



sindzicat 12 февраля 2019 в 12:03

# Создание и настройка портативной сборки Jupyter Notebook и Lab на Windows. Часть 1

Python

Tutorial

Всем привет. Когда я начинал изучение Python, устанавливал впервые Jupyter Notebook, потом пытался передать с созданное в нём приложение на предприятие, я часто сталкивался с различными проблемами. То кириллица в имени пользователя мешает, то настройки не перенеслись, то ещё чего-то. Все эти проблемы я преодолел в основном самостоятельно, используя Google и затратив немало времени на их решение.

По мере роста опыта я научился создавать папку, в которой лежит переносимое с одного компьютера на другой виртуальное окружение Python, настройки Jupyter и Matplotlib, портативные программы (ffmpeg и др.) и шрифты. Я мог написать дома программу, скопировать всю эту папку на компьютер предприятия, и быть уверенным, что ничего не потеряется и не сломается на ровном месте. Потом я подумал, что такую папку можно дать и новичку в Python, и он получит полностью настроенную и переносимую среду.

## Оглавление

- **Введение**
- Краткая инструкция по созданию портативной сборки Jupyter
- Установка Miniconda (Python 3.7)
- Создание структуры каталогов
- Создание переносимого виртуального окружения Python
  - Создание виртуального окружения с помощью conda
  - Исправление ошибки HTTP 000 CONNECTION FAILED при создании виртуального окружения
  - Активация виртуального окружения
  - Установка пакетов Python в виртуальном окружении
  - Выход из виртуального окружения Python
- Подготовка портативной сборки Jupyter к запуску
  - Настройка переменных окружения для Jupyter, IPython и Matplotlib
  - Создание файла для запуска Jupyter с настройками пользователя
  - Дополнительные файлы для выполнения служебных действий
- Заключение

## Введение

В последние годы Python стал популярным языком программирования. Его часто используют для написания математических расчётов, анализа больших данных, машинного обучения и построения нейросетей. После появления конструкций `async` и `await` стало возможным написания быстрых веб-фреймворков. Производительность Python постепенно повышается из релиза в релиз, а использование Cython или Numba может сделать приложение даже более быстрым, чем на других языках программирования. Например, скорость работы веб-фреймворка Vibora<sup>(en)</sup> сопоставима со скоростью работы решений на Go<sup>(en)</sup>. В 2018 году Python официально стал языком для изучения в школах и вузах Франции<sup>(en)</sup> и Казахстана<sup>(en)</sup>. В России как минимум некоторые кафедры перешли на Python, например, кафедра РК-6<sup>(ru)</sup> в МГТУ им. Н.Э. Баумана.

Приступая к изучению Python, новые пользователи порой сталкиваются с трудностями при установке необходимых библиотек и настройке среды программирования. Если имя пользователя Windows содержит не латинские символы, некоторые библиотеки могли не установиться или не запускаться. У начинающих пользователей могут возникать проблемы с настройкой Jupyter Notebook на локальном компьютере. Если он установлен на диске C:\, как открыть файл на диске D:\? Когда я делал первые шаги в Python, мне тоже приходилось преодолевать эту трудности.

Наконец, если все проблемы позади, могут возникнуть трудности передать приложение другому пользователю. Я сталкивался с ситуацией, когда созданное мною виртуальное окружение для Python отказывалось работать на другом компьютере. Кроме того, Jupyter Notebook и Matplotlib хранят свои настройки в папке пользователя, что усложняет перенос приложений, использующих специфичные настройки.

Решением описанных выше проблем будет создание полностью портативной сборки Jupyter Notebook и/или Jupyter Lab на Windows. Она хранит в себе интерпретатор Python, его библиотеки и настройки, настройки всех необходимых сторонних библиотек, включая Matplotlib и Jupyter, не привязано к имени пользователя и не будет ругаться, если вы запустите её на другом компьютере. Мы можем упаковать такую сборку в архив, либо написать скрипт или программу, которая создаст такую же сборку на компьютере абсолютного новичка. Более продвинутым пользователям портативная сборка может быть полезна тем, что она позволяет хранить окружение Python и настройки библиотек в разных местах. Вы можете разместить папку с настройками в специальное место, которое синхронизируется с облачным хранилищем: Dropbox, облако Mail.ru\*, Яндекс или Google. За счёт этого на всех компьютерах автоматически получится локально работающая среда с одинаковыми настройками.

\*Да, то самое, клиент которого под Linux больше не коннектится<sup>(ru)</sup>. Если уберут аналогичный под Windows, мне придётся искать замену. 1 Тб на дороге бесплатно не валяется.

Для простоты восприятия материала я решил описать создание портативной сборки под Windows. Но эта инструкция с минимальными изменениями годится для создания сборки на Linux и Mac OS. Статья в первую очередь предназначена для новичков, поэтому я постарался описать как можно подробнее и проще для восприятия.

Статья состоит из двух частей. В первой части мы создадим портативную сборку, во второй займёмся настройками для Jupyter Notebook, Jupyter Lab, IPython и Matplotlib.

## Краткая инструкция по созданию портативной сборки Jupyter

1. Создайте папку C:\Dev. В ней будут установлены Miniconda и портативная сборка Jupyter\*.

\*Здесь и далее Jupyter = Jupyter Notebook + Jupyter Lab.

2. Скачайте инсталлятор Miniconda с сайта <https://conda.io/miniconda><sup>(en)</sup>. Выберите Python 3 для Windows 64 бит или 32 бит в зависимости от разрядности вашей операционной системы. Установите Miniconda в папку C:\Dev\Miniconda3.

3. Создайте следующую структуру каталогов для портативной сборки Jupyter:

```
C:\
├── Dev\
│   ├── Jupyter\
│   │   ├── dist\
│   │   │   ├── apps\
│   │   │   ├── conf\
│   │   │   │   ├── backup\
│   │   │   │   ├── ipython\
│   │   │   │   ├── jupyter\
│   │   │   │   └── matplotlib\
│   │   ├── fonts\
│   │   └── projects\
```

4. Создайте виртуальное окружение для Python с помощью conda\*:

```
C:\Dev\Miniconda3\Scripts\conda.exe create -p C:\Dev\Jupyter\dist\pyenv3.7-win64 --copy --yes python=3 conda
```

\*Вы можете использовать канал conda-forge для установки более свежих библиотек, добавив аргумент -c conda-forge:

```
C:\Dev\Miniconda3\Scripts\conda.exe create -p C:\Dev\Jupyter\dist\pyenv3.7-win64 --copy --yes -c conda-forge python=3 conda
```

5. Активируйте окружение и установите пакеты Python с помощью pip\*:

```
C:\Dev\Jupyter\dist\pyenv3.7-win64\Scripts\activate  
pip --no-cache-dir install numpy scipy matplotlib jupyter jupyterlab
```

Примечание: если вам необходимо установить Numpy и Scipy, которые используют библиотеку MKL от Intel для ускорения расчётов, используйте<sup>(en)</sup> intel-numpy вместо numpy и intel-scipy вместо scipy (устанавливается только в Python 3.6!):

```
pip --no-cache-dir install intel-numpy intel-scipy matplotlib jupyter jupyterlab
```

После установки выполните:

```
conda.bat deactivate
```

\*Если возникнут ошибки при установке, попробуйте так:

```
C:\Dev\Jupyter\dist\pyenv3.7-win64\Scripts\activate  
conda config --add channels conda-forge  
conda install numpy scipy matplotlib jupyter jupyterlab
```

и после окончания установки

```
conda.bat deactivate
```

6. В папке C:\Dev\Jupyter\dist создайте файл setenv.bat, который будет управлять тем, где Jupyter и Matplotlib будут хранить свои настройки:

```
@echo off  
  
set conf_path=%~dp0\conf  
  
set JUPYTER_CONFIG_DIR=%conf_path%\jupyter  
set JUPYTER_DATA_DIR=%conf_path%\jupyter\data  
set JUPYTER_RUNTIME_DIR=%conf_path%\jupyter\data\runtime  
set IPYTHONDIR=%conf_path%\ipython  
set MPLCONFIGDIR=%conf_path%\matplotlib  
  
REM Matplotlib search FFMPEG in PATH variable only!  
set PATH=%~dp0\apps\ffmpeg\bin;%PATH%
```

7. В папке C:\Dev\Jupyter\dist создайте файл run\_jupyter\_notebook.bat для запуска Jupyter Notebook с заданными параметрами:

```
@echo off

call %~dp0\setenv.bat
call %~dp0\pyenv3.7-win64\Scripts\jupyter-notebook.exe --notebook-dir=%1
```

8. Аналогично, в папке C:\Dev\Jupyter\dist создайте файл run\_jupyter\_lab.bat для запуска Jupyter Lab с заданными параметрами:

```
@echo off

call %~dp0\setenv.bat
call %~dp0\pyenv3.7-win64\Scripts\jupyter-lab.exe --notebook-dir=%1
```

9. В папке C:\Dev\Jupyter\dist создайте файл enable\_extension.bat, который активирует заданное расширение в Jupyter Notebook:

```
@echo off

REM Enable extension in Jupyter Notebook.
REM Example:
REM enable_extension.bat widgetsnbextension

call %~dp0\setenv.bat
call %~dp0\pyenv3.7-win64\Scripts\jupyter-nbextension.exe enable %1
```

10. Предположим, что рабочие файлы находятся в папке D:\my-projects. В этой папке создайте ярлыки на файлы run\_jupyter\_notebook.bat и run\_jupyter\_lab.bat. После создания каждого из ярлыков зайдите в его свойства и очистите строку «Рабочая папка». Если не очистить — Jupyter не увидит нужную вам папку!

