Tinkoff Python

Лекция 3

Сеть. Простое веб-приложение

Пара слов обо мне

занимаюсь чат-ботами

Палий Антон

Тест

С чего начинается веб-запрос?

Пара слов про транспортный уровень

TCP/UDP

TCP	**UDP**
Transmission	User
Control	Datagram
Protocol	Protocol

TCP

UDP

Прикладной уровень

DNS

tinkoff.ru -> 178.248.236.218

HTTP

Hypertext Transfer Protocol

```
~$ telnet google.com 80
Trying 64.233.165.139...
Connected to google.com.
Escape character is '^]'.
GET /
HTTP/1.0 200 OK
Date: Sat, 22 Feb 2020 18:19:03 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-
```

```
~$ telnet tinkoff.ru 80
Trying 178.248.236.218...
Connected to tinkoff.ru.
Escape character is '^]'.
GET /
Connection closed by foreign host.
```

Что же пошло не так?

HTTPS

HTTP, в котором
данные шифруются

Заголовки

key: value

Пара ключ-значения для передачи дополнительной инфомации между клиентом и сервером

1. General Headers— есть в любом сообщении клиента и сервера.
2. Request Headers — только в запросах клиента.
3. Response Headers — только для ответов от сервера.
4. Entity Headers— сопровождают каждую сущность сообщения.

404

слайд не найден

200

https://httpstatuses.com

- 1×× Informational
- 2×× Success
- **200** OK
- 3×× Redirection
- **301** Moved Permanently
- **307** Temporary Redirect
- **308** Permanent Redirect

- 4×× Client Error
- **400** Bad Request
- **401** Unauthorized
- **403** Forbidden
- **404** Not Found
- **405** Method Not Allowed
- **418** I'm a teapot
- 5×× Server Error
- **500** Internal Server Error
- **501** Not Implemented
- **502** Bad Gateway
- **503** Service Unavailable
- **504** Gateway Timeout

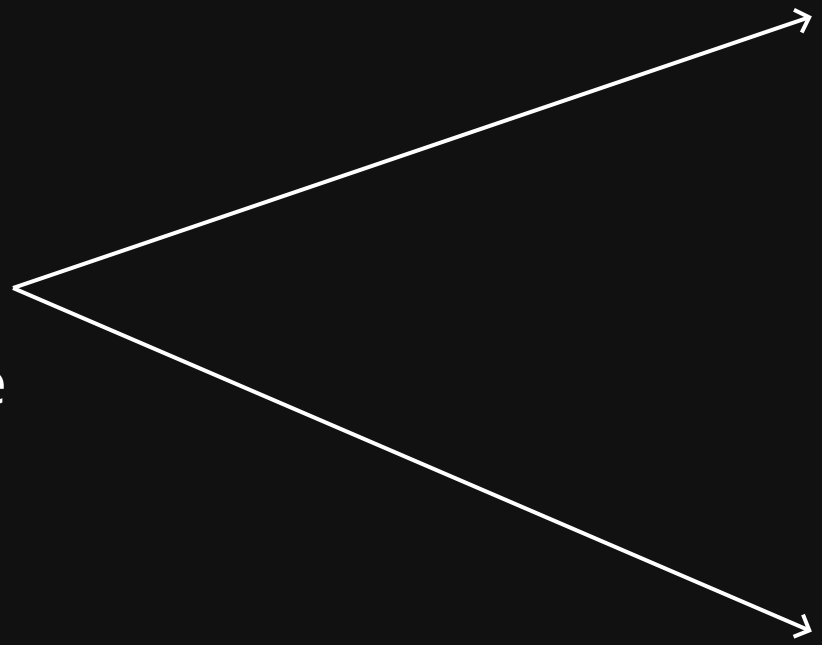
```
from http import HTTPStatus
```

Как из программы получить доступ к сетевому интерфейсу?

Socket

Общее

- Создание
- Отправка
- Получение
- Закрытие



Сервер

- Бинд
- Слушать
- Подтвердить

Клиент

- Подключиться

```python
import socket


HOST = '127.0.0.1'
PORT = 1025

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

```
~$ telnet localhost 1025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /
GET /
```

Программное выполнение запросов

curl

Консольная утилита для выполнения запросов.

