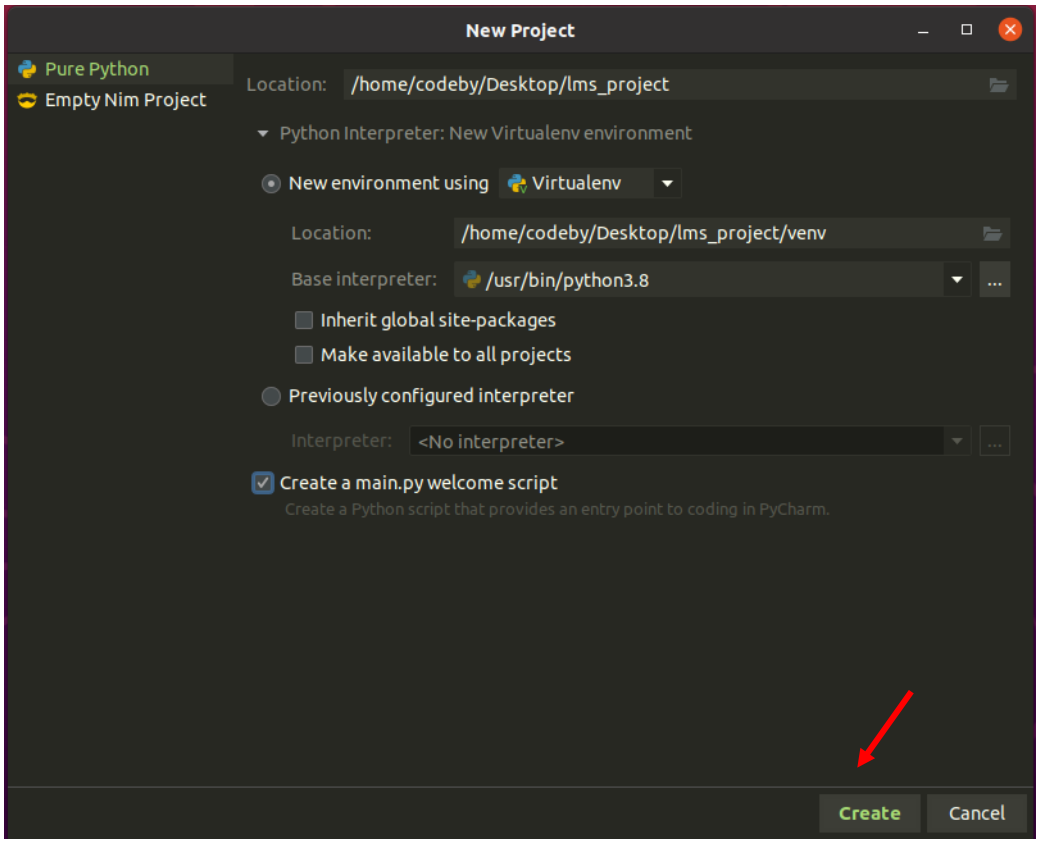


Для этого нажмите на кнопку «New Project»

Появится меню для указания настроек проекта

В окошке для Location укажем имя нашего проекта. В ходе данного курса мы разработаем прототип веб-приложения для системы онлайн-обучения (LMS). Поэтому назовём его «lms_project». Вы можете выбрать для себя произвольное название.