Все потоки   Разработка   Администрирование   Дизайн   Менеджмент   Маркетинг   Научпоп



Войти

Регистрация

следующей командой:

```
C:\Dev\Miniconda3\Scripts\conda.exe clean --all
```

После выполнения данной команды нужно зайти в папку C:\Dev\Miniconda3\pkgs и очистить содержимое папки .trash. Только тогда мы действительно сократим размер папки Miniconda3.

## Установка Miniconda (Python 3.7)

Давайте создадим в корне диска C:\ папку Dev. В этой папке я складываю все программы и инструменты для разработки, которые почему-то предпочитают устанавливаться не в C:\Program Files. Например, туда я устанавливаю Ruby, Go, Python, Jupyter, Msys, SQLite Studio и т.д.

Сначала нам необходимо установить Python. У Python есть две ветки: Python 2 и Python 3. Python 2 поддерживается<sup>(en)</sup> до 2020 года, поэтому будем ставить только Python 3.

Для установки Python 3 обычно обращаются к официальному сайту [python.org](https://python.org)<sup>(en)</sup>, откуда скачивают его и устанавливают. Однако мы хотим получить *переносимую* сборку, поэтому поступим иначе: мы скачаем и установим Miniconda.

Что такое Miniconda? По факту это Python с предустановленным и настроенным менеджером пакетов conda. Консольная программа conda позволит нам создать папку, в которой будет Python нужной нам версии вне зависимости от того, какая версия Python идёт в составе Miniconda. Также с помощью conda в эту папку можно установить практически все известные библиотеки для Python: Numpy, Scipy, Matplotlib, SymPy и т.д. Папка, в которую установлен Python и его библиотеки, называется виртуальным окружением. Библиотеки для Python поставляются в форме специальных архивов, которые называются пакетами.

У conda есть отличительные особенности, из-за которой она удобна и для начинающих и опытных пользователей:

- Пакеты Python, которые устанавливаются через conda, уже скомпилированы под Windows. Меньше вероятность, что попытка установить его завершится ошибкой\*.
- Вы можете создать виртуальное окружение с той версией Python, которая вам нужна. Не имеет значения, какая версия Python установлена с Miniconda.

\*Надо отметить, что ситуация с установкой пакетов в Python из года в год улучшается. Несколько лет назад я не смог установить Numpy через pip (выдавалась ошибка), и я использовал conda. В 2018 году я попробовал последнюю версию pip, и скачался файл с расширением .whl (так называемое «колесо») с уже скомпилированным Numpy, и всё установилось прекрасно.

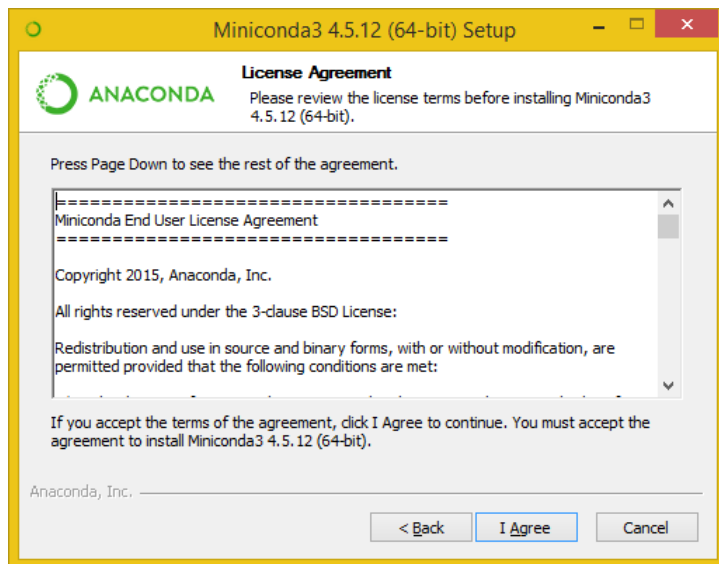
Итак, нам нужно скачать и установить Miniconda. Для этого пройдем на <https://conda.io/miniconda><sup>(en)</sup> и выберем 64-битную версию для Windows на Python 3. Если у вас 32-битный компьютер, вам следует скачать 32-битную версию.

Miniconda ставится так же, как и обычное Windows приложение:

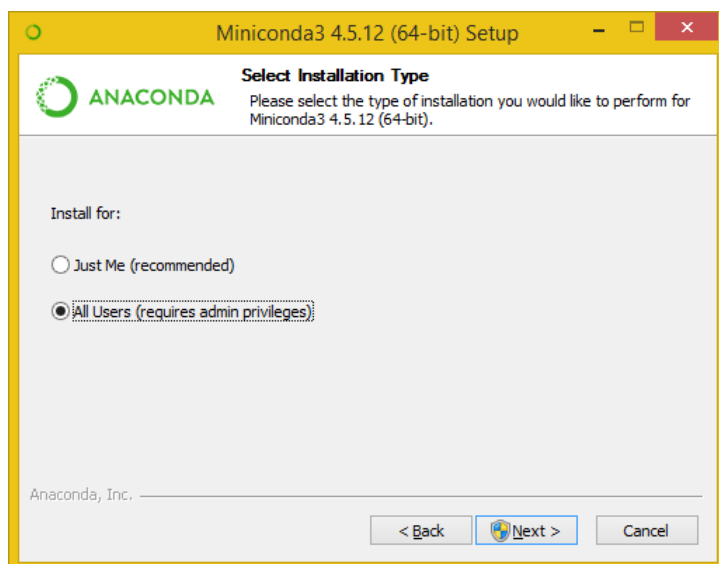
1. Запускаем инсталлятор, жмём Next



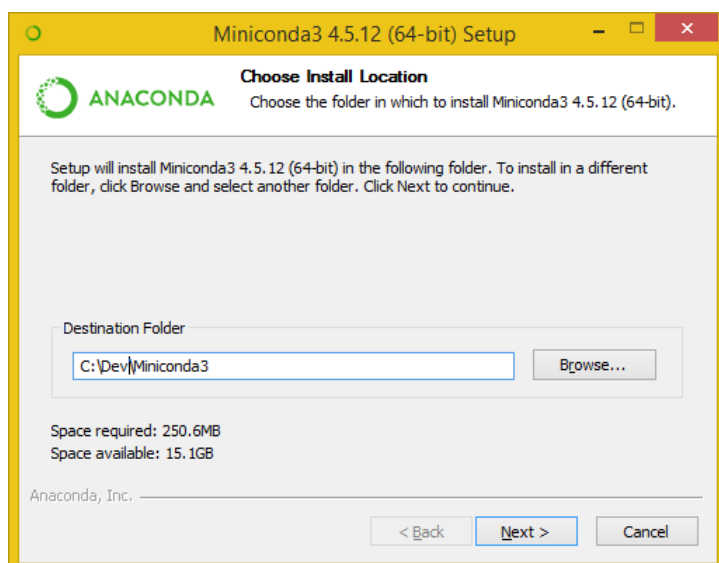
2. Соглашаемся с лицензионным соглашением I Agree



3. Я предпочитаю установку для всех пользователей, потому что это даст мне возможность указать путь для установки. Выбираем пункт «All users»:



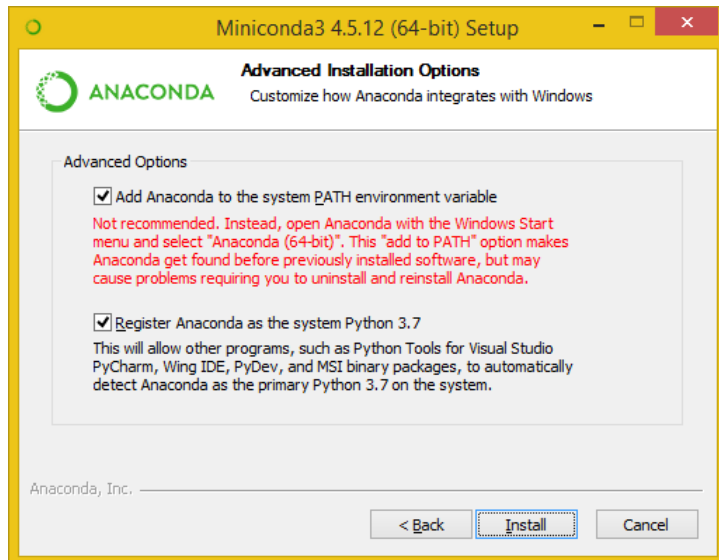
4. Корректируем путь для установки на C:\Dev\Miniconda3:



