

# Проверить, идет ли запись!





## Преподаватель



### Андрей Гридяев

- Опыт в автоматизации 5 лет
- Языки: Python, Java
- Фреймворки: pytest, selenide

# Правила вебинара



Активно участвуем



Задаем вопрос в чат



Off-topic обсуждаем в Slack в канале #general



Вопросы вижу в чате, могу ответить не сразу

# Маршрут вебинара



### **Цели вебинара** | После занятия вы сможете

Собирать и публиковать Python packages

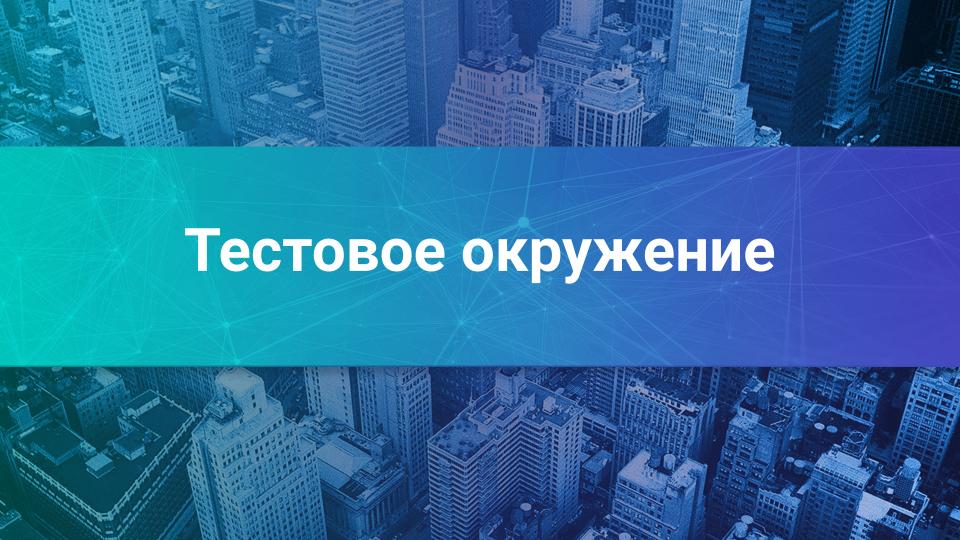
Oбирать и устанавливать deb packages

Эффективно распространять свой код

## Смысл | Зачем вам это уметь

Настраивать тестовое окружение

Решать ряд проблем с Linux-окружением



## Тестовое окружение

**Тестовое окружение** - аппаратное и программное обеспечение и другие средства, необходимые для выполнения тестов

- Docker container
- Виртуальная машина
- Физический сервер
- Комбинации

## Исходный код тестов

#### Где хранить исходный код тестов?

- Вместе с продуктом
- Отдельно от продукта
- Тесты разных уровней (приемочные, интеграционные) вместе/отдельно от продукта?

Какой язык использовать для разработки тестов?

- Такой же как язык разработки
- Более удобный в зависимости от инструментария

# Где запускать тесты?

#### Где запускать тесты?

- Локально
- Билд-агенты
- Изолированные окружения (контейнеры, виртуальные машины)



# PyPI и pip

**PyPI** (**Py**thon **P**ackage Index) - репозиторий, содержащий пакеты, которые разрабатываются Python community

**рір** - пакетный менеджер для Python, который позволяет устанавливать пакеты, которые не входят в стандартную библиотеку Python

#### **Установка**

```
$ python3 -m pip install requests
$ python3 -m pip install requests==2.24.0
$ python3 -m pip install -i https://custom-pypi.org/simple my_package
$ python3 -m pip install
git+https://github.com/psf/requests.git#egg=requests
```

### **Distribution**

Source distribution - формат распространения пакетов, включающий метаданные и исходные файлы, которые используются при установке пакета с помощью рір или при генерации built distribution

**Built distribution** - формат распространения пакетов, включающий метаданные и файлы, которые необходимо поместить в определенное место в системе для корректной установки пакета

### Source distribution

source distribution - архив с исходными файлами, который используется для самостоятельной установки с помощью **pip** либо для сборки **wheel** 

Представляет из себя **tar.gz**-архив

```
$ tar -tf six-1.15.0.tar.gz
```

#### Сборка

\$ python3 setup.py sdist

#### **Установка**

\$ python3 -m pip install six-1.15.0.tar.gz

### Wheels

wheels - формат пакета, созданный для распространения модулей и скомпилированных расширений

Представляет из себя **zip**-архив

\$ unzip -l six-1.15.0-py2.py3-none-any.whl

### Сборка

\$ python3 setup.py bdist\_wheel

#### **Установка**

\$ python3 -m pip install six-1.15.0-py2.py3-none-any.whl

### Именование wheels

#### six-1.15.0-py2.py3-none-any.whl

- **six** имя пакета
- **1.15.0** версия пакета
- py2.py3 Python tag, поддержка Python 2 и Python 3
- none ABI tag
- **any** поддерживаются любые платформы

pure-Python wheel - Jinja2-3.0.0a1-py3-none-any.whl universal wheel - chardet-3.0.4-py2.py3-none-any.whl platform wheel - numpy-1.18.4-cp38-cp38-win32.whl