```
~$ curl -i tinkoff.ru
HTTP/1.1 301 Moved Permanently
Server: QRATOR
Date: Mon, 24 Feb 2020 16:03:37 GMT
Content-Length: 0
Connection: keep-alive
Keep-Alive: timeout=15
Location: https://tinkoff.ru/
```

Позволяет	**Не позволяет**
Делать http запросы	работать из python
Подключаться по https	
Удобный для консоли синтаксис	
Действие в одну команду	

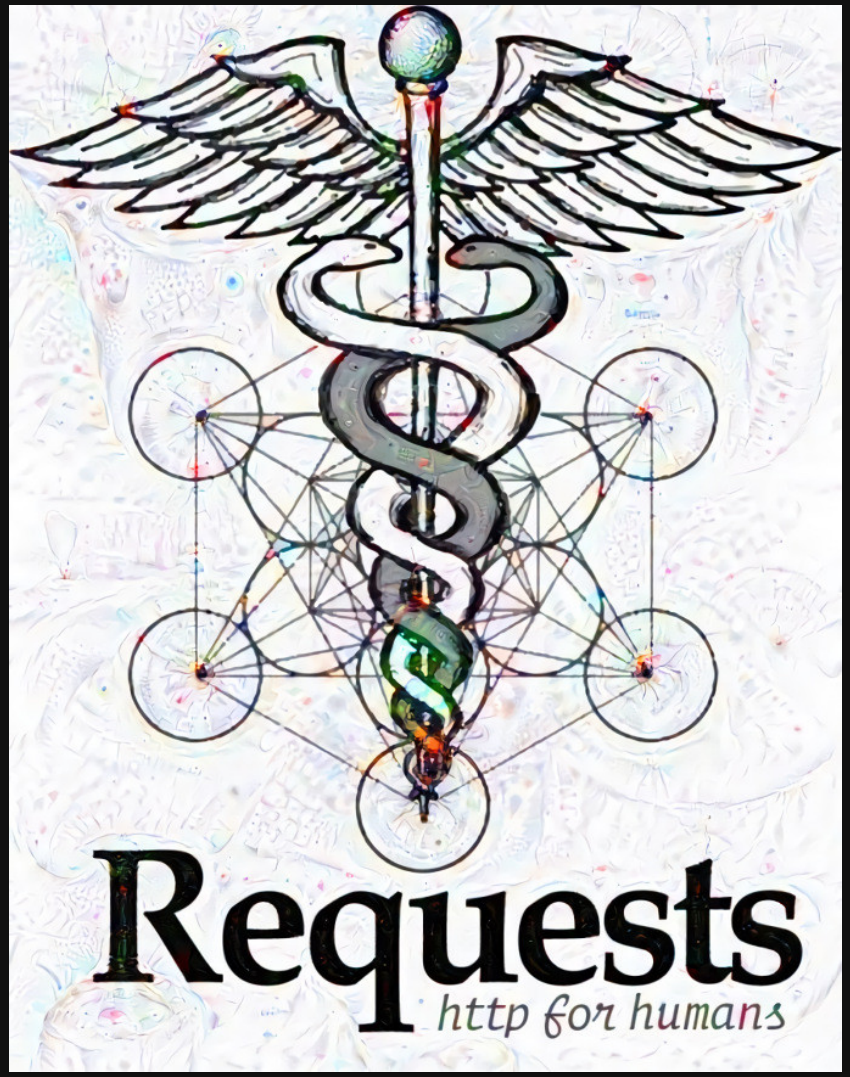
urllib.request

Стандартное решение из коробки

```
In [1]: import urllib

In [2]: f = urllib.request.urlopen('https://tinkoff.ru/')

In [3]: f
Out[3]: <http.client.HTTPResponse at 0x7f461c847d50>

In [4]: print(f.read(100).decode('utf-8'))
<!DOCTYPE html>
<html class="no-js" lang="ru">
<head>
<meta charset="UTF-8">
<title data-meta-dynami
```

При работе имеет множество ньюансов, да и в своей же документации рекомендует использовать

Requests