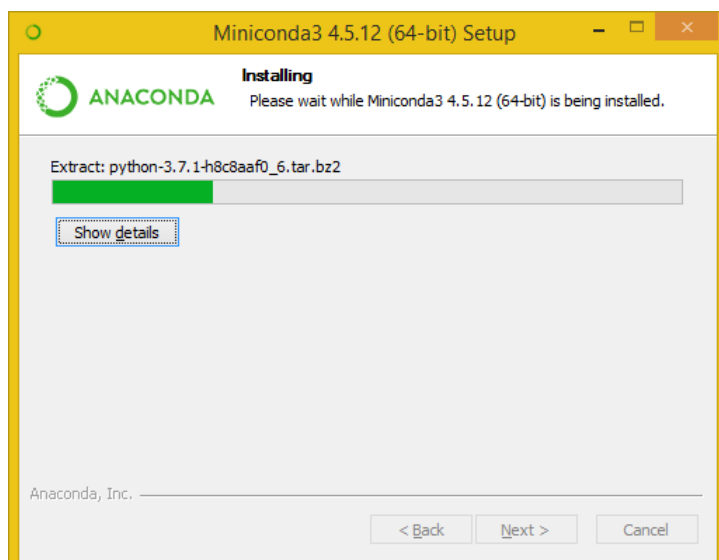
5. Здесь я ставлю оба флажка. Флажок «Add Anaconda to the system PATH environment variable» делает команду `conda` доступной в терминале из любого каталога. Если вы этот флажок не поставите, единственное, что изменится, это то, что в терминале вместо `conda` вам понадобится набрать полный путь к `conda.exe`. Я не устанавливаю Anaconda, потому что она мне ставит много чего лишнего, поэтому я игнорирую нежелательность установки данного флажка. Если вы поставите этот флажок, а после установки передумаете, вы можете просто удалить `conda` из системных переменных. Это просто. Но если не знаете, можете загуглить или спросить. Контакты в конце статьи.

Я также ставлю флажок «Register Anaconda as the system Python 3.7». Если какой-то программе вдруг понадобится Python, она будет использовать Python, установленный вместе с Miniconda. Также данный флажок делает команду `python` доступной в терминале из любой папки. Данный флажок желательно поставить, если до этого вы не устанавливали Python. Если уже какой-то Python установлен, я бы не советовал ставить этот флажок сейчас, а скорректировать системные переменные при необходимости.

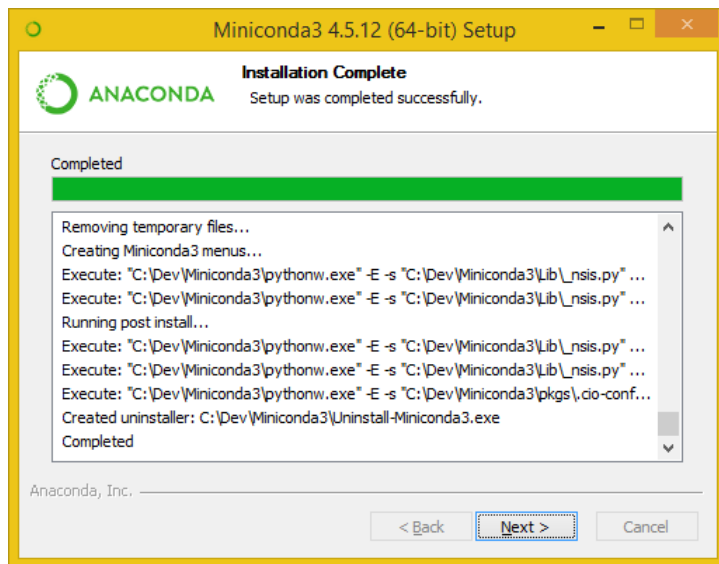
После этого нажимаем Install и начнётся процесс установки:



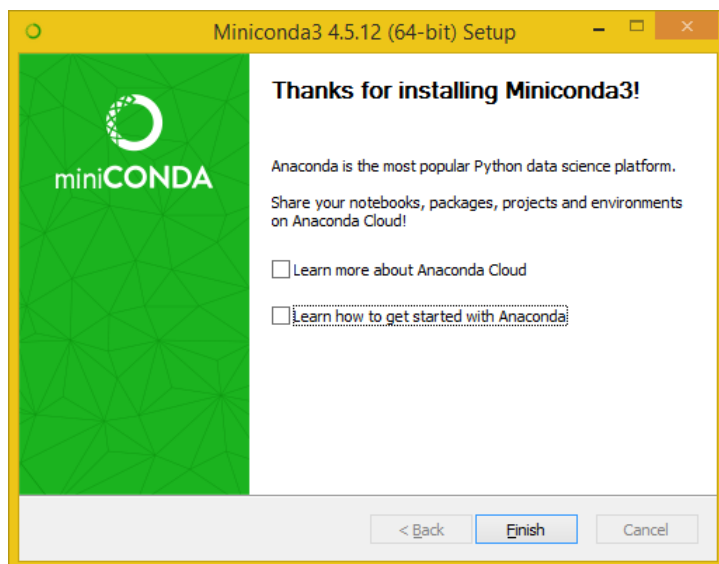
6. Во время установки можете нажать Show details. Тем самым вы увидите больше информации о том, что именно происходит во время установки. Но это не обязательно.



7. Когда установка закончится, появится фраза «Completed», а кнопка Next станет доступной. Жмём Next



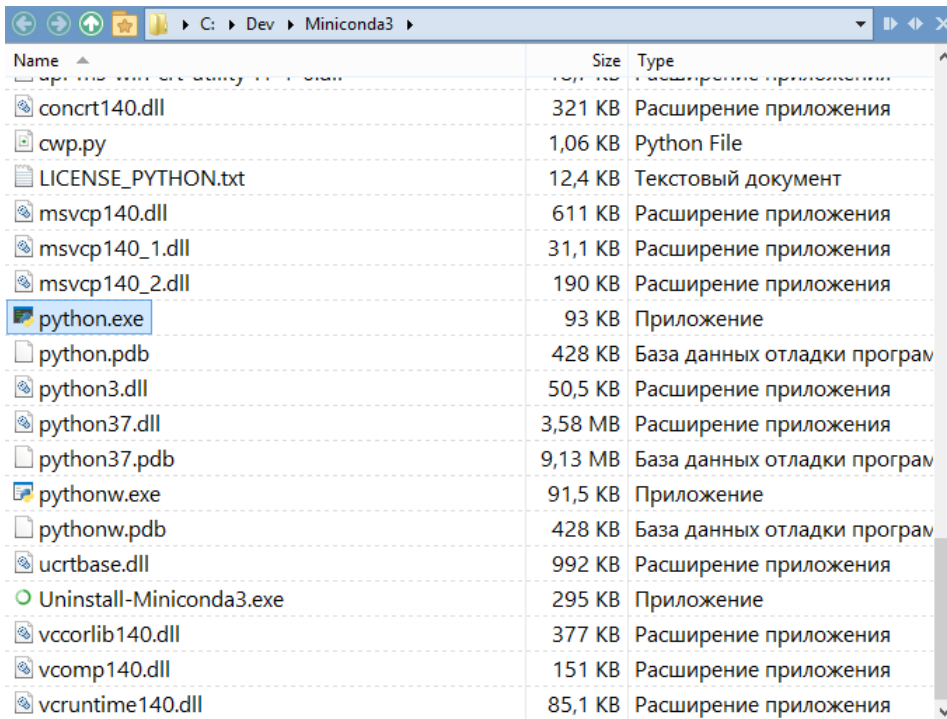
8. В последнем окне нам предлагается узнать про Anaconda Cloud (это первый флажок) и как начать работу с Anaconda (второй флажок). Мне ничего из этого не нужно, поэтому я снимаю все флажки и нажимаю Finish. Установка Miniconda завершена.



После установки Miniconda в папке C:\Dev мы увидим новую папку miniconda весом примерно 340 Мб. Да, это немало, и она ещё будет раздуваться. Позже я покажу, как быстро и безопасно уменьшать её объём.

