**2023/12/22 00:00|Лекция. Обзор сторонних библиотек. Часть 1.**

**Обзор сторонних библиотек. Часть 1**

На новом модуле, посвящённом сторонним библиотекам языка python, мы будем изучать:

* как можно искать наши сторонние библиотеки;
* как их можно импортировать в свой проект;
* как их правильно импортировать;
* какие именно библиотеки или пакеты импортировать;
* как можно обустроить для себя виртуальное окружение.

В общем, все, что вот с этим связано будем потихонечку проходить поэтому нахватайтесь терпения.

И так с самого начала необходимо рассказать про различия понятий, а именно: пакет, модуль, фреймворк, библиотека и т.д., то есть вообще, чем отличаются данные понятия друг от друга(рис.1).



Рис.1

Почему вообще эти понятия часто путают? У нас есть самый главный подпункт— это модуль. **Модуль** — это какой-то обычно один файл с расширением .py, то есть обычный Python файл. Он содержит в себе различные классы, функции и так далее. Если захотим импортировать модуль, то они будут маленькие. Там он будет немного помогать решать некоторую важную функцию или то, что до вас давно уже придумали, решили и так далее. То есть это обычный один такой маленький Python файл.

Что такое пакет? **Пакет** — это уже как раз сборник разных модулей. Это сборник разных маленьких, а может быть и больших Python файлов. Они вместе с собой как-то работают, взаимодействуют и так далее. Причём термин пакет в Python равносилен термину библиотека.**Так что, если будем говорить либо пакет, либо библиотека, это будет значить одно и то же.**Между пакетом и библиотекой почти различий нет, и там, и там могут быть свои подпакеты, то есть и у библиотеки могут быть свои подпакеты, которые содержат одни Python файлы, так и у пакета могут быть свои подпакеты, которые содержат уже свои Python файлы, то есть вот такая вот архитектура.

Ещё есть такая штука как фреймворк. **Фреймворк** — это то же самое, что и пакет библиотека. Есть маленькие фреймворки, есть большие фреймворки, состоящие из некоторых библиотек. Точно также и библиотека, может состоять из некоторых библиотек. В чем большое различие между фреймворком и пакетом с библиотекой?

Во-первых, зачастую фреймворк намного больше, чем библиотека, а библиотека зачастую реально больше, чем пакет. Обычно принято называть библиотеками то, что побольше, чем пакет. Первым идёт у нас фреймворк, потом идёт уже библиотека и дальше пакет, но категорично это условное различие. Если вы назовёте какой-нибудь пакет, то же самое библиотека или наоборот, все будет нормально, но между фреймворком и библиотека вместе с этим пакетом есть важное различие. Фреймворк сам определяет те правила, по которым будет работать программист. То есть, если взять популярный django, то там невозможно будет пойти против его правил. Сам django декларирует нам правила, что сперва необходимо создать главное приложение, вообще активировать это приложение, создавать под приложение и так далее. То есть мы играем по правилам фреймворк. Можно им пользоваться, но адаптировать полностью под себя - нет.

В случае с разными сторонними библиотеками, мы можем спокойно их взять и просто внедрить в свой проект, каким бы он там ни был, большим или маленьким, и просто использовать те функции,находящиеся в этих пакетах или библиотеках. В этом и есть важное различие.

Теперь осталось понять, как скачать разные пакеты, библиотеки или даже, может быть, модули прям в наше виртуальное окружение, то есть именно к нашему Python так, чтобы у нас все скачалось правильно, все работало правильно, и могли импортировать все как надо.

Чтобы это узнать, перед этим **рассмотрим этапы создания стороннего модуля.** Допустим мы программисты, написали свою программу, может быть даже пакет, и хотим его закинуть на какие-нибудь сервисы, которые там предоставляют этот код всем и его мог использовать каждый человек.

1. Для начала пишется код каким-либо разработчикам, и зачастую этот код пишется с системой контроля файлов, ну как, например, git.
2. Далее этот код, эта программа, эта папка загружается на сервера, поддерживающие Open Source. Такими серверами, например, является GitHub. Там содержится очень много проектов. Любой человек может их использовать, добавить к себе репозиторий и даже может его видоизменять.
3. Так как таких проектов на GitHub много и там есть очень много различных языков, GitHub работает со всеми языками не только с python, и найти как бы нормальный работающий пакет или не пакет тяжело, поэтому разработчики придумали свой сайт PyPi.org. Все крупные проекты, которые в будущем будут использоваться как пакеты, загружаются на PyPi.org и оттуда их легко уже скачать с помощью pip.

**Давайте посмотрим основные сайты, которые сегодня уже упоминались.**

Первый сайт — это естественно, GitHub. Здесь показано его главное меню(рис.2) и чтобы что-то здесь найти, можем просто в поиске ввести. Для примера нам нужна библиотека. Она будет работать, допустим, с mp3 файлами. Так и пишем название - mp3(рис.3), то есть пытаемся что-нибудь с этим найти связанное. Указываем обязательно python(рис.4) и видим здесь очень много разного всего, что-то можно найти в принципе.

