академия больших данных

mail.ru group

Production Python code

Михаил Марюфич, MLE

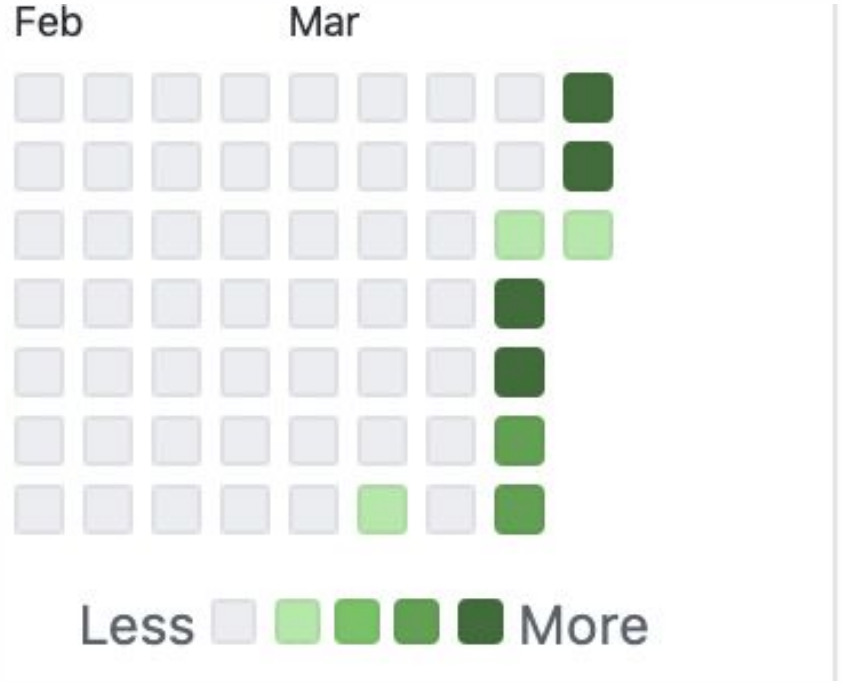

Q&A

Результаты

131	06.04.2021 8:27:20	**Клестов Андрей Владимирович**
132	06.04.2021 10:19:10	**Латыпов Ильяс Дамирович**
133	06.04.2021 11:32:17	**Махмутова Лилия Равильевна**
134	06.04.2021 12:44:59	**Иванкин Михаил**
135	06.04.2021 13:22:36	**Ермаков Владимир Николаевич**
136	06.04.2021 13:44:16	Горностаев Дмитрий Александрович
137	06.04.2021 14:27:34	Шевчук Алексей
138		
139	06.04.2021 15:45:10	Мурзина Александра Ивановна
140	06.04.2021 17:04:08	Солиев Музаффар Анваржон угли
141	06.04.2021 18:07:22	Юнусов Егор Викторович
142	06.04.2021 18:16:17	Опалёв Роман Сергеевич
143	06.04.2021 18:47:13	Войтешонок Вячеслав Сергеевич
144		

Результаты

Reviewed 76 pull requests in 75 repositories



Ошибки

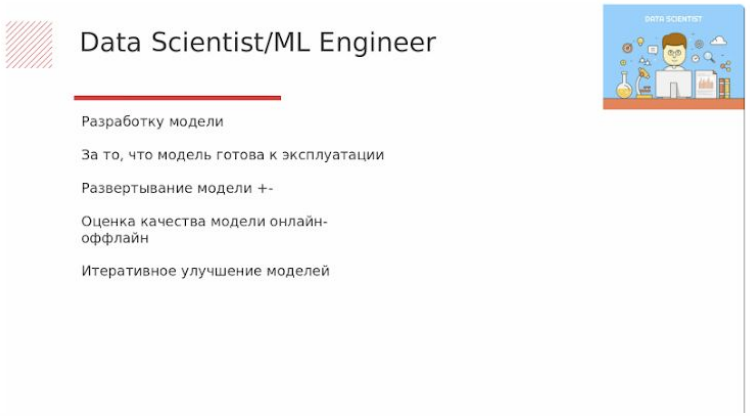
1) Создать репозиторий не в зоне https://github.com/made-ml-in-prod-2021
2) Замерджить без approve
3) Написать в readme что-то вроде ololo kek
4) Добавиться в форму и сразу после этого написать в чатик

Вопрос

Добрый день! Хотел задать вам небольшой вопрос по первой лекции.

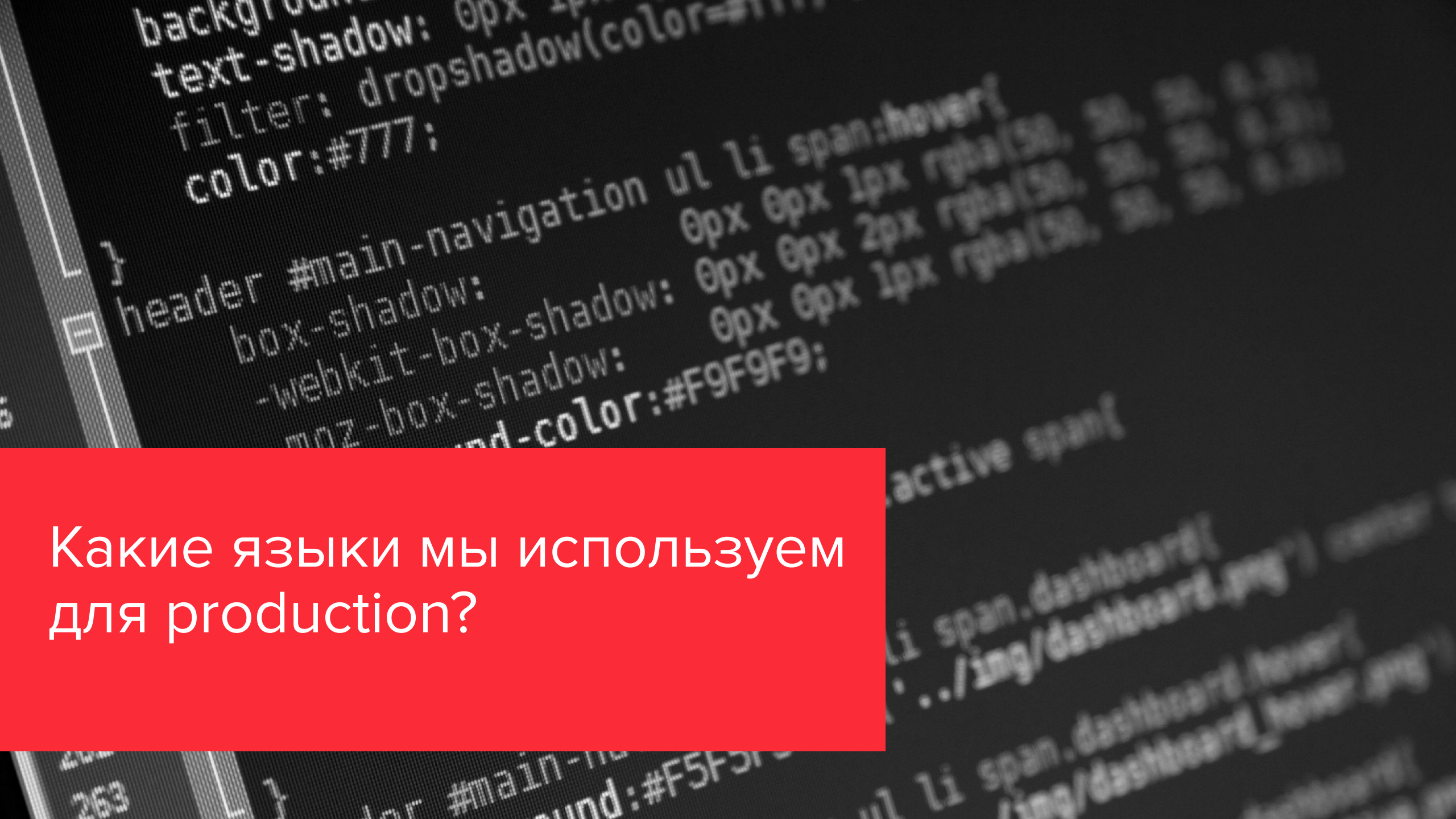
Не совсем понял, чем отличаются ML-инженеры от DS (на ваших слайдах они были упомянуты через /, создалось впечатление, что они являются синонимами в некотором роде).

Я всегда представлял, что ML-инженер -- это Software-инженер, специализирующийся на разработке ML-сервисов или ML-подмодулей программ. Т.е он проводит весь цикл подготовки данных, экспериментов с моделями, но затем реализует это в виде продакшен кода, а не ограничивается добавлением модельки в условный Model Registry, о котором вы рассказывали. Так ли это?

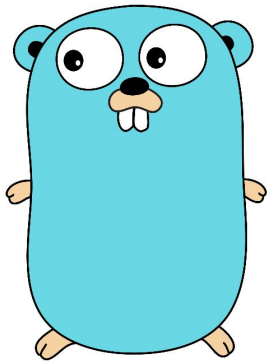


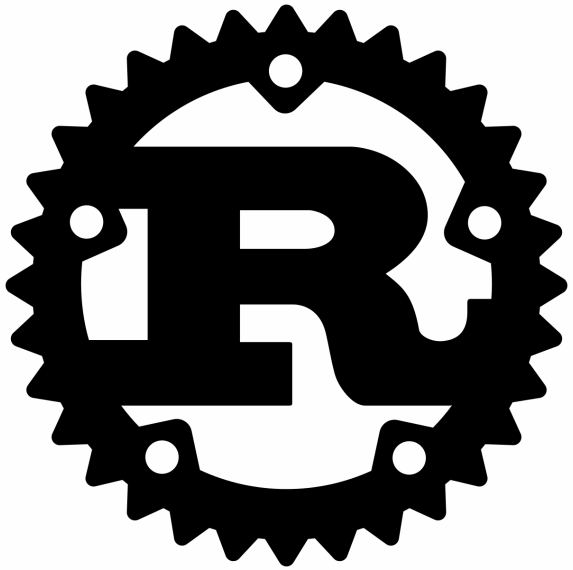
Содержание занятия

- Почему, как и зачем DS используют python?
- Советы по читаемому коду на python
- Организация и дистрибьюция кода на python
- Виртуальные окружения и менеджеры зависимостей
- Демо проект с регрессией


Какие языки мы используем для production?

Языки для production





Python

- Динамическая типизация
- Интерпретируемый

Python

- Динамическая типизация
- Интерпретируемый
- Огромное количество готовых пакетов
- В ML стандарт de facto

Jupyter

Jupyter Notebook

jupyter BOT CLASSIFICATION MODEL -- BEST ATTEMPT Last Checkpoint: 8 минут назад (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

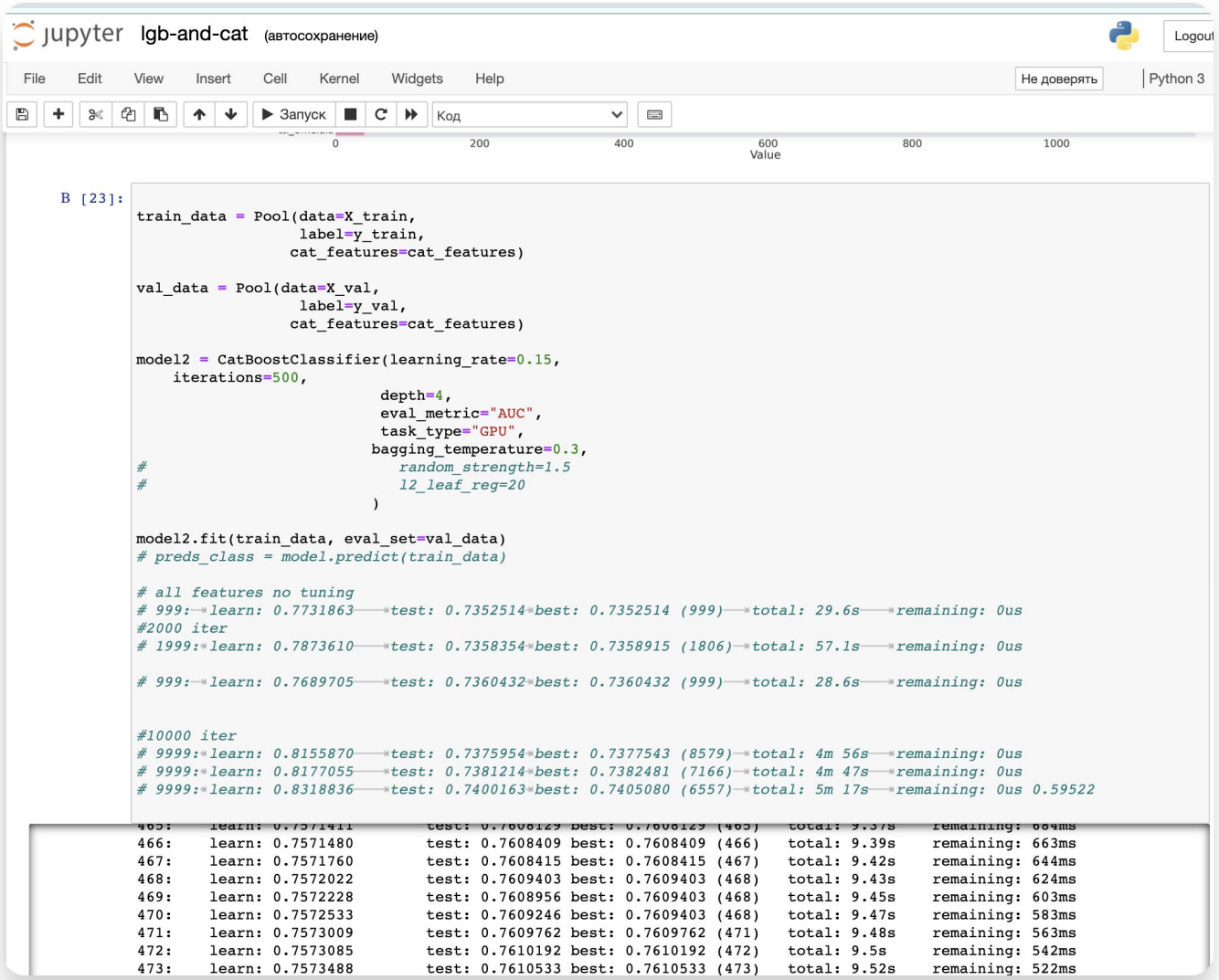
In [1]:
```
import data_processing
```

In [2]:
```
import pandas as pd
```

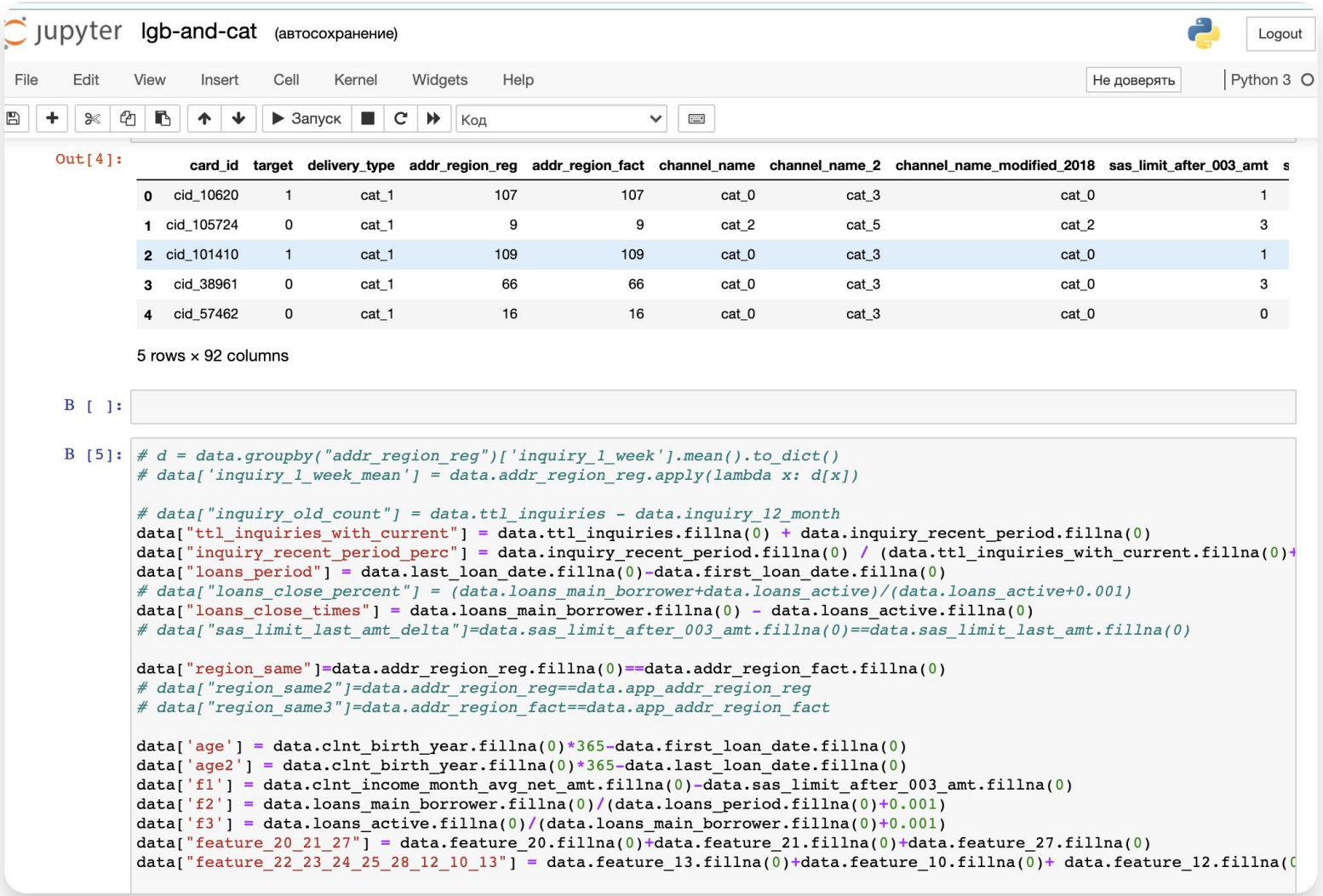
In [3]:
```
data = pd.read_csv("data/behaviour_public_2018-2019_filtered_raw.csv", "\t")
features = data_processing.generate_features(data)
target = data_processing.get_target(data)
model = BotClassificationPipeline()
model.fit(features, target)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-c940d1691b65> in <module>
      2 features = data_processing.generate_features(data)
      3 target = data_processing.get_target(data)
----> 4 model = BotClassificationPipeline()
      5 model.fit(features, target)

NameError: name 'BotClassificationPipeline' is not defined
```

Jupyter Notebook



Jupyter Notebook



Out[4]:

	card_id	target	delivery_type	addr_region_reg	addr_region_fact	channel_name	channel_name_2	channel_name_modified_2018	sas_limit_after_003_amt	s
0	cid_10620	1	cat_1	107	107	cat_0	cat_3	cat_0	1	
1	cid_105724	0	cat_1	9	9	cat_2	cat_5	cat_2	3	
2	cid_101410	1	cat_1	109	109	cat_0	cat_3	cat_0	1	
3	cid_38961	0	cat_1	66	66	cat_0	cat_3	cat_0	3	
4	cid_57462	0	cat_1	16	16	cat_0	cat_3	cat_0	0	

