## 3 Виртуальное окружение

Для каждого отдельного проекта рекомендуется создавать так называемое виртуальное окружение. И сделать это можно средствами самого Python.

То есть, нужно:

1. **Создать папку**, где будет находиться ваш проект.
2. **Открыть эту папку в VS Code** (не обязательно в VS Code, конечно, можно в любом другом редакторе кода).
3. В терминале, **находясь в папке проекта**, **создать для проекта виртуальное окружение**, выполнив команду:

python3.11 -m venv venv

Здесь флаг -m говорит интерпретатору запустить инструмент создания виртуального окружения venv как исполняемый модуль, а второй venv - название папки, в которую будет установлено виртуальное окружение. Оно может быть любым другим, но обычно принято называть его либо venv, либо env, от английского - virtual environment или просто environment.

## *Комментарий пользователя*:

Чтобы установить виртуальное окружение в папку с нашим проектом, в терминале сначала нужно **зайти в корневой каталог нашего проекта**, потом только нужно прописывать:

**python -m venv venv** (или открыть терминал прямо из данной папки, чтоб сразу находиться в данной папке, а не прописывать его). А после установки, вроде нужно ещё активировать данное вирт. окружение командой: **venv\scripts\activate**  .

Если я правильно рассуждаю, то в идеале построить последовательность действий по созданию вирт. окружения, чтоб не играть в угадайку))

**Пример последовательности действий для создания вирт. окружения (windows):**

1) Открываем папку с будущим проектом

2) Нажимаем правую кнопку мыши на пустое пространство в папке и выбираем: "Открыть окно PowerShell здесь"

3) Пишем: "**python -m venv venv**"

4) Активируем установленное вирт. окружение, прописав следом следующую команду: "**venv\scripts\activate**"

## Запуск и работа в виртуальном окружении

После того, как мы создали виртуальное окружение - его необходимо запустить (активировать), чтобы все библиотеки, которые дополнительно нужно будет устанавливать для проекта, ставились именно в виртуальное окружение, а не глобально. Запуск на macOS и Ubuntu немного отличается от запуска на Windows.

На macOS и Ubuntu виртуальное окружение активируется командой:

source venv/bin/activate

После ее выполнения, вы должны увидеть название вашего виртуального окружения в скобках, перед приглашением ко вводу команд в командную строку:

Air-Mihail:Python mikhail$ source venv/bin/activate

(venv) Air-Mihail:Python mikhail$

На Windows в командной строке терминала cmd нужно выполнить:

venv\Scripts\activate.bat

На Windows в командной строке терминала Power Shell:

venv\Scripts\activate.ps1

Если после выполнения команды на Windows ничего не произошло - то есть, перед приглашением к работе с командной строкой не появилось в скобках название виртуального окружения - переходите на следующий шаг - там инструкция, что с этим делать.

Если же виртуальное окружение запустилось, давайте убедимся, что устанавливаемые через менеджер пакетов pip библиотеки, действительно ставятся в виртуальное окружение, а не глобально. Выполните команду: pip list

Вы должны получить список установленных пакетов, типа:

Package Version

----------------- -------

flake8 5.0.4

mccabe 0.7.0

mypy 0.982

mypy-extensions 0.4.3

pip 22.3

pycodestyle 2.9.1

pyflakes 2.5.0

setuptools 65.5.0

typing\_extensions 4.4.0

Теперь давайте поставим, например, библиотеку numpy, которая очень часто используется в статистике, аналитике, машинном обучении и еще много-много где, где требуется работа с матрицами.

pip install numpy

После установки давайте снова вызовем команду: pip list

И убедимся, что библиотека numpy присутствует в виртуальном окружении:

Package Version

----------------- -------

flake8 5.0.4

mccabe 0.7.0

mypy 0.982

mypy-extensions 0.4.3

numpy 1.23.4

pip 22.3

pycodestyle 2.9.1

pyflakes 2.5.0

setuptools 65.5.0

typing\_extensions 4.4.0

А теперь нужно **выйти из виртуального окружения**. Делается это вводом команды:

deactivate

После ее выполнения название виртуального окружения в скобках пропадет из командной строки. А, значит, теперь мы находимся в глобальном окружении. И если теперь ввести команду, чтобы посмотреть на список установленных библиотек (команда pip list), то увидим, что он отличается, и библиотека numpy не установлена.

Итак, на всякий случай еще раз повторю.

|  |  |
| --- | --- |
| Активация виртуального окружения для macOS и Ubuntu | source venv/bin/activate |
| Активация для Windows (cmd): | venv\Scripts\activate.bat |
| Активация для Windows (PowerShell): | venv\Scripts\activate.ps1 |
| Выход из виртуального окружения - команда: | deactivate |

## Удаление виртуального окружения

Если, вдруг, вы почему-то решили удалить из проекта виртуальное окружение, сделать это можно просто удалив папку venv (или с другим названием, если вы, при создании виртуального окружения, называли ее по-другому).