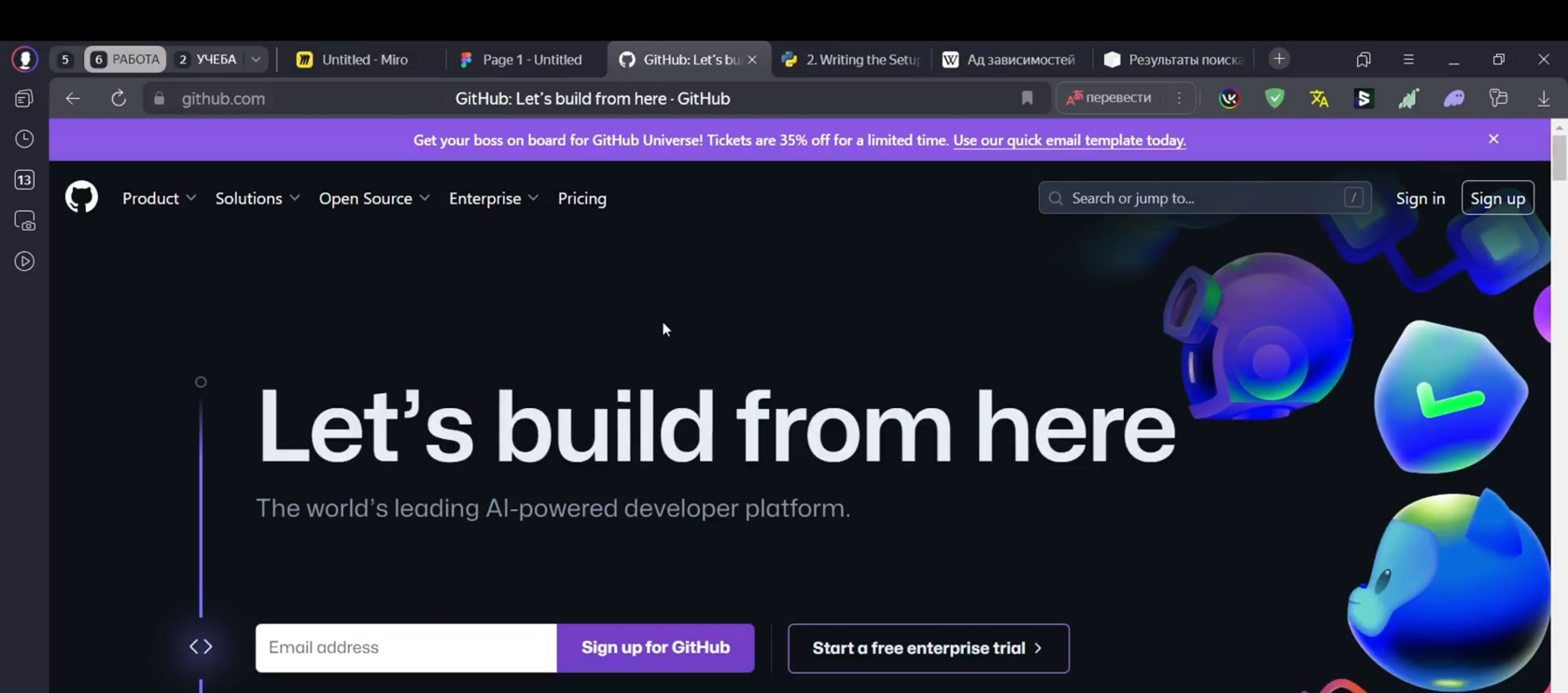


Рис.2

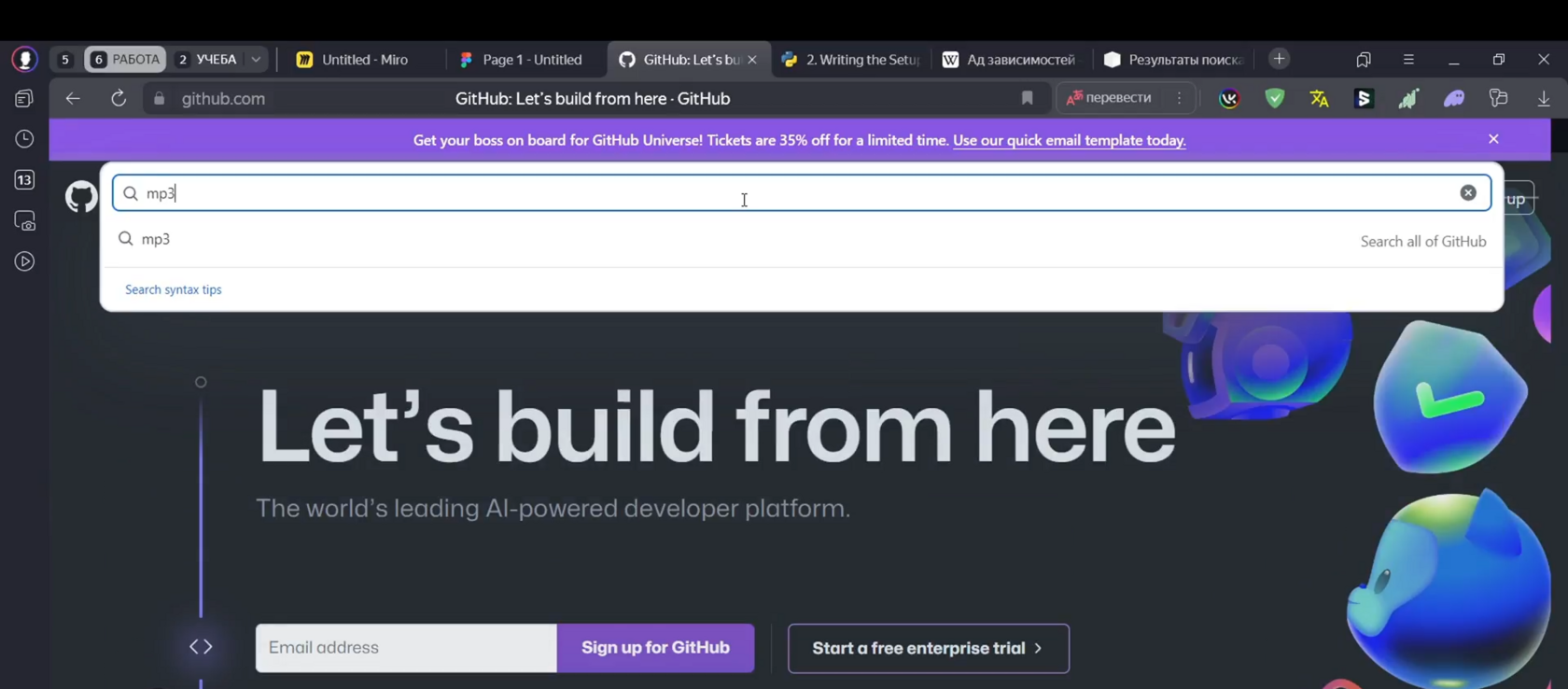


Рис.3

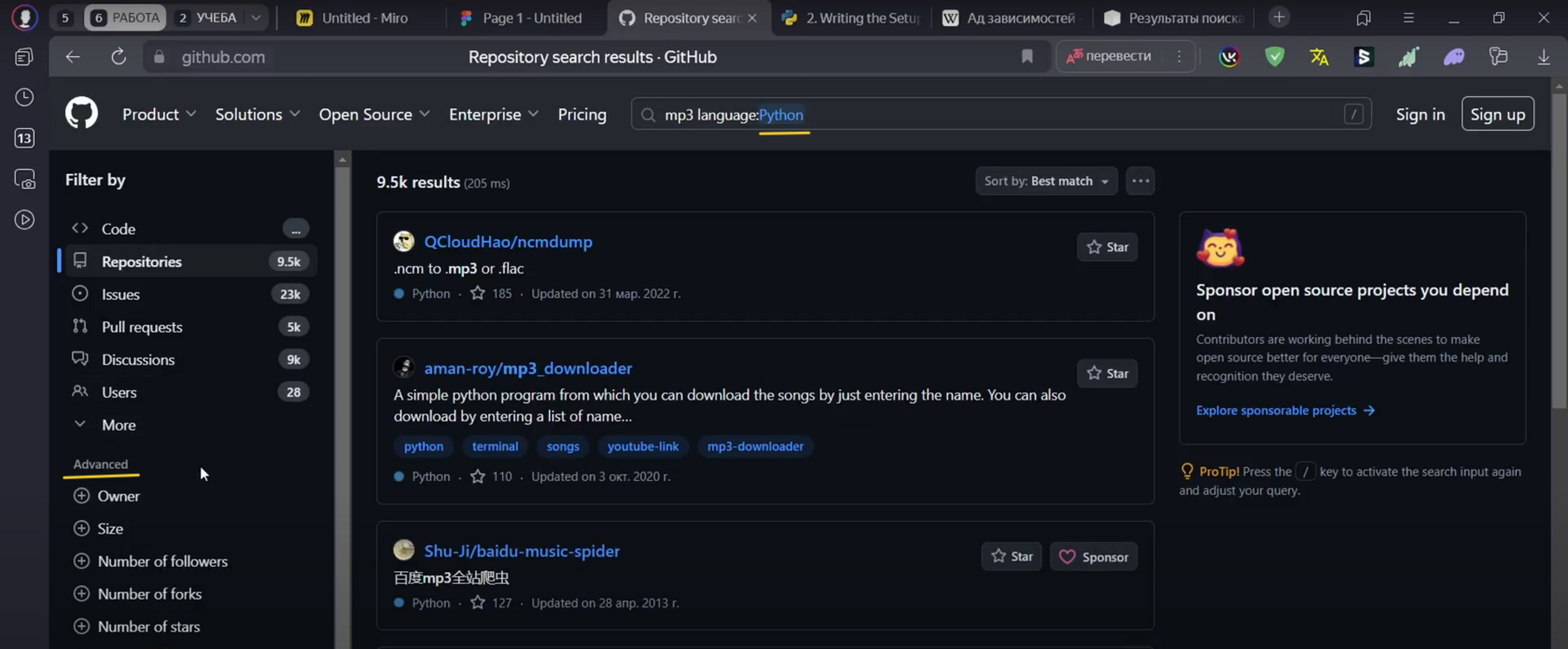


Рис.4

**Как довериться тому, что здесь написано?** Потому что любой разработчик может запустить сюда свой код. Как проверить, вообще, качественное ли это что-то или некачественно? В принципе, это нужно самому скачивать этот код, то есть перейдём на какой-то проект, качать нужно файл, как видим, он был сделан очень давно(рис.5), но зато у него очень много звёздочек. Просто качайте к себе и работайте с этим кодом, но это тоже нужно ещё уметь отдельно качать эти файлы, дублировать к себе в репозиторий. В общем, не самая удобная штука.