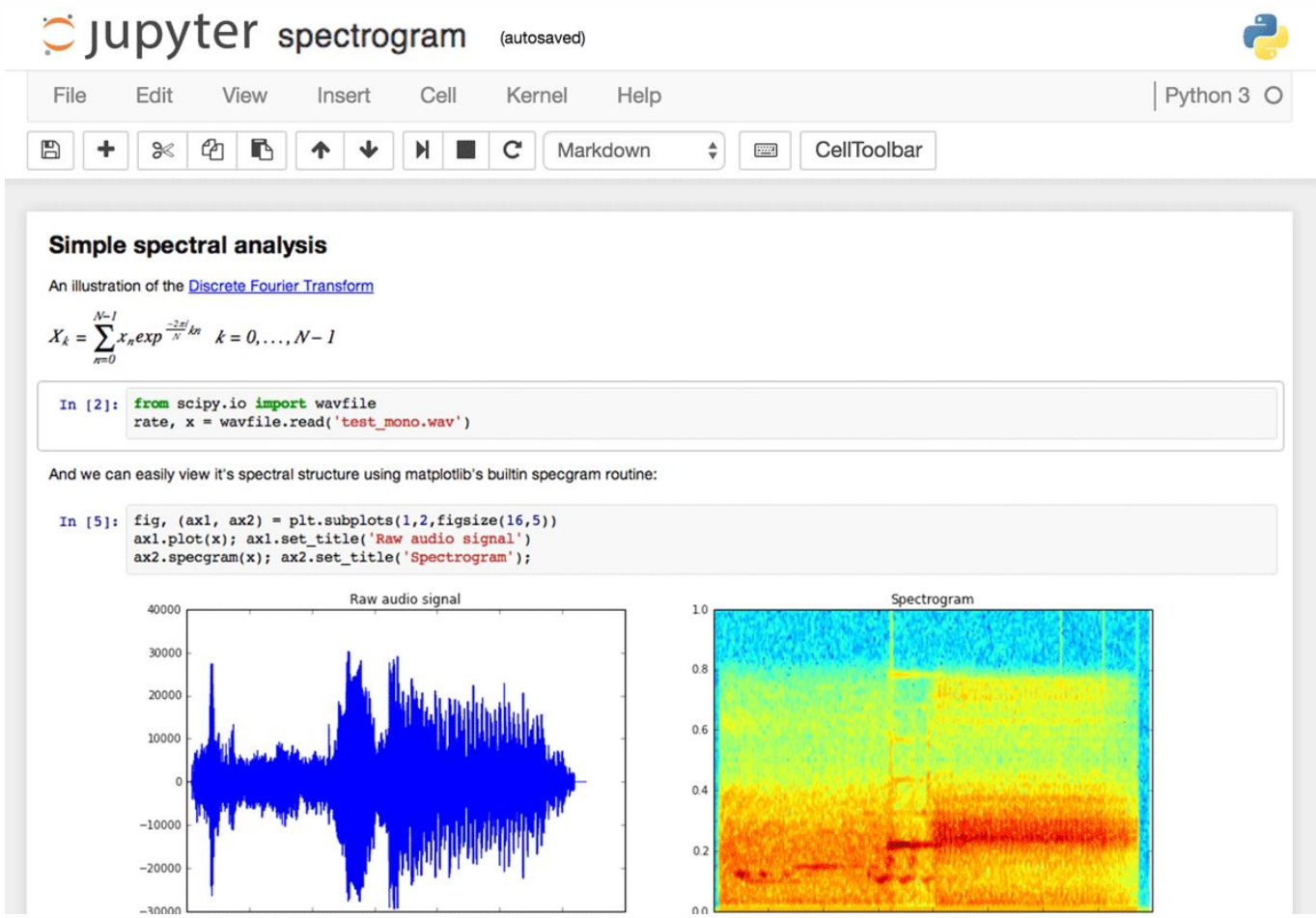
5 rows × 92 columns

In [ ]:

In [5]:
```
# d = data.groupby("addr_region_reg")['inquiry_1_week'].mean().to_dict()
# data['inquiry_1_week_mean'] = data.addr_region_reg.apply(lambda x: d[x])

# data["inquiry_old_count"] = data.ttl_inquiries - data.inquiry_12_month
data["ttl_inquiries_with_current"] = data.ttl_inquiries.fillna(0) + data.inquiry_recent_period.fillna(0)
data["inquiry_recent_period_perc"] = data.inquiry_recent_period.fillna(0) / (data.ttl_inquiries_with_current.fillna(0)+
data["loans_period"] = data.last_loan_date.fillna(0)-data.first_loan_date.fillna(0)
# data["loans_close_percent"] = (data.loans_main_borrower+data.loans_active)/(data.loans_active+0.001)
data["loans_close_times"] = data.loans_main_borrower.fillna(0) - data.loans_active.fillna(0)
# data["sas_limit_last_amt_delta"]=data.sas_limit_after_003_amt.fillna(0)==data.sas_limit_last_amt.fillna(0)

data["region_same"]=data.addr_region_reg.fillna(0)==data.addr_region_fact.fillna(0)
# data["region_same2"]=data.addr_region_reg==data.app_addr_region_reg
# data["region_same3"]=data.addr_region_fact==data.app_addr_region_fact

data['age'] = data.clnt_birth_year.fillna(0)*365-data.first_loan_date.fillna(0)
data['age2'] = data.clnt_birth_year.fillna(0)*365-data.last_loan_date.fillna(0)
data['f1'] = data.clnt_income_month_avg_net_amt.fillna(0)-data.sas_limit_after_003_amt.fillna(0)
data['f2'] = data.loans_main_borrower.fillna(0)/(data.loans_period.fillna(0)+0.001)
data['f3'] = data.loans_active.fillna(0)/(data.loans_main_borrower.fillna(0)+0.001)
data["feature_20_21_27"] = data.feature_20.fillna(0)+data.feature_21.fillna(0)+data.feature_27.fillna(0)
data["feature_22_23_24_25_28_12_10_13"] = data.feature_13.fillna(0)+data.feature_10.fillna(0)+ data.feature_12.fillna(0
```

Jupyter Notebook - достоинства

- Можно строить графики
- Удобно делать отчеты
- Удобно изучать новые пакеты
- Прост в установке
- Легко развернуть на удаленном сервере

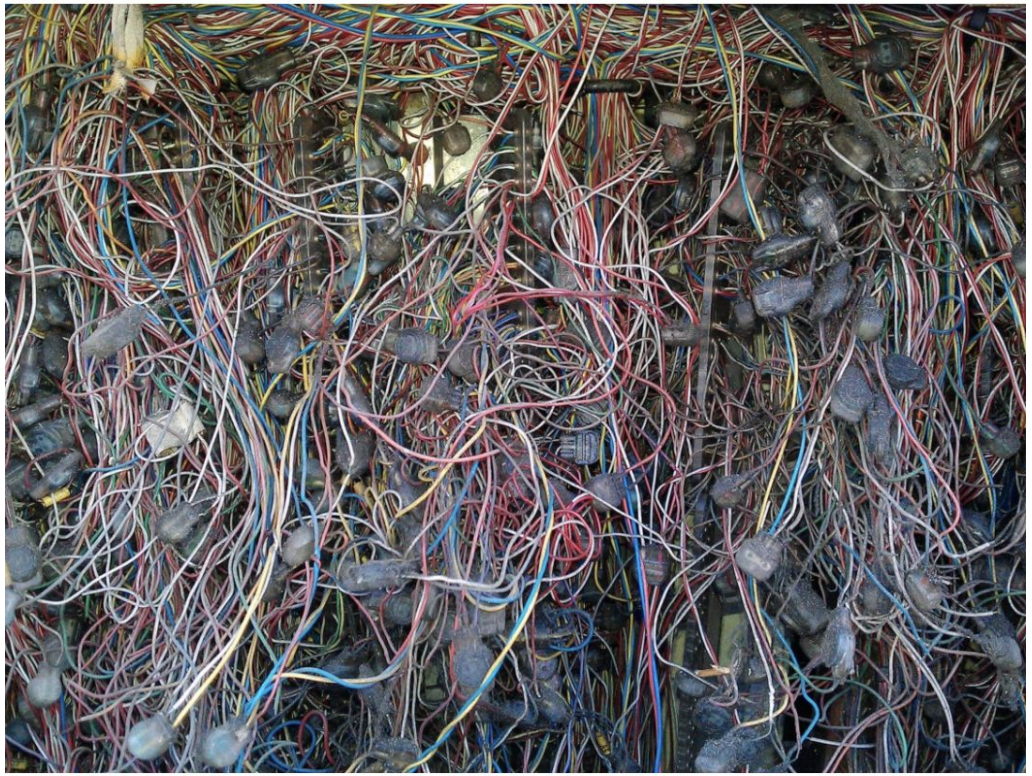


Jupyter Notebook - недостатки

- Проблемы с версионированием в гите
- Слабая IDE
- Сложно тестировать код
- Позволяет выполнять ячейки в произвольном порядке
-
- Плохой пользовательский опыт при запуске длительных вычислений

5 reasons why jupyter notebooks suck

Alexander Mueller Mar 24, 2018 · 3 min read ★



How it feels like managing jupyter notebooks (Complexity © https://www.flickr.com/photos/bitterjug/7670055210)

https://towardsdatascience.com/5-reasons-why-jupyter-notebooks-suck-4dc201e27086

Еще про ноутбуки

Яндекс Облако

Yandex
DataSphere

Amazon
SageMaker

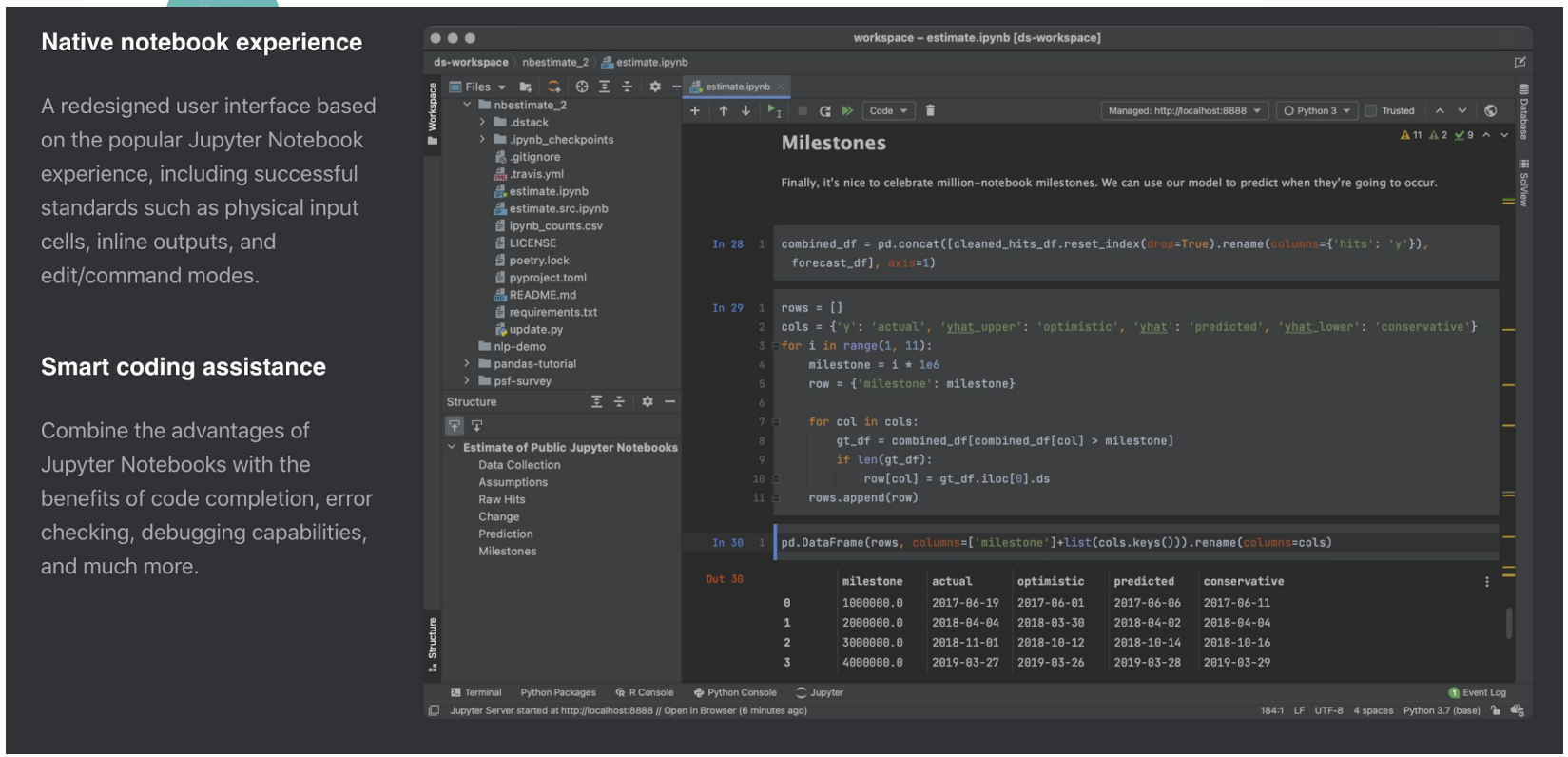
aws

databricks

Еще про ноутбуки

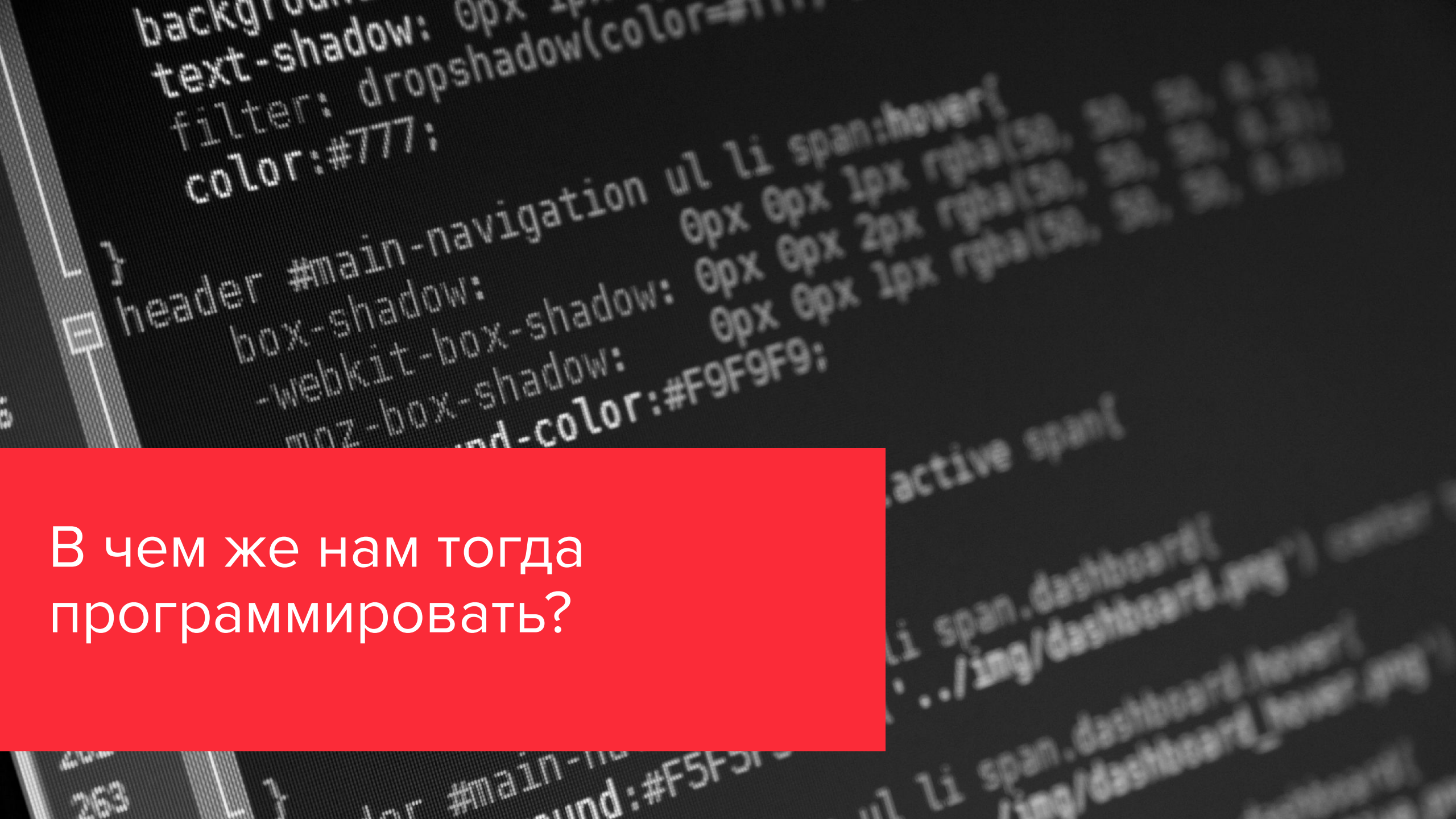
JetBrains DataSpell

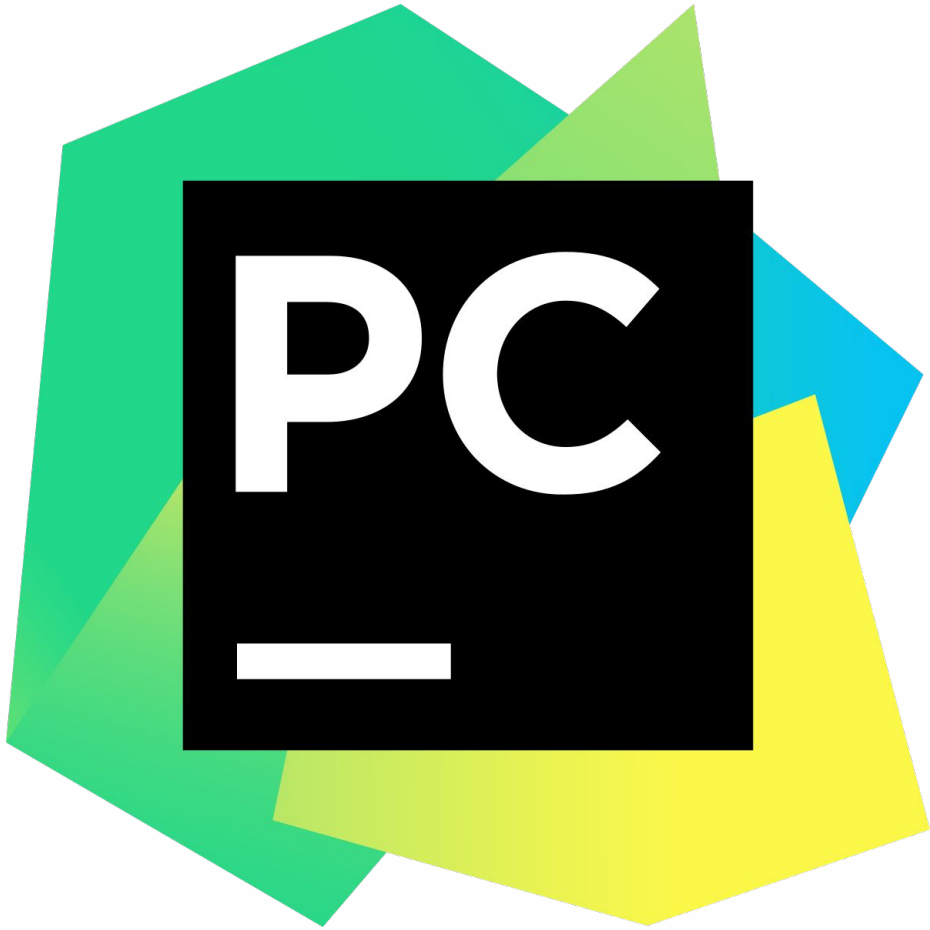
A new IDE by JetBrains for
data science. Join the preview!



