Programowanie aplikacji webowych w Pythonie

Wrocław, 13 maja 2015

Piotr Wasilewski

email: info@piotrek.io

www: http://piotrek.io

Pyramid™

Bottle

WEB2PY

Tornado

python™

django

web.py


Pyramid™


Bottle

WEB2PY

Tornado


WERKZEUG


python™


django

web.py

TURBO
GEARS

Python – w skrócie

- www.python.org
- najnowsza wersja: 3.4.3 (CPython)
- dokumentacja: docs.python.org/3/
- Package Index: pypi.python.org/pypi
- pip: pip.pypa.io/en/stable/
- Zen Of Python:
  www.python.org/dev/peps/pep-0020/

$3 > 2$

Python - podstawy

- typy
- funkcje
- dektoratory
- instrukcje if i for
- klasy
- dziedziczenie
- wyjątki
- importowanie, moduły i paczki

Wcięcia!

```
if x > 9000:
    print("It's over nine thousand!")
```

```
>>> from __future__ import braces
  File "<stdin>", line 1
SyntaxError: not a chance
```

REPL (read-eval-print loop)

```
$ python3
Python 3.4.3 (default, May 10 2015, 20:27:16)
[GCC 4.6.3] on linux
Type "help", "copyright", "credits" or "license"
for more information.
>>> print("Hello, World!")
Hello, World!
>>>
```

Zobacz też:

- ipython
- bpython

Typy

```
>>> type("Hello, World!")
<class 'str'>
>>> type(2015)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> type(1+2j)
<class 'complex'>
>>> type([1, 2, 'three', [4]])
<class 'list'>
>>> type(('a', 'b', 'c'))
<class 'tuple'>
>>> type({1, 2, 3})
<class 'set'>
>>> type({'a': 1, 2: 'b'})
<class 'dict'>
```

Typowanie, zarządzanie pamięcią

```
>>> a = 123
>>> b = a
>>> id(a), id(b)
(8756800, 8756800)
```

Typowanie, zarządzanie pamięcią

```
>>> a = 123
>>> b = a
>>> id(a), id(b)
(8756800, 8756800)
```

```
>>> a = "parrot"
>>> a, b
('parrot', 123)
>>> id(a), id(b)
(140207744565744, 8756800)
```

Referencje i garbage collector

```
>>> a = "Monty Python and the Holy Grail"
>>> sys.getrefcount(a)
2
>>> b = a
>>> sys.getrefcount(a)
3
>>> del b
>>> sys.getrefcount(a)
2
```

Referencje i garbage collector

```
>>> a = "Monty Python and the Holy Grail"
>>> sys.getrefcount(a)
2
>>> b = a
>>> sys.getrefcount(a)
3
>>> del b
>>> sys.getrefcount(a)
2
```

```
>>> sys.getrefcount(3)
48
```

Funkcje

```
def say_hello():
    print("Hello!")
```

```
>>> say_hello()
Hello!
```

Funkcje – argumenty i zwracanie

```
def add(a, b, c):
    return a + b + c
```

```
>>> x = add(3, 5, 10)
>>> print(x)
18
```

Funkcje – argumenty i zwracanie

```
def add(a, b, c):
    return a + b + c
```

```
>>> x = add(3, 5, 10)
>>> print(x)
18
```

```
def add(a, b, c=10):
    return a + b + c
```

```
>>> x = add(3, 5)
>>> print(x)
18
```

Funkcje – args

```
def mean(*args):
    print(args)
    return sum(args) / len(args)
```

```
>>> x = mean(1, 2, 3, 4, 5)
(1, 2, 3, 4, 5)
>>> print(x)
3.0
```

```
>>> l = [1, 2, 3, 4, 5]
>>> x = mean(*l)
(1, 2, 3, 4, 5)
```

Funkcje – kwargs

```
def show_message(**kwargs):
    who = kwargs['who']
    msg = kwargs['message']
    print('{}: {}'.format(who, msg))
```

```
>>> show_message(who='me', message='hello!')
me: hello!
```

```
>>> d = {'who': 'you', 'message': 'hi!'}
>>> show_message(**d)
you: hi!
```

Funkcje – args+kwargs

```
def f(a, b, c=0, *args, **kwargs):
    print(a)
    print(b)
    print(c)
    print(args)
    print(kwargs)
```

```
>>> f(1, 2, 3, 4, d=5)
1
2
3
(4,)
{'d': 5}
```