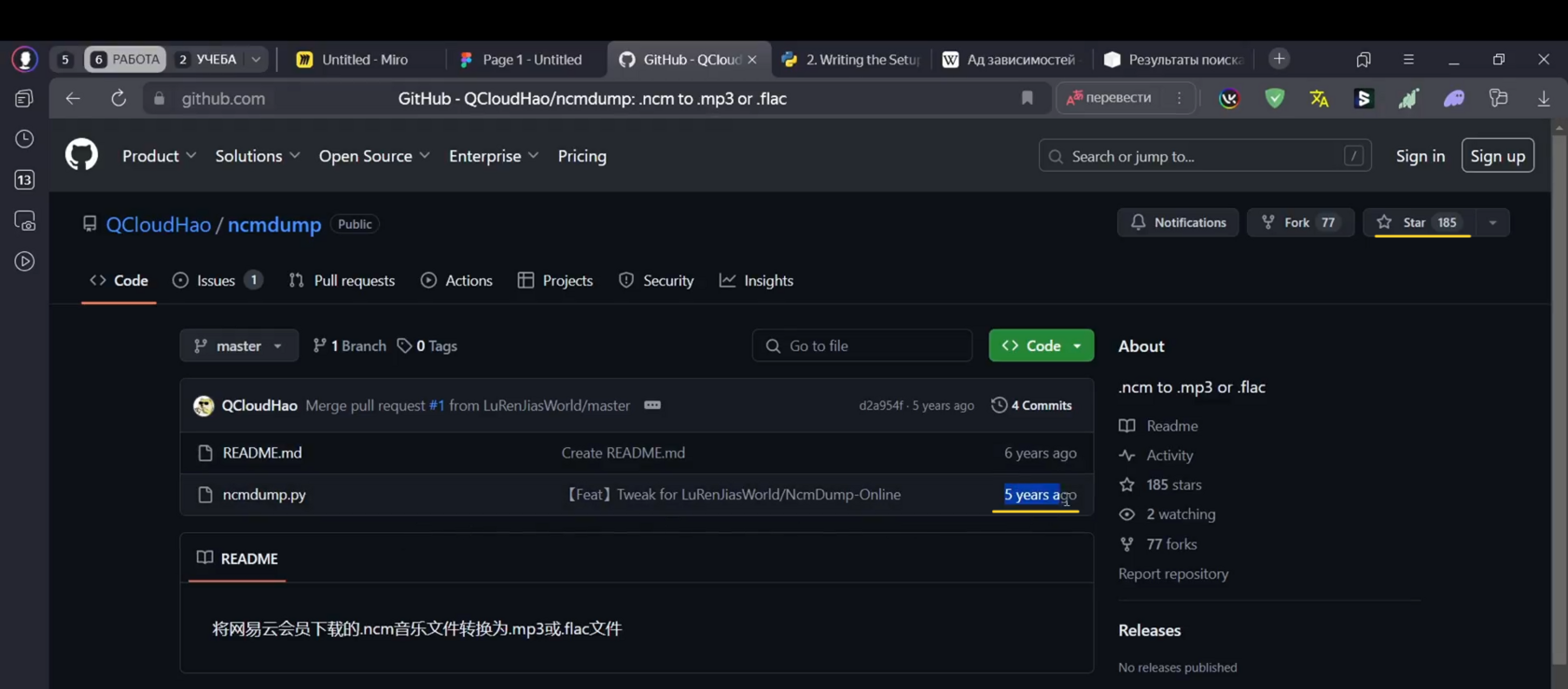


Рис.5

**Давайте познакомимся с PyPi.org.**Вот так вот выглядит наш главный сайт “Ищите, устанавливайте, публикуете Python пакеты в индексе пакетов”(рис.6). Почему именно этот сайт удобный? Потому что потом в дальнейшем сможем качать отсюда все сторонние библиотеки с помощью всего лишь одной команды встроенной в терминал, и у нас не будет сложностей с импортированием документов к себе в проект. Здесь мы можем найти любой проект.

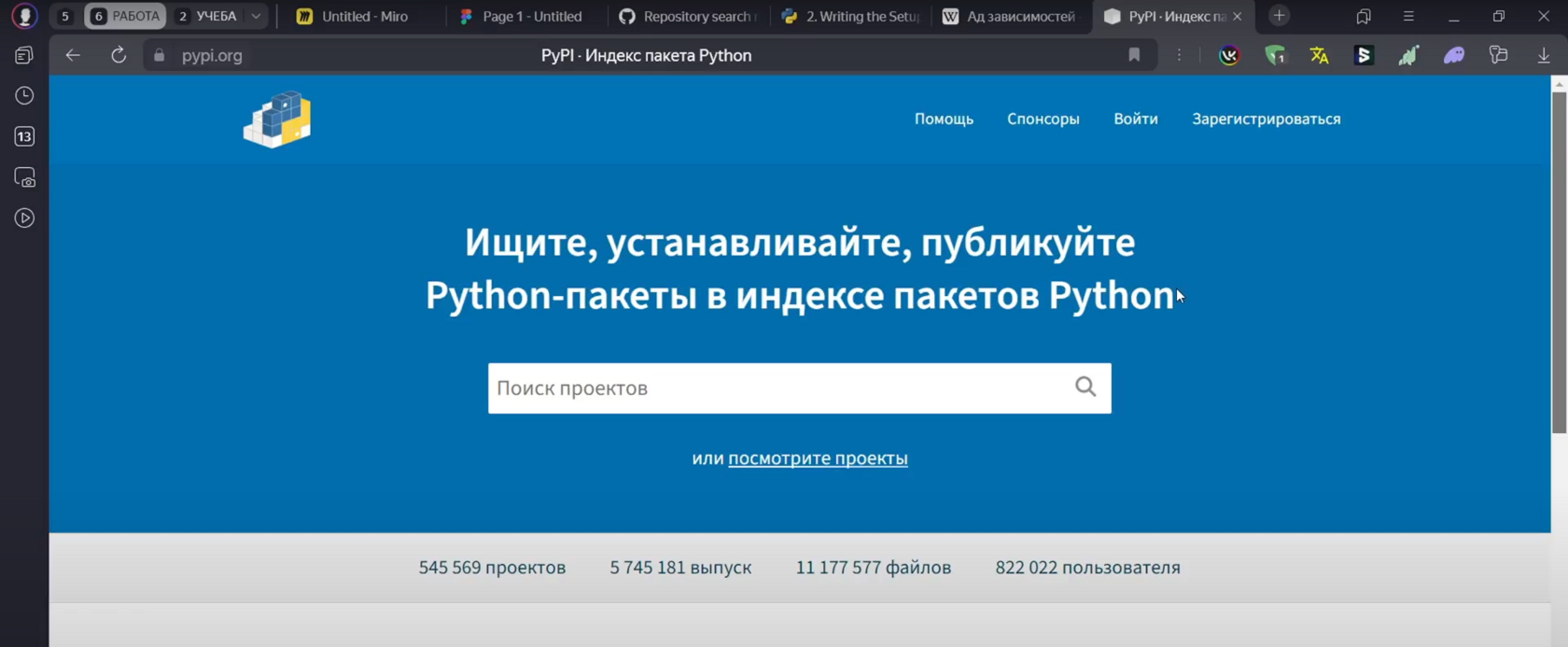


Рис.6

Ну, допустим, хотим скачать библиотеку matplotlib. Она работает с построением графиков и с матанализом. Ищем здесь название matplotlib и первая же ссылка — это и есть основной пакет matplotlib(рис.7). Есть ещё какие-то сторонние, но они не нужны, потому что у нас есть matplotlib последней версии, его и будем использовать. Тут есть подробное описание проекта. Если его здесь нет, то всегда можно перейти на ссылку проекта, почитать про этот проект. Обычно Homepage(рис.8) для небольшого проекта является ссылка на GitHub, но matplotlib настолько огромная библиотека, что у неё есть собственный сайт, поэтому в принципе туда и ссылается. Есть Source Code на GitHub, можем его посмотреть, можем его почитать, если нам интересно. Ну, в принципе, это можно и в Python сделать.

