

PYTHON 2 (2022)

Ннну, давайте знакомиться!



Ннну, давайте знакомиться!



Поехали!

DISCLAIMER: ПРОБЛЕМА ПОСТРОЕНИЯ КУРСА

DISCLAIMER: ПРОБЛЕМА ПОСТРОЕНИЯ КУРСА

Это первая итерация элективного курса

Выше уровень слушателей:

Выше уровень слушателей:

- сложнее дать что-то новое и интересное

Выше уровень слушателей:

- сложнее дать что-то новое и интересное
- выше дисперсия покрытия знаний

Выше уровень слушателей:

- сложнее дать что-то новое и интересное
- выше дисперсия покрытия знаний
- локальный “оптимум” – давать узкоспециализированные и часто плохо тиражируемые “приколюхи”

Если давать приколюхи:

Если давать приколюхи:

- сильные студенты останутся сильными

Если давать приколюхи:

- сильные студенты останутся сильными
- слабые – слабыми

Решение: ни вашим, ни нашим

СДЕЛАТЬ НЕ ИНТЕРЕСНО



И НЕ ДАТЬ УРОВЕНЬ

Что мы с этим делаем

- **Будем повторять**, с разных сторон и разной глубиной
- Надеемся на сознательность и самостоятельную работу
- Сегодня посемлируем из разных частей