Еще про ноутбуки

https://github.com/nteract/papermill


В чем же нам тогда программировать?



PyCharm

VS CODE

Характеристики продакшен кода

Качественный продакшн код должен обладать следующими характеристиками (top-5):

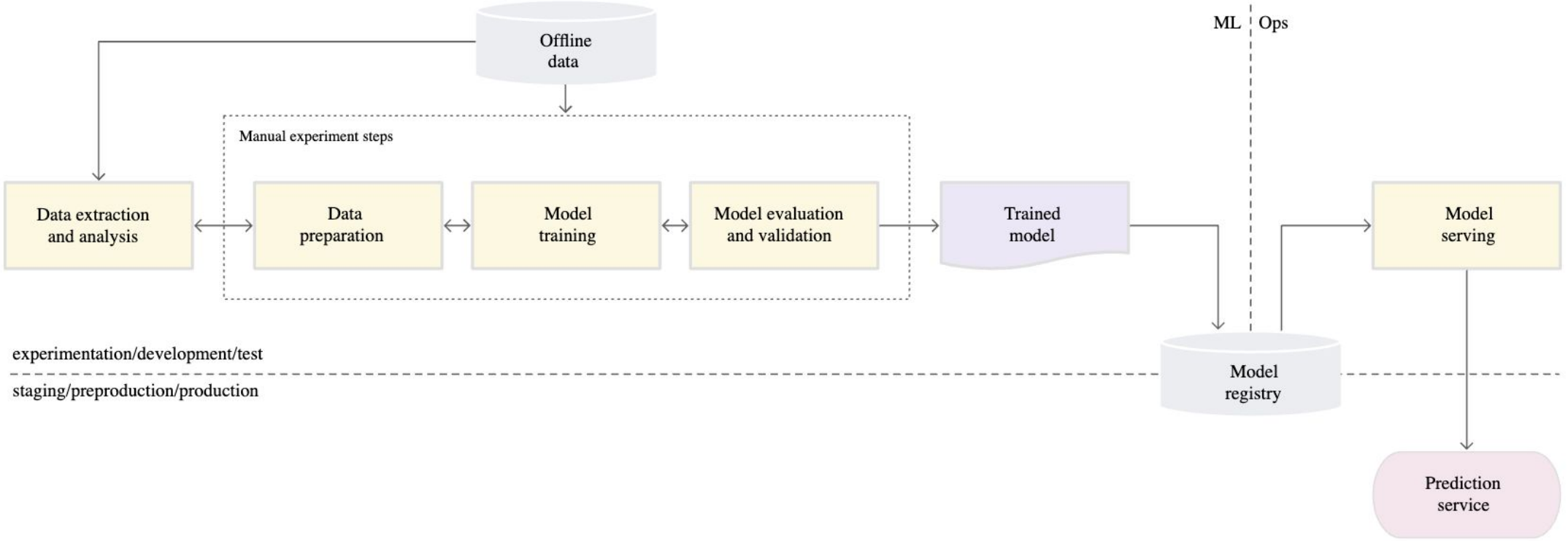
- Читаемый
- Эффективный
- Модульный
- Содержащий тесты и прописанные exceptions
- Задокументированный
-

Характеристики продакшн проекта:

- **Версионирование**
- **Удобная структура**
- **Явные зависимости**

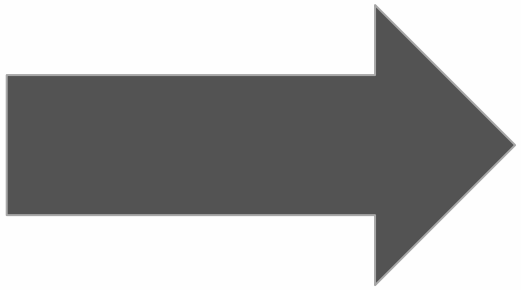
MLOPS Level 0

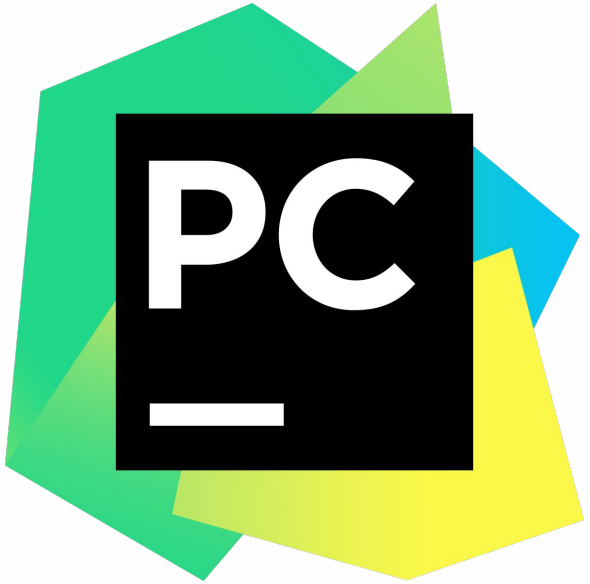
DS делает модели



https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Типичная картина



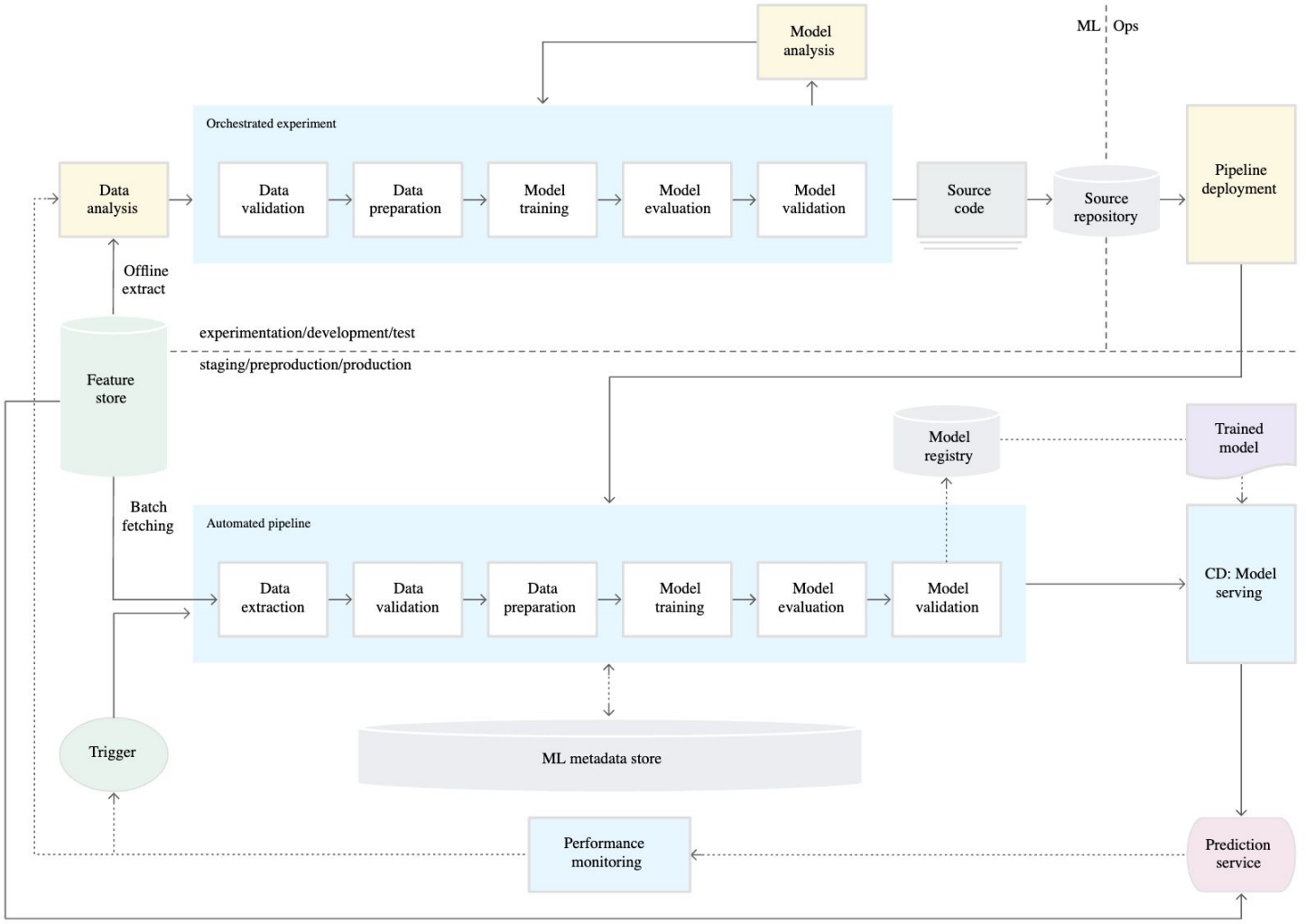


Код построения модели
- не поревьюен
- не воспроизводим
- не обобщаемый

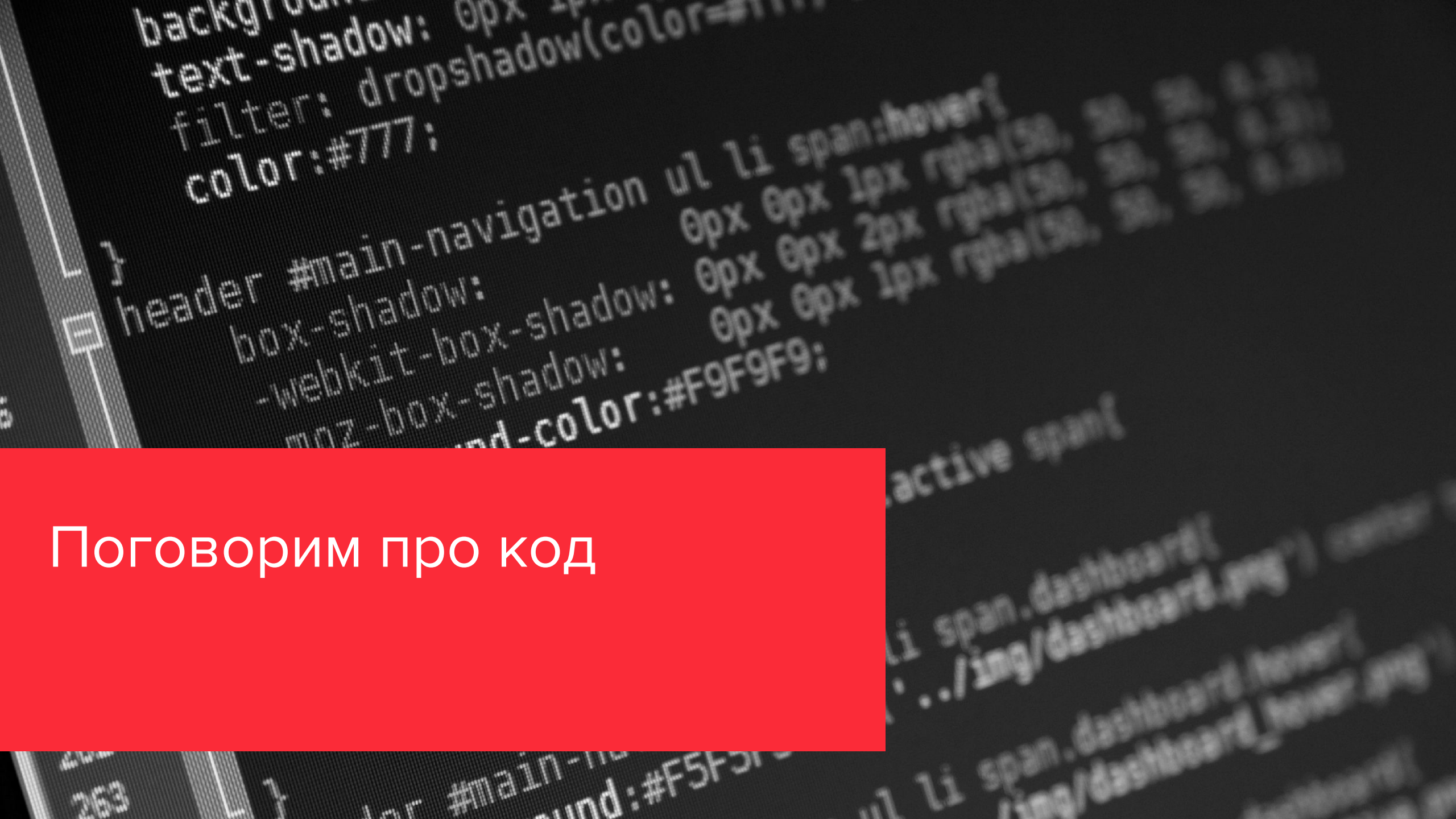
Код использования модели
- Написан бэкендером
- Он норм