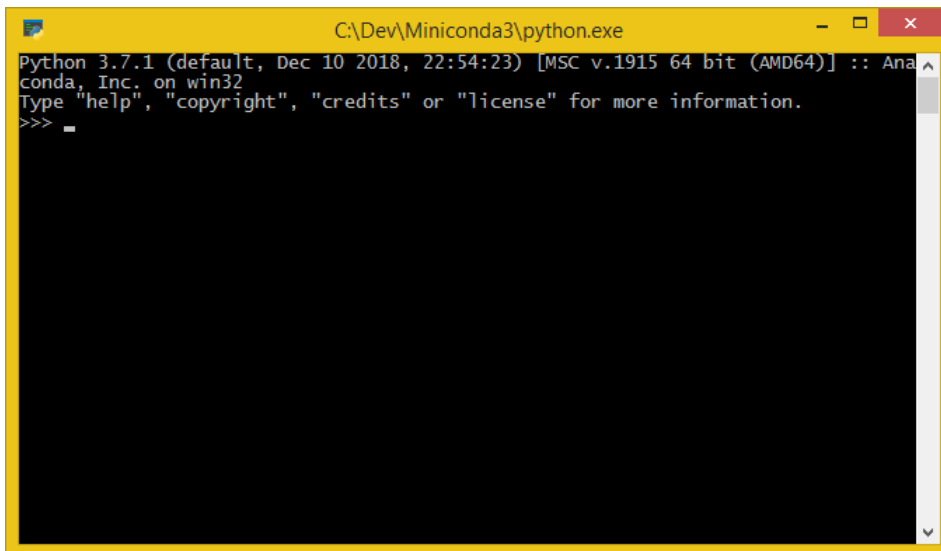
Зайдём в папку miniconda. Немного прокрутив список файлов, мы увидим python.exe. Тот самый Python 3.7, который установился в моём случае (на скриншоте Directory Opus).





Name	Size	Type
concrtdll	321 KB	Расширение приложения
cwp.py	1,06 KB	Python File
LICENSE_PYTHON.txt	12,4 KB	Текстовый документ
msvcp140.dll	611 KB	Расширение приложения
msvcp140_1.dll	31,1 KB	Расширение приложения
msvcp140_2.dll	190 KB	Расширение приложения
python.exe	93 KB	Приложение
python.pdb	428 KB	База данных отладки програм
python3.dll	50,5 KB	Расширение приложения
python37.dll	3,58 MB	Расширение приложения
python37.pdb	9,13 MB	База данных отладки програм
pythonw.exe	91,5 KB	Приложение
pythonw.pdb	428 KB	База данных отладки програм
ucrtbase.dll	992 KB	Расширение приложения
Uninstall-Miniconda3.exe	295 KB	Приложение
vccorlib140.dll	377 KB	Расширение приложения
vcomp140.dll	151 KB	Расширение приложения
vcruntime140.dll	85,1 KB	Расширение приложения

Если дважды кликнуть по `python.exe` — запустится консольное окно, в котором можно вводить команды Python.



Вы можете для теста после `>>>` ввести:

```
import antigravity
```

и нажать Enter. Откроется браузер по умолчанию с комиксом про Python на `xkcd`.

В папке `C:\Dev\Miniconda\Scripts` мы найдём `conda.exe`. Эта та самая консольная команда, с помощью которой мы будем создавать виртуальное окружение Python.

## Создание структуры каталогов

Теперь у нас всё готово для того, чтобы начать создание портативной сборки Jupyter Notebook. Для начала создадим следующую структуру каталогов:

```
C:\
  Dev\
    Jupyter\
```

```
dist\  
  apps\  
  conf\  
    backup\  
    ipython\  
    jupyter\  
    matplotlib\  
  fonts\  
  projects\
```

В папке Dev создайте папку Jupyter. В свою очередь в папке Jupyter создайте папки dist и projects. В папке dist будет виртуальное окружение Python со всеми необходимыми библиотеками, файлы настроек, дополнительные программы, шрифты — всё, что необходимо для нашей разработки на Python в среде Jupyter Notebook или Jupyter Lab. Папка projects — это место по умолчанию для проектов. Сам я эту папку обычно не использую, и она остаётся пустой. Но если мне понадобится передать программу другому пользователю вместе с настроенным Jupyter, я положу свою программу в эту папку projects, сделаю архив всей папки Jupyter и отправлю архив пользователю.

Папка apps содержит вспомогательные программы. Например, я часто кладу туда портативную версию FFmpeg, которая нужна Matplotlib для создания анимации.

Папка conf содержит настройки различных библиотек. В нашем случае для IPython, Jupyter и Matplotlib.

В папку conf\backup я кладу копии своих файлов настроек на случай, если в будущем где-то напортачу с настройками.

Папка fonts содержит шрифты, которые могут быть использованы, например, в Matplotlib. Лично мне понравились Roboto и PT Serif.

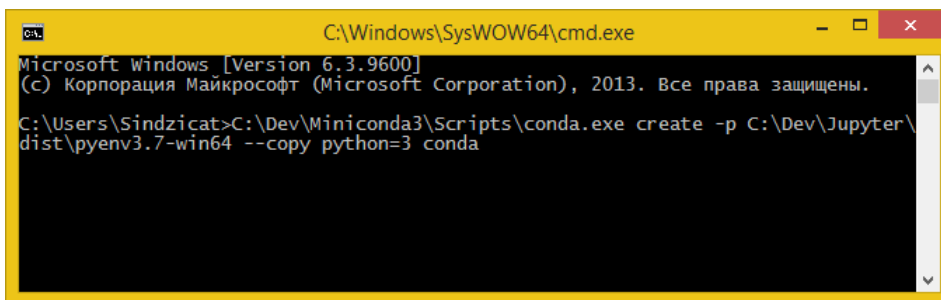
Кроме папок apps, conf и fonts вы можете создать и другие папки на своё усмотрение. Например, папку temp для временных файлов.

## Создание переносимого виртуального окружения Python

### Создание виртуального окружения с помощью conda

Откройте командную строку (Win+R → cmd.exe → Enter) и введите\*:

```
C:\Dev\Miniconda3\Scripts\conda.exe create -p C:\Dev\Jupyter\dist\pyenv3.7-win64 --copy --yes python=3 conda
```



\*Для установки более свежих версий библиотек можно подключить канал conda-forge через аргумент -c conda-forge:

```
C:\Dev\Miniconda3\Scripts\conda.exe create -p C:\Dev\Jupyter\dist\pyenv3.7-win64 --copy --yes -c conda-forge python=3 conda
```

Если потом понадобится удалить канал conda-forge, зайдите в Проводнике в папку %userprofile%, найдите в ней файл .condarc, откройте его блокнотом и удалите строку conda-forge.

Рассмотрим эту команду. Сначала идёт полный путь к `conda.exe`. Если при установке Miniconda вы поставили галочку «Add Anaconda to the system PATH environment variable», вместо полного пути достаточно написать просто `conda`.

Слово `create` даёт команду создания нового окружения. Аргумент `-p` говорит о том, что это окружение должно быть создано там, где мы укажем, а не в папке `C:\Dev\Miniconda3\envs`. В примере прописан полный путь и название будущей папки `pyenv3.7-win64` (расшифровка: `python 3.7 environment for Windows 64-bit`). Если у вас командная строка открыта в папке `dist` или вы с помощью команды `cd` заранее перешли в эту папку, вместо полного пути можно было написать просто `pyenv3.7-win64`.

Аргумент `--copy` сообщает `conda`, что в виртуальном окружении должны быть установлены сами пакеты. В противном случае пакет будет установлен в папке `C:\Dev\Miniconda3`, а в виртуальном окружении будет ссылка на него. Вы не заметите эту подмену, пока не попытаете запустить виртуальное окружение на другом компьютере.

Далее идёт перечисление пакетов. Прежде всего мы должны установить сам Python третьей версии. Также я обязательно указываю `conda`. Т.е. программа `conda` будет установлена дважды: в составе Miniconda и в виртуальном окружении. Установка `conda` в виртуальном окружении увеличивает его размер совсем чуть-чуть, но даст возможность пользователю обновить пакеты в виртуальном окружении на компьютере, где Miniconda не установлена. Это делает виртуальное окружение полностью автономным. Вы можете даже деинсталлировать Miniconda после создания виртуального окружения, и оно продолжит работать как ни в чём не бывало. Я, правда, оставляю Miniconda на тот случай, если какому-то приложению понадобится Python.

Вообще, кроме Python и `conda` можно было сразу указать необходимые пакеты, но в 2018 году я перестал так делать и вместо этого стал использовать для установки пакетов `pip`. Во-первых, новейшие версии `pip` стали скачивать `.whl` файлы с уже скомпилированными библиотеками, и проблемы с установкой ряда библиотек исчезли. Во-вторых, размер виртуального окружения при установке пакетов через `pip` получается в 3 раза меньше, чем при установке пакетов через `conda`.