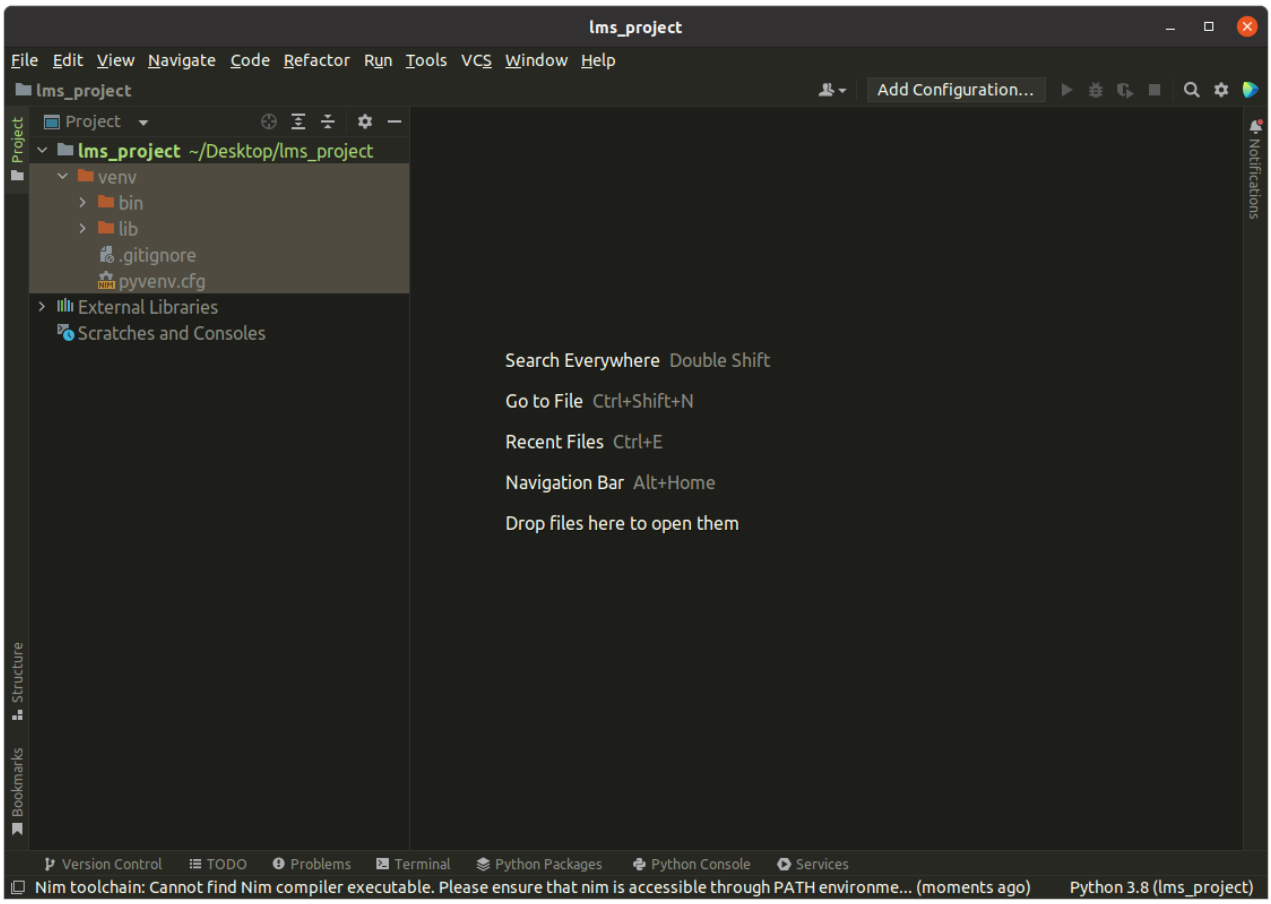
Далее выберите Python-интерпретатор для создаваемого виртуального окружения (рекомендую в проекте использовать Python 3.x версию, так как самые свежие версии Django поддерживают именно её). Также желательно НЕ выбирать глобальные python-пакеты, потому что есть вероятность возникновения конфликтов между версиями библиотек (например, в глобальном есть Django 2.2, а Вам нужна Django 3.2.2)



В итоге у нас получилась следующая настройка проекта:

Далее нажимаем «Create» в правом нижнем углу

Появляется основное окно IDE, в котором можно начинать писать проект.



Поздравляю! Вы только что успешно завершили установку IDE и настроили свой первый проект.

Чтобы начать использовать Django, нам нужно ещё его установить из питоновского магазина библиотек PyPI (Python Package Index).