MLOPS Level 1

Automated pipeline = prod code

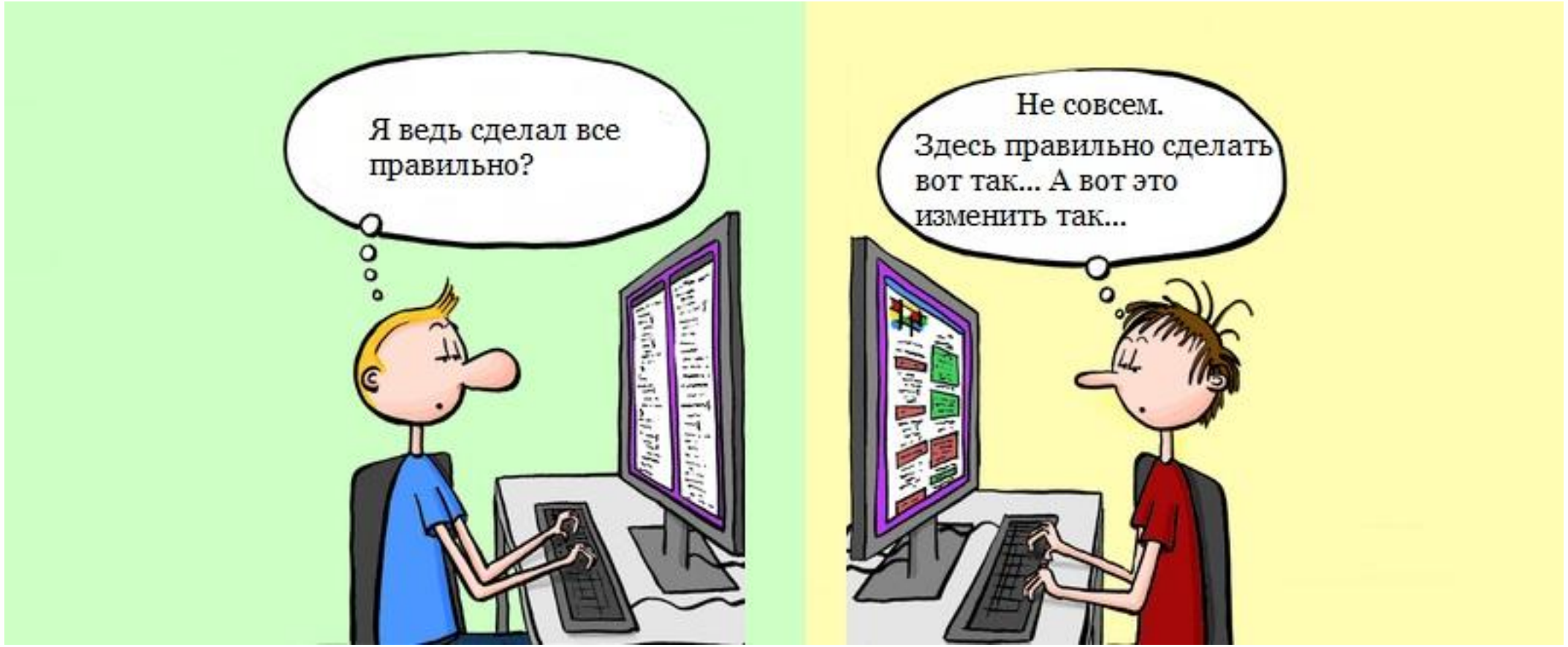


https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning



Поговорим про код

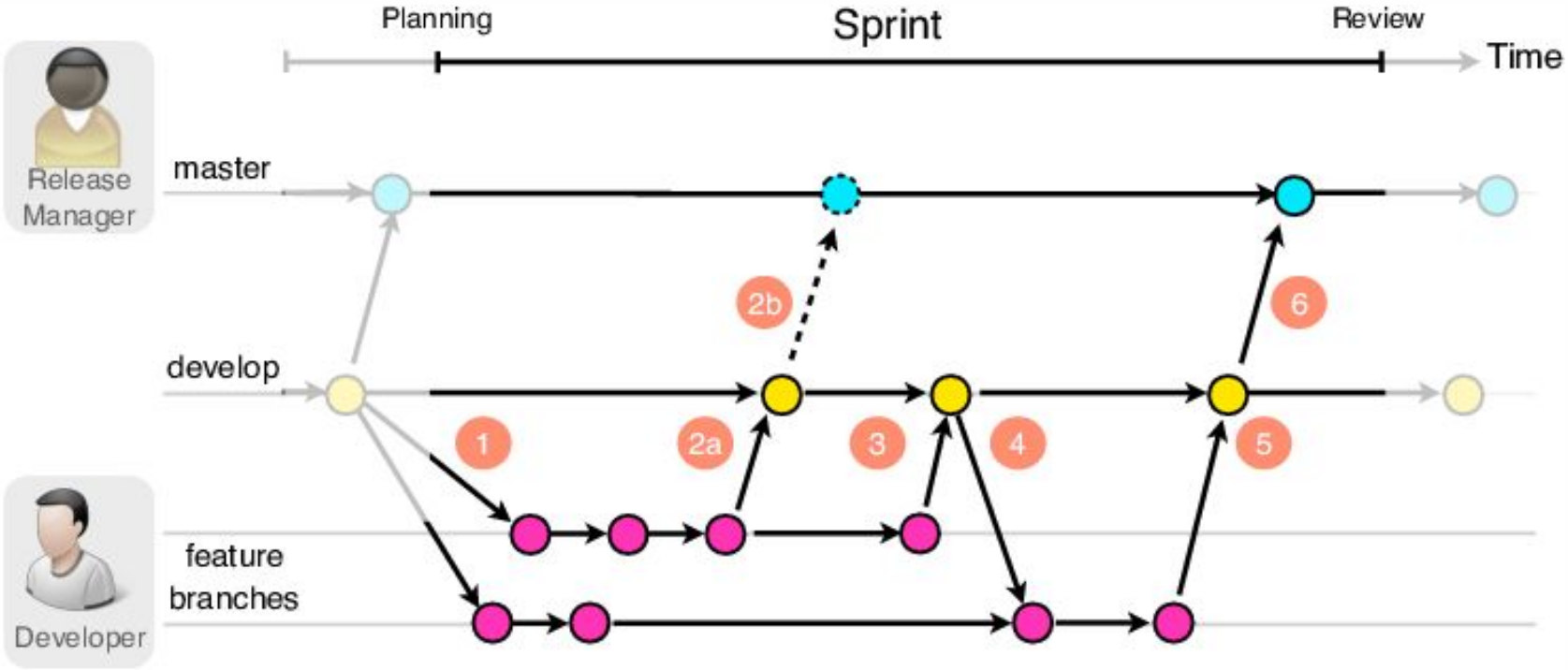
Читаемый код



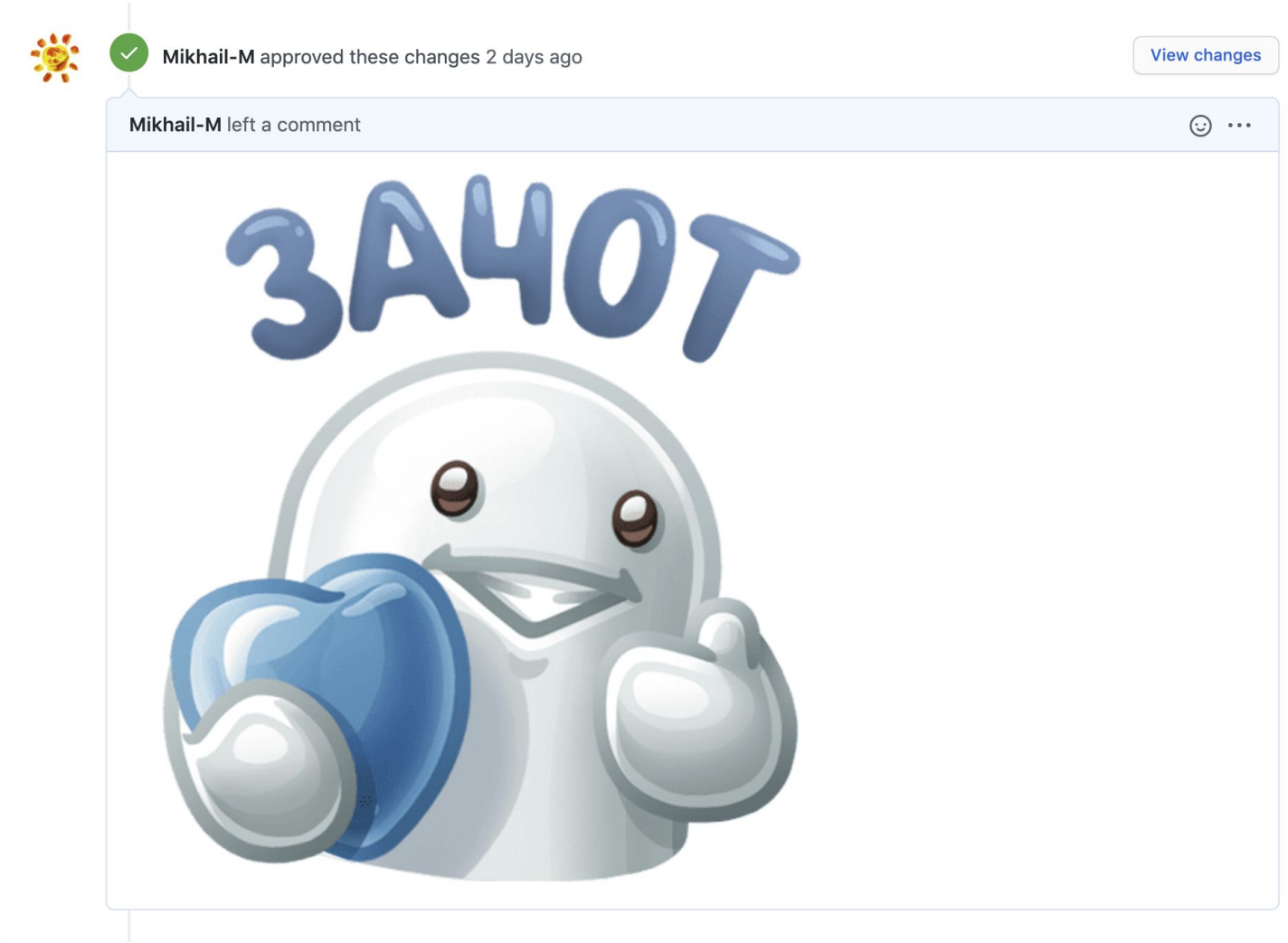
Ревью

Репозиторий

Создаем репозиторий до начала работы, а не в её конце!



PULL REQUEST



Читаемые названия

```
for i in range(n):
    for j in range(m):
        for k in range(l):
            temp_value = X[i][j][k] * 12.5
            new_array[i][j][k] = temp_value + 150
```

https://towardsdatascience.com/data-scientists-your-variable-names-are-awful-heres-how-to-fix-them-89053d2855be

Читаемые названия

```
for i in range(n):
    for j in range(m):
        for k in range(l):
            temp_value = X[i][j][k] * 12.5
            new_array[i][j][k] = temp_value + 150
```

- магические константы
- короткие/непонятные/не привязанные к задаче имена переменных

https://towardsdatascience.com/data-scientists-your-variable-names-are-awful-heres-how-to-fix-them-89053d2855be

Читаемые названия

```
PIXEL_NORMALIZATION_FACTOR = 12.5
PIXEL_OFFSET_FACTOR = 150

for row_index in range(row_count):
    for column_index in range(column_count):
        for color_channel_index in range(color_channel_count):
            normalized_pixel_value = (
                original_pixel_array[row_index][column_index][color_channel_index]
                * PIXEL_NORMALIZATION_FACTOR
            )
            transformed_pixel_array[row_index][column_index][color_channel_index] = (
                normalized_pixel_value + PIXEL_OFFSET_FACTOR
            )
```

https://towardsdatascience.com/data-scientists-your-variable-names-are-awful-heres-how-to-fix-them-89053d2855be

Нет портянкам

jupyter lgb-and-cat (автосохранение) Logout

File Edit View Insert Cell Kernel Widgets Help Не доверять Python 3

▶ Запуск Код

Out[4]:

	card_id	target	delivery_type	addr_region_reg	addr_region_fact	channel_name	channel_name_2	channel_name_modified_2018	sas_limit_after_003_amt	s
0	cid_10620	1	cat_1	107	107	cat_0	cat_3	cat_0	1	
1	cid_105724	0	cat_1	9	9	cat_2	cat_5	cat_2	3	
2	cid_101410	1	cat_1	109	109	cat_0	cat_3	cat_0	1	
3	cid_38961	0	cat_1	66	66	cat_0	cat_3	cat_0	3	
4	cid_57462	0	cat_1	16	16	cat_0	cat_3	cat_0	0	

5 rows × 92 columns

In [ ]:

In [5]:
```
# d = data.groupby("addr_region_reg")['inquiry_1_week'].mean().to_dict()
# data['inquiry_1_week_mean'] = data.addr_region_reg.apply(lambda x: d[x])

# data["inquiry_old_count"] = data.ttl_inquiries - data.inquiry_12_month
data["ttl_inquiries_with_current"] = data.ttl_inquiries.fillna(0) + data.inquiry_recent_period.fillna(0)
data["inquiry_recent_period_perc"] = data.inquiry_recent_period.fillna(0) / (data.ttl_inquiries_with_current.fillna(0)+
data["loans_period"] = data.last_loan_date.fillna(0)-data.first_loan_date.fillna(0)
# data["loans_close_percent"] = (data.loans_main_borrower+data.loans_active)/(data.loans_active+0.001)
data["loans_close_times"] = data.loans_main_borrower.fillna(0) - data.loans_active.fillna(0)
# data["sas_limit_last_amt_delta"]=data.sas_limit_after_003_amt.fillna(0)==data.sas_limit_last_amt.fillna(0)

data["region_same"]=data.addr_region_reg.fillna(0)==data.addr_region_fact.fillna(0)
# data["region_same2"]=data.addr_region_reg==data.app_addr_region_reg
# data["region_same3"]=data.addr_region_fact==data.app_addr_region_fact

data['age'] = data.clnt_birth_year.fillna(0)*365-data.first_loan_date.fillna(0)
data['age2'] = data.clnt_birth_year.fillna(0)*365-data.last_loan_date.fillna(0)
data['f1'] = data.clnt_income_month_avg_net_amt.fillna(0)-data.sas_limit_after_003_amt.fillna(0)
data['f2'] = data.loans_main_borrower.fillna(0)/(data.loans_period.fillna(0)+0.001)
data['f3'] = data.loans_active.fillna(0)/(data.loans_main_borrower.fillna(0)+0.001)
data["feature_20_21_27"] = data.feature_20.fillna(0)+data.feature_21.fillna(0)+data.feature_27.fillna(0)
data["feature_22_23_24_25_28_12_10_13"] = data.feature_13.fillna(0)+data.feature_10.fillna(0)+ data.feature_12.fillna(
```

Нет портянкам

- Используйте функции и классы, они помогут сделать код обозримым

```
def run_training() -> NoReturn:
    data = load_dataset(file_name=config.TRAINING_DATA_FILE)

    X_train, X_test, y_train, y_test = train_test_split(
        data[config.FEATURES], data[config.TARGET], test_size=0.1, random_state=0
    )

    pipeline.fit(X_train[config.FEATURES], y_train)

    _logger.info(f"saving model version: {_version}")
    save_pipeline(pipeline_to_persist=pipeline.price_pipe)
```

Маленькие функции

- Если функция занимает более 50 строк, то вероятно, её лучше разбить
- Если вы копируете код - пора создать функцию. (но можно и раньше)

SOLID

- The Single-responsibility principle: a class should only have a single responsibility
- The Open–closed principle: software entities … should be open for extension, but closed for modification.
- The Liskov substitution principle: objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program
- The Interface segregation principle: many client-specific interfaces are better than one general-purpose interface.
- The Dependency inversion principle: depend upon abstractions, [not] concretions

https://en.wikipedia.org/wiki/SOLID

Работа с ресурсами

```
f = open("hello.txt", "w")
f.write("hello!")
f.close()
```

Работа с ресурсами

```
f = open("hello.txt", "w")
try:
    f.write("hello!")
finally:
    f.close()
```

Работа с ресурсами

```
with open("hello.txt", "w") as f:
    f.write("hello!")
```

Typing

```
def fit_on_data(model, features, target):
    model.fit(features.to_numpy(), target.to_numpy())
    return model
```

https://docs.python.org/3/library/typing.html

Typing

```
def fit_on_data(model, features, target):
    model.
    mode                                                    numpy())
    retu
```

```
fit
if                                          if expr
ifn                                 if expr is None
ifnn                          if expr is not None
len                                         len(expr)
```

https://docs.python.org/3/library/typing.html

Typing

```python
def fit_on_data_typing(
    model: ForestClassifier, features: pd.DataFrame, target: pd.DataFrame
) -> ForestClassifier:
    model.fit(features.to_numpy(), target.to_numpy())
    return model
```

https://docs.python.org/3/library/typing.html

Typing

```
def fit_on_data_typing(
    model: ForestClassifier, features: pd.DataFrame, target: pd.DataFrame
) -> ForestClassifier:
    model.
    mode                                                    _numpy())
    retu
```

```
fit(self, X, y, samp…   BaseForest
estimators_             BaseForest
verbose                 BaseForest
if                         if expr
ifn                 if expr is None
apply(self, X)          BaseForest
bootstrap               BaseForest
class_weight            BaseForest
classes_            ForestClassifier
```

https://docs.python.org/3/library/typing.html

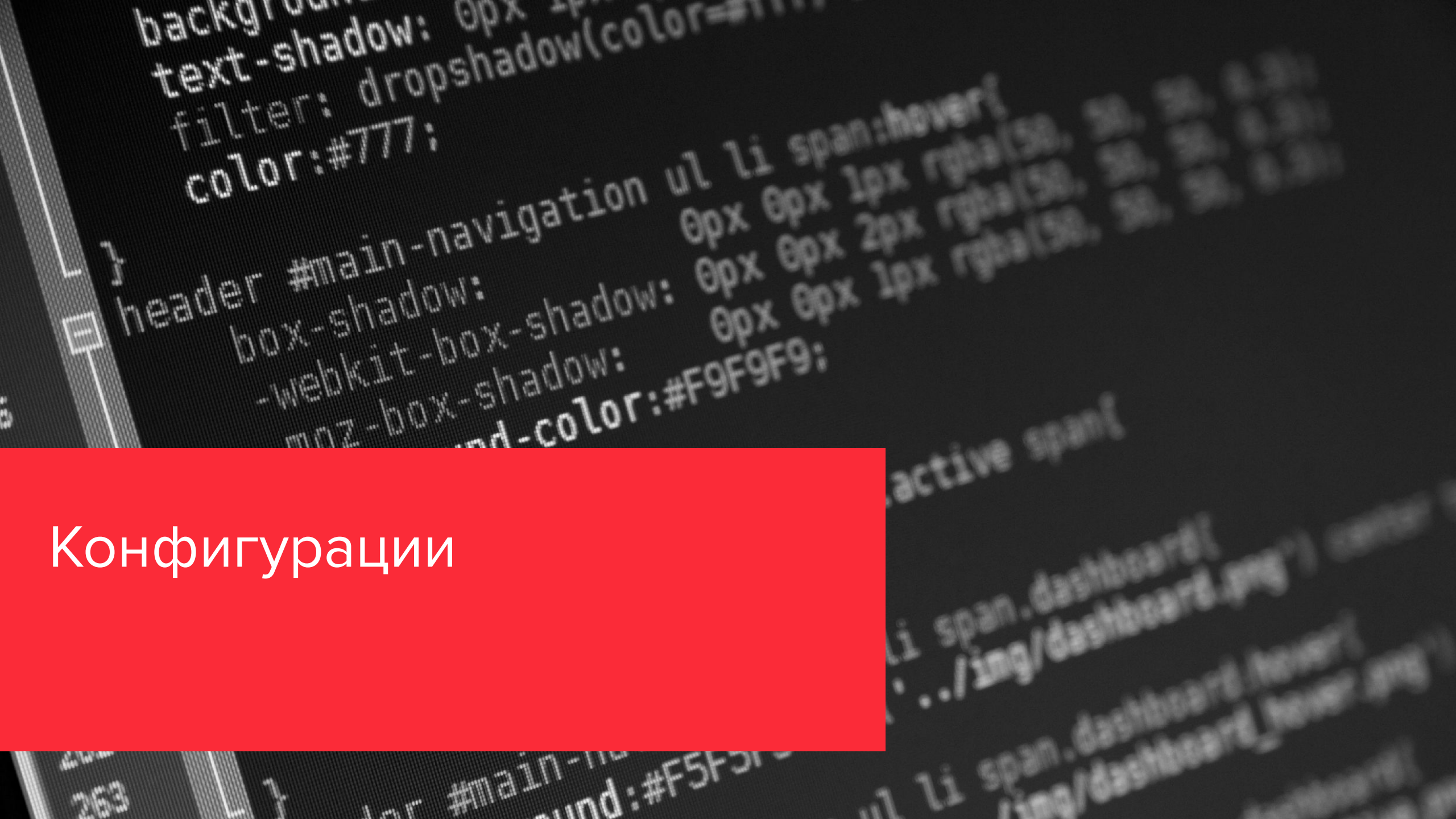
Typing

```
def fit_on_data_typing(
    model: ForestClassifier, features: pd.DataFrame, target
) -> ForestClassifier:
    features.
    model.f                                                    npy())
    return
```

```
to_numpy(self, dtype,…   DataFrame
info                     DataFrame
join(self, other, on,…   DataFrame
append(self, other, i…   DataFrame
shape                    DataFrame
```

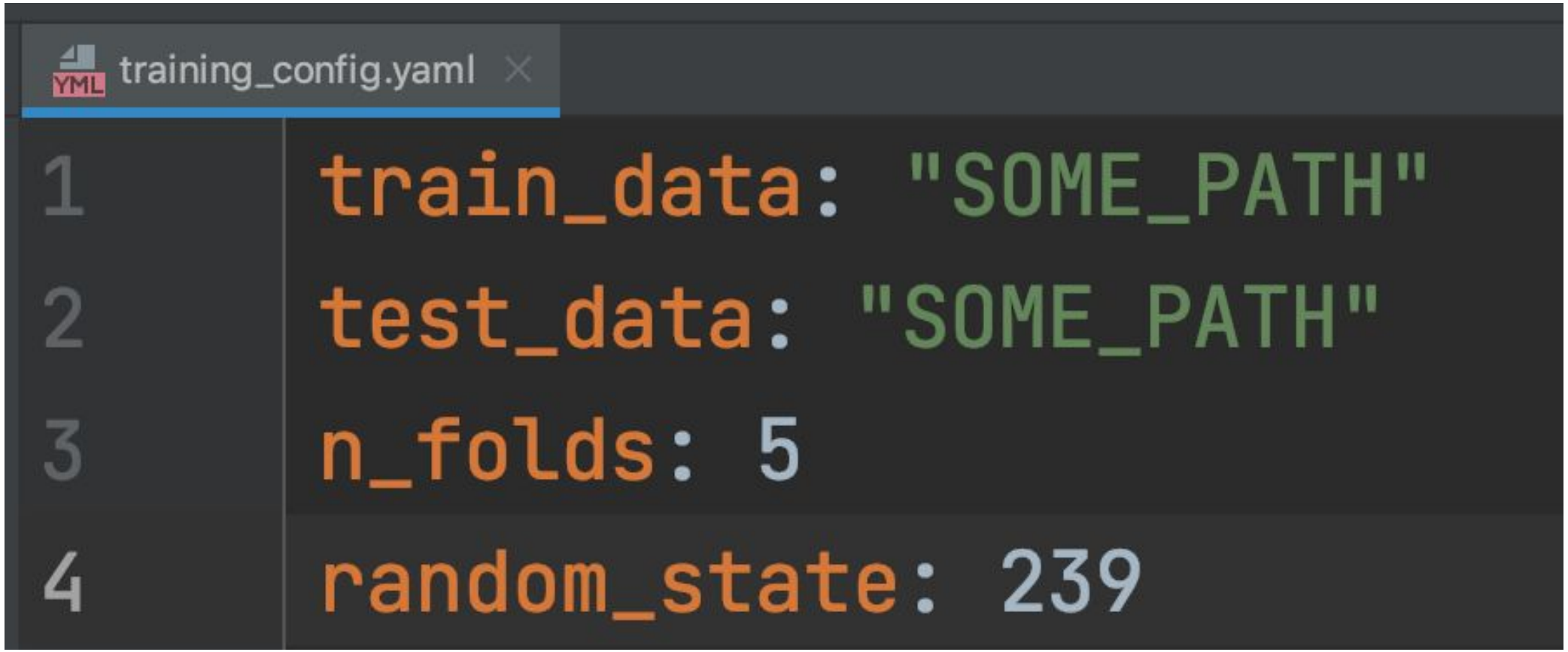
Проверка типов в python

https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html



Конфигурации

Конфиг в yaml



Считывание конфига из yaml

```
def read_config(config_path: str) -> Dict[str, Union[int, str]]:
    with open(config_path, "r") as input_stream:
        config = yaml.safe_load(input_stream)
        return config
```