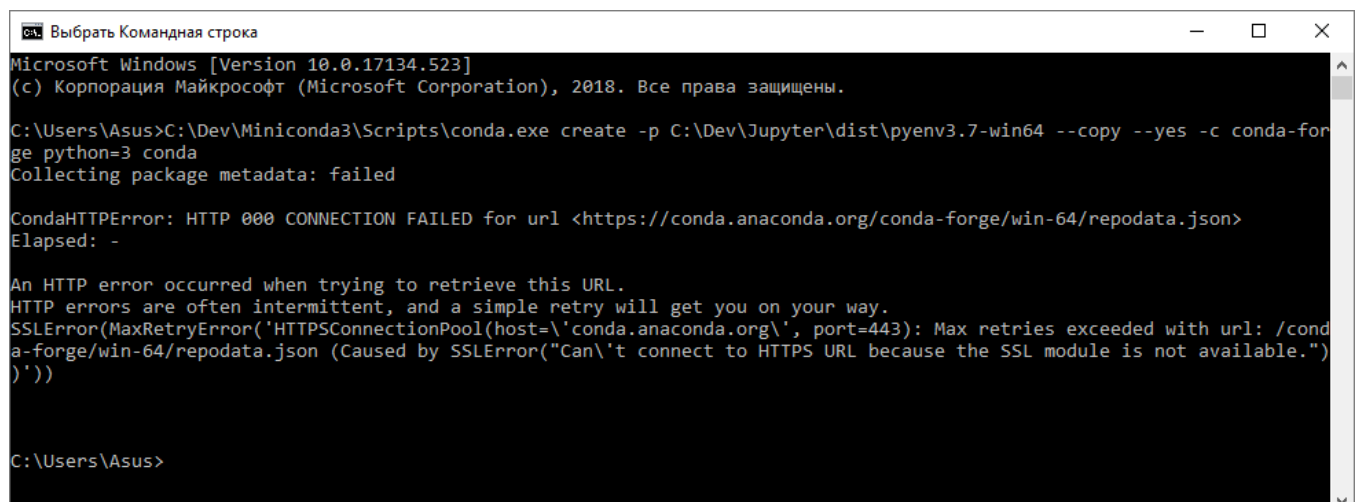
## Исправление ошибки HTTP 000 CONNECTION FAILED при создании виртуального окружения

У одного из пользователей при выполнении команды

```
C:\Dev\Miniconda3\Scripts\conda.exe create -p C:\Dev\Jupyter\dist\pyenv3.7-win64 --copy --yes -c conda-forge python=3 conda
```

столкнулся с ошибкой следующего содержания:

```
> C:\Users\Asus>C:\Dev\Miniconda3\Scripts\conda.exe create -p C:\Dev\Jupyter\dist\pyenv3.7-win64 --copy --yes -c conda-forge python=3 conda
Collecting package metadata: failed
CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://conda.anaconda.org/conda-forge/win-64/repo_data.json>
Elapsed: -
An HTTP error occurred when trying to retrieve this URL.
HTTP errors are often intermittent, and a simple retry will get you on your way.
SSLError(MaxRetryError('HTTPSConnectionPool(host='conda.anaconda.org', port=443): Max retries exceeded with url: /conda-forge/win-64/repo_data.json (Caused by SSLError("Can't connect to HTTPS URL because the SSL module is not available.")))
```



```
Выбрать Командная строка
Microsoft Windows [Version 10.0.17134.523]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

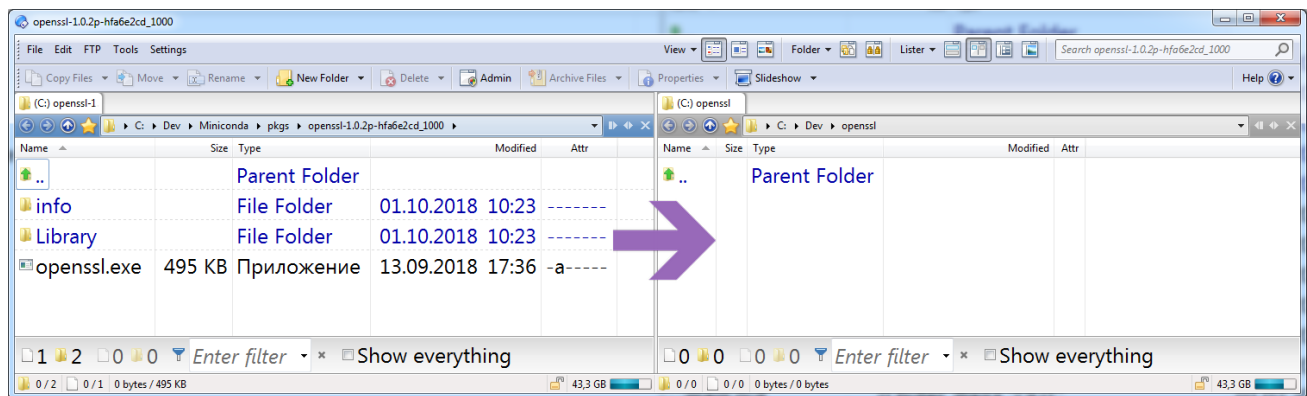
C:\Users\Asus>C:\Dev\Miniconda3\Scripts\conda.exe create -p C:\Dev\Jupyter\dist\pyenv3.7-win64 --copy --yes -c conda-forge python=3 conda
Collecting package metadata: failed
CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://conda.anaconda.org/conda-forge/win-64/repo_data.json>
Elapsed: -
An HTTP error occurred when trying to retrieve this URL.
HTTP errors are often intermittent, and a simple retry will get you on your way.
SSLError(MaxRetryError('HTTPSConnectionPool(host='conda.anaconda.org', port=443): Max retries exceeded with url: /conda-forge/win-64/repo_data.json (Caused by SSLError("Can't connect to HTTPS URL because the SSL module is not available.")))

C:\Users\Asus>
```

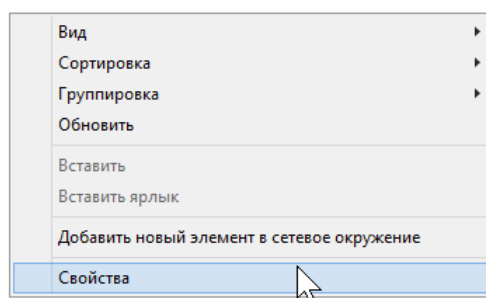
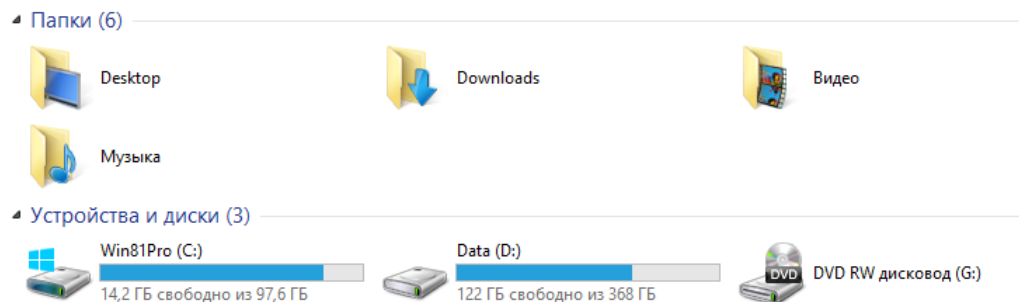
Мне потребовался не один час, чтобы разобраться с ней, потому что на первый взгляд проблема либо с некорректной установкой Miniconda либо с сетью. У некоторых корпоративных пользователей действительно был заблокирован этот ресурс, но проблема происходила у пользователя дома. Переустановка Miniconda не помогла.

В итоге оказалось, что данная ошибка означает, что `conda.exe` не нашло файл `openssl.exe`. В итоге было применено следующее решение:

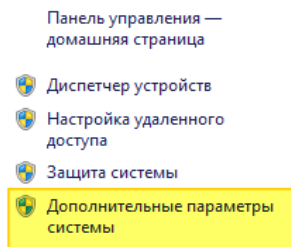
1. Создали папку `C:\Dev\openssl`.
2. В папке `C:\Dev\Miniconda3\pkgs` нашли папку, название которой начинается с `openssl`. Например, `openssl-1.1.1a-he774522_0`. Если папок несколько, выбираем ту, у которой в названии номер больше.
3. В найденной папке ищем файл `openssl.exe` и копируем `openssl.exe` и **все файлы и папки**, которые лежат вместе с `openssl.exe`, в `C:\Dev\openssl`.