```
In [1]: import requests

In [2]: from http import HTTPStatus

In [3]: r = requests.get('https://tinkoff.ru/')

In [4]: r.status_code
Out[4]: 200

In [5]: assert r.status_code == HTTPStatus.OK

In [5]: r.headers
Out[5]: {'Date': 'Tue, 25 Feb 2020 08:30:31 GMT', 'Content-Type'

In [6]: r.text
Out[6]: '<!DOCTYPE html>\n<html class="no-js" lang="ru">\n<head>'
```

timeout

всегда указывайте явно

```
In [1]: import requests

In [2]: r = requests.get('https://tinkoff.ru/')

In [3]: r
Out[3]: <Response [200]>

In [3]: r = requests.get('https://tinkoff.ru/', timeout=0.01)
---------------------------------------------------------------------------
timeout                                   Traceback (most recent call last)
```

Перерыв

Объединим сеть и сокет

Выберем фреймфорк

- Django
- Flask
- Sanic
- FastAPI
- Bottle
- Falcon

Flask

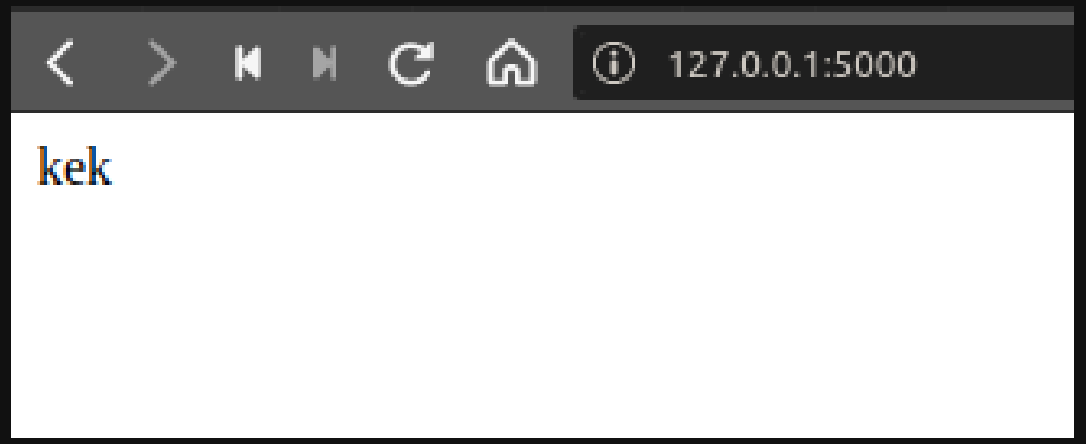
web development,
one drop at a time

Напишем простое приложение

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def main():
    return "kek"

```

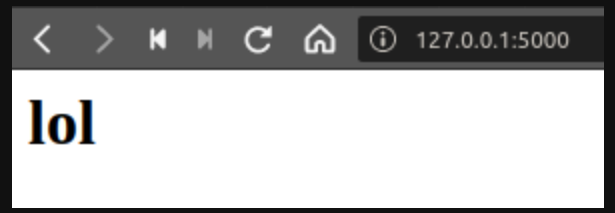
```
FLASK_APP=app.py PYTHONPATH=$(pwd)/app_module flask run
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to us
 * Serving Flask app "app.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

```
telnet localhost 5000
Connected to localhost.
Escape character is '^]'.
GET /

kekConnection closed by foreign host.
```


127.0.0.1:5000
kek

Добавим немного красоты

```python
# добавим h1 тег к нашему тексту
from flask import Flask

app = Flask(__name__)

@app.route('/')
def main():
    return "<h1>lol</h1>"
```



Тесты

```python
import pytest

from app import app


@pytest.fixture
def client():
    app.config['TESTING'] = True
    with app.test_client() as client:
        yield client


def test_connection(client):
    result = client.get('/')
    assert result.status == '200 OK'
    assert result.data == b'<h1>lol</h1>'
```

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def main():
    return "kek"