Используем конфиги из yaml

```python
def test_read_config(config_path: str):
    config = read_config(config_path)
    assert config["n_folds"] == 5
    assert config["test_data"]
    all_fields = ["n_folds", "test_data", "train_data", "random_state"]

    assert set(all_fields) == set(config.keys())
```

DataClasses

```python
@dataclass
class TrainingParams:
    train_data: str
    test_data: str
    random_state: int
    n_folds: int = field(default=5)


TrainingParamsSchema = class_schema(TrainingParams)


def read_config_to_dataclass(config_path: str) -> TrainingParams:
    params_dict = read_config(config_path)
    return TrainingParamsSchema().load(params_dict)
```

DataClasses

```
def test_read_config_dataclass(config_path: str):
    params: TrainingParams = read_config_to_dataclass(config_path)
    assert params.n_folds == 5
    assert params.
```

```
n_folds                TrainingParams
test_data              TrainingParams
par                          (expr)
```

Python Object Serialization libraries

serpyco

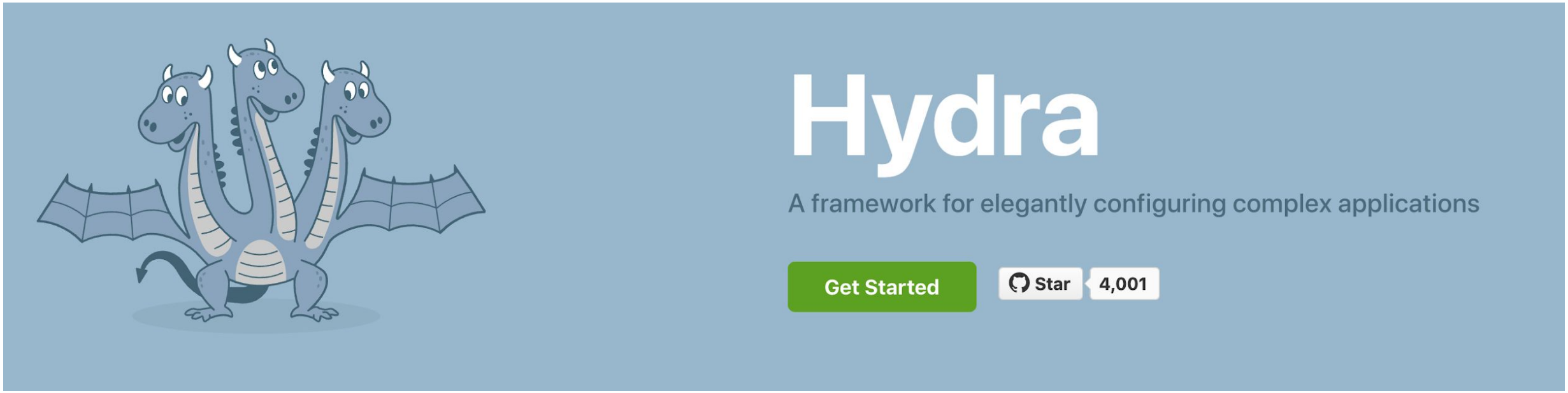
Project ID: 9168810

256 Commits **3** Branches **50** Tags **1.5 MB** Files **321.5 MB** Storage

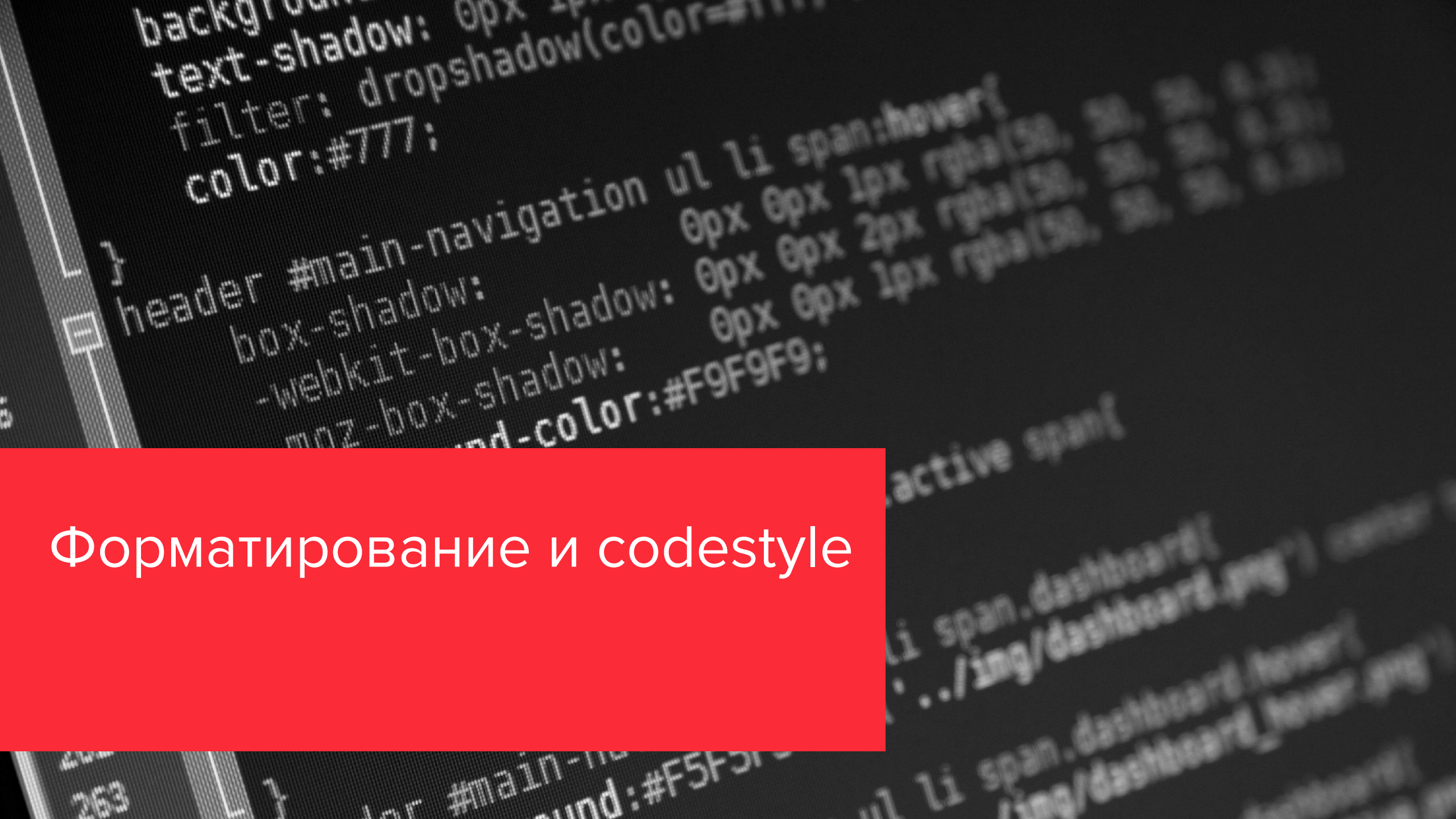
Fast serialization of dataclasses using Cython.

Посмотрим на конфиг

Hydra



https://hydra.cc/docs/intro/



Форматирование и codestyle

Python convention

Naming Styles

The table below outlines some of the common naming styles in Python code and when you should use them:

Type	Naming Convention	Examples
Function	Use a lowercase word or words. Separate words by underscores to improve readability.	`function`, `my_function`
Variable	Use a lowercase single letter, word, or words. Separate words with underscores to improve readability.	`x`, `var`, `my_variable`
Class	Start each word with a capital letter. Do not separate words with underscores. This style is called camel case.	`Model`, `MyClass`
Method	Use a lowercase word or words. Separate words with underscores to improve readability.	`class_method`, `method`
Constant	Use an uppercase single letter, word, or words. Separate words with underscores to improve readability.	`CONSTANT`, `MY_CONSTANT`, `MY_LONG_CONSTANT`
Module	Use a short, lowercase word or words. Separate words with underscores to improve readability.	`module.py`, `my_module.py`
Package	Use a short, lowercase word or words. Do not separate words with underscores.	`package`, `mypackage`

Строки в питоне

```
s1 = "QWERTY"
s2 = 'QWERTY'
```

Строки в питоне

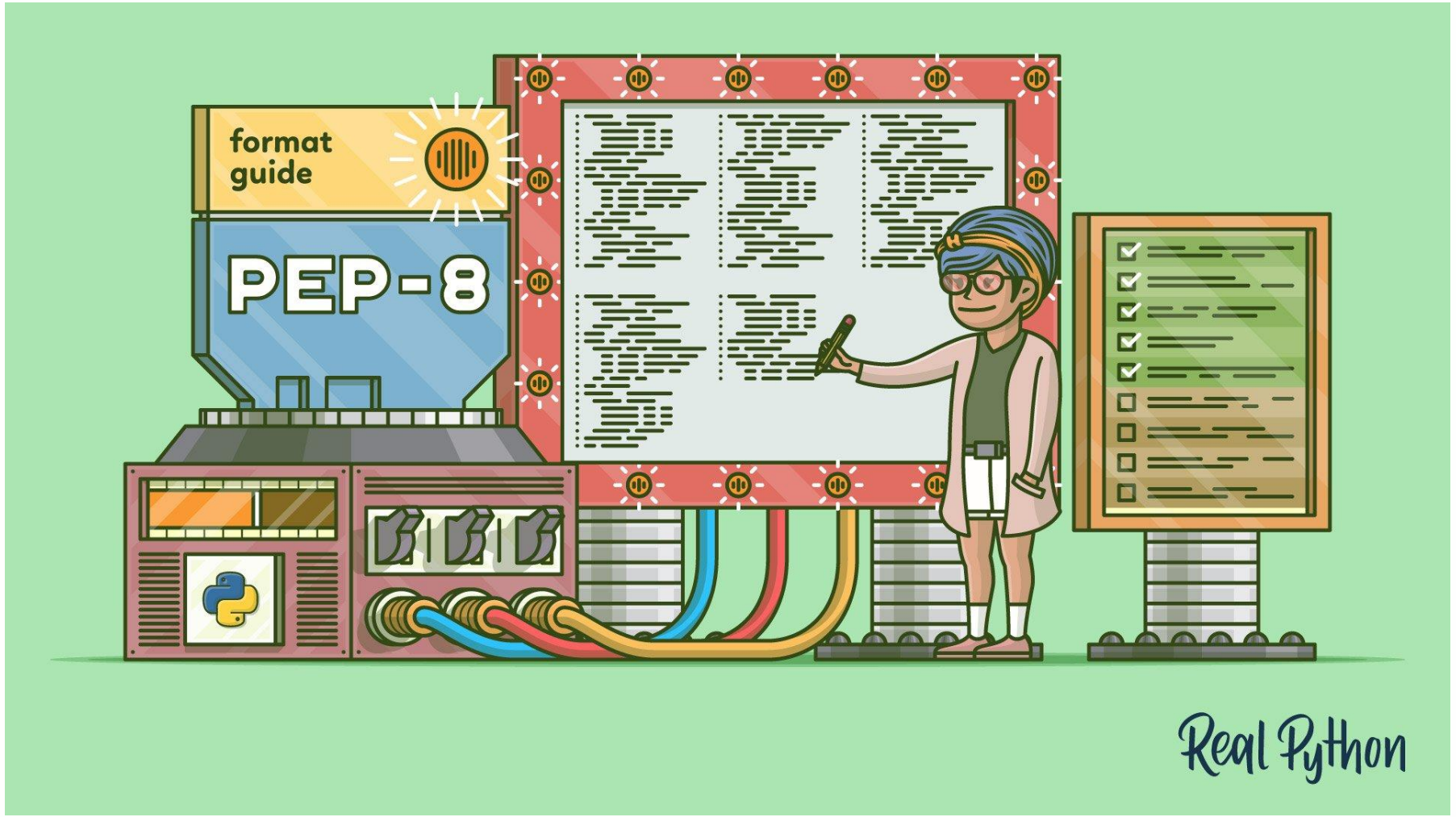
```
s1 = "QWERTY"
s2 = 'QWERTY'
```

Всегда используйте что-то одно

Форматирование строк в питоне

```python
def test_string_format():
    name = "Misha"
    s0 = "Hello, " + name
    s1 = "Hello, %s" % name  # classic
    s2 = "Hello, {}".format(name)  # py3 format
    s3 = f"Hello, {name}"  # fstring

    assert s0 == s1 == s2 == s3
```

PEP8



https://www.python.org/dev/peps/pep-0008/

BLACK

```
def very_important_function(template: str, *variables, file: os.PathLike,
                            engine: str, header: bool = True, debug: bool = False):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, 'w') as f:
        ...

# out:

def very_important_function(
    template: str,
    *variables,
    file: os.PathLike,
    engine: str,
    header: bool = True,
    debug: bool = False,
):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
        ...
```

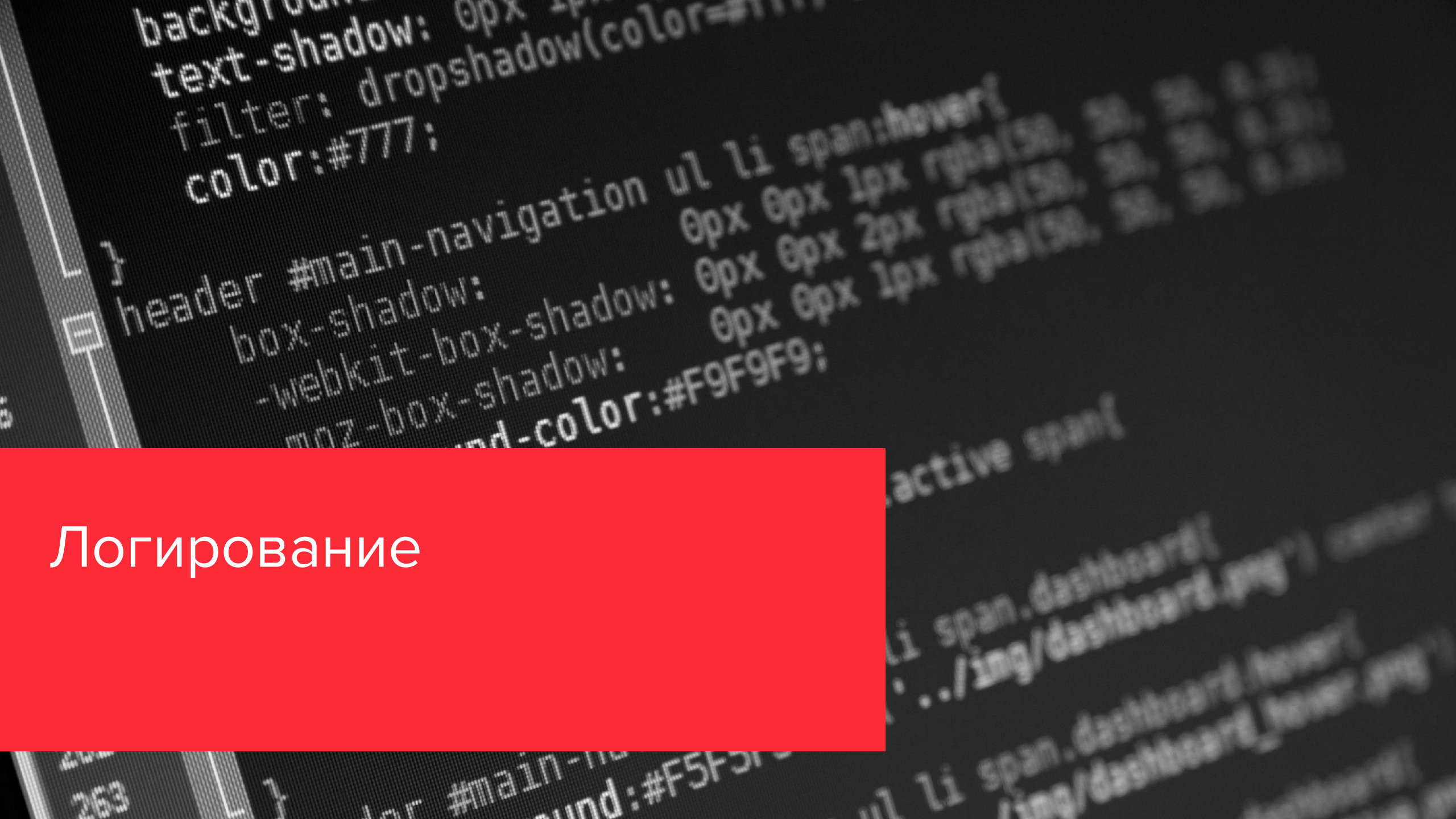
https://github.com/psf/black

BLACK

```
# in
j = [1,
     2,
     3
]

# out:

j = [1, 2, 3]
```

BLACK

```
# in

ImportantClass.important_method(exc, limit, lookup_lines, capture_locals, extra_argument)

# out:

ImportantClass.important_method(
    exc, limit, lookup_lines, capture_locals, extra_argument
)
```



