



Pip и виртуальные окружения

работа с пакетным менеджером, работа с виртуальными окружениями, как сделать так, чтобы пакеты одного проекта не мешали пакетам другого проекта



Что такое pip

pip – пакетный менеджер, который используется для установки и управления программными пакетами, написанными на Python. Пакеты, которые устанавливаются с помощью pip, можно найти на сайте PyPi (Python Package Index): <https://pypi.org/>

- Версии языка начиная с Python 2.7.9 и Python 3.4 содержат в себе pip (pip3) по умолчанию

У некоторых языков программирования есть свои аналоги pip:

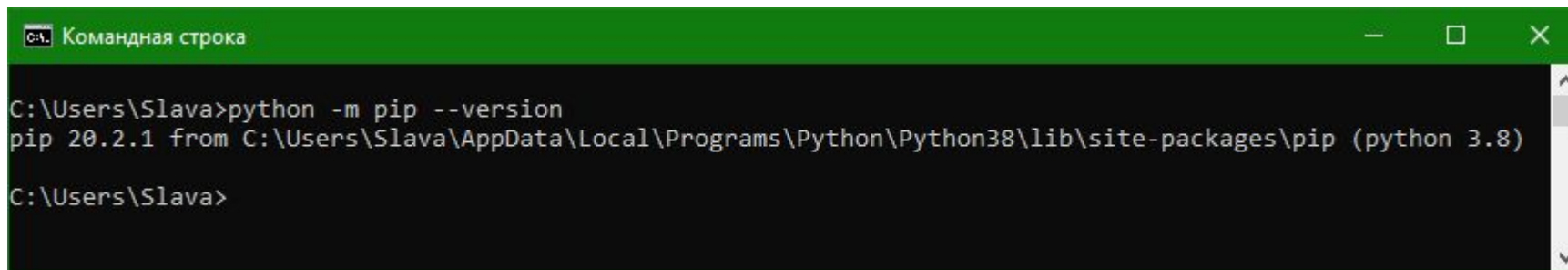
- JavaScript – NPM
- Ruby – Gem
- PHP – Composer



Проверка установки

Для того, чтобы проверить, установлен ли pip, необходимо выполнить команду:

- `pip --version`



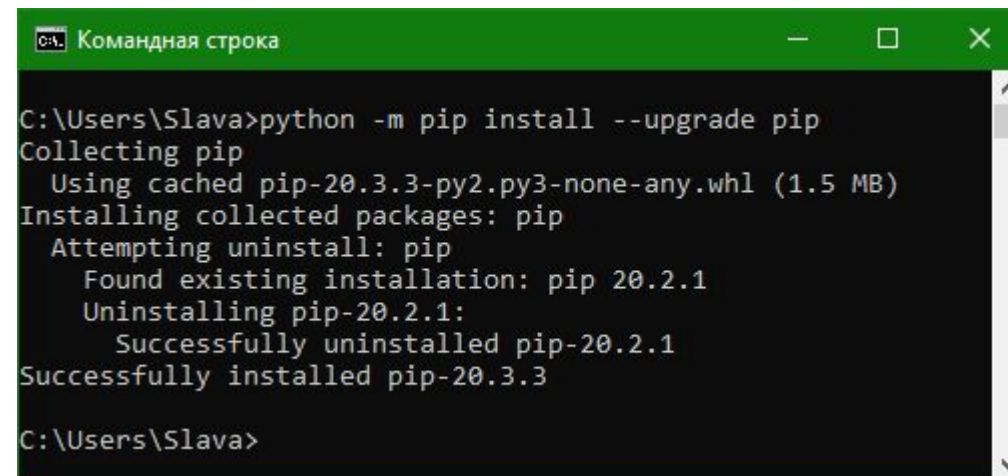
```
C:\Users\Slava>python -m pip --version
pip 20.2.1 from C:\Users\Slava\AppData\Local\Programs\Python\Python38\lib\site-packages\pip (python 3.8)
C:\Users\Slava>
```



Обновляем pip до последней версии

Для обновления версии pip необходимо выполнить команду

- `pip install --upgrade pip`



```
Командная строка
C:\Users\Slava>python -m pip install --upgrade pip
Collecting pip
  Using cached pip-20.3.3-py2.py3-none-any.whl (1.5 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.1
    Uninstalling pip-20.2.1:
      Successfully uninstalled pip-20.2.1
Successfully installed pip-20.3.3
C:\Users\Slava>
```



Установка pip в Windows

Если pip не установлен, то необходимо выполнить следующие действия:

1. Скачать установочный скрипт: <https://bootstrap.pypa.io/get-pip.py>
2. Открыть командную строку
3. Перейти в каталог со скачанным скриптом
4. Выполнить команду: `python get-pip.py`



Установка pip в Linux (Ubuntu)

Если pip не установлен, то сперва необходимо попробовать установить его с помощью системного пакетного менеджера:

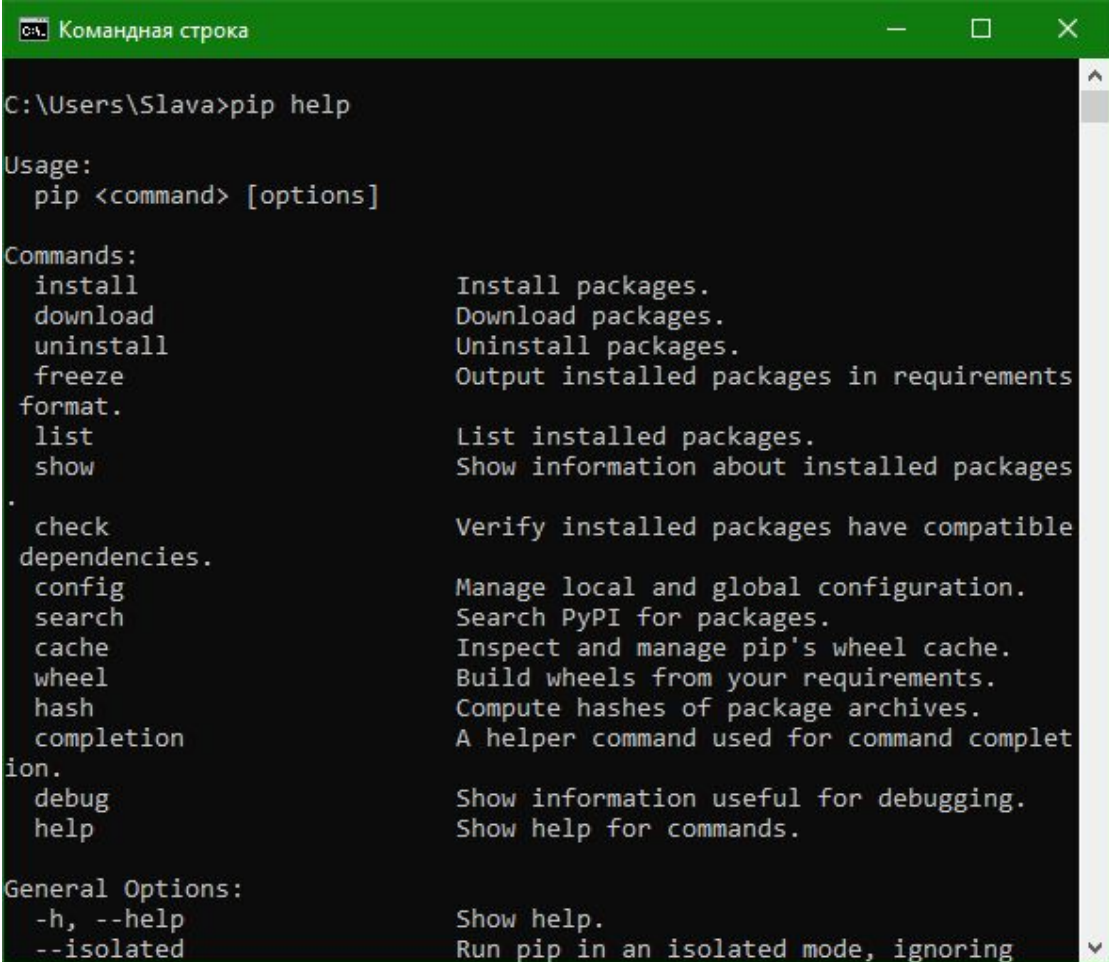
- Для Python 2: `sudo apt-get install python-pip`
- Для Python 3: `sudo apt-get install python3-pip`

Если установить не получилось, то выполнить те же действия, что и для ОС Windows



Просмотр списка основных команд и опций pip

Для просмотре списка основных команд и опций pip, необходимо выполнить в консоли или терминале: `pip help`



```
C:\Users\Slava>pip help

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements
  format            format.
  list              List installed packages.
  show              Show information about installed packages
  .
  check             Verify installed packages have compatible
dependencies.
  config            Manage local and global configuration.
  search            Search PyPI for packages.
  cache             Inspect and manage pip's wheel cache.
  wheel             Build wheels from your requirements.
  hash              Compute hashes of package archives.
  completion        A helper command used for command complet
ion.
  debug             Show information useful for debugging.
  help              Show help for commands.

General Options:
  -h, --help        Show help.
  --isolated         Run pip in an isolated mode, ignoring
```

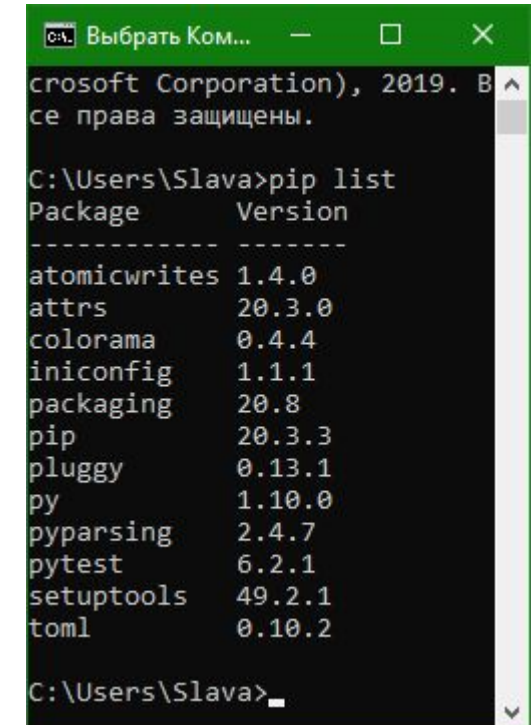


Список установленных пакетов

Для того, чтобы просмотреть, какие пакеты установлены, необходимо выполнить команду: `pip list`

Полезные опции:

- `-o` или `--outdated` – список установленных пакетов, для которых доступны обновления
- `-u` или `--uptodate` – список пакетов не требующих обновления
- `-l` или `--local` – список пакетов виртуального окружения (virtualenv)
- `--user` – список пакетов, установленных в окружении пользователя



```
Microsoft Corporation), 2019. В
се права защищены.

C:\Users\Slava>pip list
Package      Version
-----
atomicwrites 1.4.0
attrs        20.3.0
colorama     0.4.4
iniconfig    1.1.1
packaging    20.8
pip          20.3.3
pluggy       0.13.1
py           1.10.0
pyparsing    2.4.7
pytest       6.2.1
setuptools   49.2.1
toml         0.10.2

C:\Users\Slava>
```