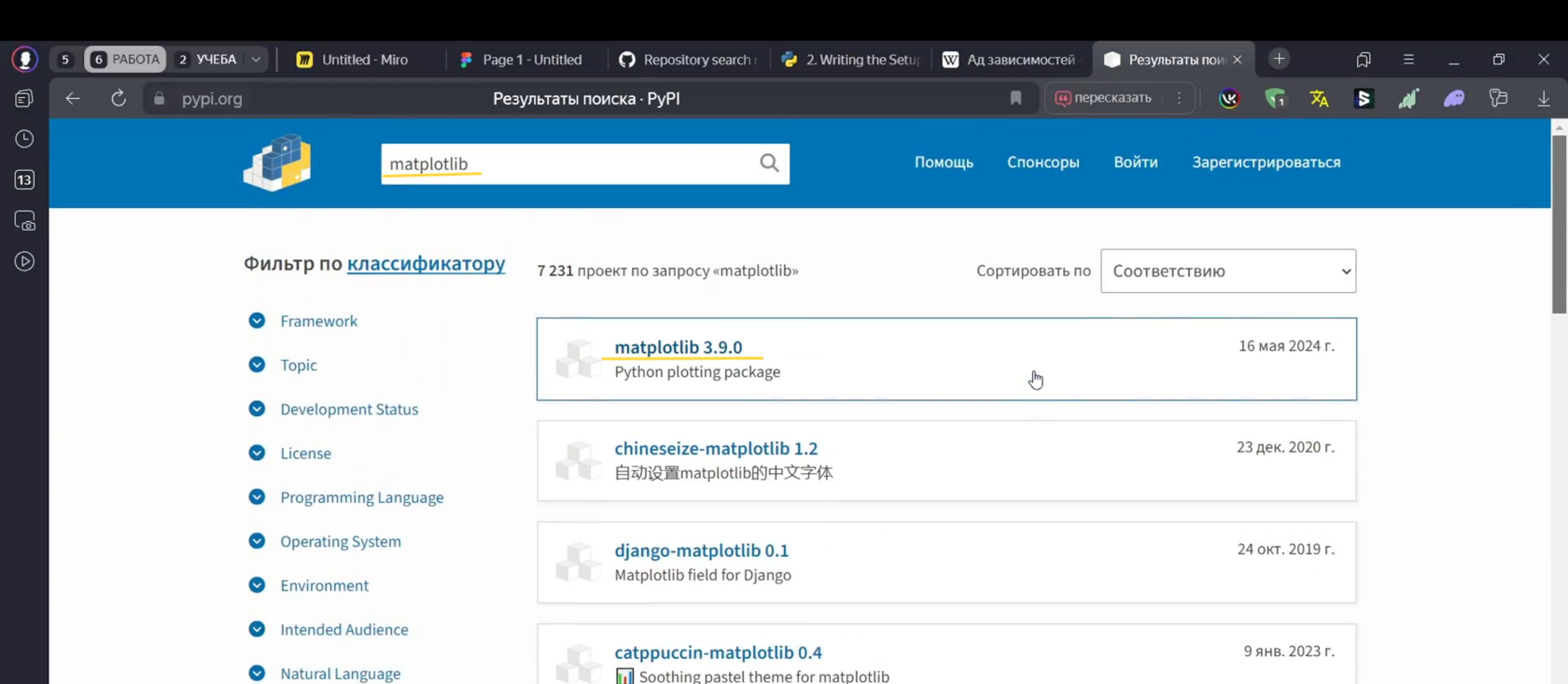


Рис.7

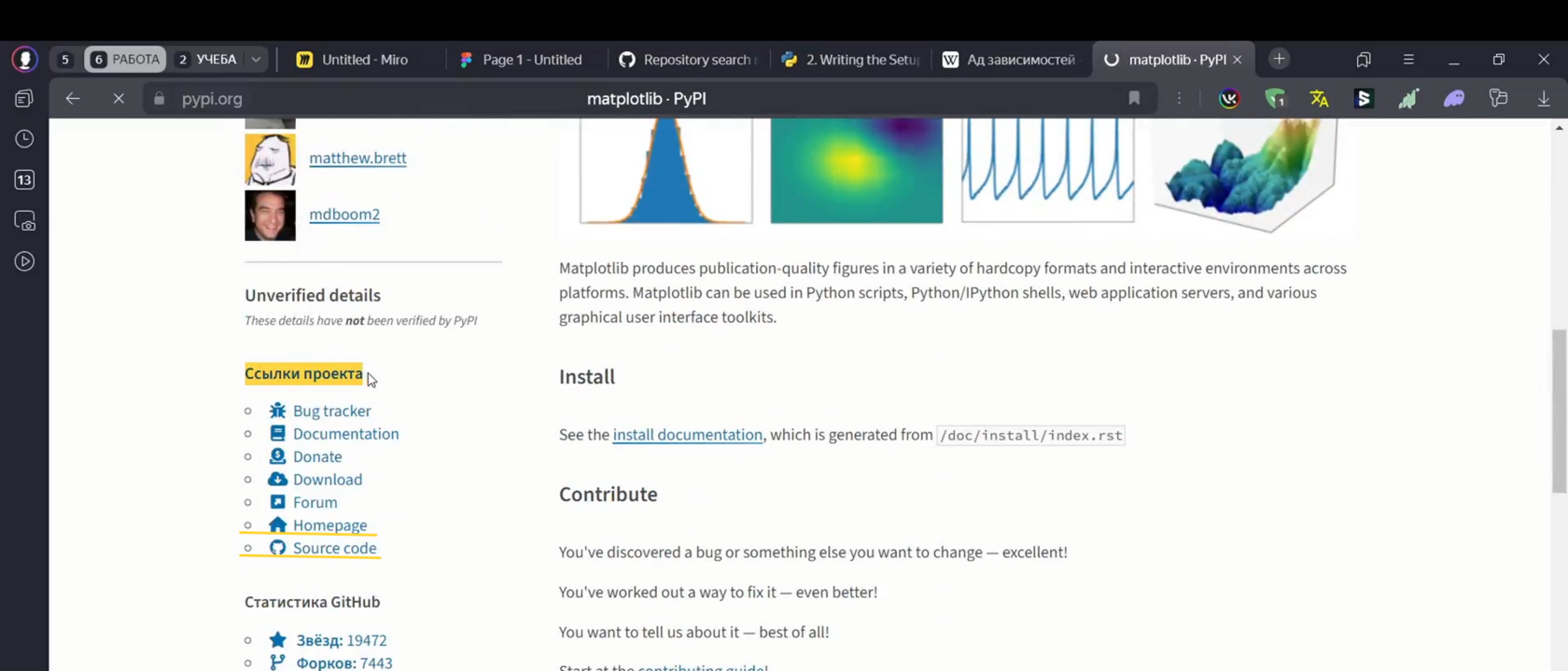


Рис.8

**Как установить библиотеку?** Вообще для этого используется pip — это когда-то был сторонний модуль. Этот модуль просто брал и импортировал вот эти вот библиотеки с таких открытых источников, но сейчас это встроенная программа, встроенный пакет. Его достаточно очень просто использовать.

Смотрите, мы уже перешли в наш любимый Pycharm. Здесь у нас открыт 1 файл main.py, тут просто написаны комментарии(рис.9), подсказка для того, как нужно работать с этой программой pip. Программа pip не десктопная (Desctop), то есть она не имеет какого-то отдельного окна. Она у нас запускается из терминала, работает в терминале. То есть переходим в терминал, у Pycharm терминал находится снизу слева(рис.10), кликаем на него и теперь можем здесь внутри написать программу, которая связана с pip.