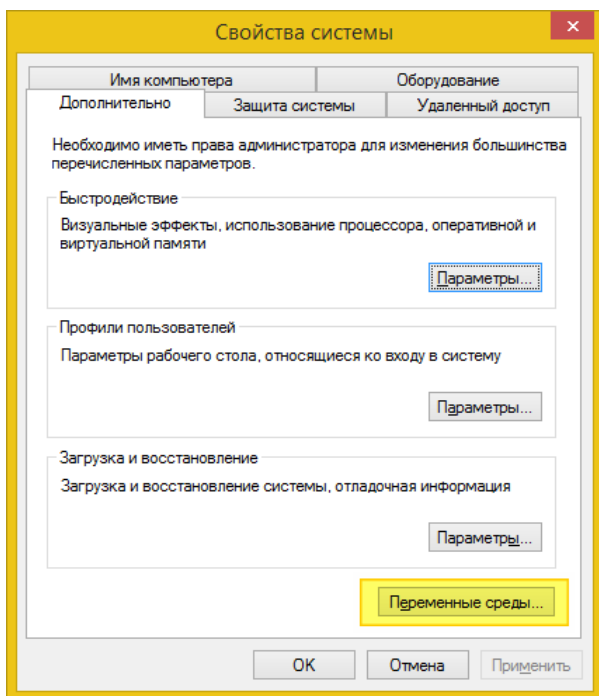
4. В Проводнике Windows заходим в «Этот компьютер» (где перечисляются все диски на компьютере). В свободном месте правым кликом мыши открываем контекстное меню и выбираем в самом низу пункт «Свойства».



5. В открывшемся окне находим «Дополнительные параметры системы»:

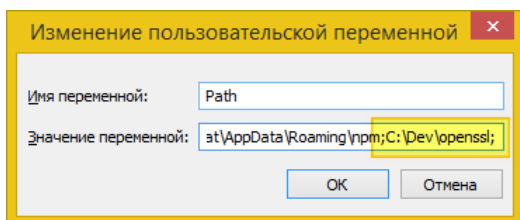


6. На вкладке «Дополнительно» находим кнопку Переменные среды:

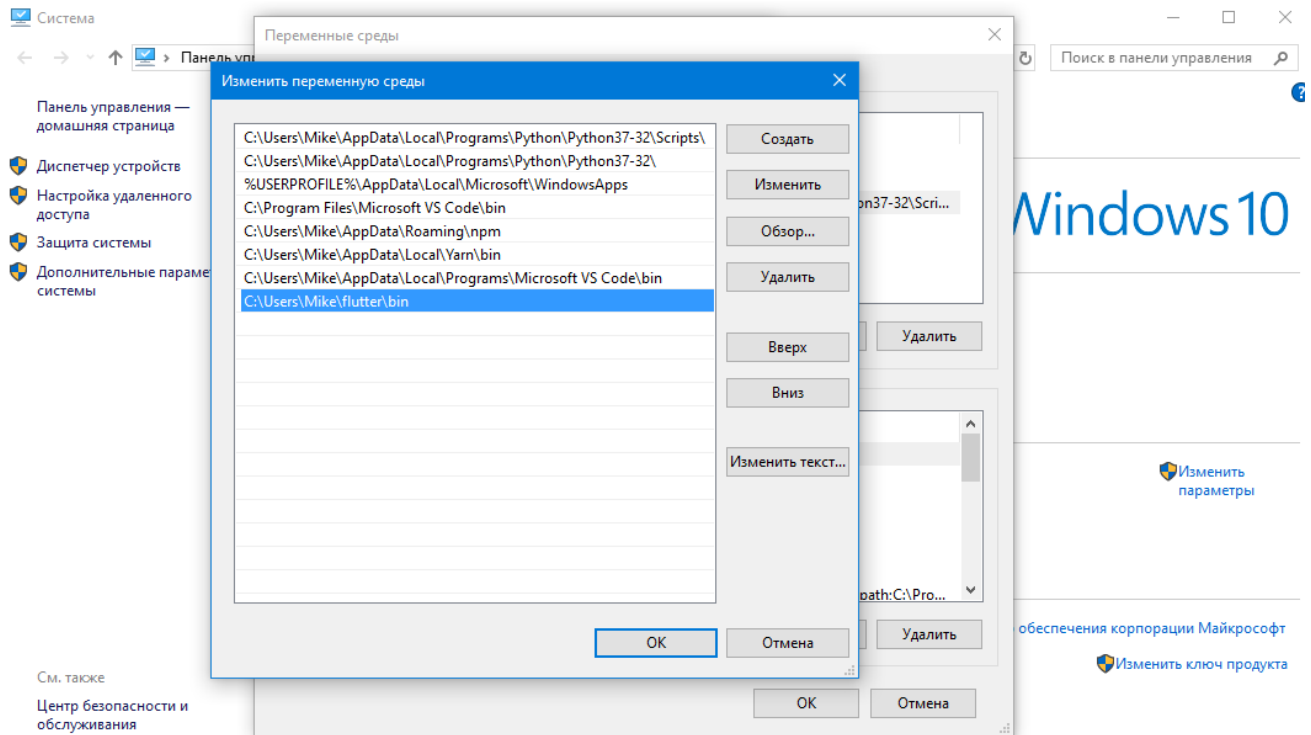


7. **Для Windows 7 и 8:** в разделе «Переменные среды для пользователя» дважды кликаем по переменной Path. Если в конце строки отсутствует точка с запятой, поставим её, и в конце этой строки допишем:

```
C:\Dev\openssl;
```



**Для Windows 10:** в разделе «Переменные среды для пользователя» дважды кликаем по переменной Path. В итоге должно появиться такое окно:



Нажимаем кнопку «Создать» и вставляем путь `C:\Dev\openssl`.

8. Закройте и откройте командную строку снова. Теперь всё должно работать. Если вдруг не заработало — надо гуглить ошибку или обращаться на форумы.

## Активация виртуального окружения

Когда создание виртуального окружения закончится, окно будет выглядеть примерно так:

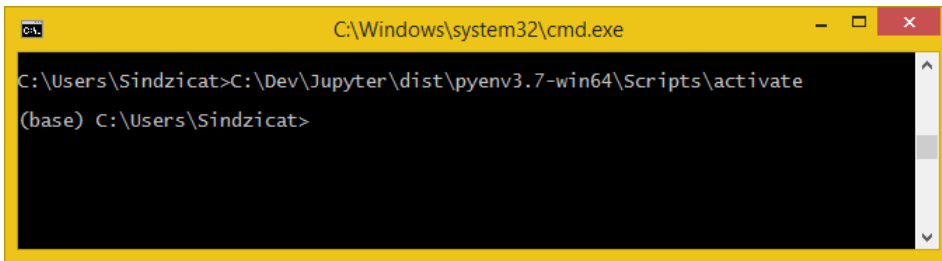
```

C:\Windows\system32\cmd.exe
yml-0.1.7      221 KB  ##### 100%
urllib3-1.24.1 148 KB  ##### 100%
win_inet_pton-1.0.1 5 KB    ##### 100%
six-1.12.0     21 KB   ##### 100%
requests-2.21.0 84 KB   ##### 100%
certifi-2018.11.29 144 KB  ##### 100%
ruamel_yaml-0.15.71 271 KB  ##### 100%
pycosat-0.6.3  98 KB   ##### 100%
chardet-3.0.4  184 KB  ##### 100%
cryptography-2.5 552 KB  ##### 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use:
# > activate C:\Dev\Jupyter\dist\pyenv3.7-win64
#
# To deactivate an active environment, use:
# > deactivate
#
# * for power-users using bash, you must source
#
C:\Users\Sindzicat>
  
```

После создания виртуального окружения установим пакеты через `pip`. Сначала необходимо активировать виртуальное окружение. Для этого в окне команд введите:

```
C:\Dev\Jupyter\dist\pyenv3.7-win64\Scripts\activate
```

В результате вы должны получить примерно следующее:

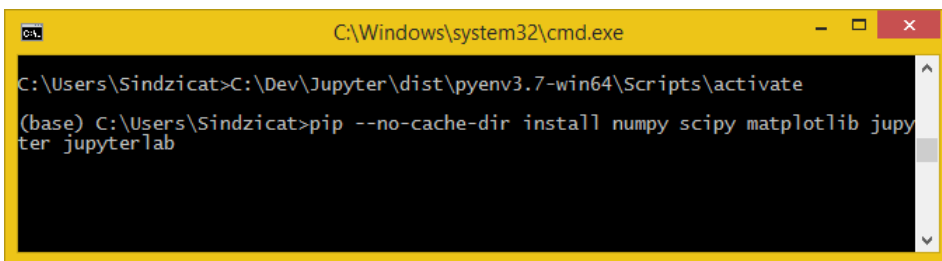


Слово (base) в начале строки как указывает на то, что мы вошли в нужное нам виртуальное окружение.

## Установка пакетов Python в виртуальном окружении

Теперь можно установить пакеты\*:

```
pip --no-cache-dir install numpy scipy matplotlib jupyter jupyterlab
```



Аргумент --no-cache-dir сообщает pip, что не следует кешировать скачанные пакеты. Это позволит нам не увеличивать размер папки виртуального окружения.