```

```
FLASK_APP=app.py PYTHONPATH=$(pwd)/app_module flask run
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to us
 * Serving Flask app "app.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

WSGI

Стандарт взаимодействия для веб-серверов типа nginx и веб-приложений на языке python.

Является низкоуровневым описанием работы приложения.

Pep 3333

- должно быть вызываемым (callable) объектом (обычно это функция или метод)
- принимать два параметра:
  - словарь переменных окружения (environ)
  - обработчик запроса (start_response)
- вызывать обработчик запроса с кодом HTTP-ответа и HTTP-заголовками
- возвращать итерируемый объект с телом ответа

```
1  HELLO_WORLD = b"Hello world!\n"
2
3
4  def simple_app(environ, start_response):
5      """Simplest possible application object"""
6      status = '200 OK'
7      response_headers = [('Content-type', 'text/plain')]
8      start_response(status, response_headers)
9      return [HELLO_WORLD]
10
11
12 if __name__ == '__main__':
13     from wsgiref.simple_server import make_server
14     srv = make_server('localhost', 8080, simple_app)
15     srv.serve_forever()
```

Gunicorn

Web Server Gateway Interface HTTP server

Gunicorn решает несколько проблем

1. Запуск приложения
2. Масштабирование приложения
3. Конфигурирование приложения
4. Балансировка трафика
5. Контроль приложения (Перезагрузки, размер очереди и пр)
6. SSL
7. Обработка сигналов
8. Работа с веб серверами
9. и пр

```
gunicorn \
  --workers 4 \
  --timeout 2000 \
  --max-requests 100000 \
  --bind 0.0.0.0:5000 \
    app_module:app()
```

Напишем приложение посложнее

HTML

```python
from uuid import uuid4

from flask import Flask

app = Flask(__name__)
response_base_body = """<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=
    <title>Document</title>
</head>
<body>
    {}
</body>
</html>"""

@app.route('/')
def main():
    return response_base_body.format(uuid4())
```

Теперь мы отдаём правильный HTML, но красивый ли код?

Файлы?

Модули?

Шаблоны!

jinja

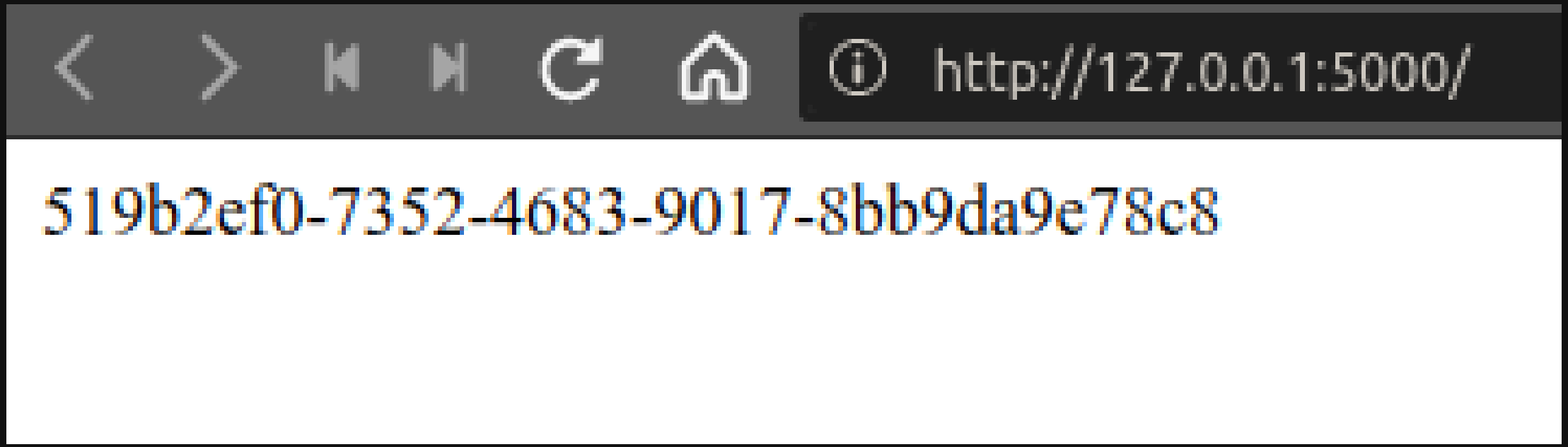
Jinja позволяет

- Автоматизировать работу с шаблонами
- Автоматически избегать XSS инъекции
- Наследовать шаблоны
- Шаблозизировать
- Избавлять код от html