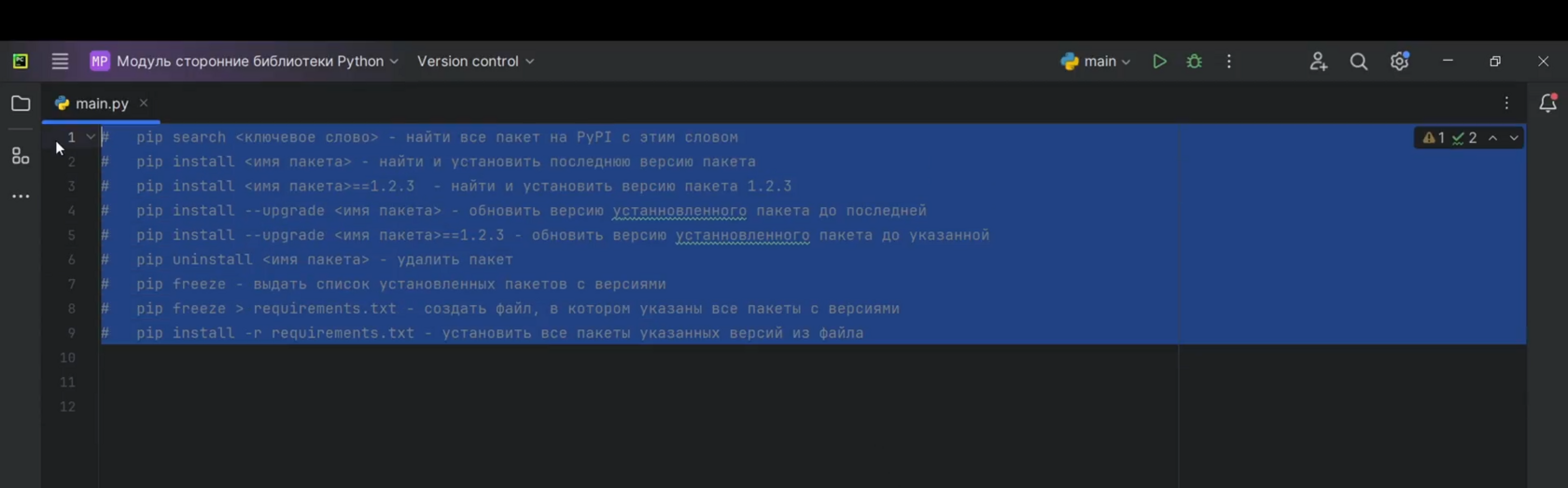


Рис.9

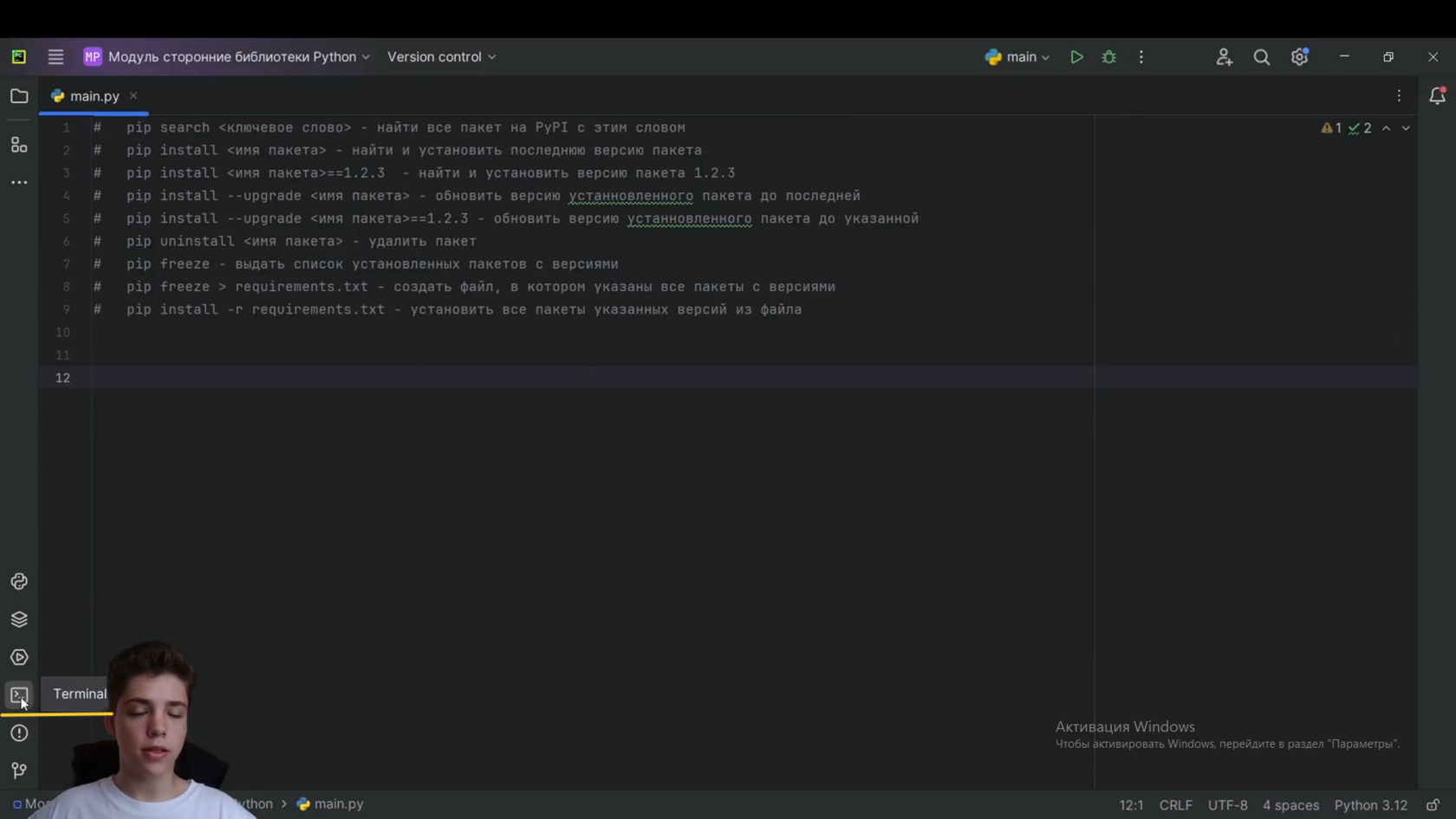


Рис.10

Чтобы установить какую-либо библиотеку, чтобы нам её скачать, просто пишем команду **pip install**(рис.11). Можем написать pip install и передать сюда название нашего проекта, название какого-то пакета, который хотим скачать.

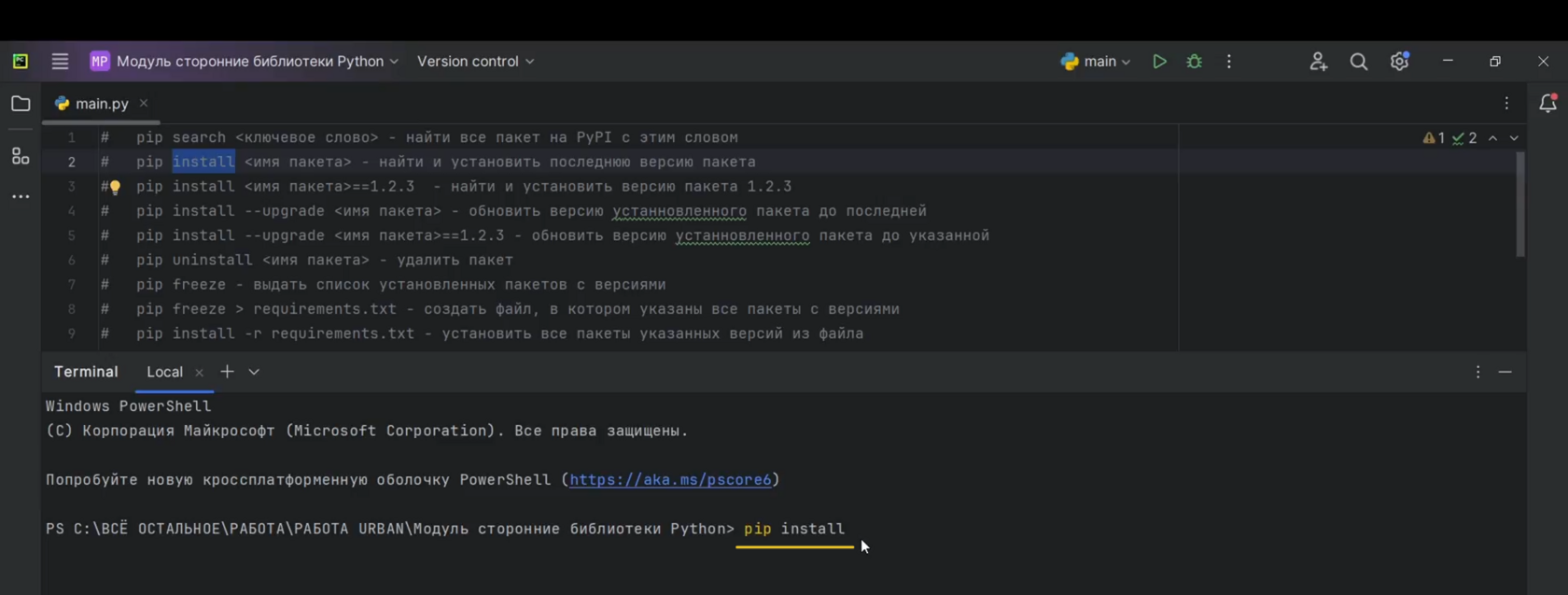


Рис.11

Ну, допустим, даже вот какая-нибудь библиотека, связанная с mp3, давайте заново найдём. Возьмем пакет “convert your media to mp3”. Вот так у нас качается программа: “pip install 2mp3”(рис.12). Мы передаём её в наш терминал, и у нас спокойно качается эта библиотека(рис.13). Этот пакетик сейчас у нас загружается, у каждого пакета разная загрузка.