\*Существует разработанная Intel библиотека MKL (Math Kernel Library)<sup>(en)</sup>, которая ускоряет работу на больших данных для популярных библиотек Python, в частности, Numpy и Scipy. Если вы хотите установить Numpy и Scipy, которые используют MKL, следует использовать<sup>(en)</sup> intel-numpy вместо numpy и intel-scipy вместо scipy:

```
pip --no-cache-dir install intel-numpy intel-scipy matplotlib jupyter jupyterlab
```

У меня получилось установить intel-numpy и intel-scipy только в виртуальном окружении с Python 3.6. Если вы хотите использовать Numpy и Scipy с MKL в окружении с Python 3.7, необходимо использовать команду:

```
conda install numpy scipy
```

Если вы не уверены, что ставить, используйте просто numpy и scipy.

Если в процессе установки через pip возникнут ошибки, попробуйте установить проблемные пакеты через conda. Пример:

```
conda install numpy scipy matplotlib jupyter jupyterlab
```

## Выход из виртуального окружения Python

После того, как установка завершена, необходимо выйти из виртуального окружения. Для этого в командной строке наберите\*:

```
conda.bat deactivate
```

\*Раньше я набирал просто deactivate, но это почему-то устарело, и надо набирать conda.bat deactivate. Даже conda deactivate будет неправильно.

## Подготовка портативной сборки Jupyter к запуску

Создадим несколько .bat файлов, которые будут заставить Jupyter и Matplotlib хранить настройки в папке dist\config, а также будут управлять запуском Jupyter Notebook и Jupyter Lab.

### Настройка переменных окружения для Jupyter, IPython и Matplotlib

Каталоги размещения настроек определяются переменными среды Windows. Изменив эти переменные, мы заставим Jupyter и Matplotlib хранить свои файлы там, где это нужно именно нам. В папке C:\Dev\Jupyter\dist создайте файл setenv.bat следующего содержания:

```
@echo off

set conf_path=%~dp0\conf

set JUPYTER_CONFIG_DIR=%conf_path%\jupyter
set JUPYTER_DATA_DIR=%conf_path%\jupyter\data
set JUPYTER_RUNTIME_DIR=%conf_path%\jupyter\data\runtime
set IPYTHONDIR=%conf_path%\ipython
set MPLCONFIGDIR=%conf_path%\matplotlib

REM Matplotlib search FFMPEG in PATH variable only!
set PATH=%~dp0\apps\ffmpeg\bin;%PATH%
```

Разберём, что делается в этом файле.

Команда @echo off необходима для того, чтобы в командной строке не выводилось сообщение при выполнении каждой строки нашего файла.

Команда set создаёт переменную. Конструкция %~dp0 означает полный путь к setenv.bat. Обратите внимание, что пробелов до и после знака = быть не должно.

Затем мы настраиваем переменные для Jupyter:

- JUPYTER\_CONFIG\_DIR — папка для файлов конфигурации Jupyter (документация<sup>(en)</sup>),
- JUPYTER\_DATA\_DIR — папка для устанавливаемых файлов данных (расширения и ядра (kernel) для Jupyter) (документация<sup>(en)</sup>),
- JUPYTER\_RUNTIME\_DIR — папка для исполняемых файлов Jupyter (runtime files) (документация<sup>(en)</sup>),
- IPYTHONDIR — папка для файлов конфигурации IPython (документация<sup>(en)</sup>),
- MPLCONFIGDIR — папка, где Matplotlib хранит свои настройки и кэш (документация<sup>(en)</sup>).

Если вы планируете создавать анимации с Matplotlib, вам понадобится FFMPEG<sup>(ru)</sup>. Я скачиваю<sup>(en)</sup> zip архив FFMPEG, распаковываю его содержание C:\Dev\Jupyter\dist\apps\ffmpeg.

Строка, которая начинается с REM — комментарий. Matplotlib почему-то ищет FFMPEG только в %PATH%. Я записываю путь к FFMPEG в начало %PATH, а не в его конец, чтобы при поиске первым нашёлся тот FFMPEG, который я положил в dist\apps.

### Создание файла для запуска Jupyter с настройками пользователя

В папке C:\Dev\Jupyter\dist создайте файл run\_jupyter\_notebook.bat следующего содержания:



```
@echo off

call %~dp0\setenv.bat
call %~dp0\pyenv3.7-win64\Scripts\jupyter-notebook.exe --notebook-dir=%1
```

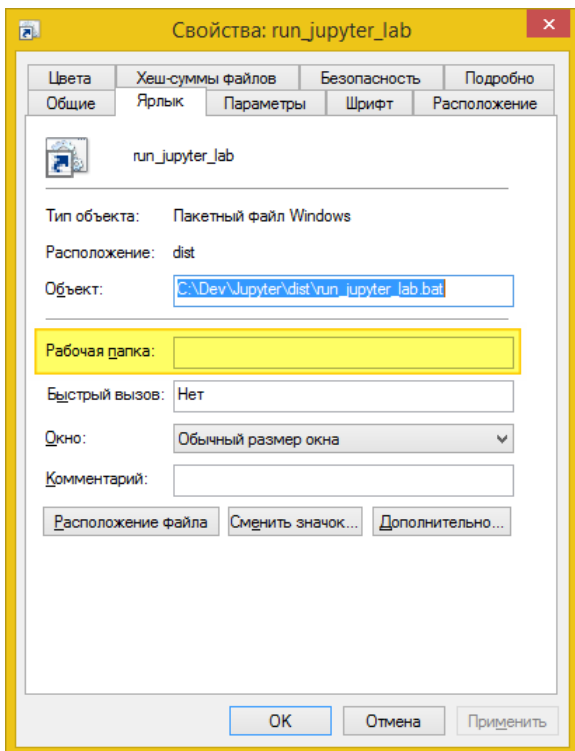
Аналогично, в папке C:\Dev\Jupyter\dist создайте файл run\_jupyter\_lab.bat следующего содержания:

```
@echo off

call %~dp0\setenv.bat
call %~dp0\pyenv3.7-win64\Scripts\jupyter-lab.exe --notebook-dir=%1
```

Каждый из этих файлов сначала выполняет setenv.bat, т.е. настраивает переменные окружения, потом запускает Jupyter Notebook или Jupyter Lab и указывает ему, где папка с нашими файлами для проекта.

Предположим, что есть папка D:\my-projects, в которой мы будем хранить файлы Jupyter Notebook или Lab. В этой папке создайте ярлыки на файлы run\_jupyter\_notebook.bat и run\_jupyter\_lab.bat. После этого в обязательном порядке откройте свойства каждого из этих ярлыков и сделайте пустой строку «Рабочая папка». Если вы этого не сделаете — Jupyter не увидит вашу папку!



После того, как это сделали, можете кликнуть дважды по любому из ярлыков. Сначала появится новое окно командной строки, потом откроется браузер по умолчанию и в нём запустится Jupyter Notebook или Lab в новой вкладке. Поздравляю: квест пройден!

## Дополнительные файлы для выполнения служебных действий

Для Jupyter Notebook написаны расширения (о них будет подробнее в части 2). Но их недостаточно установить. Их ещё надо активировать. Согласно документации, вам нужно сделать следующее (не выполняйте эту команду!):

```
jupyter nbextension enable <nbextension require path>
```

Но мы не можем выполнить команду в таком виде, потому что настройки окажутся вне портативной сборки. Мы должны сделать иначе:

```
C:\Dev\Jupyter\dist\setenv.bat
C:\Dev\Jupyter\dist\pyenv3.7-win64\Scripts\jupyter.exe nbextension enable <nbextension require path>
```

Чтобы упростить себе задачу, мы можем в папке C:\Dev\Jupyter\dist создать файл enable\_extension.bat следующего содержания:

```
@echo off

REM Enable extension in Jupyter Notebook.
REM Example:
REM enable_extension.bat widgetsnbextension

call %~dp0\setenv.bat
call %~dp0\pyenv3.7-win64\Scripts\jupyter-nbextension.exe enable %1
```

В итоге наша запись в окне командной строки сократится и станет такой:

```
C:\Dev\Jupyter\dist\enable_extension.bat <nbextension require path>
```

Если вам время от времени в окне команд нужно запускать различные действия с jupyter, можно создать в папке C:\Dev\Jupyter\dist файл jupyter.bat следующего содержания:

```
@echo off
call %~dp0\setenv.bat
call %~dp0\pyenv3.7-win64\Scripts\jupyter.exe %1 %2 %3 %4 %5 %6 %7 %8 %9
```

Аналогично можно сделать для запуска IPython и других случаев.

## Заключение

Часть 1 подошла к концу. Мы создали полностью портативную и переносимую сборку Jupyter и можем работать с файлами, которые лежат в нужной нам папке. Для этого создаём ярлыки на run\_jupyter\_notebook.bat и run\_jupyter\_lab.bat, в свойствах ярлыков обязательно очищаем строку «Рабочая папка», и всё готово к работе.