Логирование

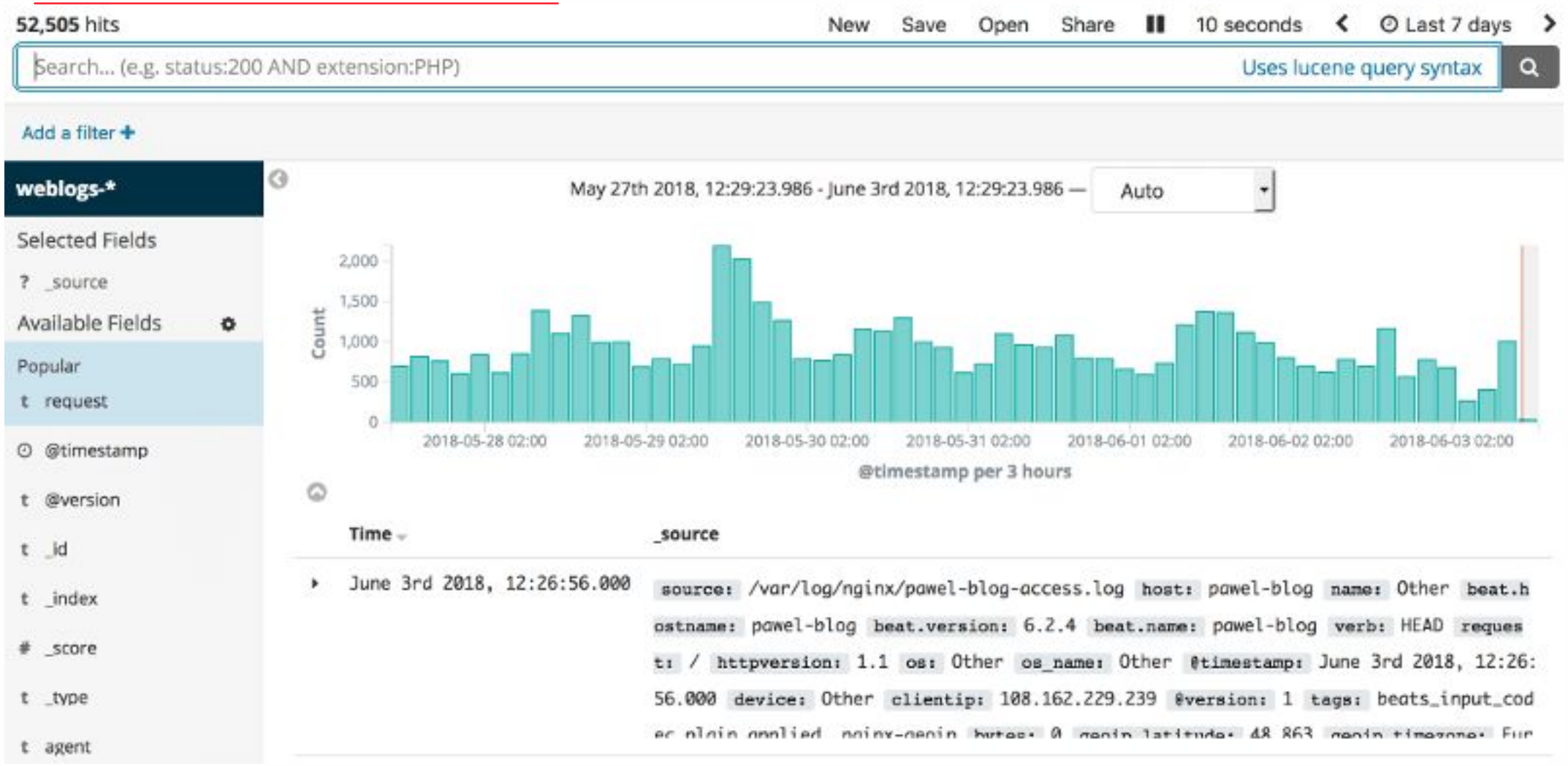
Print vs Logging

```
def some_function_with_print():
    print("start doing")
    try:
        do_stuff()
    except RuntimeError as e:
        print(e)
    print("end doing")
```

Print vs Logging

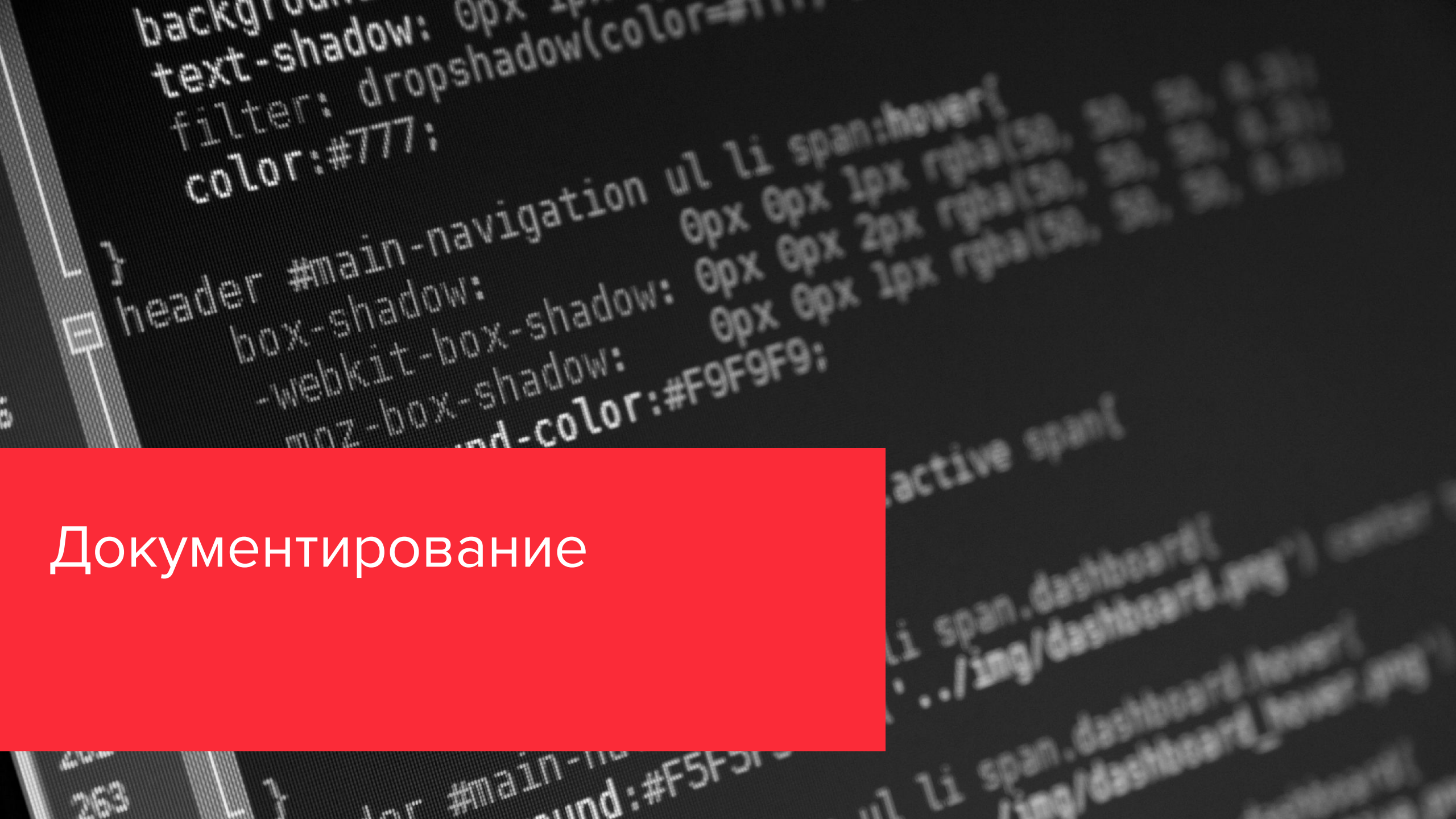
```
def some_function_with_logging():
    logging.info("start doing")
    try:
        do_stuff()
    except RuntimeError as e:
        logging.error(e)
    logging.info("end doing")
```

Kibana


52,505 hits
New
Save
Open
Share
10 seconds
Last 7 days
Search... (e.g. status:200 AND extension:PHP)
Uses lucene query syntax
Add a filter +
weblogs-*
Selected Fields
? _source
Available Fields
Popular
t request
@timestamp
t @version
t _id
t _index
_score
t _type
t agent
May 27th 2018, 12:29:23.986 - June 3rd 2018, 12:29:23.986 —
Auto
Count
2,000
1,500
1,000
500
0
2018-05-28 02:00
2018-05-29 02:00
2018-05-30 02:00
2018-05-31 02:00
2018-06-01 02:00
2018-06-02 02:00
2018-06-03 02:00
@timestamp per 3 hours
Time
_source
June 3rd 2018, 12:26:56.000
source: /var/log/nginx/pawel-blog-access.log host: pawel-blog name: Other beat.h
ostname: pawel-blog beat.version: 6.2.4 beat.name: pawel-blog verb: HEAD reques
t: / httpversion: 1.1 os: Other os_name: Other @timestamp: June 3rd 2018, 12:26:
56.000 device: Other clientip: 108.162.229.239 @version: 1 tags: beats_input_cod

PLEASE, DON’T

```
print = logging.info


def some_function_with_print():
    print("start doing")
    try:
        do_stuff()
    except RuntimeError as e:
        print(e)
    print("end doing")
```



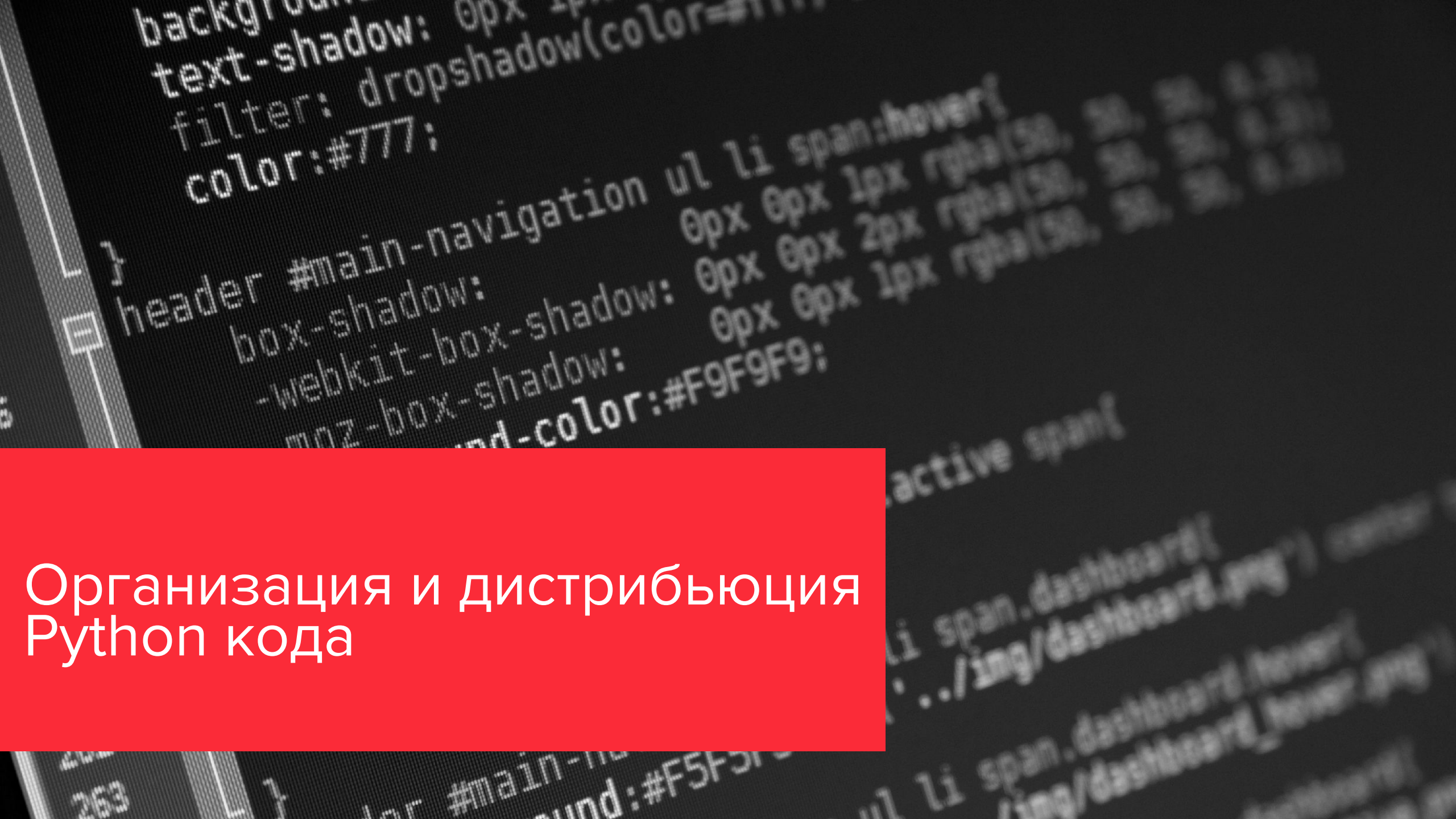
Документирование

SELF-DOCUMENTED

Самодокументируемый код

Документация

```
class ExampleError(Exception):
    """Exceptions are documented in the same way as classes.

    The __init__ method may be documented in either the class level
    docstring, or as a docstring on the __init__ method itself.

    Either form is acceptable, but the two should not be mixed. Choose
    convention to document the __init__ method and be consistent with i

    Note:
        Do not include the `self` parameter in the ``Args`` section.

    Args:
        msg (str): Human readable string describing the exception.
        code (:obj:`int`, optional): Error code.

    Attributes:
        msg (str): Human readable string describing the exception.
        code (int): Exception error code.

    """

    def __init__(self, msg, code):
        self.msg = msg
        self.code = code
```



Организация и дистрибьюция Python кода

Модуль

Модуль это просто файл .py

Там могут быть
- импорты других модулей
- объявления функций, классов, переменных
- исполняемый код

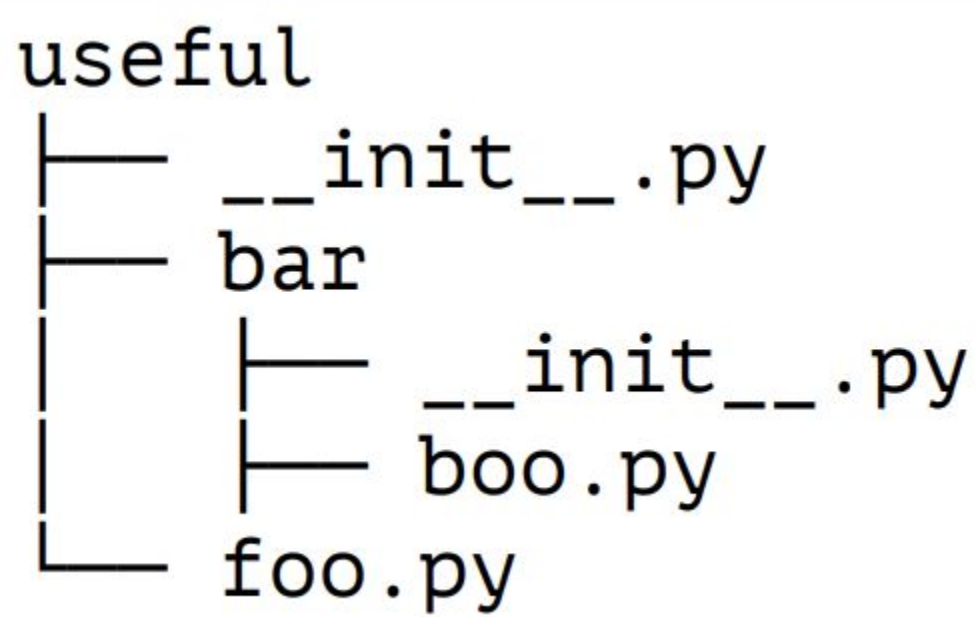
Модуль

```python
import os


def example_fun() -> str:
    return os.curdir


if __name__ == "__main__":
    print(example_fun())
```

Пакет

Пакет это просто папка с модулями

Там могут быть
- Модули
- Вложенные пакеты
- __init__.py



__init__.py

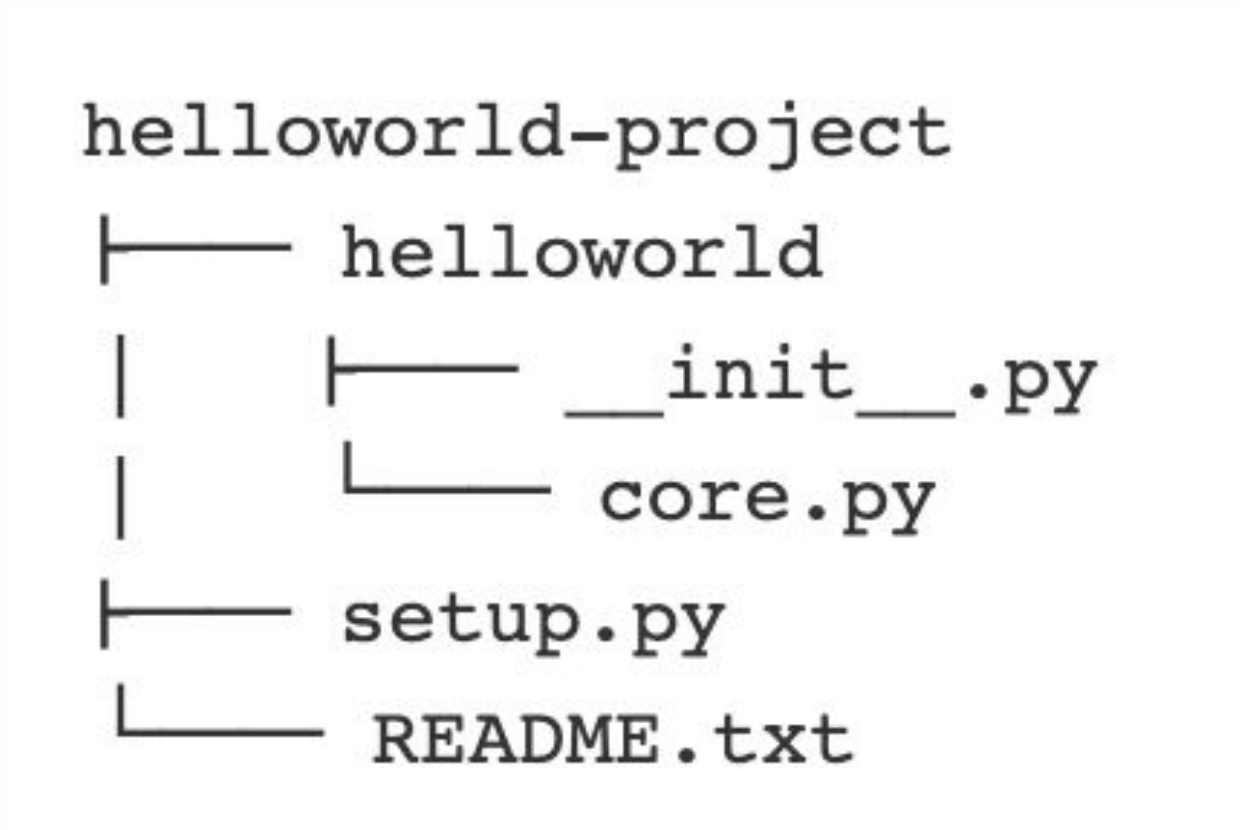
- При импорте пакета (**import package, from my_cool_lib import package**) импортируется только __init__.py (выполняется все то, что написано внутри)