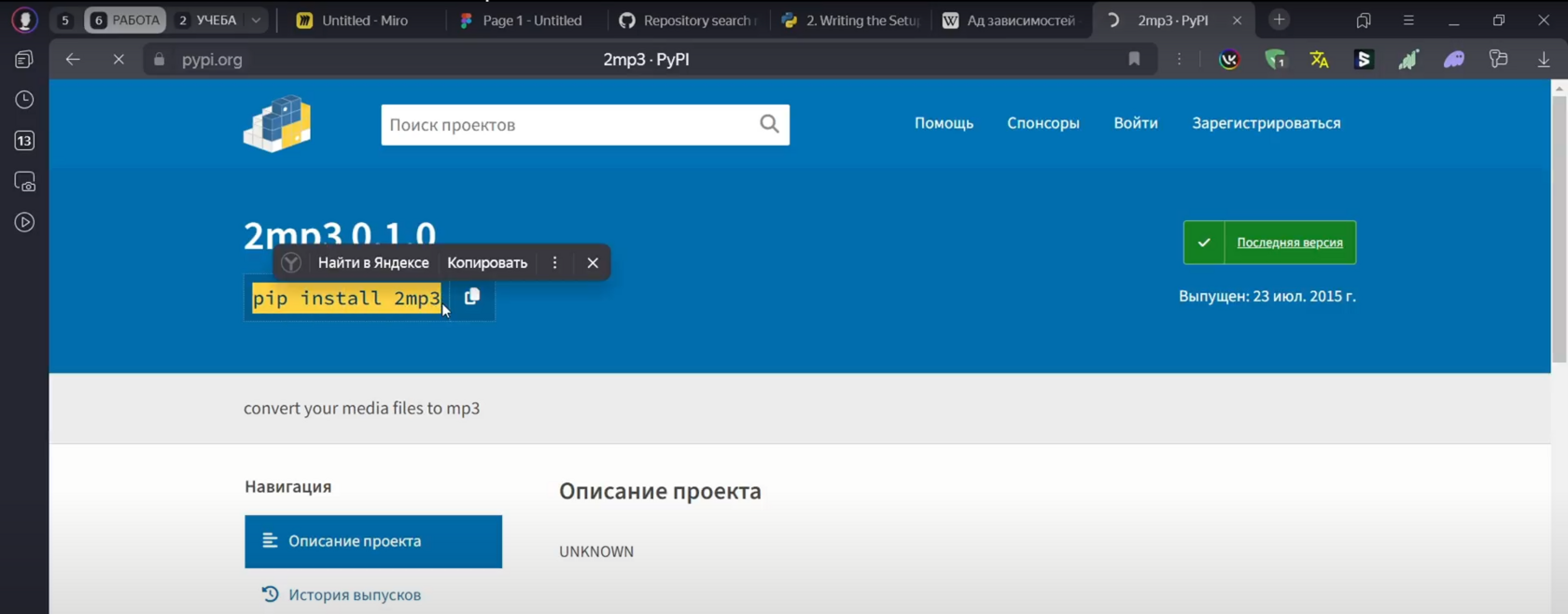


Рис.12

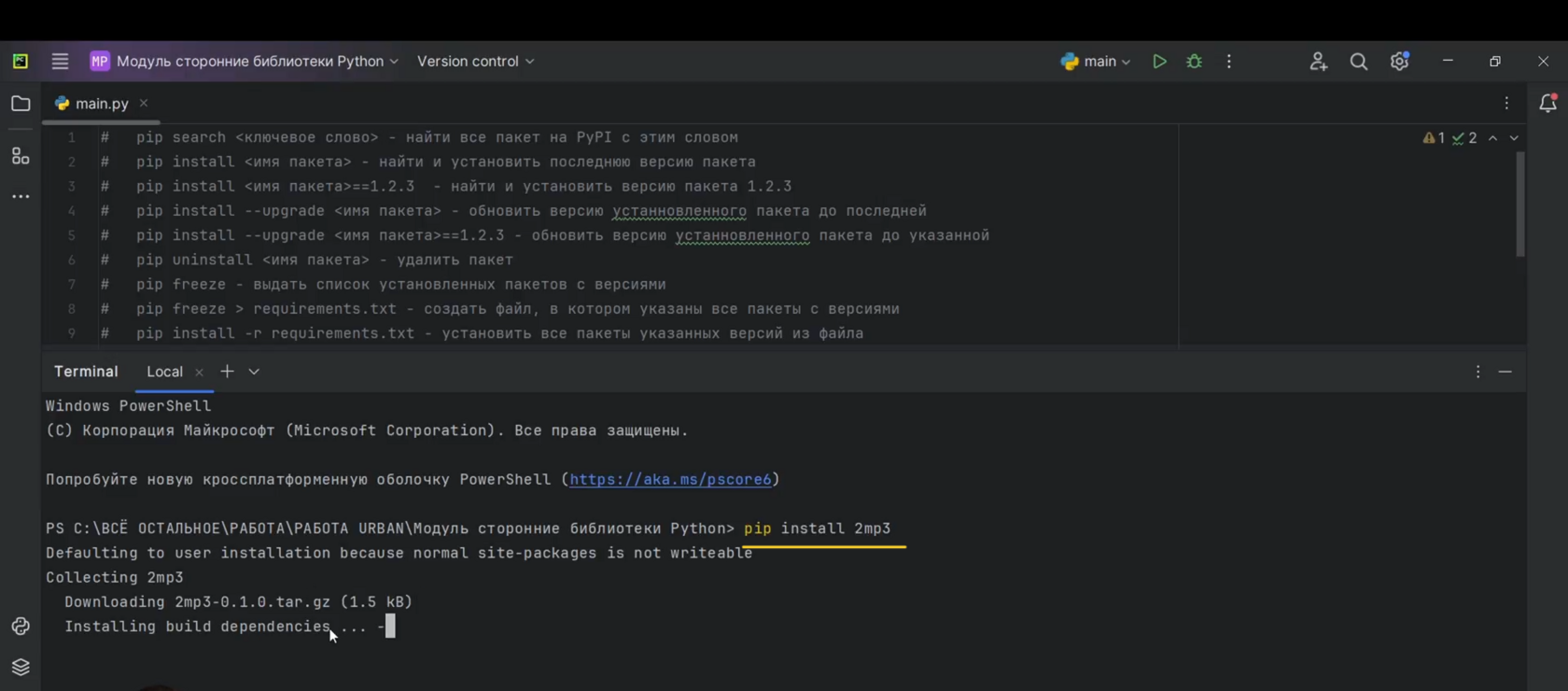


Рис.13

**Как понять, что загрузка выполнилась?** У нас опять высветится наша строка для терминала, то есть сможем снова в ней что-то писать. Также у нас перечислены различные программы. Например, можем установить определённую версию нашего пакета(рис.14). Пакет обновляется, и у них есть различные версии. Зачем нам качать определённую версию? Затем, что какой-либо проект, который уже в продакшене, был написан на достаточно старых библиотеках или фреймворках, и все зависимости, находящиеся там, они достаточно уже устарели и чтобы мы могли продолжать работать с этим приложением, а не писать какое-нибудь новое, чтобы там все менять, нам нужно качать определённую какую-то версию нашего пакета.

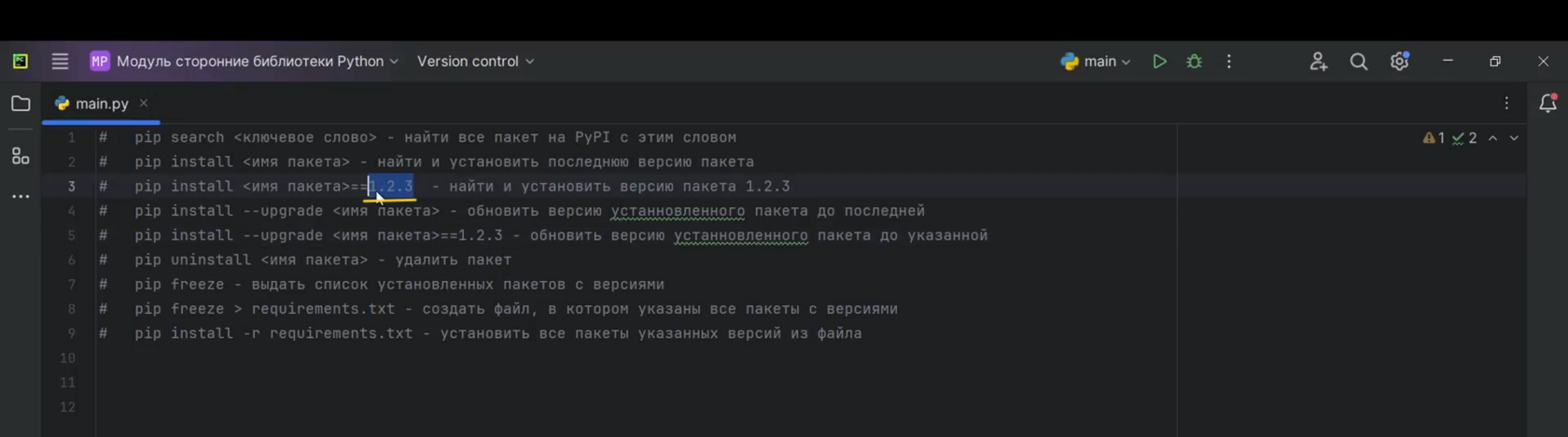


Рис.14

Зачастую пакет качается не один. Вместе с ним есть ещё и пакет зависимостей, то есть это какие-то другие сторонние библиотеки. Почему они качаются? Потому что у нас эта библиотека, которую качаем, была построена на других уже в библиотеках или пакет. Поэтому они сперва качаются с помощью программы, а потом уже качается необходимый нам пакет.

Есть ещё важная функция. **Она убирает, то есть удаляет наш проект**. Если запустим наш терминал, запустим **pip install**, то есть 2mp3, но вместо install напишем uninstall(рис.15), то у нас пакет просто удалится. Все, мы его убрали из нашего виртуального окружения, чтобы он нам не мешался.