В части 2 будут рассмотрены различные вопросы кастомизации Jupyter Notebook, Jupyter Lab и Matplotlib. Научимся подключать расширения в Jupyter Notebook и Jupyter Lab, настраивать сочетания клавиш, размер шрифта и прочие настройки, напишем свои «магические команды».

Если у вас остались вопросы, но нет аккаунта на Хабре, вот мои контакты:

ВК: <https://vk.me/sindzicat>

Telegram: <https://t.me/sindzicat>

E-mail: [sh.andr.gen@yandex.ru](mailto:sh.andr.gen@yandex.ru)

Теги: [python](#), [jupyter](#), [jupyter notebook](#), [jupyterlab](#)

Хэбы: [Python](#)

↑ +13 ↓ 87 23,1k 8 Поделиться

**Андрей Ширшов @sindzicat**

Пользователь

[ВКонтакте](#) [Github](#) [Telegram](#)

## ПОХОЖИЕ ПУБЛИКАЦИИ

9 октября 2017 в 16:55

**Как получать оповещения от Jupyter notebook в Telegram?**

↑ +22 👁 11,6k 📖 118 💬 17

25 сентября 2017 в 14:00

**Визуализация результатов выборов в Москве на карте в Jupyter Notebook**

↑ +57 👁 29,5k 📖 176 💬 45

21 марта 2017 в 11:47

**Использование ArcGIS API for Python в Jupyter Notebook**

↑ +23 👁 13,9k 📖 96 💬 7

## ЗАКАЗЫ

**Необходимо сделать парсер по Вконтакте (или доработать существующий)**

30 000 Р за проект • 6 откликов • 59 просмотров

**Разработка модуля заявок для системы-маркетплейса ( Python / Flask )**

30 000 Р за проект • 4 отклика • 51 просмотр

**Проверка работ студентов по курсу Python-разработчик**

30 000 Р за проект • 15 откликов • 76 просмотров

**Решить пример для курсовой на языке Python**

500 Р за проект • 7 откликов • 49 просмотров

**Разработать платформу для работы с базами данных**

1 000 Р за час • 16 откликов • 84 просмотра

Больше заказов на Хабр Фрилансе

## Комментарии 8

**emkh** 12 февраля 2019 в 13:51 🗨 📖

↑ 0 ↓

Ссылка на телеграмм у вас стоит по умолчанию на telegram.

**sindzicat** 12 февраля 2019 в 14:52 🗨 📖 🔄 ⬆

↑ 0 ↓

Спасибо! Исправил.

**mgstr93** 13 февраля 2019 в 10:08 🗨 📖

↑ 0 ↓

А почему бы не создать докер контейнер для всего этого? В плане использования будет гораздо проще чем следовать пошаговой инструкции.

 **sindzicat** 13 февраля 2019 в 10:16 0 

Спасибо за интересную идею! Я знаю про Docker, но вот насчёт того, что это будет проще, у меня сомнения:

1. Я могу создать портативную сборку и скинуть её пользователю. Ему только распаковать, и он может работать. Ничего дополнительно, включая Docker, ему ставить не надо.
2. Docker нормально поддерживает Windows 10. Что касается Windows 7 или 8, я пытался установить в одно время и это даже работало, но выглядело это всё как костыли. Я пока не понял, чем это лучше обычной виртуалки.
3. Если пользователь (особенно начинающий) решит поправить что-то в файлах, ему достаточно зайти в папку dist. В случае с Docker ему ещё надо разобраться, как зайти в файловую систему.
4. Созданная мною портативная сборка позволит писать программы под Windows. У Docker обычно Linux в образах...

Мнение моё конечно субъективно, но я считаю, что лучше минимум прослоек. Особенно для начинающих.

 **mgstr93** 13 февраля 2019 в 23:08 0 

Ему только распаковать, и он может работать. Ничего дополнительно, включая Docker, ему ставить не надо.

Да Docker поставить придется, но на мой взгляд он как универсальный инструмент все равно должен быть установлен на любой машине.

А если серьезно то это дополнительная зависимость, но зато она позволяет избавиться от всех других зависимостей.

И пользователю после этого достаточно будет передать короткую команду вида `docker run --name jupyter -d -p 8888:8888 jupyter/minimal-notebook`.

Такую команду можно отправить письмом, в мессенджере и СМСкой. Понятно что пользователю придется скачать имидж, но это все сделает докер и думать о том как передать сотни мегабайтов не нужно.

На Windows 7 нормально работает Docker for Windows Toolbox. Использовать его или Docker for Windows это предпочтения виртуализации — Hyper V или Virtual Box, дело вкуса.

Если пользователь (особенно начинающий) решит поправить что-то в файлах, ему достаточно зайти в папку dist. В случае с Docker ему ещё надо разобраться, как зайти в файловую систему.

Зайти очень просто `"docker exec -it /bin/bash"` но делать этого не надо.

Идеология Докера в том, чтобы создать новый контейнер на основе имиджа, а делается это всего несколькими строками Dockerfile.

```
FROM jupyter/minimal-notebook
EXPOSE 8888
RUN pip install --upgrade pip \
    && pip install pandas
ENTRYPOINT ["/opt/conda/bin/jupyter", "notebook", "--NotebookApp.token='']
```

Теперь получился имидж основанный на minimal-notebook в который поставлена pandas. Очень просто и наглядно.

Созданная мною портативная сборка позволит писать программы под Windows. У Docker обычно Linux в образах..

Да в Докере Линукс под капотом, но если ставится Jupyter то какая разница где бежит кросс-платформенный Питон? Если же речь о том, чтобы редактировать файлы в любимом редакторе, то сделав shared volume для докер контейнера — можно редактировать файлы в любом редакторе проинсталлированным под Windows, а код запускать внутри контейнера, где все окружение стандартизировано.

Весьма рекомендую взглянуть на решение основанное на докере. На работе мы уже давно используем Jupyter, вернее Jupyterhub, потому что хотели получить многопользовательское решение с нулевым footprint на машине пользователей.

 **pavelkargashin** 13 февраля 2019 в 10:27 0 

Интересно, мне и полезно очень оказалось. Автору огромное спасибо.

Есть ли возможность пояснить как настраивать pandas. Я часто использую numpy-pandas.

Правильно ли я понял, что папки создаются под отдельные пакеты?



sindzicat

13 февраля 2019 в 10:40



Каждая папка в `dist\config` действительно под отдельный пакет. Эта структура не жёсткая. Например, вы можете использовать два виртуальных окружения с различными версиями Python, и структура файлов в папке `dist` может быть примерно такой:

```
C:\
  Dev\
    Jupyter\
      dist\
        apps\
        conf\
          py2.7\
            ipython\
            jupyter\
            matplotlib\
          py3.7\
            ipython\
            jupyter\
            matplotlib\
        fonts\
        tmp\
        pyenv2.7-win64
        pyenv3.7-win64
        projects\
```

Только не забудьте подредактировать пути в `setenv.bat`.

Вообще, Matplotlib, Jupyter и IPython по умолчанию создают папки `.matplotlib`, `.jupyter` и `.ipython`. Имена этих папок начинаются с точки, что соответствует скрытому каталогу в Linux. На Windows можно создать папки, название которой начинается с точки, но либо консольной командой `mkdir`, либо в специальных файловых менеджерах, например, Directory Opus. Я решил избавить новичков от таких нюансов, поэтому в инструкции папки без точки в начале названия.

Что касается Pandas, я сходу не нашёл, какие переменные окружения использует эта библиотека. Честно говоря, я с большим трудом нашёл про переменные окружения у Matplotlib. Подскажите пожалуйста, где в документации найти это, и я добавлю про Pandas в свою инструкцию.



sindzicat

13 февраля 2019 в 10:45



Мне сейчас удалось найти раздел `Setting Startup Options in python/ipython Environment`. Это будет учтено во второй части данной статьи.

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ

- Сутки
- Неделя
- Месяц

Тёмная сторона работы в Яндекс.Маркете

+272 46,3k 86 339

Советы по выбору усилителя сигнала сотовой связи 2G/3G/4G/5G

+35 22,9k 151 68

Apple предупредила мародёров, что отследит все украденные телефоны

+32 20,3k 14 77

Хватит натягивать сову на глобус

+41 23,4k 42 149

13 человек получили президентские грамоты за заслуги в становлении Рунета

+22 11,5k 3 31

Ваш аккаунт	Разделы	Информация	Услуги
Войти	Публикации	Устройство сайта	Реклама
Регистрация	Новости	Для авторов	Тарифы
	Хабы	Для компаний	Контент
	Компании	Документы	Семинары
	Пользователи	Соглашение	Мегaproекты
	Песочница	Конфиденциальность	

Если нашли опечатку в посте, выделите ее и нажмите Ctrl+Enter, чтобы сообщить автору.