Сделать это можно так же 2 способами:

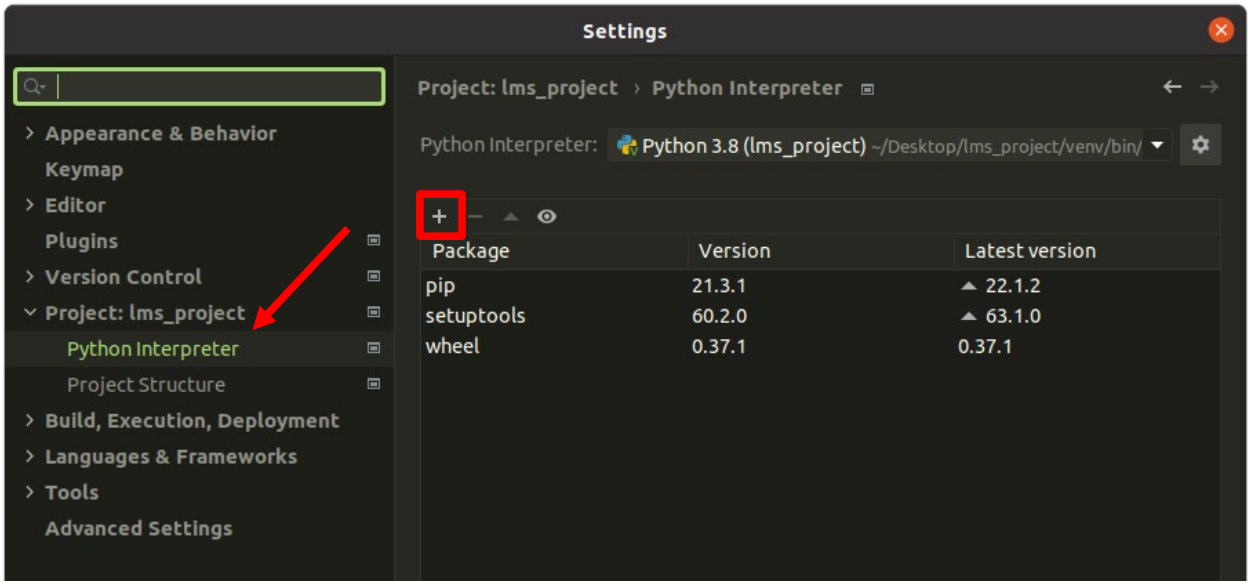
- через интерфейс IDE PyCharm

- через терминал с помощью пакетного менеджера pip

1 способ:

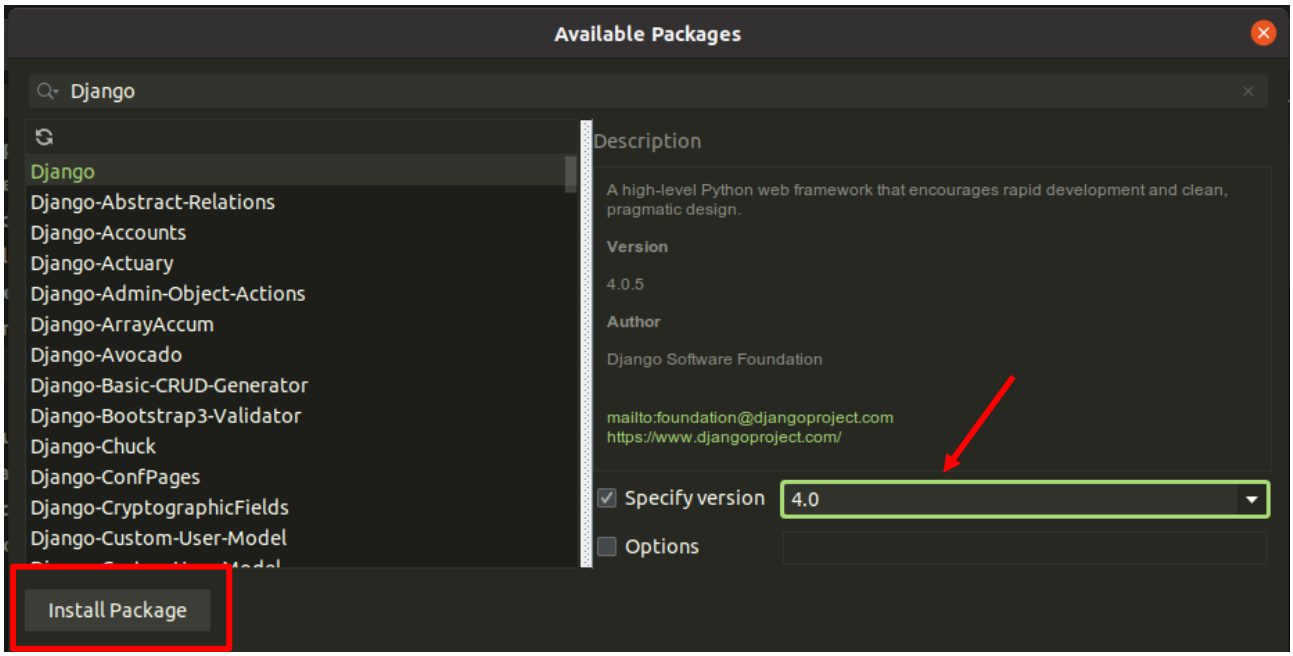
Перейдите в пункт меню «File → Settings» или используйте сочетание клавиш Ctrl+Alt+S