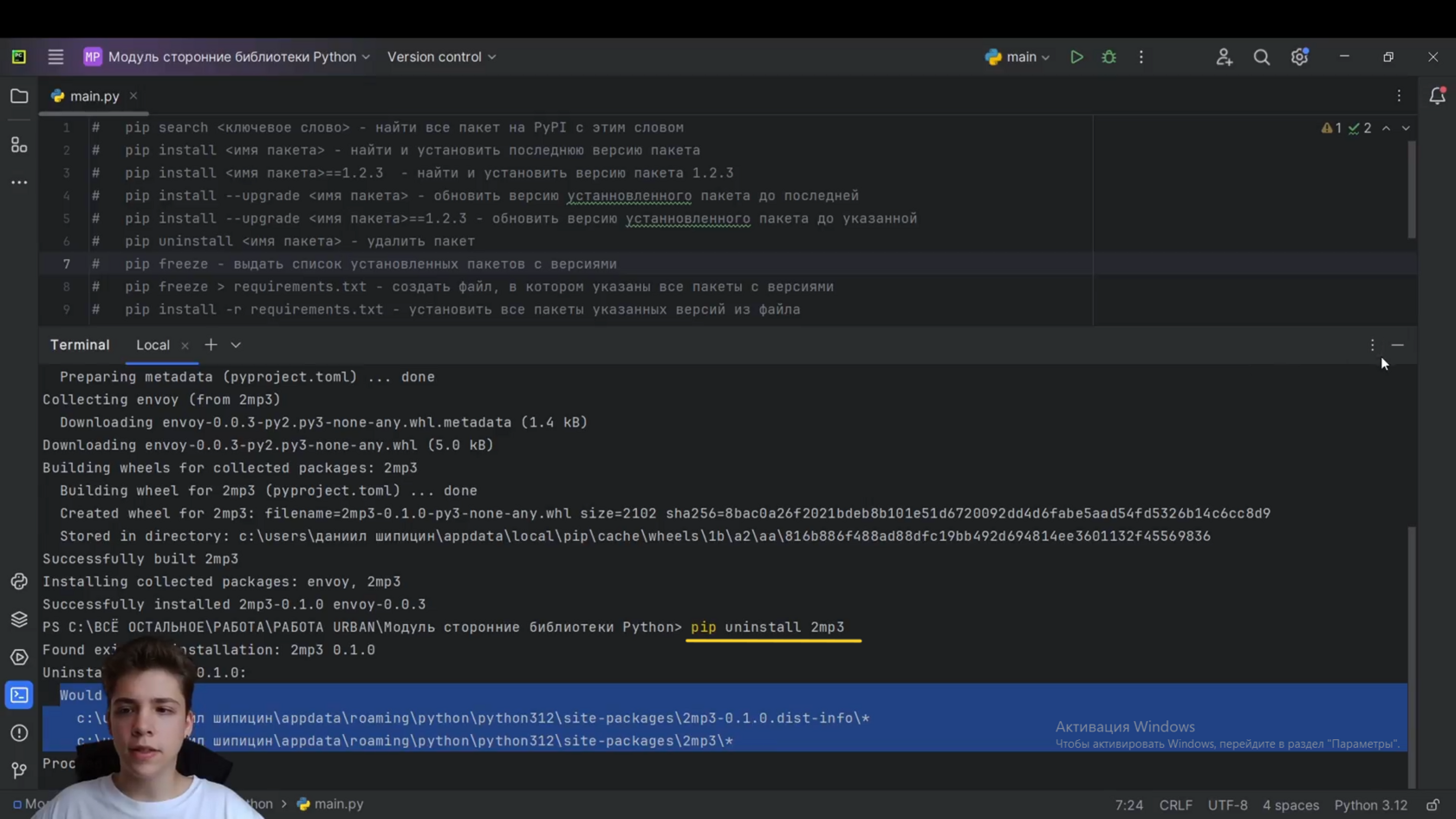


Рис.15

Есть ещё программа **pip freeze**, которая выдаёт список установленных пакетов с версиями, причём есть программа pip freeze > requirements.txt — она берет и создаёт текстовый файл со всеми библиотеками. Они у нас установлены именно в это виртуальное окружение.

**Давайте посмотрим, как работает эта программа.**  
Для начала очистим терминал(рис.16).

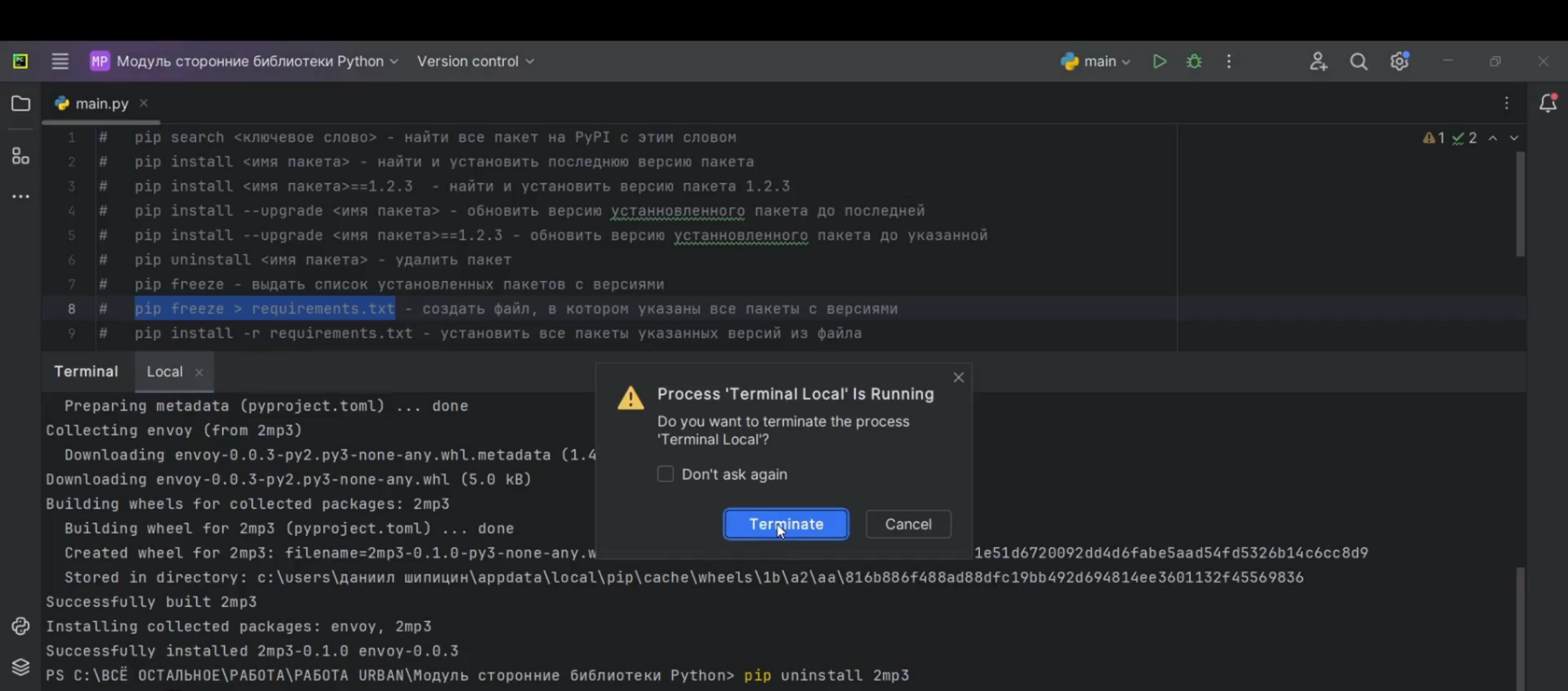


Рис.16

Вот новый терминал и пишем программу pip freeze > requirements.txt(рис.17), у нас создаётся в корневой папке, где у нас сохранён файл main.py - requirements.txt(рис.18). Здесь у нас перечислены все библиотеки, которые уже были установлены. Вот этот весь список, у нас уже сейчас был здесь. В папке Lib, которая есть в Python, перечислены все основные библиотеки и пакеты(рис.19).