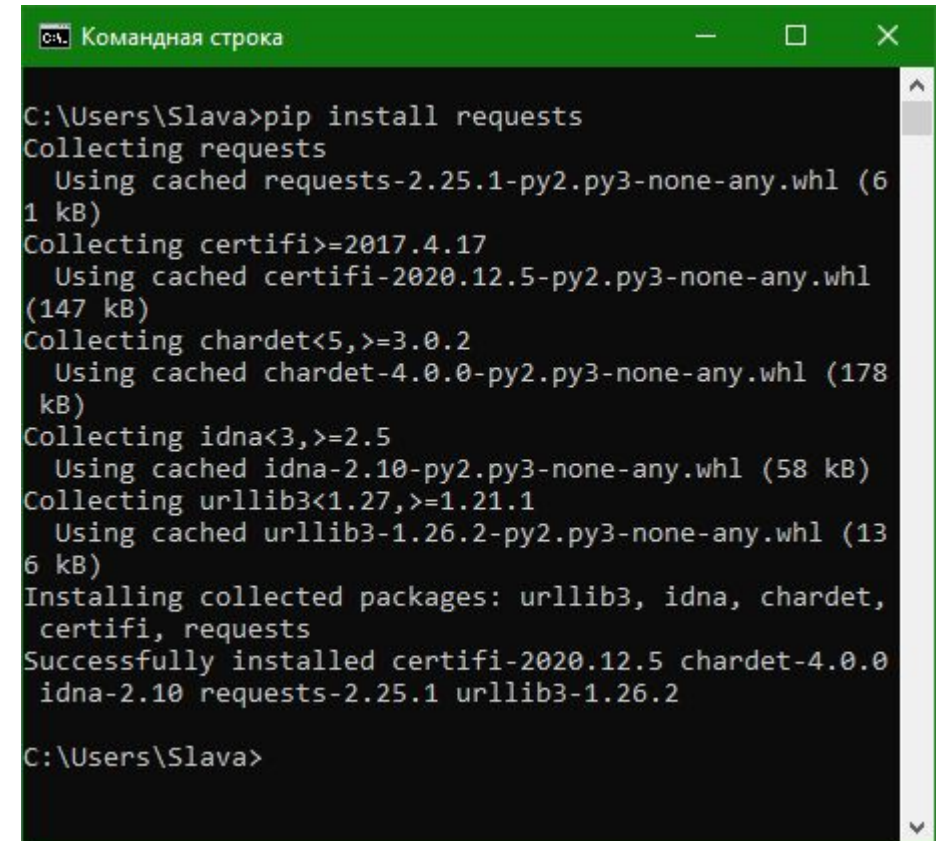


Установка стороннего пакета

Для установки стороннего пакета, необходимо выполнить команду: `pip install имя_пакета`

Полезные опции:

- `-r, --requirement <file>` – установка пакетов по списку из указанного requirement-файла
- `-U, --upgrade` – обновление пакета до последней доступной версии
- `-I (ай большое), --ignore-installed` – игнорировать уже установленные пакеты, не переустанавливать и не



```
C:\Users\Slava>pip install requests
Collecting requests
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting chardet<5,>=3.0.2
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.2-py2.py3-none-any.whl (136 kB)
Installing collected packages: urllib3, idna, chardet, certifi, requests
Successfully installed certifi-2020.12.5 chardet-4.0.0 idna-2.10 requests-2.25.1 urllib3-1.26.2

C:\Users\Slava>
```

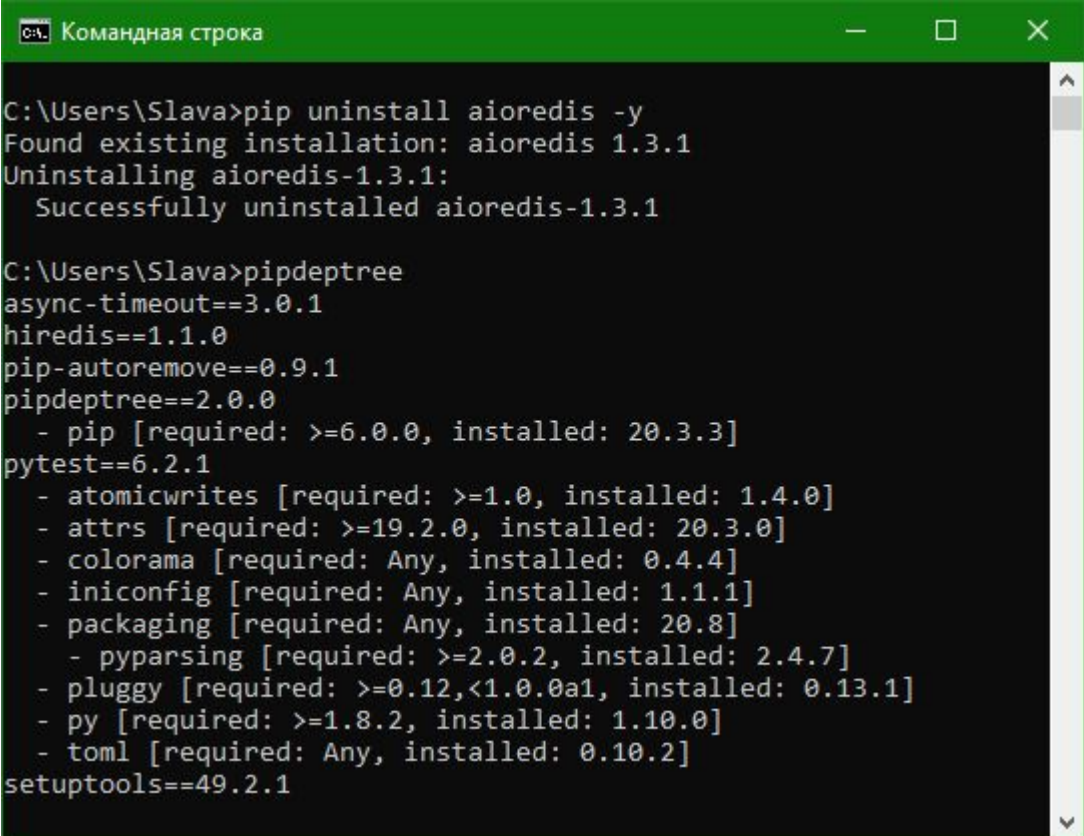


Удаление стороннего пакета

Чтобы удалить пакет, необходимо выполнить команду: `pip uninstall имя_пакета`. Установленные с пакетом зависимости **не удаляются**

Полезные опции:

- `-y, --yes` – `pip` не будет запрашивать подтверждений, автоматическое согласие на все действия
- `-r, --requirement <file>` - удалить все пакеты, которые перечислены в `requirement` файле