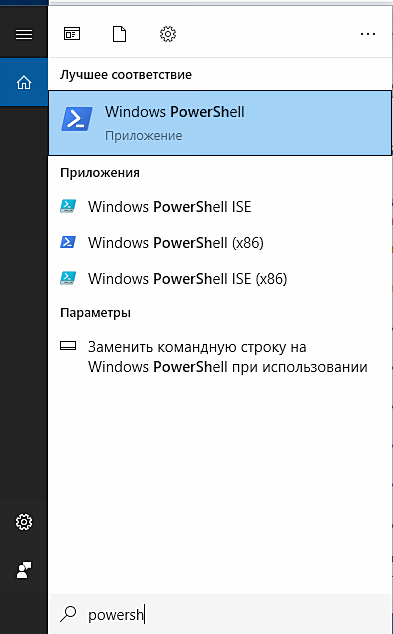
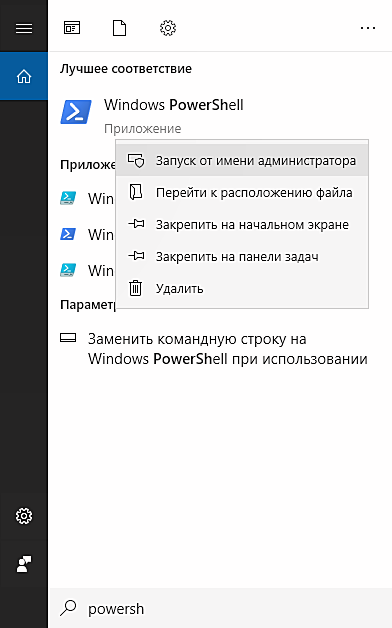
## Если виртуальное окружение не запускается в Windows

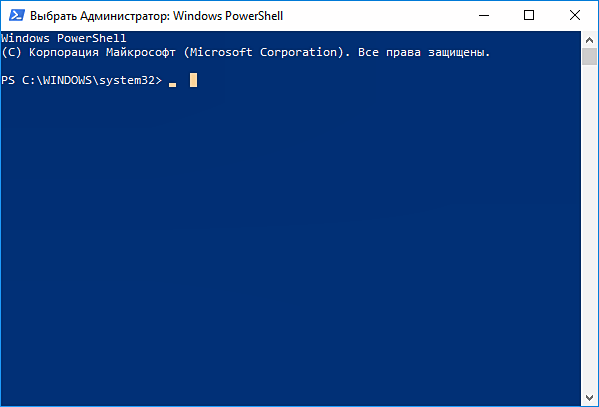
Возможна ситуация, при которой у вас не запустится виртуальное окружение под Windows, хотя в директории проекта вы увидите, что папка с вашим виртуальным окружением есть. При этом терминал может не сообщать никаких ошибок, просто выполнение команды:

venv\Scripts\activate.bat ни к чему не приводит.

Это происходит из-за настроек безопасности (политики выполнения), которые ограничивают исполнение потенциально опасных сценариев. Чтобы исправить ситуацию нужно найти в поиске Windows Power Shell (именно Power Shell, a не cmd), запустить его от имени администратора и выполнить команду:

Set-ExecutionPolicy unrestricted



В процессе выполнения команды, нужно будет подтвердить действие по изменению политики выполнения:

PS C:\WINDOWS\system32> Set-ExecutionPolicy unrestricted

Изменение политики выполнения

Политика выполнения защищает компьютер от ненадежных сценариев. Изменение

политики выполнения может поставить под угрозу безопасность системы, как описано

в разделе справки, вызываемом командой about\_Execution\_Policies и расположенном

по адресу https:/go.microsoft.com/fwlink/?LinkID=135170 . Вы хотите изменить

политику выполнения?

[Y] Да - Y [A] Да для всех - A [N] Нет - N [L] Нет для всех - L

[S] Приостановить - S[?] Справка (значением по умолчанию является "N"): y

PS C:\WINDOWS\system32>

После этого попробуйте снова запустить в терминале VS Code виртуальное окружение.

**Установка библиотеки aiogram**

Собственно, **AIOgram** - это ядро данного курса по разработке телеграм-ботов. На старте курса я использовал библиотеку версии 2.24, но в процессе его развития и общения со студентами, решил перейти на aiogram 3, не смотря на то, что библиотека до сих пор находится в beta-версии. Причины перехода следующие:

* Создатель библиотеки сказал, что aiogram 2 больше развиваться не будет
* У второй библиотеки не было и уже, скорее всего, не будет нормальной документации
* aiogram 3, не смотря на то, что до сих пор нет стабильной версии, уже довольно широко используется разработчиками ботов и серьезных проблем там не выявлено, а те баги, которые, вдруг, всплывают - очень быстро исправляются автором при поддержке комьюнити
* По отзывам тех, кто разрабатывал и на второй и на третьей версии, третья лишена некоторых недостатков второй, а также обладает новыми возможностями, которых в двойке уже точно не будет

Чтобы установить библиотеку aiogram, находящуюся в beta-версии - при запущенном виртуальном окружении выполните в терминале команду:

pip install -U --pre aiogram

Обычно никаких проблем не возникает и aiogram нормально устанавливается со всеми зависимостями.

Если у вас не было проблем с установкой aiogram или вы с ними успешно справились - теперь, при просмотре списка установленных зависимостей с помощью команды: pip list

Вы должны увидеть, что aiogram доступен:

(venv) mike@MacBook-Air-Mihail-2 aiogram\_course\_bot\_template % pip list

Package Version

------------------ ---------

aiofiles 22.1.0

aiogram 3.0.0b6

aiohttp 3.8.3

aiosignal 1.3.1

async-timeout 4.0.2

attrs 22.2.0

autopep8 2.0.1

certifi 2022.12.7

charset-normalizer 2.1.1

flake8 6.0.0

frozenlist 1.3.3

idna 3.4

magic-filter 1.0.9

marshmallow 3.19.0

mccabe 0.7.0

multidict 6.0.4

mypy 0.991

mypy-extensions 0.4.3

packaging 23.0

pip 22.3.1

pycodestyle 2.10.0

pydantic 1.10.4

pyflakes 3.0.1

setuptools 65.6.3

typing\_extensions 4.4.0

yarl 1.8.2