В окне Settings найдите меню Python Interpreter



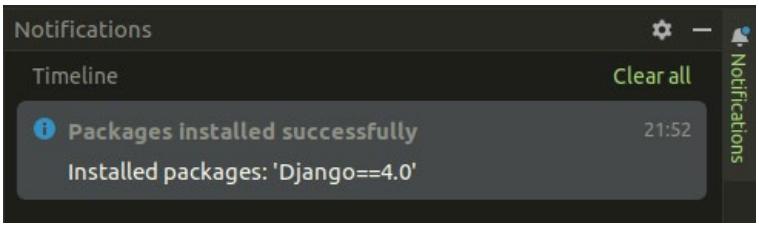
Появится список установленных библиотек в ранее созданном виртуальном окружении. Пока что установлены библиотеки по умолчанию. Для установки Django нажмите на «+».

В окне «Avaliable Packages» Вы можете найти все нужные библиотеки, написанные на Python и хранящиеся в PyPI.



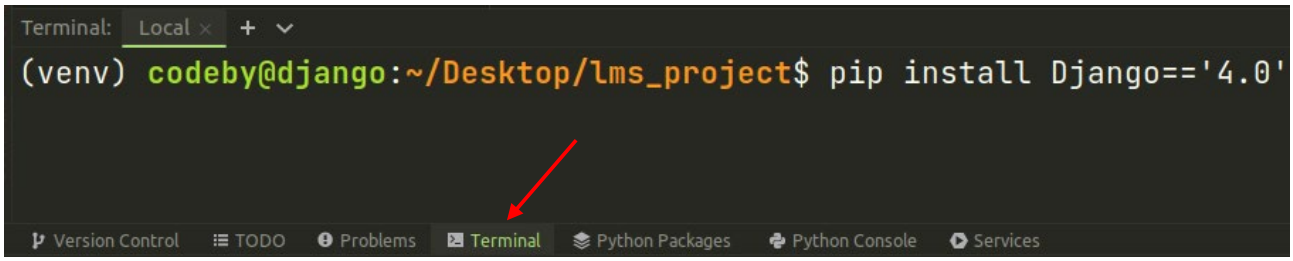
В поисковой строке ищем Django. Наш проект мы будем писать с использованием версии Django 4.0 — это самая свежая и стабильная версия на данный момент.