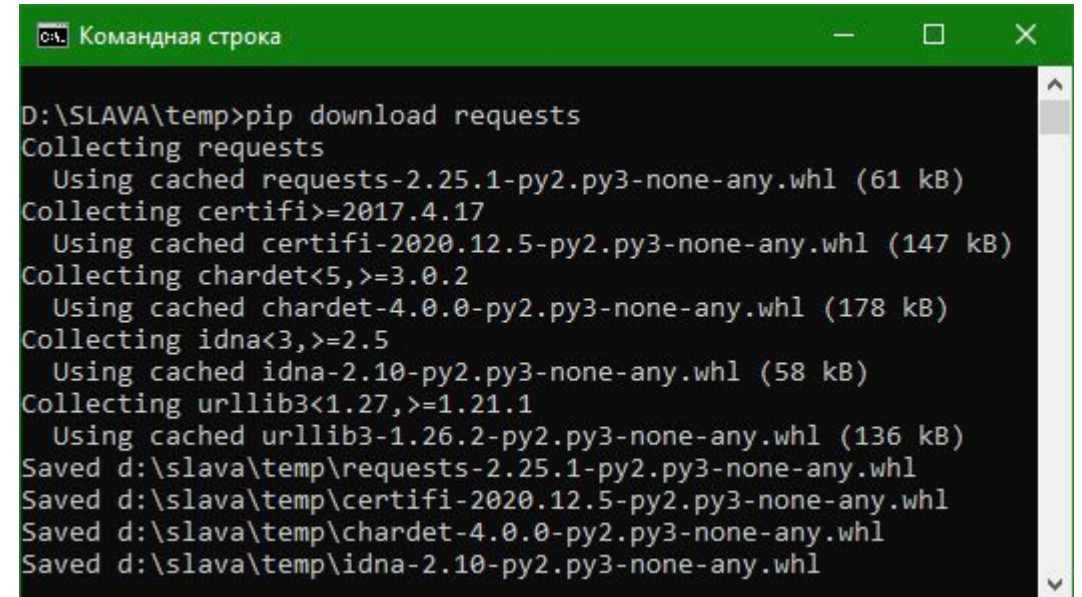
```
C:\Users\Slava>pip uninstall aioredis -y
Found existing installation: aioredis 1.3.1
Uninstalling aioredis-1.3.1:
  Successfully uninstalled aioredis-1.3.1

C:\Users\Slava>pipdeptree
async-timeout==3.0.1
hiredis==1.1.0
pip-autoremove==0.9.1
pipdeptree==2.0.0
- pip [required: >=6.0.0, installed: 20.3.3]
pytest==6.2.1
- atomicwrites [required: >=1.0, installed: 1.4.0]
- attrs [required: >=19.2.0, installed: 20.3.0]
- colorama [required: Any, installed: 0.4.4]
- iniconfig [required: Any, installed: 1.1.1]
- packaging [required: Any, installed: 20.8]
- pyparsing [required: >=2.0.2, installed: 2.4.7]
- pluggy [required: >=0.12,<1.0.0a1, installed: 0.13.1]
- py [required: >=1.8.2, installed: 1.10.0]
- toml [required: Any, installed: 0.10.2]
setuptools==49.2.1
```



Как скачать пакет

Для того, чтобы скачать пакет для дальнейшей установки там, где нет доступа к PyPI, необходимо выполнить команду: `pip download`. Также скачиваются зависимые пакеты



```
Командная строка
D:\SLAVA\temp>pip download requests
Collecting requests
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting chardet<5,>=3.0.2
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.2-py2.py3-none-any.whl (136 kB)
Saved d:\slava\temp\requests-2.25.1-py2.py3-none-any.whl
Saved d:\slava\temp\certifi-2020.12.5-py2.py3-none-any.whl
Saved d:\slava\temp\chardet-4.0.0-py2.py3-none-any.whl
Saved d:\slava\temp\idna-2.10-py2.py3-none-any.whl
```



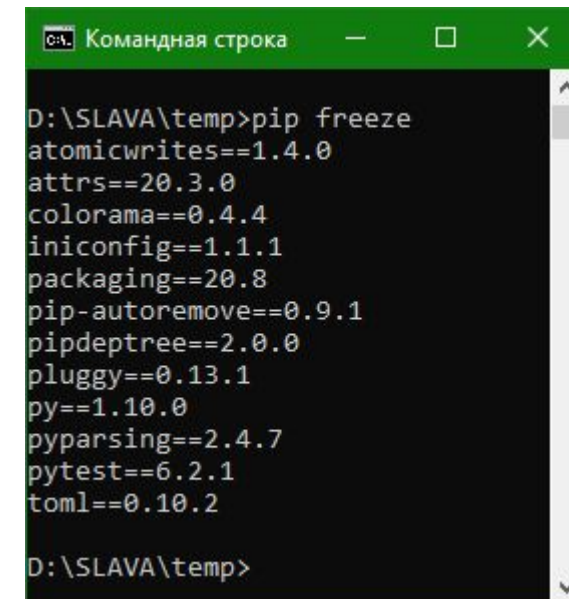
Создание файла requirements

Для просмотра списка установленных пакетов вместе с их версиями используется команда, которая делает вывод в специальном формате: `pip freeze`

Результат вывода можно скопировать в файл `requirements` в windows, либо если MacOS или Linux выполнить команду, которая создаст файл с выводом из freeze: `pip freeze > requirements.txt`

Полезные опции:

- `-l`, `--local` – в список пакетов будут включены только пакеты виртуального окружения (virtualenv)
- `--user` – в список пакетов будут включены только пакеты, установленные в пространстве текущего пользователя



```
Командная строка
D:\SLAVA\temp>pip freeze
atomicwrites==1.4.0
attrs==20.3.0
colorama==0.4.4
iniconfig==1.1.1
packaging==20.8
pip-autoremove==0.9.1
pipdeptree==2.0.0
pluggy==0.13.1
py==1.10.0
pyparsing==2.4.7
pytest==6.2.1
toml==0.10.2
D:\SLAVA\temp>
```

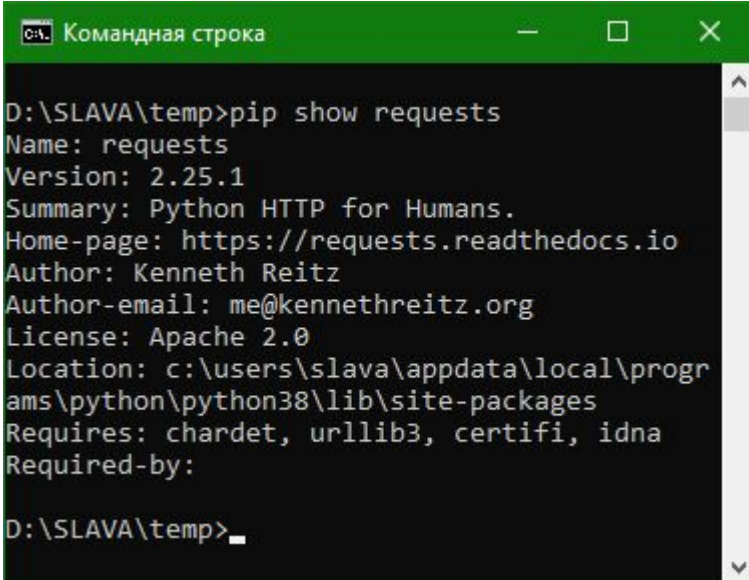


Получить информацию об установленном пакете

Для просмотра информации об установленном пакете, необходимо выполнить команду: `pip show имя_пакета`

Полезные опции:

- `-f, --files` – добавляет к основной информации полный список установленных файлов указанного пакета