```
project
    app.py
templates
    base.html
```

```python
from uuid import uuid4

from flask import Flask, render_template

server = Flask('some_name')

@server.route('/')
def main():
    return render_template('base.html', data=uuid4())
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>Document</title>
</head>
<body>
        {{ data }}
</body>
</html>
```


http://127.0.0.1:5000/
519b2ef0-7352-4683-9017-8bb9da9e78c8

Что-то сложнее

id	data
0	70757c53-0130-4d40-b087-faba5c813829
1	450eb1b9-1fef-4820-8277-8b70d73f3a29
2	80b1e683-ac3b-4871-805d-ad19f18ff71d
3	7cd2c5a4-529e-40af-a2c4-e443f7774e39
4	bca9a317-18d1-4cf2-bad9-8eafce221571
5	4baabcdd-c1a0-4c18-b07e-2f72aa374352
6	66f6ecd8-fa13-4425-a7ba-fa065d08eca1
7	521945be-970d-4dbf-bfdb-54d44bbcf22b
8	4d9db140-b6f6-494d-8f98-cf48bf136a79
9	35bedfc1-b1bc-4cd7-8500-1a1f71c7d702
10	9dcfe59d-bf45-49d0-b5c7-2c58ab3ba04b
11	00ca6469-112f-4895-b2f9-95b11ba93fc3
12	c1eb928f-3830-43c3-9c27-c466a8422ba6
13	36dceb58-0fe8-4b9b-bb11-319db1931ff9
14	ce3c880c-c8e3-484b-b9b8-b5ce1c4c24a1
15	85b0dbff-ef95-4792-9f93-fe3f32c9a675
16	3968bc85-cdd0-40ed-8181-e08ef1434946
17	05d258e1-f6c9-416b-9bbc-2073d3923846
18	cdb889a5-d79e-4ab0-bb5e-25ae341ef89c
19	7a78402f-35c3-4818-863f-66c0f92484ab
20	ab4d965d-6116-4459-b183-35160adeef67
21	fccf2766-4d62-4d07-bc6b-65cdbbc229fa
22	7a9410c5-5e84-4224-970f-a222c1670461
23	cad54244-b9d8-42ec-ba3d-25ac6552e8b7
24	9bc81a93-26b8-4697-b798-3d1c68f37cf6
25	95b1d922-400b-43a3-babe-975b5a398e2b
26	3b2cf17b-c461-4a00-93ae-eb8f7230bf0a
27	c0ce065e-d9ef-4416-ae13-9e342a2e9b73
28	492205cc-fad7-49ea-adf2-18d56552e72a
29	bec8f1c6-eec5-405a-b96b-67bfab6013c0
30	fa32e128-dc41-42f1-af85-3301c3d1e9bf
31	6ca0a147-fa5e-4b57-8380-c35ca9f8758f

```
project
└── app.py
templates
└── base.html
```

```python
@server.route('/')
def main():
    users = {}
    for i in range(100):
        users[i] = uuid4()
    return render_template('base.html', users=users)
```

```html
<body>
  <table style="width:100%">
    <tr>
      <th>id</th>
      <th>data</th>
    </tr>
    {% for user_id, user_name in users.items() %}
    <tr>
      <th> {{ user_id }} </th>
      <th> {{ user_name }} </th>
    </tr>
    {% endfor %}
  </table>
</body>
```

Основные разделители в jinja