- Служит фасадом

```
from ._dict_vectorizer import DictVectorizer
from ._hash import FeatureHasher
from .image import img_to_graph, grid_to_graph
from . import text

__all__ = ['DictVectorizer', 'image', 'img_to_graph', 'grid_to_graph', 'text',
           'FeatureHasher']
```

https://stackoverflow.com/questions/448271/what-is-init-py-for

Структура проекта



https://klen.github.io/create-python-packages.html

setup.py

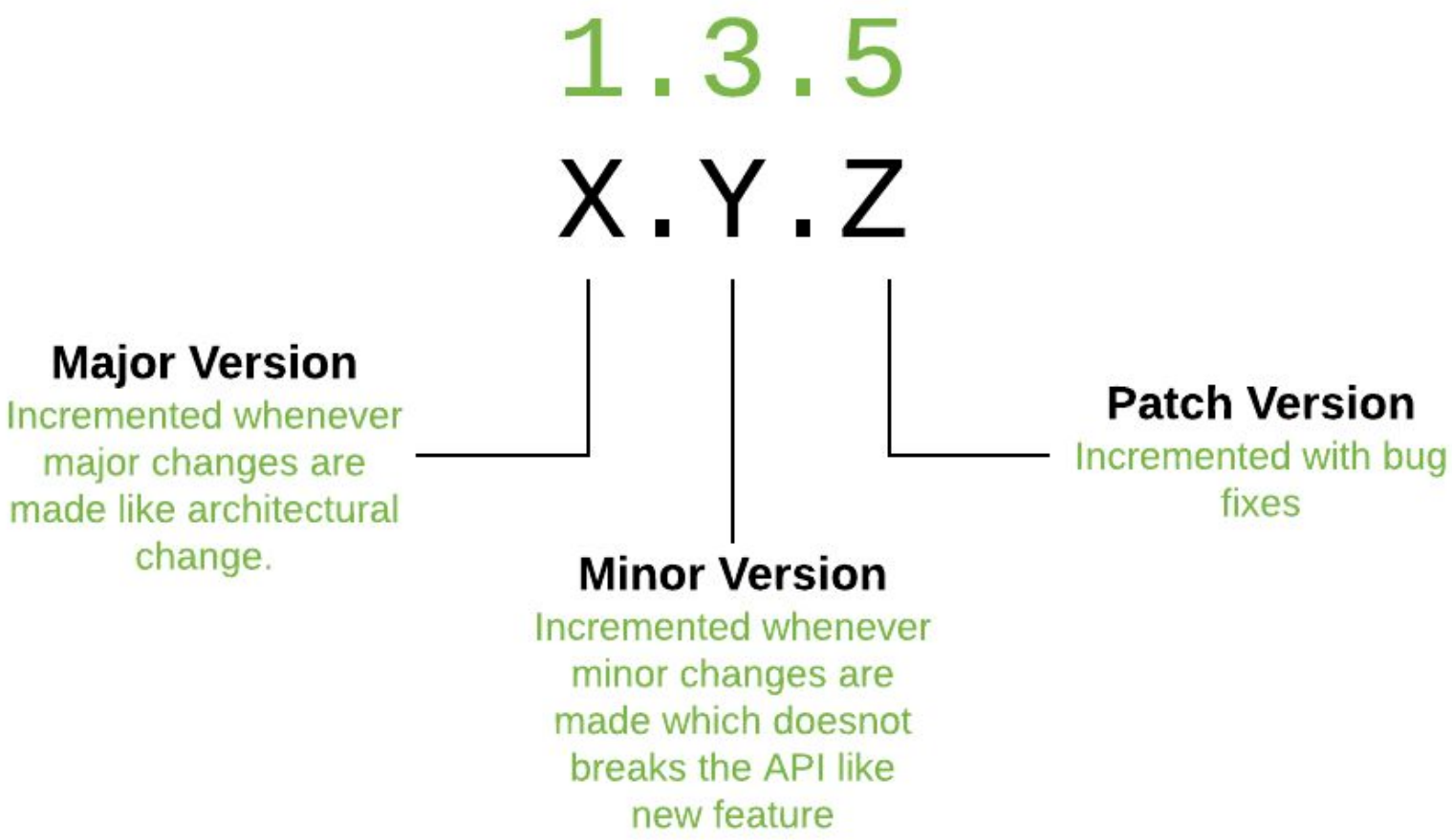
```
#!/usr/bin/env python

from distutils.core import setup

setup(name='Distutils',
      version='1.0',
      description='Python Distribution Utilities',
      author='Greg Ward',
      author_email='gward@python.net',
      url='https://www.python.org/sigs/distutils-sig/',
      packages=['distutils', 'distutils.command'],
     )
```

https://docs.python.org/3/distutils/setupscript.html

setup.py

```
setup(
    name="dvc",
    version=version,
    description="Git for data scientists - manage your code and data together",
    long_description=open("README.rst", "r", encoding="UTF-8").read(),
    author="Dmitry Petrov",
    author_email="dmitry@dvc.org",
    download_url="https://github.com/iterative/dvc",
    license="Apache License 2.0",
    install_requires=install_requires,
    extras_require={
        "all": all_remotes,
        "gs": gs,
        "gdrive": gdrive,
        "s3": s3,
        "azure": azure,
        "oss": oss,
        "ssh": ssh,
        "ssh_gssapi": ssh_gssapi,
        "hdfs": hdfs,
        "webhdfs": webhdfs,
        "webdav": webdav,
        "tests": tests_requirements,
    },
    keywords="data-science data-version-control machine-learning git"
    " developer-tools reproducibility collaboration ai",
    python_requires=">=3.6",
    classifiers=[
        "Development Status :: 4 - Beta",
        "Programming Language :: Python :: 3",
        "Programming Language :: Python :: 3.6",
        "Programming Language :: Python :: 3.7",
        "Programming Language :: Python :: 3.8",
    ],
    packages=find_packages(exclude=["tests"]),
    include_package_data=True,
    url="http://dvc.org",
```

Semantic Versioning

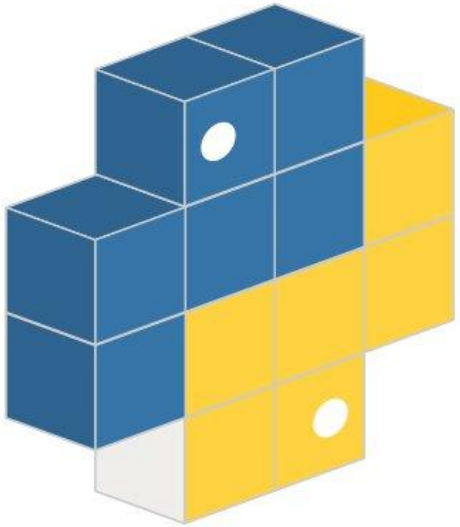


Дистрибьюция

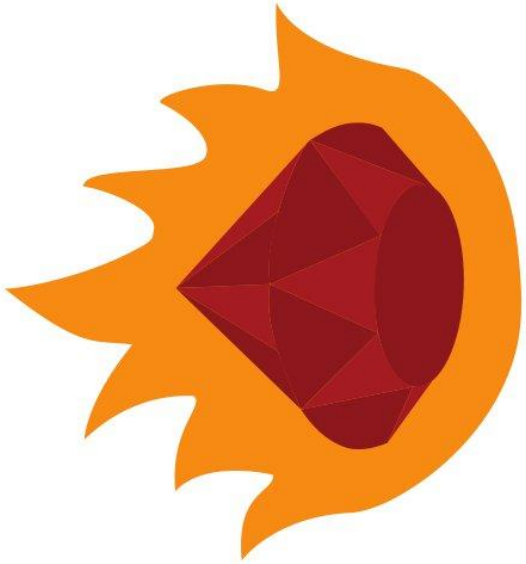
Проект, содержащий setup.py можно собрать
> **python setup.py sdist** -> my_project.tar.gz

Этот пакет можно устанавливать и распространять
> **pip install my_project.tar.gz**

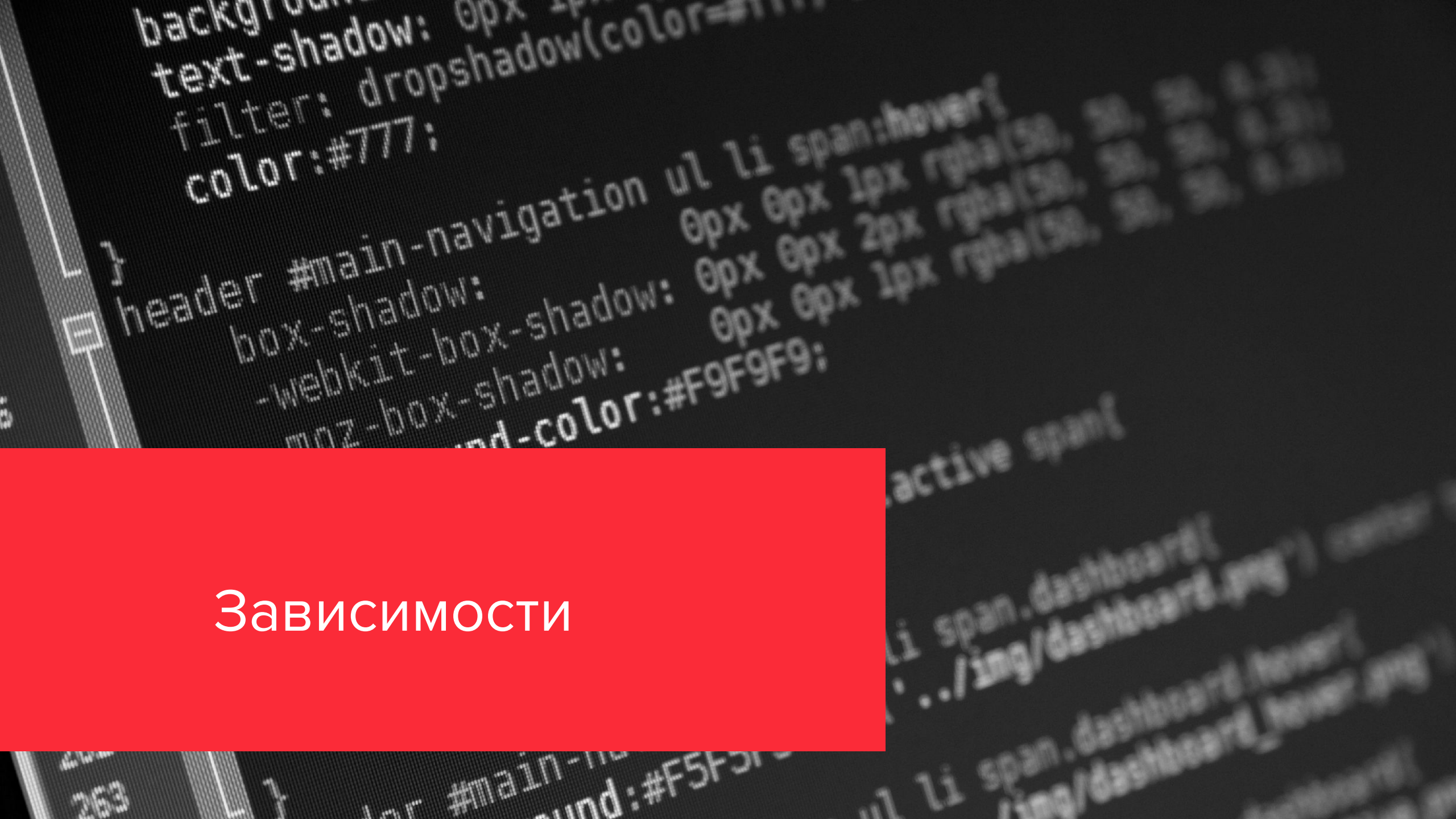
Репозитории пакетов



Pypi



Gemfury



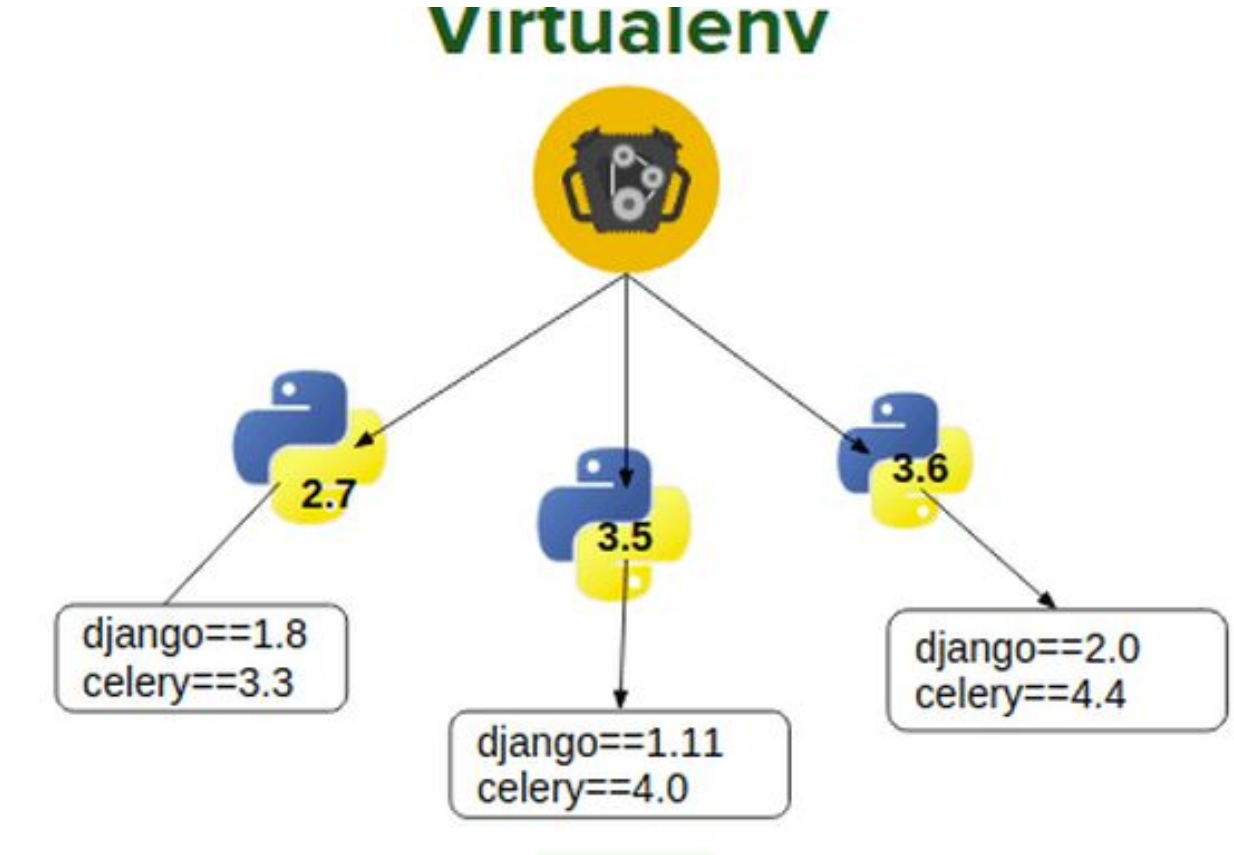
Зависимости

Зависимости

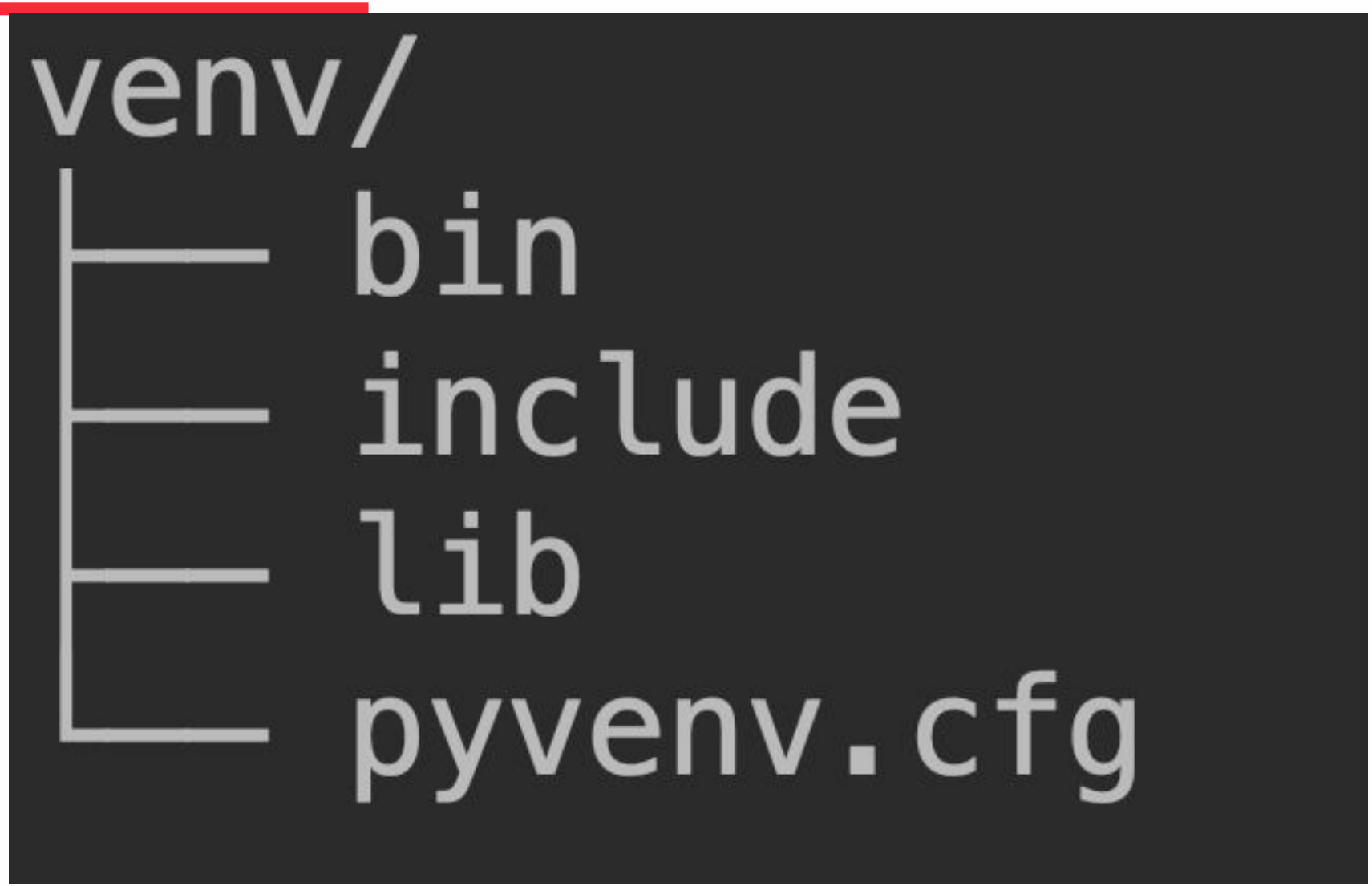
pip install -r requirements.txt

```
sklearn==1.6.0
dataclasses=0.8
pyyaml==3.11
marshmallow-dataclass==8.3.0
torch>=1.5
```