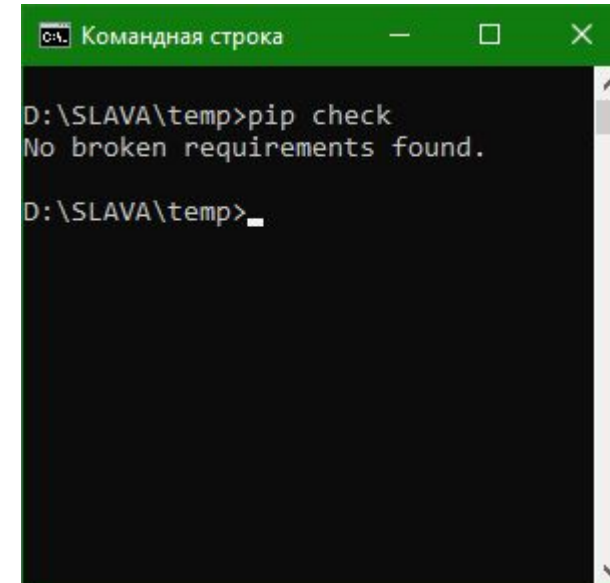


```
Командная строка
D:\SLAVA\temp>pip show requests
Name: requests
Version: 2.25.1
Summary: Python HTTP for Humans.
Home-page: https://requests.readthedocs.io
Author: Kenneth Reitz
Author-email: me@kennethreitz.org
License: Apache 2.0
Location: c:\users\slava\appdata\local\programs\python\python38\lib\site-packages
Requires: chardet, urllib3, certifi, idna
Required-by:
D:\SLAVA\temp>
```



Проверка пакетов на совместимость

Для проверки совместимости установленных пакетов, нежно выполнить команду: `pip check`



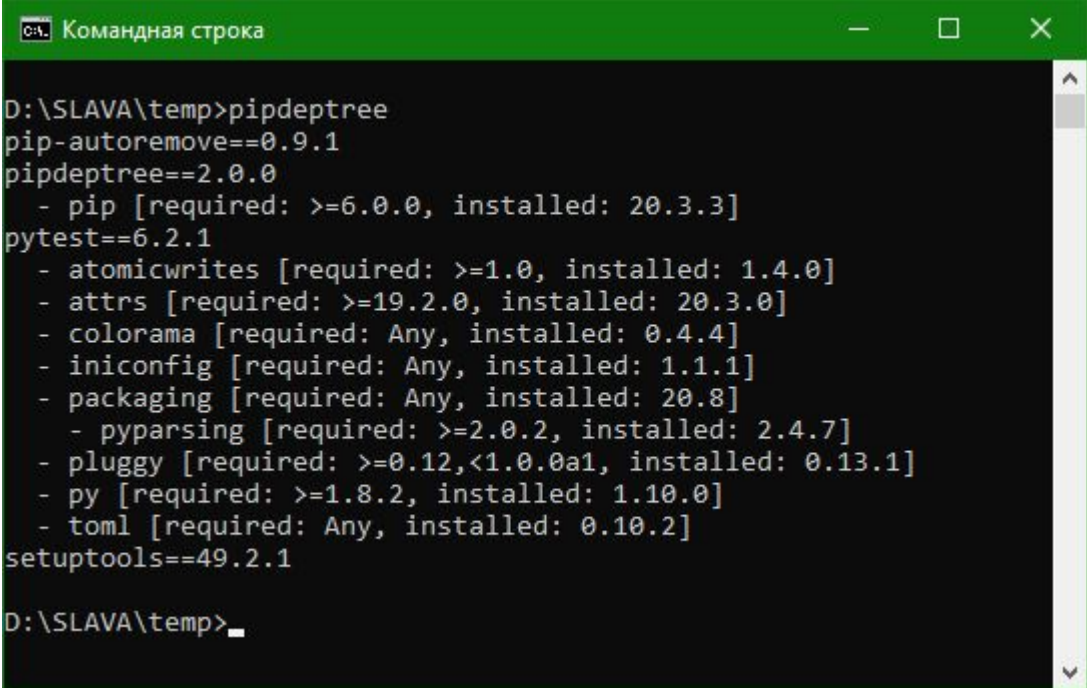
```
Командная строка
D:\SLAVA\temp>pip check
No broken requirements found.
D:\SLAVA\temp>_
```



Удобный просмотр списка пакетов

Для того, чтобы наглядно видеть какие установленные пакеты от каких зависят – можно воспользоваться пакетом `pipdeptree`:

- Установить пакет командой: `pip install pipdeptree`
- Выполнить команду: `pipdeptree`

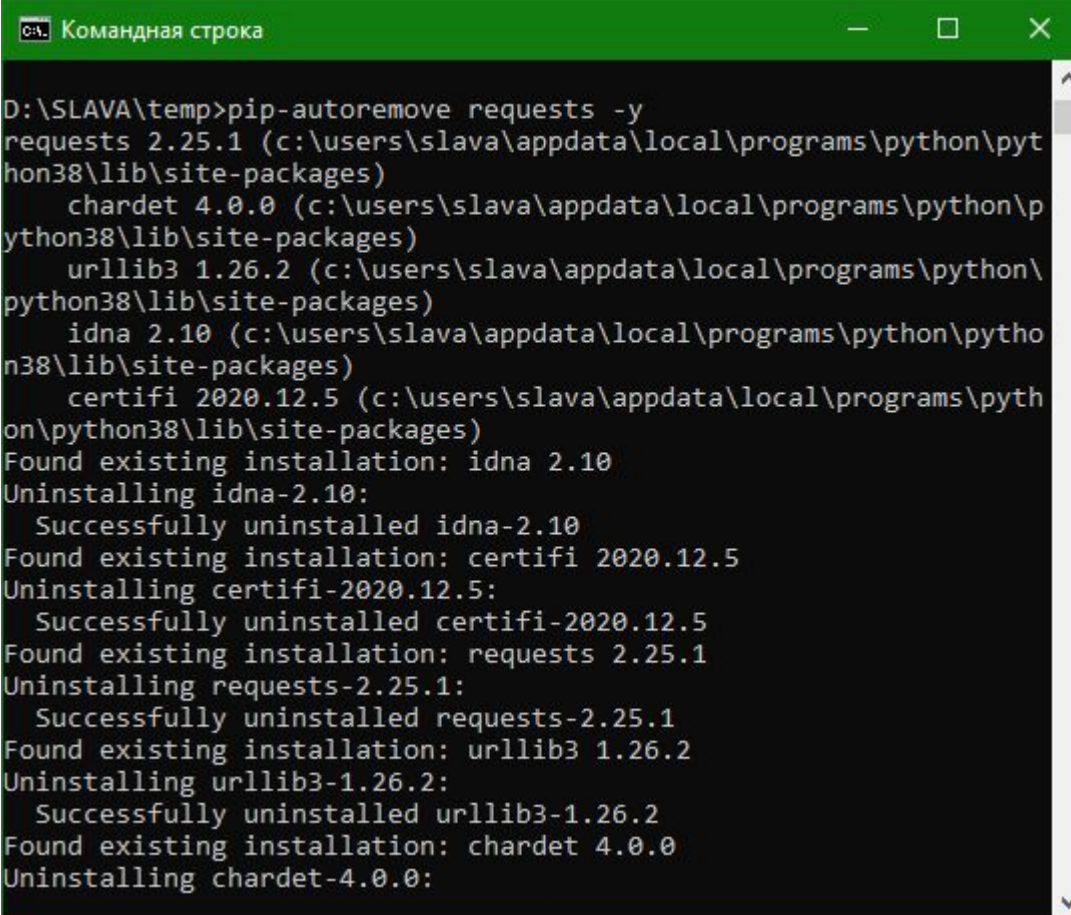


```
D:\SLAVA\temp>pipdeptree
pip-autoremove==0.9.1
pipdeptree==2.0.0
- pip [required: >=6.0.0, installed: 20.3.3]
pytest==6.2.1
- atomicwrites [required: >=1.0, installed: 1.4.0]
- attrs [required: >=19.2.0, installed: 20.3.0]
- colorama [required: Any, installed: 0.4.4]
- iniconfig [required: Any, installed: 1.1.1]
- packaging [required: Any, installed: 20.8]
- pyparsing [required: >=2.0.2, installed: 2.4.7]
- pluggy [required: >=0.12,<1.0.0a1, installed: 0.13.1]
- py [required: >=1.8.2, installed: 1.10.0]
- toml [required: Any, installed: 0.10.2]
setuptools==49.2.1

D:\SLAVA\temp>
```