### Семантическое версионирование

### 1.15.0

- 1 "Major" номер версии. Увеличивается, когда API библиотеки или приложения меняется обратно несовместимым образом
- **15** "**Minor**" номер версии. Увеличивается, когда в API добавляются новые функции без нарушения обратной совместимости. При увеличении "**Major**" принято сбрасывать "**Minor**" в 0
- **0** "**Patch**" номер версии. Увеличивается при исправлении багов, рефакторинге и прочих изменениях, которые ничего не ломают, но и новых фичей не добавляют. При увеличении "**Major**" или "**Minor**" принято сбрасывать "**Patch**" в 0

### wheels vs source distributions

#### wheels

- Устанавливаются быстрее, чем source distributions
- Имеют меньший размер, чем source distributions (six)
   https://pypi.org/project/six/#files
- При установке wheels не запускается setup.py
- Для установки расширений не требуется компилятор

#### source distributions

- Можно воспользоваться если соответствующий wheel "сломан"
- Нужно добавить небольшой патч и затем установить пакет

# Публикация Python package

#### Сборка пакета

```
setuptools, wheel - сборка пакета (<u>setup.py</u>)
```

\$ python3 setup.py sdist bdist\_wheel

### Проверка wheel (<u>check-wheel-contents</u>)

\$ check-wheel-contents dist/agrpkg-0.0.4-py3-none-any.whl

#### Загрузка пакета

twine - загрузка пакета на PyPI

\$ python3 -m twine upload --repository testpypi dist/\*



### Пакет

**Пакет** - архивный файл, содержащий исполняемый двоичный файл, файл конфигурации и иногда информацию о зависимостях

Представляет из себя ar-архив

```
$ ar -t python3-six 1.15.0-2 all.deb
```

- debian-binary текстовый файл, содержащий версию формата debпакета
- control.tar tar-архив, содержащий информацию и скрипты установки пакета
- data.tar tar-архив, содержащий дерево устанавливаемых файлов пакета

```
$ dpkg-deb -c python3-six_1.15.0-2_all.deb
```

## Менеджер пакетов

**Менеджер пакетов** позволяет устанавливать, удалять или обновлять любой пакет или группы пакетов, выполняя автоматически все необходимые для этого процедуры (доставку пакетов из удаленных репозиториев, вычисление зависимостей и установку требуемых по ним пакетов, удаление замещаемых пакетов и т. п.)

- Yum и RPM Red Hat (Fedora, RHEL, openSUSE)
- dpkg Debian (Ubuntu)
- pacman Arch Linux
- brew OSX

## Сборка deb пакетов

**fpm** - инструмент, который позволяет собирать пакеты для различных систем

- \$ sudo apt-get update
- \$ sudo apt-get install ruby-dev build-essential
- \$ sudo gem install fpm

### Сборка python-пакета

\$ fpm -s python -t deb setup.py

## Добавление в РАТН

Не всегда нужно устанавливать пакеты, иногда достаточно поместить исполняемый файл в директорий, который указан в **РАТН** 

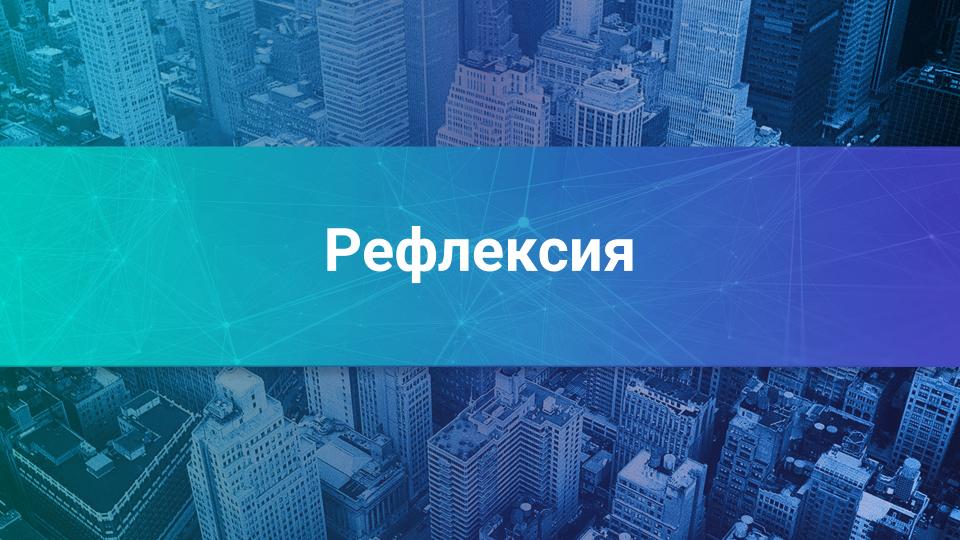
```
$ echo $PATH

$ sudo curl -L
https://github.com/docker/compose/releases/download/1.27.4/docker-compose
-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

$ 11 /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose

$ docker-compose --version
```



### **Цели вебинара** | После занятия вы сможете

Cобирать и публиковать Python packages

Oбирать и устанавливать deb packages

Эффективно распространять свой код

# Материалы

### Документация

- Python Packaging User Guide
- Writing the Setup Script
- setuptools
- <u>distutils Building and installing Python modules</u>
- What Are Python Wheels and Why Should You Care?

### Видео

The Black Magic of Python Wheels