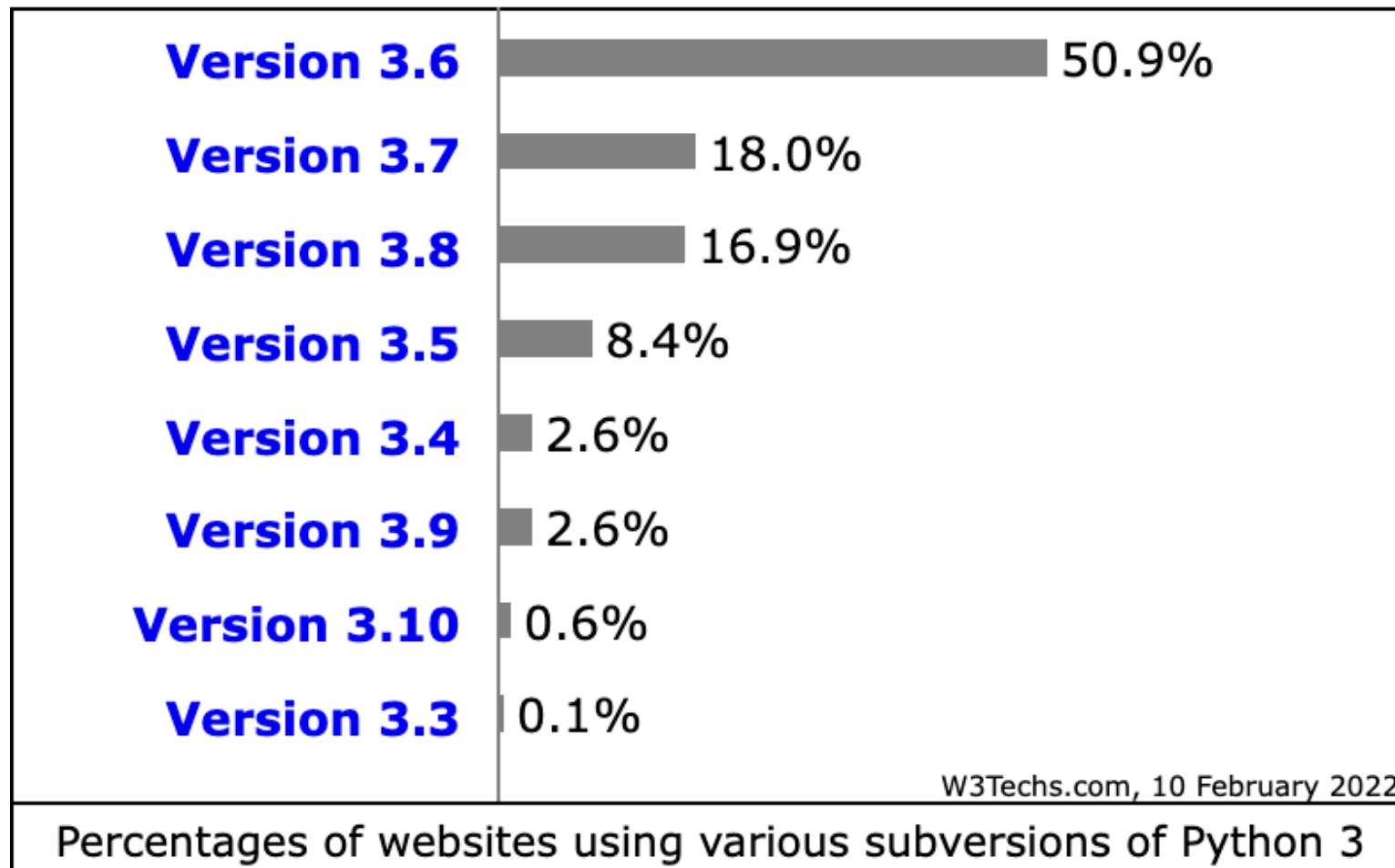
Disclaimer 2: уникальность контента

Бессовестный плагиат

Рекомендации по самостоятельной работе

- Исследовать ресурсы, где написано “почему”, а не только лишь “как”
 - Официальная документация Python (!)
 - **Google Python Styleguide** – пожалуйста, ознакомьтесь с ним
 - PEPs
- Книги:
 - **Fluent Python**

Ремарка про версии Python



ЭКОСИСТЕМА И ТУЛИНГ

ЭКОСИСТЕМА И ТУЛИНГ

- IDE

ЭКОСИСТЕМА И ТУЛИНГ

- IDE
- Форматтеры

ЭКОСИСТЕМА И ТУЛИНГ

- IDE
- Форматтеры
- Линтеры

ЭКОСИСТЕМА И ТУЛИНГ

- IDE
- Форматтеры
- Линтеры
- Обвязка

ЭКОСИСТЕМА И ТУЛИНГ

- IDE
- Форматтеры
- Линтеры
- Обвязка
- Менеджмент пакетов и питонов OMG

IDE

- VsCode
- PyCharm

VsCode

VsCode

- легковесный

VsCode

- легковесный
- шикарно работает по ssh, имея на компьютере
только тонкого клиента

VsCode

- легковесный
- шикарно работает по ssh, имея на компьютере только тонкого клиента
- Мощная поддержка Python с помощью Python Language Server

PyCharm

PyCharm

- потяжелее, но умеет вообще все (в платной версии), лучше продуман

PyCharm

- потяжелее, но умеет вообще все (в платной версии), лучше продуман
- разработка через ssh не такая шикарная и только платно

Vim / Neovim c LSP

Vim / Neovim с LSP

- есть на каждой linux машине это не совсем правда

Vim / Neovim с LSP

- есть на каждой linux машине это не совсем правда
- разработка по ssh

Vim / Neovim с LSP

- есть на каждой linux машине это не совсем правда
- разработка по ssh
- универсальный инструмент редактирования
чего угодно

Vim / Neovim с LSP

- есть на каждой linux машине это не совсем правда
- разработка по ssh
- универсальный инструмент редактирования
чего угодно
- недоЙДЕ это не совсем правда

ФОРМАТТЕРЫ

Их несколько популярных (autorep8, yapf, black, isort), мы с вами будем использовать `black`

ФОРМАТТЕРЫ

Их несколько популярных (autoper8, yapf, black, isort), мы с вами будем использовать `black`
хорош тем, что практически не конфигурируется

ЛИНТЕРЫ



ЛИНТЕРЫ



С ними больно, они постоянно ошибаются, но на средне-крупных проектах без них хуже

ЛИНТЕРЫ



С ними больно, они постоянно ошибаются, но на средне-крупных проектах без них хуже

Обычно один раз надо все настроить, а потом тиражировать по репозиториям

Мы с вами будем использовать flake8 и тур

- flake + plugins – будем использовать
- тур – будем использовать

В средне-больших проектах линтеров никогда
не бывает много

Обвязка

- Тестирование – unittest, pytest
- Сборка пакета – [setup.py](#), pbr, poetry
- CI/CD – github action, gitlab ci

**ВИРТУАЛЬНЫЕ
ОКРУЖЕНИЯ,
ПАКЕТНЫЕ
МЕНЕДЖЕРЫ**

Что такое пакет?

Что такое пакет?

- грубо говоря – распакованный архив с файликами

Что такое пакет?

- грубо говоря – распакованный архив с файликами
- подсистема импортов, поиска пакетов и модулей основана **на путях**

Что такое пакет?