Вопрос-ответ

- ? Как я могу удалить пакет со всеми его зависимостями?
- ✓ Можно воспользоваться пакетом `pip-autoremove`:
- Установить пакет: `pip install pip-autoremove`
 - Выполнить команду: `pip-autoremove имя_пакета`
 - Полезная опция: `-y`, чтобы сразу подтвердить удаление



```

D:\SLAVA\temp>pip-autoremove requests -y
requests 2.25.1 (c:\users\slava\appdata\local\programs\python\python38\lib\site-packages)
  chardet 4.0.0 (c:\users\slava\appdata\local\programs\python\python38\lib\site-packages)
  urllib3 1.26.2 (c:\users\slava\appdata\local\programs\python\python38\lib\site-packages)
  idna 2.10 (c:\users\slava\appdata\local\programs\python\python38\lib\site-packages)
  certifi 2020.12.5 (c:\users\slava\appdata\local\programs\python\python38\lib\site-packages)
Found existing installation: idna 2.10
Uninstalling idna-2.10:
  Successfully uninstalled idna-2.10
Found existing installation: certifi 2020.12.5
Uninstalling certifi-2020.12.5:
  Successfully uninstalled certifi-2020.12.5
Found existing installation: requests 2.25.1
Uninstalling requests-2.25.1:
  Successfully uninstalled requests-2.25.1
Found existing installation: urllib3 1.26.2
Uninstalling urllib3-1.26.2:
  Successfully uninstalled urllib3-1.26.2
Found existing installation: chardet 4.0.0
Uninstalling chardet-4.0.0:

```



Зачем нужны виртуальные окружения

При разработке программ на Python или использовании инструментов, созданных другими разработчиками, может возникнуть ряд проблем, связанных с использованием различных версий пакетов, тут на помощь приходят виртуальные окружения.



Основные проблемы

С какими проблемами можно столкнуться, если не использовать виртуальные окружения:

- Различные проекты могут использовать одну и ту же библиотеку, но при этом требуемые версии могут различаться
- Может возникнуть необходимость в том, чтобы запретить вносить изменения в приложение на уровне библиотек, т.е. чтобы оно работало независимо от того, обновляются у вас библиотеки или нет. Если оно будет использовать библиотеки из глобального хранилища, то со временем могут возникнуть проблемы
- У вас может просто не быть доступа к каталогу `/usr/lib/pythonXX/site-packages`



Что такое виртуальные окружения

Для решения данных вопросов используется подход, основанный на построении виртуальных окружений – своего рода песочниц, в рамках которых запускается приложение со своими библиотеками, обновление и изменение которых не затронет другие приложения, использующие те же библиотеки



Какие есть инструменты

virtualenv:

- Работает с Python 2 и Python 3. Если на проекте Python 2, то это ваш выбор
- Инструмент virtualenvwrapper – обертка на virtualenv, чтобы хранить все окружения в одном месте

venv:

- Появился в Python 3
- Встроен в интерпретатор
- Не может быть использован в Python2.
- По своему функционалу очень похож на virtualenv

pyenv:

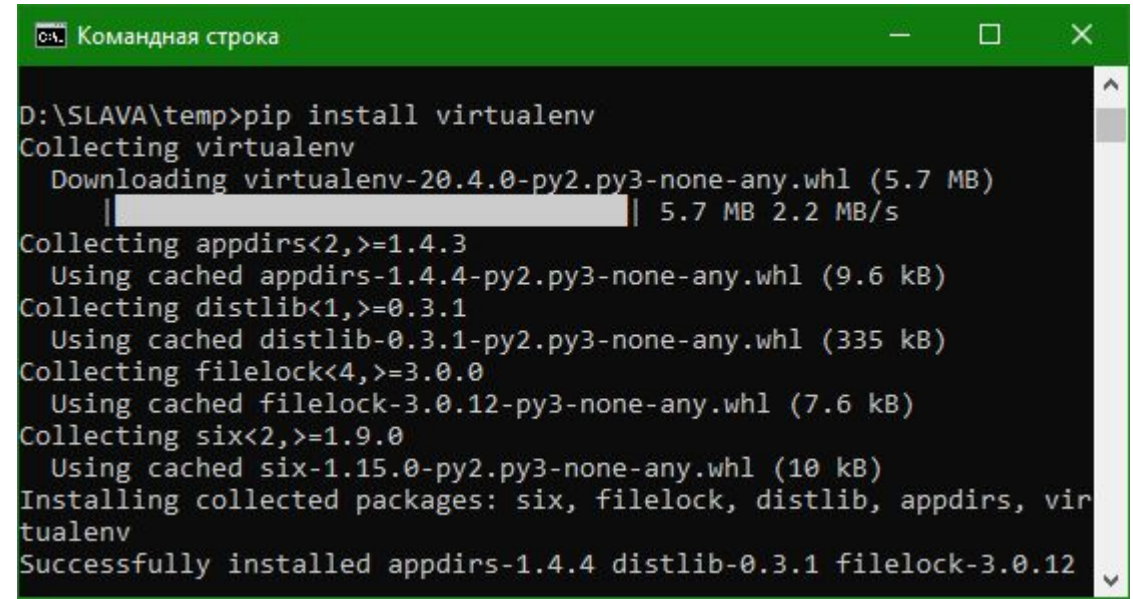
- Инструмент для изоляции версий Python
- Применяется, когда на одной машине вам нужно иметь несколько версий интерпретатора для тестирования на них разрабатываемого вами ПО