```
{% ... %} Выражения

{{ ... }} Вывод значения

{# ... #} Комментарии
```

Выражения

```
{% for var in vars %}

{% endfor %}

{% if a == '' %}

{% endif %}

{% macro input(name) -%}
    <div> "{{ name }}" </div>
{%- endmacro %}

{% extends "base.html" %}

{% block title %}some data{% endblock %}
```

Вывод значения

```
1  {{ name }}
2
3  {{ 1 + 1 }}
4
5  {{ var is some_test }}
6  {% if variable is defined %}
7
8  {{ name | filter }}
9  {{ my_variable|default('kek') }}
```

Линтеры

Flake8

```
$ flake8 path/to/code/
app/__init__.py:1:1: W391 blank line at end of file
app/utils.py:7:2: E225 missing whitespace around operator
app/utils.py:10:1: E402 module level import not at top of
file
app/utils.py:10:1: F401 'os' imported but unused
app/__main__.py:1:1: F401 '.utils' imported but unused
```

https://pypi.org/project/flake8/

Flake8 plugins

- flake8-builtins
- flake8-comprehensions
- flake8-eradicate
- flake8-isort
- flake8-logging-format
- flake8-pytest
- pep8-naming
- etc.

Pylint

```
$ pylint path/to/code
************* Module app.utils
app/utils.py:7:1: C0326: Exactly one space required around assignment
x=2
 ^ (bad-whitespace)
app/utils.py:1:0: C0111: Missing module docstring (missing-docstring)
app/utils.py:5:0: C0103: Constant name "x" doesn't conform to
UPPER_CASE naming style (invalid-name)
app/utils.py:7:0: C0103: Constant name "x" doesn't conform to
UPPER_CASE naming style (invalid-name)
app/utils.py:10:0: C0413: Import "import os" should be placed at the
top of the module (wrong-import-position)
app/utils.py:10:0: W0611: Unused import os (unused-import)
```

https://www.pylint.org

Отключение проверок

```
# my_strange_module.py
# pylint:disable=missing-docstring
import os # noqa: F401
```

Настройки линтеров

setup.cfg

```
[flake8]
enable-extensions = G
exclude = .git
ignore =
    A003 ; 'id' is a python builtin, consider renaming the class attribute
    W504 ; Line break occurred after a binary operator
max-complexity = 20
max-line-length = 80
show-source = true

[pylint]  ; `pylint --rcfile=setup.cfg my_code`
good-names=i,j,k,e,x,_,pk,id
max-module-lines=300
output-format = colorized

disable=
    C0103, ; Constant name "api" doesn't conform to UPPER_CASE naming style (invalid-name)
    C0111, ; Missing module docstring (missing-docstring)
    C0330, ; Wrong hanging indentation before block (add 4 spaces)
    E0213, ; Method should have "self" as first argument (no-self-argument) - N805 for flake8
    R0201, ; Method could be a function (no-self-use)
    R0901, ; Too many ancestors (m/n) (too-many-ancestors)
    R0903, ; Too few public methods (m/n) (too-few-public-methods)
    W0511, ; TODO needed? (fixme)
```

Автоформатирование кода

Black

```
$ black app
```

```
# BEFORE
x = {  'a':37,'b':42,

'c':927}

y = 'hello ''world'
z = 'hello '+'world'
a = 'hello {}'.format('world')
class foo  (      object  ):
  def f    (self    ):
    return          37*-+2
  def g(self, x,y=42):
      return y
def f  (    a ) :
  return        37+-+a[42-x :  y**3]
```

```
# AFTER
x = {"a": 37, "b": 42, "c": 927}

y = "hello " "world"
z = "hello " + "world"
a = "hello {}".format("world")


class foo(object):
    def f(self):
        return 37 * -+2

    def g(self, x, y=42):
        return y


def f(a):
    return 37 + -+a[42 - x : y ** 3]
```

https://github.com/ambv/black

Isort

```
$ isort --apply --recursive app
...
```

https://github.com/timothycrosley/isort

```
from my_lib import Object

print("Hey")

import os

from my_lib import Object3

from my_lib import Object2

import sys

from third_party import libA
import sys

from third_party import libB

print("yo")
```

```
import os
import sys

from third_party import libA, libB

from my_lib import Object, Object2, Object3

print("Hey")
print("yo")
```

Работа с зависимостями

Poetry

```
~/src/sdispater/demo
❯❯❯ poetry install
```