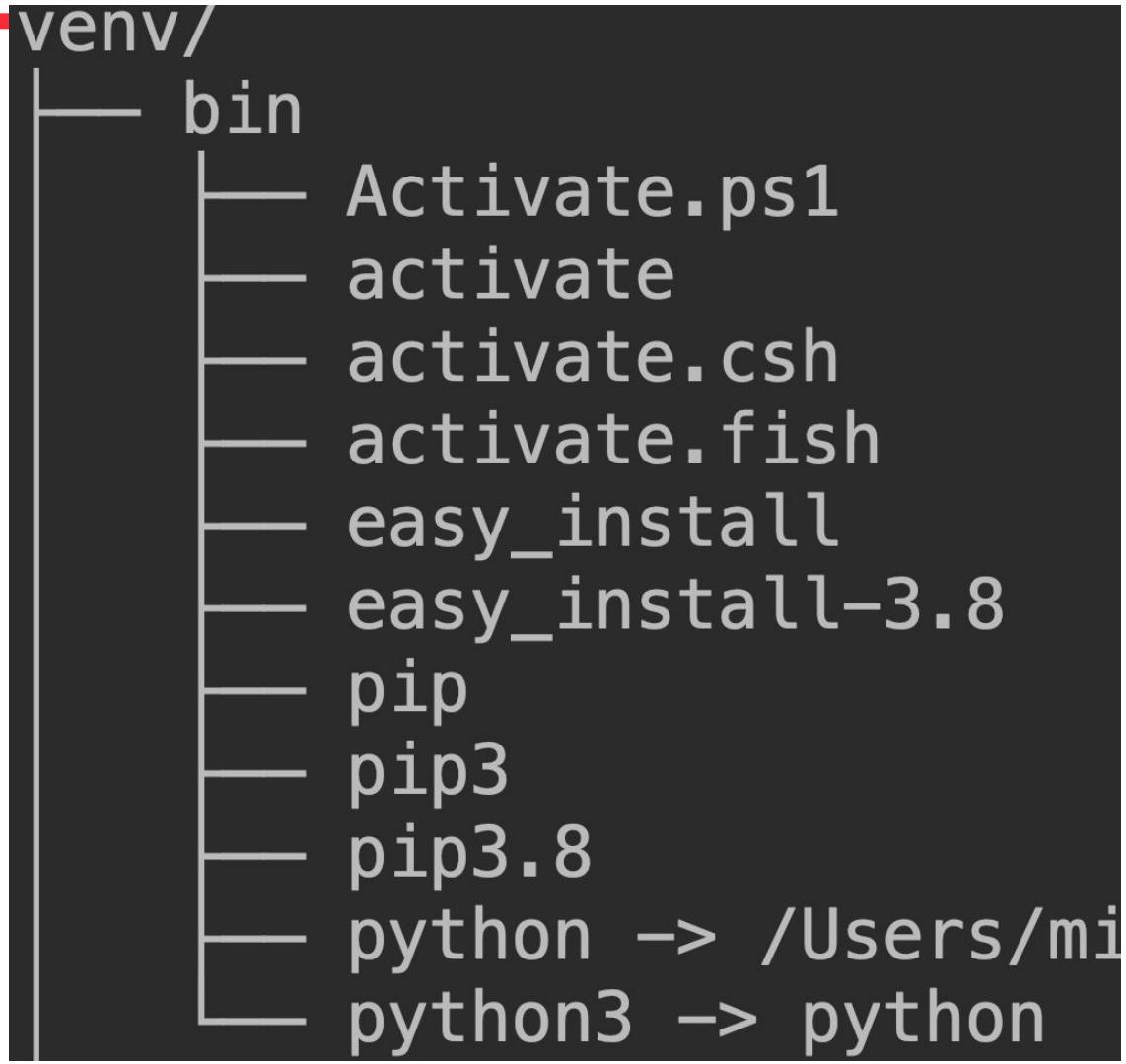
Виртуальные окружения



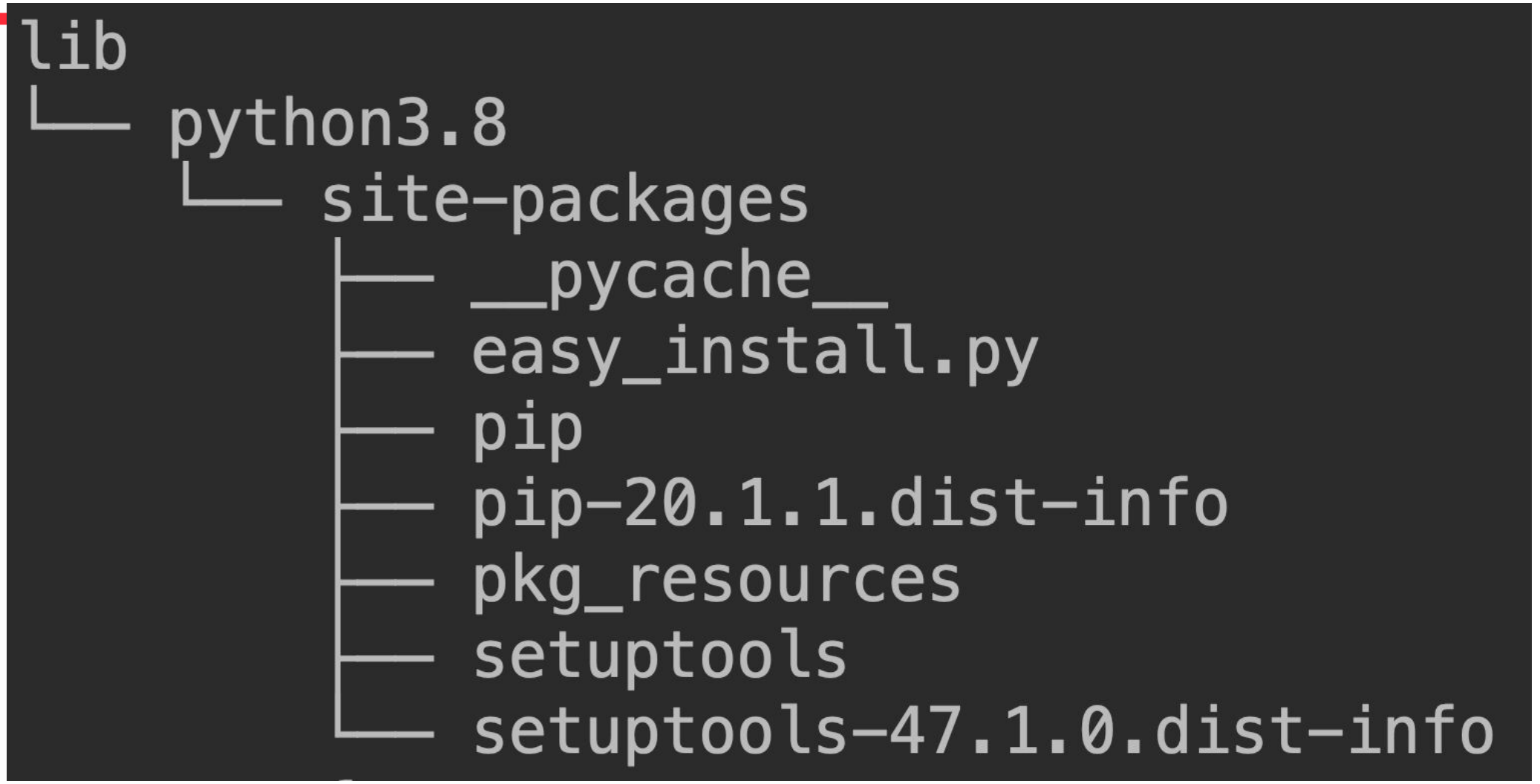
VENV структура



VENV структура - bin

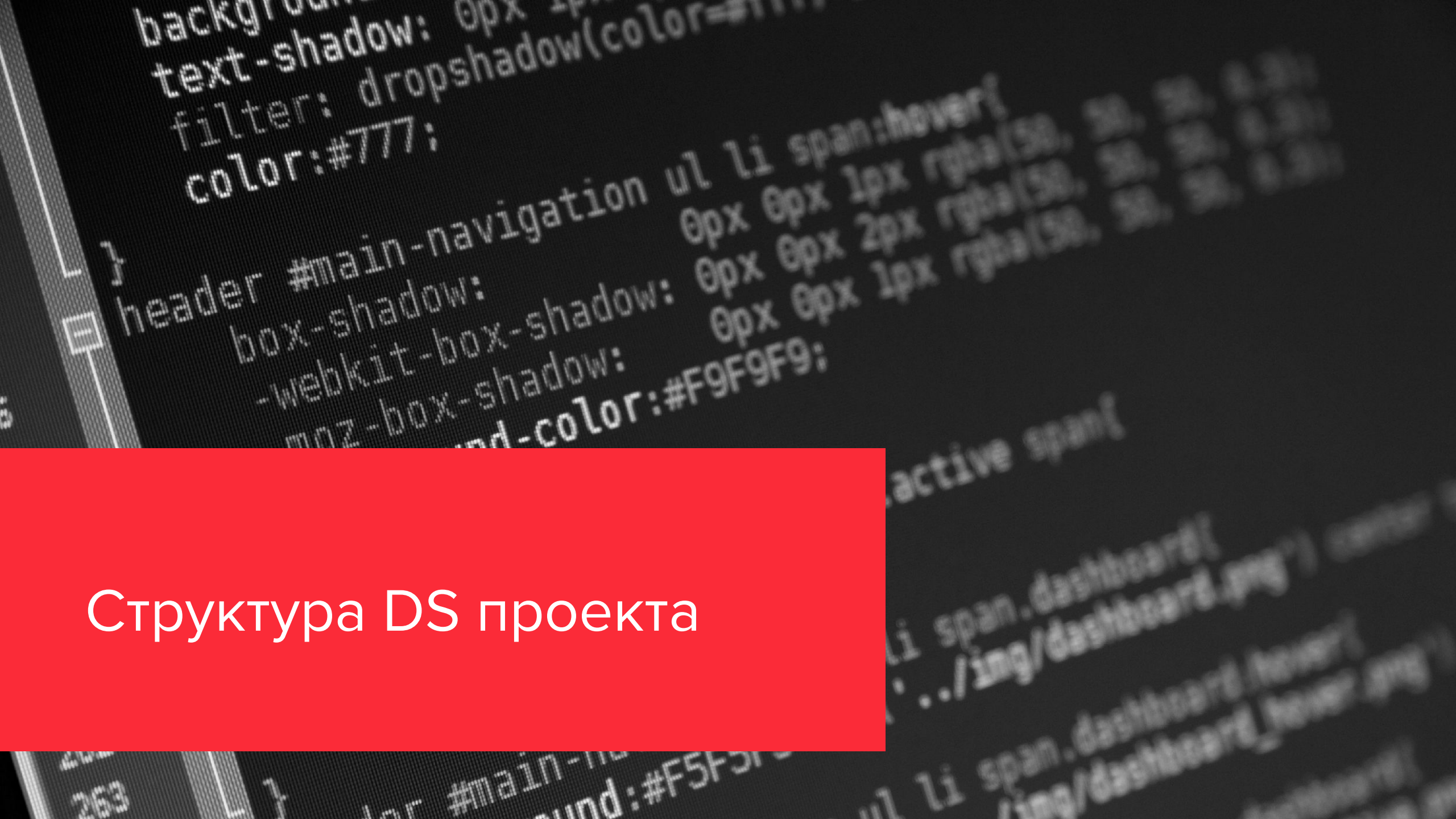


VENV структура - site-packages



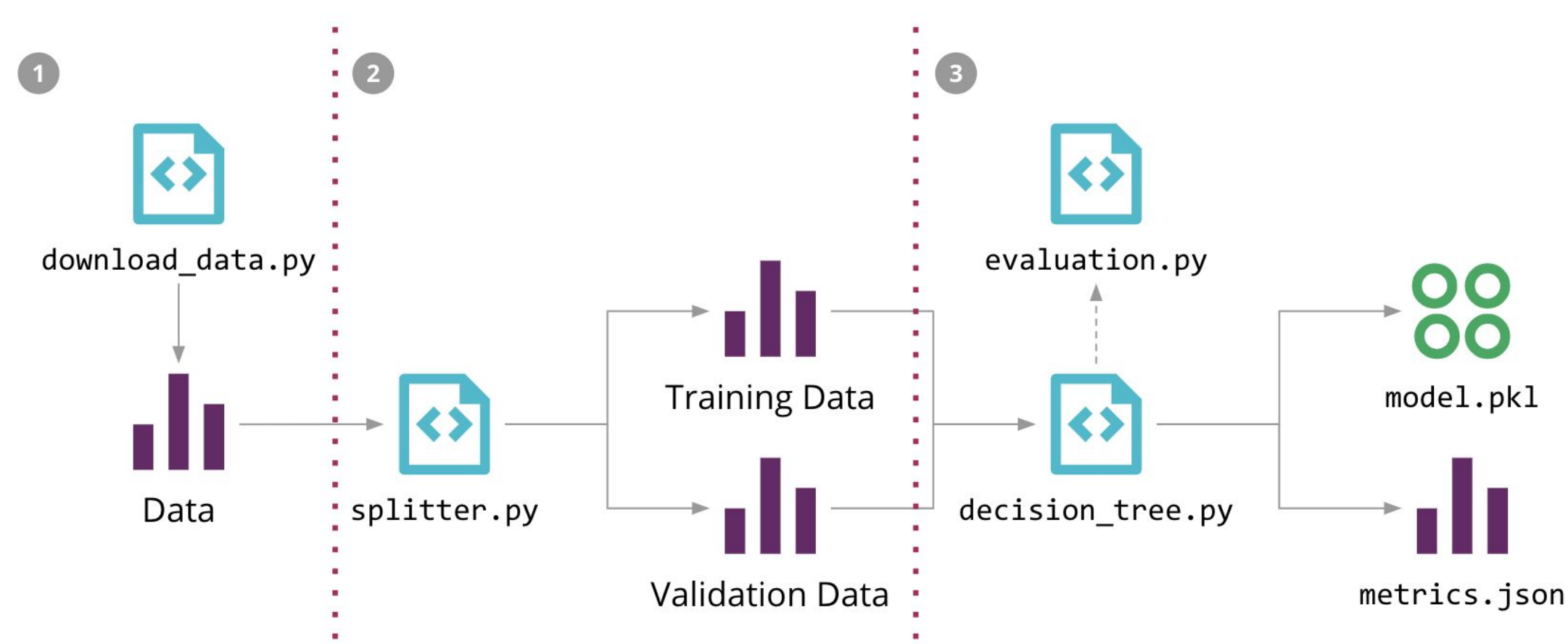
Инструменты для контроля виртуальных окружений

- **virtualenv**
- **virtualenvwrapper**
- **conda**
- **pyenv**
- **poetry**



Структура DS проекта

ML проект

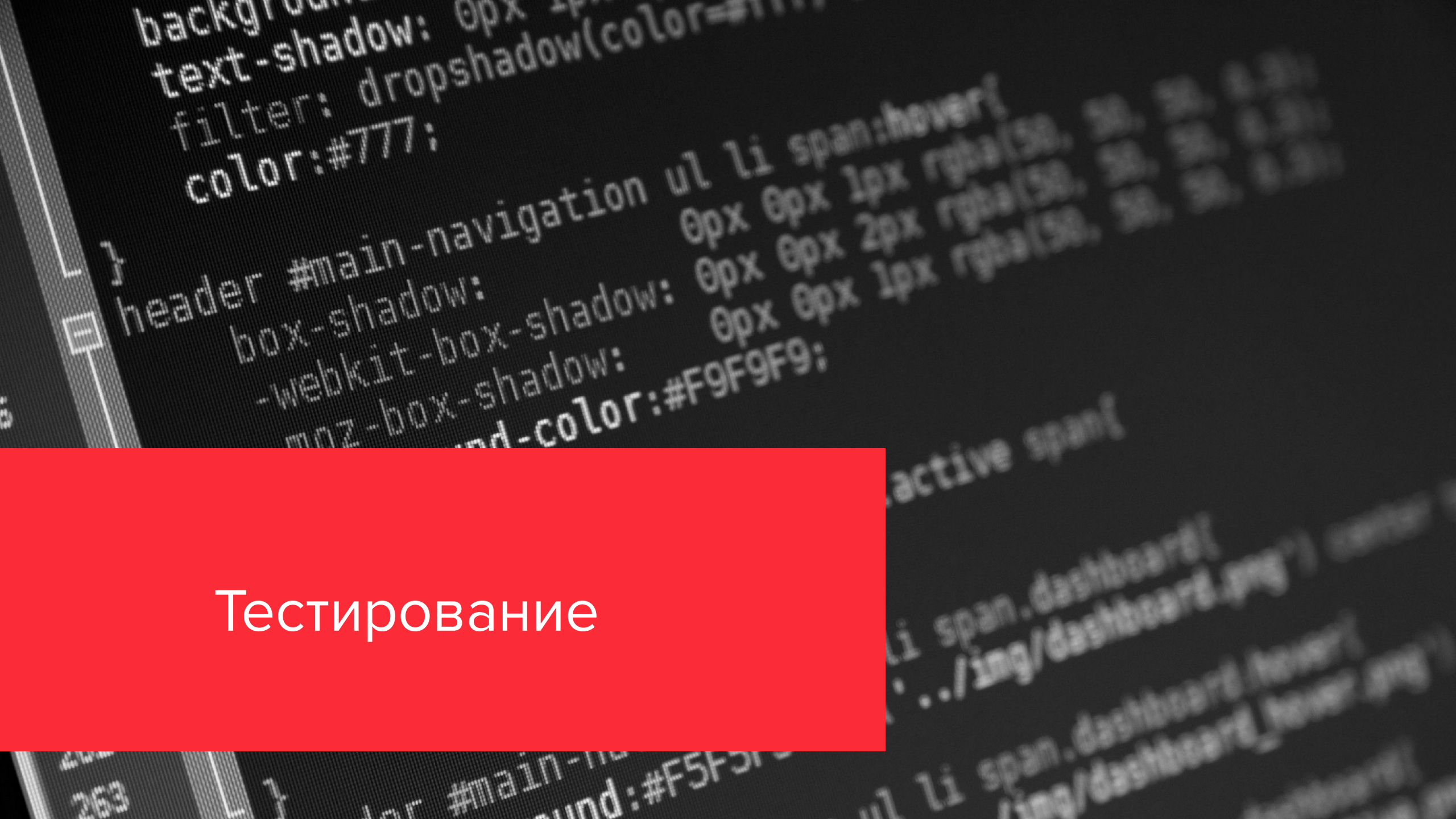

1
2
3
download_data.py
Data
splitter.py
Training Data
Validation Data
evaluation.py
decision_tree.py
model.pkl
metrics.json

Cookiecutter DS template

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
```

Cookiecutter DS template

```
|
├── setup.py           <- Make this project pip installable with `pip install -e`
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```



Тестирование

Зачем тесты?

```
# test_capitalize.py

def capital_case(x):
    return x.capitalize()

def test_capital_case():
    assert capital_case('semaphore') == 'Semaphore'
```

1. Код без тестов -- это сломанный код
2. Даже если всё работает сейчас, вы сломаете его завтра. Без тестов вы не узнаете об этом.
3. Заставляет вас писать код лучше
4. Легче владеть кодом коллективно

Статья про тестирование в ML

1) Pre-train tests to ensure correct implementation

2) Post-train tests to ensure expected learned behaviour

https://towardsdatascience.com/testing-ml-77cb2089317

Доклады про тестирование