Установка virtualenv

Для установки virtualenv
используем pip:

- `pip install virtualenv`



```
Командная строка

D:\SLAVA\temp>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.4.0-py2.py3-none-any.whl (5.7 MB)
    | 5.7 MB 2.2 MB/s
Collecting appdirs<2,>=1.4.3
  Using cached appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Collecting distlib<1,>=0.3.1
  Using cached distlib-0.3.1-py2.py3-none-any.whl (335 kB)
Collecting filelock<4,>=3.0.0
  Using cached filelock-3.0.12-py3-none-any.whl (7.6 kB)
Collecting six<2,>=1.9.0
  Using cached six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: six, filelock, distlib, appdirs, virtualenv
Successfully installed appdirs-1.4.4 distlib-0.3.1 filelock-3.0.12
```

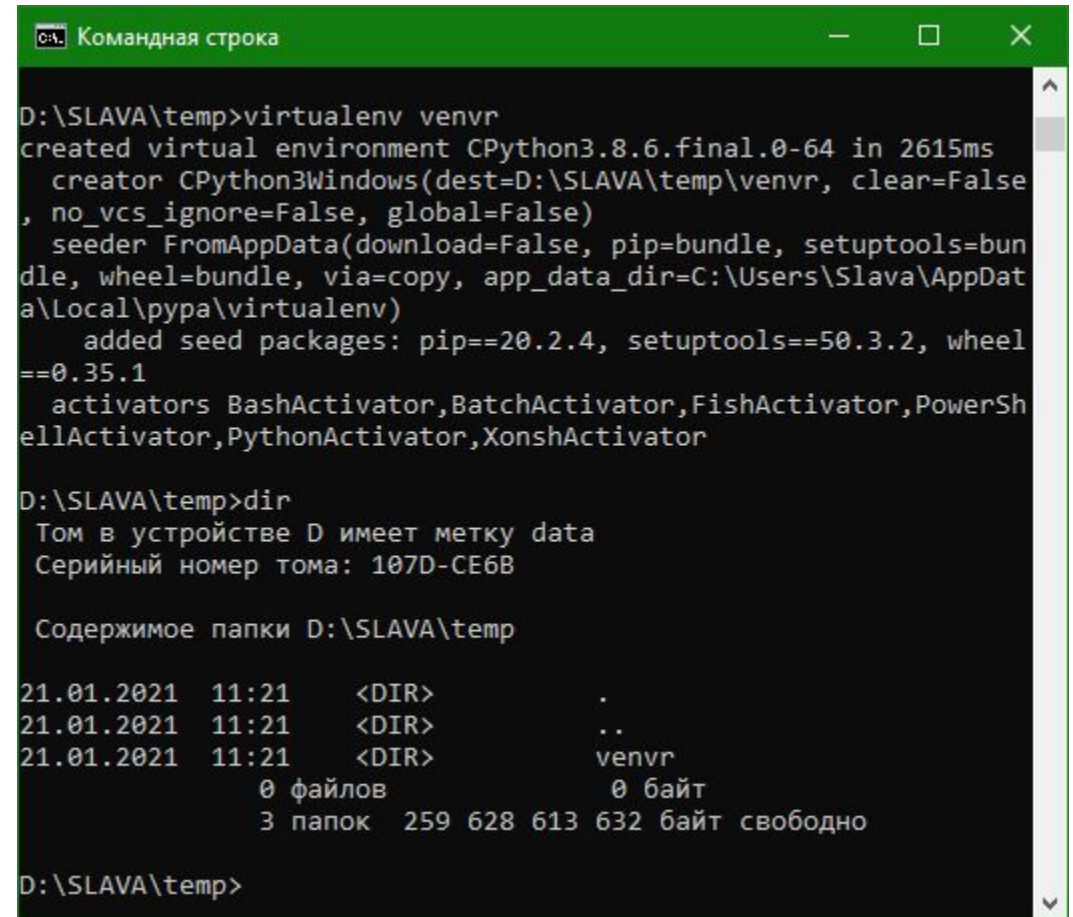


Создаем виртуальное окружение в virtualenv

Для создания виртуального окружения выполняем следующие действия:

- Переходим в директорию проекта
- Выполняем команду: `virtualenv название_окружения`

После выполнения команды в текущей директории будет создана директория с указанным названием



```
Командная строка

D:\SLAVA\temp>virtualenv venvr
created virtual environment CPython3.8.6.final.0-64 in 2615ms
  creator CPython3Windows(dest=D:\SLAVA\temp\venvr, clear=False
    , no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bun
    dle, wheel=bundle, via=copy, app_data_dir=C:\Users\Slava\AppData
    a\Local\pypa\virtualenv)
    added seed packages: pip==20.2.4, setuptools==50.3.2, wheel
    ==0.35.1
  activators BashActivator,BatchActivator,FishActivator,PowerSh
    ellActivator,PythonActivator,XonshActivator

D:\SLAVA\temp>dir
Том в устройстве D имеет метку data
Серийный номер тома: 107D-CE6B

Содержимое папки D:\SLAVA\temp

21.01.2021  11:21    <DIR>          .
21.01.2021  11:21    <DIR>          ..
21.01.2021  11:21    <DIR>          venvr
                0 файлов             0 байт
                3 папок   259 628 613 632 байт свободно

D:\SLAVA\temp>
```



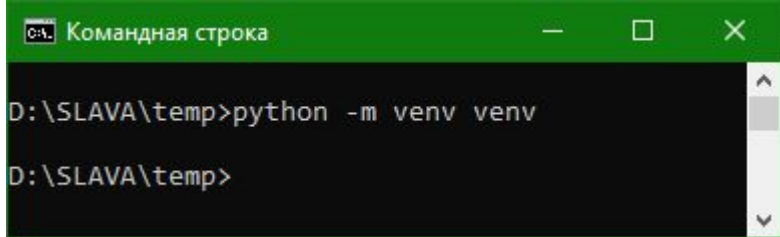
Создание окружения в venv

venv устанавливать не нужно!