После выбора нажмите «Install Package». Интерфейс IDE сообщит Вам об успешном завершении скачивания фреймворка в виде следующего уведомления:

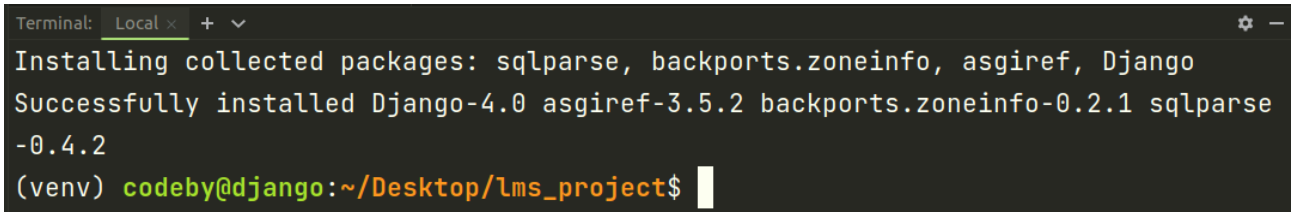


2 способ:

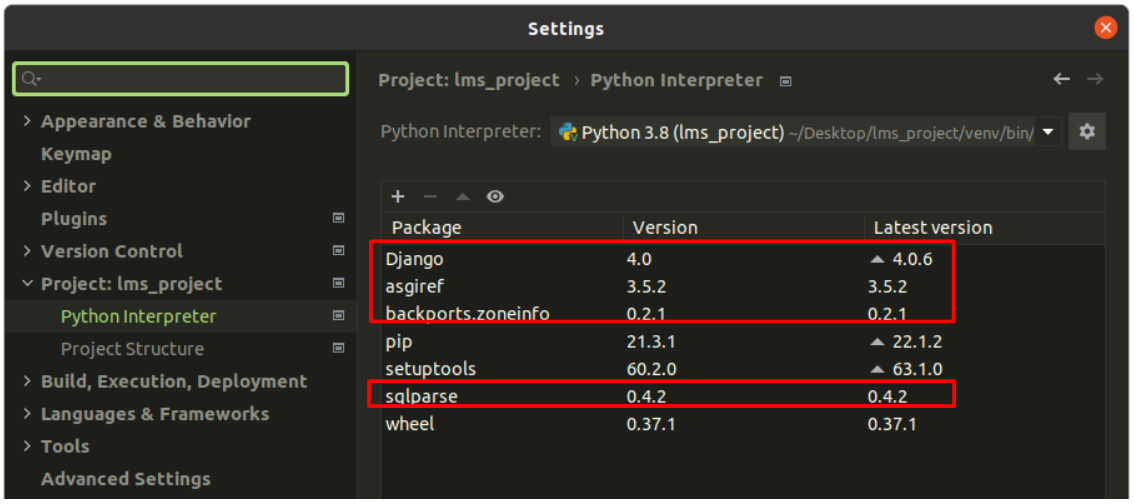
На нижней панели можно увидеть Инструменты окна, в которое нужно перейти, чтобы открыть активировать оболочку командной строки. В нём автоматически терминал перейдёт в директорию нашего проекта. После этого пишем команду для установки Django, показанную на картинке ниже.



Успешность установки в данном случае определяется конечным сообщением о статусе.



После установки мы можем убедиться, что фреймворк точно появился в виртуальном окружении. Для этого перейдите снова в «Python Interpreter» или в «Python Packages» всё в тех же Инструментах окна и затем увидите новые библиотеки.



Как мы можем увидеть, вместе с Django установились ещё 3 дополнительные библиотеки: asgiref, backports.zoneinfo, sqlparse. Для чего они?

- asgiref обеспечивает стабильную работу тестового веб-сервера с сайтом

- zoneinfo нужен для корректной работы с временными зонами

- sqlparse обеспечивает разбор SQL-кода, необходимый ORM для формирования запросов

В целом, работа по установке/настройке IDE/Django успешно завершена.

В следующем уроке мы создадим Django-проект и рассмотрим его структуру и настройки.

Домашнее задание

✓ Установите на свой ПК IDE PyCharm и Django framework, настройте проект.

✓ Внимательно изучите интерфейс IDE PyCharm: попробуйте настроить под себя: шрифт, цветовую гамму; изучите горячие клавиши для повышения эффективности вашего взаимодействия с IDE, работу с файлами (создание, удаление, переименование).