- грубо говоря – распакованный архив с файликами
- подсистема импортов, поиска пакетов и модулей основана **на путях**
- если пакетов с нужными именем несколько, интерпретатор детерминированно выбирает первый

Что такое пакет?

- грубо говоря – распакованный архив с файликами
- подсистема импортов, поиска пакетов и модулей основана **на путях**
- если пакетов с нужными именем несколько, интерпретатор детерминированно выбирает первый
- про это мы еще поговорим

Нельзя просто так взять и заимпортировать
пакет какой-то конкретной версии



Нельзя просто так взять и заимпортировать
пакет какой-то конкретной версии



не зашив версию в имя пакета

Нельзя просто так взять и заимпортировать
пакет какой-то конкретной версии



не зашив версию в имя пакета

передаем привет Golang-у, в котором это стандартный способ

VENV

самый простой способ получить чистое изолированное окружение

```
$ python3 -m venv <path_to_desired_env> # *запустить* модуль v  
$ source <path_to_desired_env>/bin/activate
```

VENV

самый простой способ получить чистое изолированное окружение

```
$ python3 -m venv <path_to_desired_env> # *запустить* модуль v  
$ source <path_to_desired_env>/bin/activate
```

activate добавляет <path_to_desired_env>/bin в переменную окружения PATH

```
$ which -a python
<path_to_desired_env>/bin/python
...
<system_python_binary>
```

```
$ deactivate  
$ which -a python # path_to_desired_env исчез!  
...  
<system_python_binary>
```

Q: почему надо делать `source .../activate`,
а не запускать его `bash .../activate`?

Q: ЧТО ЗНАЧИТ “ЗАПУСТИТЬ” МОДУЛЬ?

```
python -m package.module arg1 arg2
```

VENV: PROS & CONS

VENV: PROS & CONS

- всегда есть, входит в стандартную библиотеку (+)

VENV: PROS & CONS

- всегда есть, входит в стандартную библиотеку (+)
- максимально прямолинейный (+)

VENV: PROS & CONS

- всегда есть, входит в стандартную библиотеку (+)
- максимально прямолинейный (+)
- не мусорит в файловой системе (+)

VENV: PROS & CONS

- всегда есть, входит в стандартную библиотеку (+)
- максимально прямолинейный (+)
- не мусорит в файловой системе (+)
- засоряет текущие переменные окружения, может взорваться

VENV: PROS & CONS

- всегда есть, входит в стандартную библиотеку (+)
- максимально прямолинейный (+)
- не мусорит в файловой системе (+)
- засоряет текущие переменные окружения, может взорваться
- не является менеджером пакетов**, занимается только изоляцией (+, -)

Q: почему нельзя ставить пакеты в системный питон? А в каких случаях можно?

CONDA

- не только python!
- ставит библиотеки, бинарники
- локализован в файловой системе (легко удалить / почистить)
 - как это работает?
- супер удобен, когда ничего нельзя и нет sudo / docker

Это до какой-то степени работает, но никогда
не делайте так!

```
$ pip install conda  
... use conda
```

Если shell integration не установлены (вам их предложат поставить при установке mini-anaconda):

```
~/miniconda3/bin/conda create -n py38 -y python=3.8 nomkl scip  
source ~/miniconda3/bin/activate py38  
which python
```

Если shell integration установлены:

```
conda create -n py38 -y python=3.8 nomkl scipy numpy scikit-learn  
conda activate py38  
which python
```

Miniconda, Anaconda, conda

- conda - пакетный менеджер
- anaconda и miniconda - дистрибутивы, использующие conda в качестве пакетного менеджера

Miniconda

- по сути, инсталляция, в которой уже есть conda, Python и минимальное кол-во пакетов, которые использует сама conda
- используйте ее, когда сетеапите на сервере / просто хотите менеджер вирт. окружений и пакетов

Anaconda

- тащит кучу всего (250+ пакетов), чтобы сразу же запустить условный питон и начать работать
- тяэелая
- не рекомендуется для работы на серверах

НЮАНСЫ

- агрессивное обновление библиотек, проблематично сделать фриз
 - нужно постараться, чтобы окружение было именно то, что нужно
- может обновить часть пакетов, часть не обновить и отрапортовать с 0 exit code
- может зафейлить обновление и зафейлить откат, превратив ваш venv в тыкву

для прода не очень подходит, но в связке с conda-pack / docker / etc можно добиться

предсказуемого поведения