Для создания окружения необходимо выполнить команду:

- `python -m venv имя_окружения`

После выполнения команды в текущей директории будет создана директория с указанным названием



```
Командная строка
D:\SLAVA\temp>python -m venv venv
D:\SLAVA\temp>
```

Что в окружении

Созданное окружение содержит в себе следующее:

- имя/Lib – пакеты окружения. Новые пакеты будут установлены в каталог site-packages
- В Windows имя/Scripts, в Linux имя/bin - скрипты для активации/деактивации окружения, интерпретатор Python, используемый в рамках данного окружения, менеджер pip и ещё несколько инструментов, обеспечивающих работу с пакетами Python

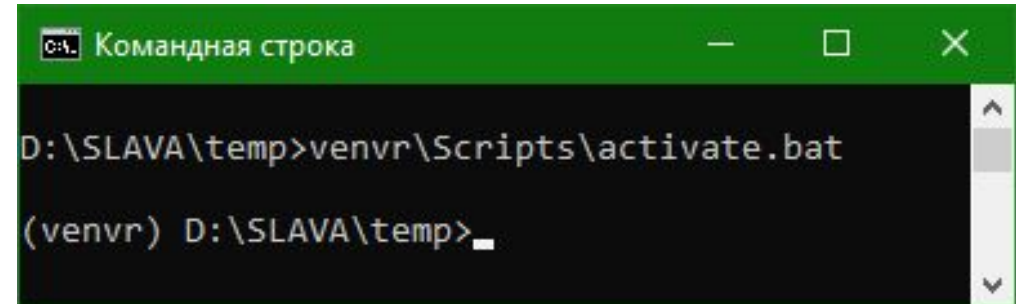


Активация виртуального окружения

Для активации:

- В Windows в командной строке нужно выполнить скрипт `имя\Scripts\activate.bat`
- В Linux выполнить команду: `source имя/bin/activate`

Когда виртуальное окружение активировано, то в строке появится его имя в скобках

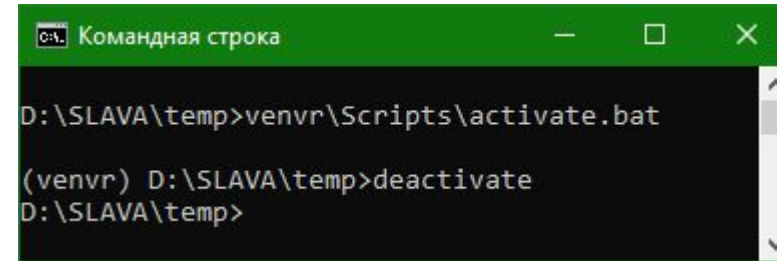


```
Командная строка
D:\SLAVA\temp>venvr\Scripts\activate.bat
(venvr) D:\SLAVA\temp>
```



Деактивация виртуального окружения

Для деактивации необходимо ввести команду: deactivate



```
C:\> Командная строка
D:\SLAVA\temp>venvr\Scripts\activate.bat
(venvr) D:\SLAVA\temp>deactivate
D:\SLAVA\temp>
```





Ваши вопросы

что необходимо прояснить в рамках
данного раздела





ОСНОВЫ Python

синтаксис, структура кода,
форматирование кода, пространства имен



Регистр

Python чувствителен к регистру

```
1 >>> myname = 'Slava'
2 >>> MyName
3 Traceback (most recent call last):
4   File "<pyshell#3>", line 1, in MyName
5   NameError: name "MyName" is not defined
```



Строки и отступы

- **Не используются скобки «{}»** для отделения блоков, вместо них двоеточия и отступы
- Количество пробелов в отступах может быть произвольным, но по договоренности равняется **4 пробелам**
- Конец строки – конец инструкции (**не требуется «;»**)

```
1  def hello_world():
2      print('hello world')
3
4      for i in range(10):
5          if i % 2 == 0:
6              print(i)
```



Кавычки

- В Python можно использовать одинарные ('), двойные (") и тройные (""" или ''') кавычки для обозначения строкового типа данных
- Для объявления многострочного текста используются **тройные** кавычки

```
1 single_quotes = 'some text'
2
3 double_quotes = "some text"
4
5 multiline_text = """
6 Hello, my friend!
7
8 I want to talk with you...
9 """
```



Комментарии

Символом решетки (#) обозначаются комментарии в Python. Любые символы после решетки игнорируются интерпретатором

```
1  # flag for bubble sort
2  flag = True
```



Структура кода Python

1. Docstring к модулю
2. Импорты из стандартной библиотеки
3. Импорты из сторонних библиотек
4. Импорты из текущей библиотеки
5. Классы, функции и т.д.



Форматирование кода Python

- У Python имеется жесткий стандарт по оформлению кода, ему надо следовать **обязательно**
- PEP 8 – style guide по оформлению кода на Python (стандарт оформления кода):
 - <https://www.python.org/dev/peps/pep-0008/>
 - <https://pep8.ru/doc/pep8/>



Вопрос-ответ

- ? Должен(а) ли я следовать PEP 8?
- ✓ Однозначно ДА! Эти рекомендации позволяют писать читаемый код. Программисты на Python следуют данному стандарту, следовательно вам будет намного проще читать чужой код





Ваши вопросы

что необходимо прояснить в рамках
данного раздела



Полезные ссылки

- Python и его документация: <https://www.python.org/>
- Roadmap для backend разработчика: <https://roadmap.sh/backend>
- IDE PyCharm: <https://www.jetbrains.com/ru-ru/pycharm/>
- Аналог pip. Poetry: <https://python-poetry.org/>
- PyPi (пакеты Python): <https://pypi.org/>
- Задаем вопросы:
 - <https://ru.stackoverflow.com/> (на русском языке)
 - <https://stackoverflow.com/> (на английском языке)
 - <https://qna.habr.com/> (на русском языке)
- Дополнительное обучение: <https://www.coursera.org/>
- популярность языков программирования:
 - Индекс TIOBE: <https://www.tiobe.com/tiobe-index//>
 - Stackoverflow: <https://insights.stackoverflow.com/survey/2019#technology>





Домашнее задание

что необходимо будет
выполнить дома

Домашнее задание

1. Работа с виртуальными окружениями и pip:
 - Создать папку на диске
 - В этой папке сделать виртуальное окружение
 - Активировать виртуальное окружение
 - С помощью pip установить пакеты requests, pipdeptree, pip-autoremove
 - Просмотреть результат выполнения pipdeptree
 - Удалить пакет request с помощью pip-autoremove
 - сохранить и скинуть скрины выполнения
2. Прочитать PEP8 (желательно несколько раз)