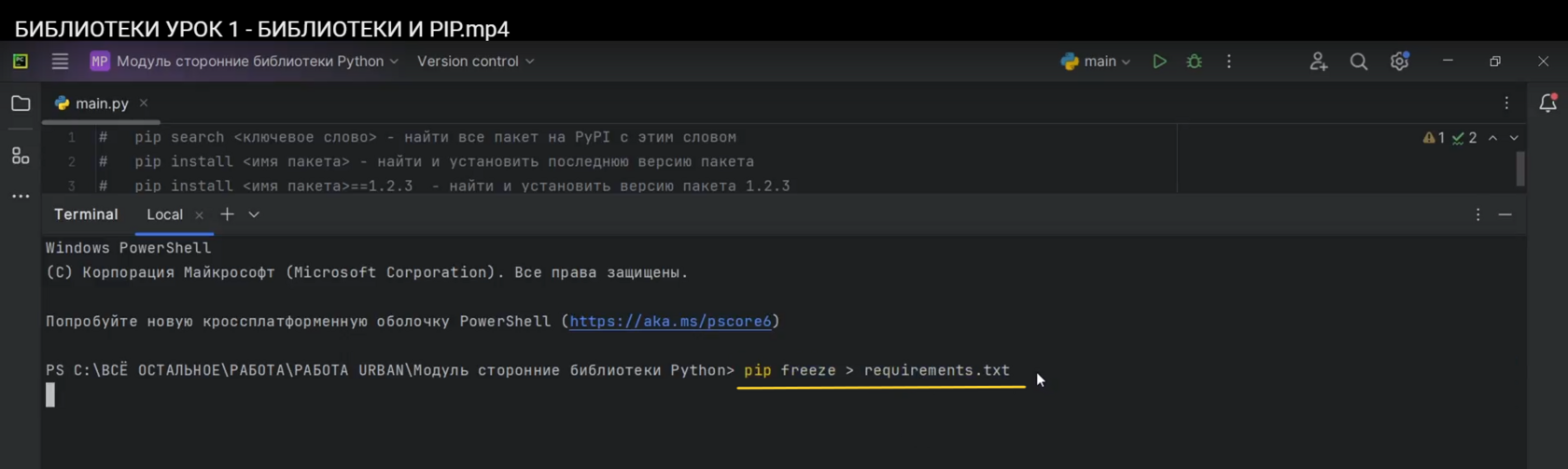


Рис.17

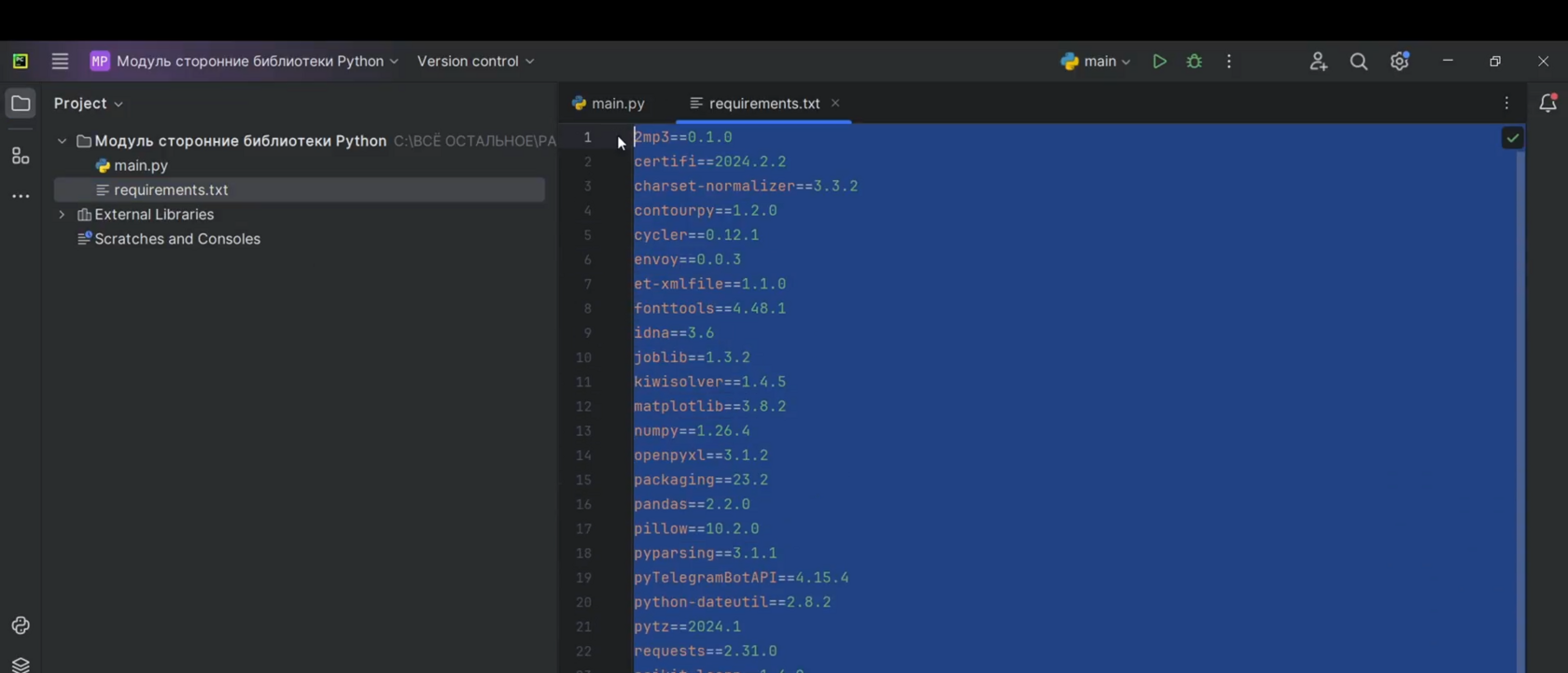


Рис.18

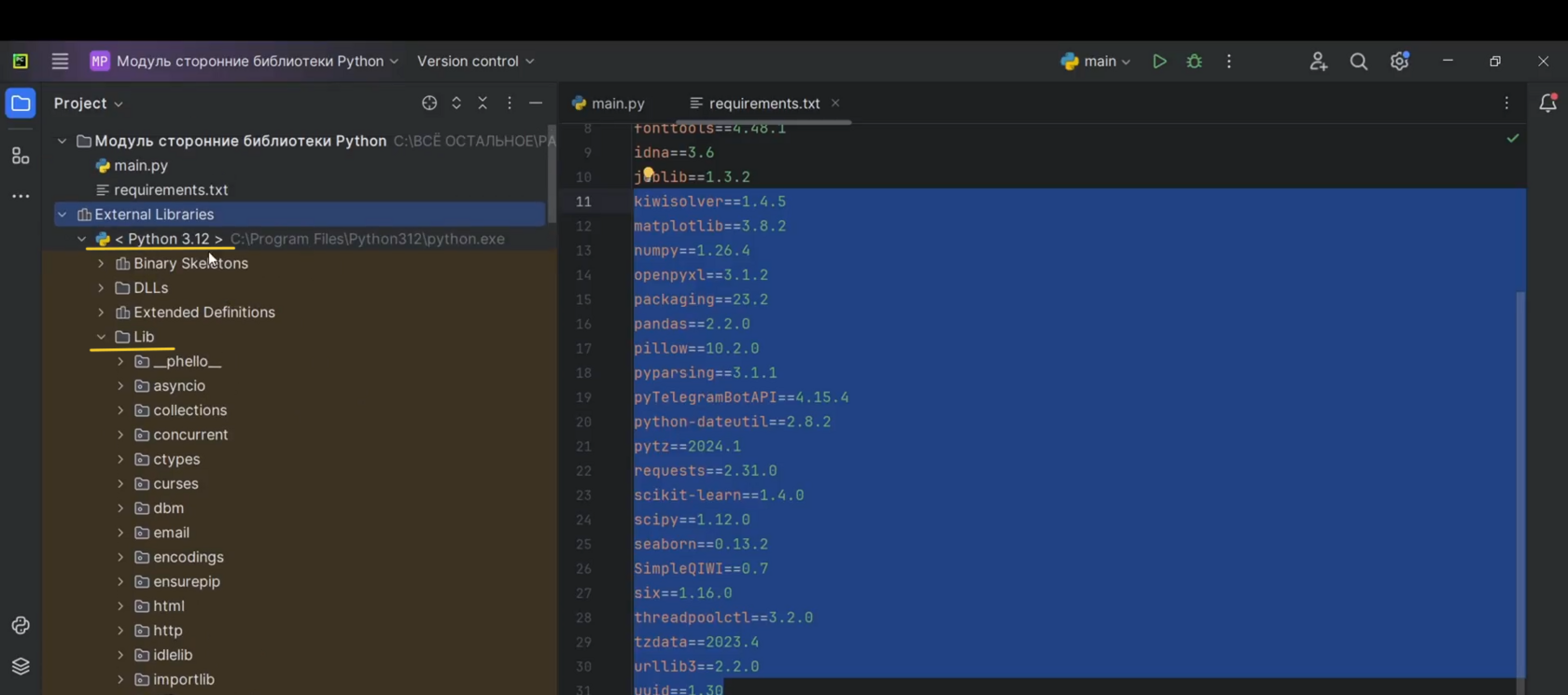


Рис.19