Dekoratory

```
def say_my_name(func):
    def wrapped(*args, **kwargs):
        print(func.__name__)
        return func(*args, **kwargs)
    return wrapped
```

```
@say_my_name
def heisenberg():
    print("You're goddamn right!")
```

```
>>> heisenberg()
heisenberg
You're goddamn right!
```

Zobacz też:
https://docs.python.org/3/library/functools.html#functools.wraps

if

```
if temperature < 10:
    print('Too cold!')
elif temperature > 30:
    print('Too hot!')
else:
    print('OK!')
```

for

```
for element in lista:
    if element > 100:
        break
else:
    print('Wszystkie elementy większe od 100')
```

Klasy

```
class Numbers:
    def __init__(self, numbers):
        self._numbers = numbers

    def total(self):
        return sum(self._numbers)

    def length(self):
        return len(self._numbers)
```

```
>>> n = Numbers([1, 2, 3, 4])
>>> n.total()
10
>>> n.length()
4
```

Dziedziczenie

```
class BetterNumbers(Numbers):
    def multiply(self, x):
        result = []
        for number in self._numbers:
            result.append(number * x)
        return result
```

```
>>> n = BetterNumbers([1, 2, 3, 4])
>>> x = n.multiply(4)
>>> print(x)
[4, 8, 12, 16]
```

Dziedziczenie - super()

```
class EvenBetterNumbers(BetterNumbers):
    def multiply(self, x):
        result = super().multiply(x)
        print(result)
        return result
```

```
>>> n = EvenBetterNumbers([1, 2, 3, 4])
>>> x = n.multiply(4)
[4, 8, 12, 16]
```

Wyjątki

```
def divide(x, y):
    if y == 0:
        raise ValueError('Boom!')
    return x / y
```

```
>>> divide(10, 5)
2.0
>>> divide(10, 0)
Traceback (most recent call last):
  ...
ValueError: Boom!
```

Wyjątki - try-except-else

```
class DeadParrotException(Exception):
    pass
```

```
try:
    raise DeadParrotException(
        'This parrot is no more!')
except DeadParrotException as ex:
    print('I wish to register a complaint:')
    print(ex)
else:
    print('No, it is resting!')
```

Duck typing

„When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."

Duck typing

```
def split_names(full_name):
    try:
        first_name, last_name = full_name.split()
    except AttributeError:
        print('String expected!')
    else:
        return first_name, last_name
```

```
>>> split_names('Jan Kowalski')
('Jan', 'Kowalski')
>>> split_names(123)
String expected!
```

import

```
import os
import os.path
from datetime import timedelta
from itertools import *
from re import match as re_match
```

```
from django import models

class User(models.Model):
    ...
```

Moduły

cheeseshop.py:

```
import datetime

spam = 'Spam! Spam! Spam!'

def opening_hours(day):
    if day.weekday() in (5, 6):
        return None
    else:
        return (
            datetime.time(10, 30),
            datetime.time(18, 0)
        )
```

```
>>> import cheeseshop
>>> cheeseshop.spam
'Spam! Spam! Spam!'
>>> day = datetime.date(2015, 5, 13)
>>> hours = cheeseshop.opening_hours(day)
```

Paczki

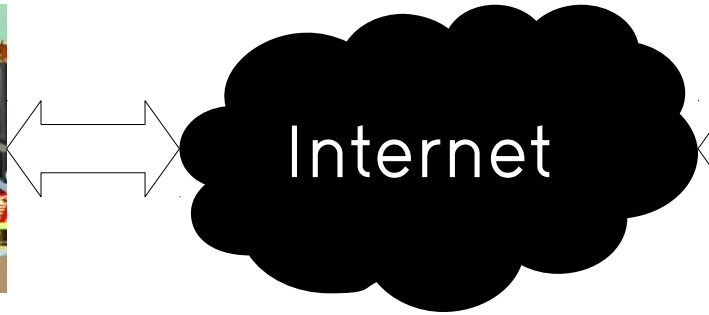
```
.
|-- test.py
|
+-- package
          |-- __init__.py  # opcjonalne w Pythonie 3
          |
          +-- module.py
```

test.py:

```
import package
import package.module
from package import module
from package import variable, Class, other_module
form package.module import variable, Class, other
```

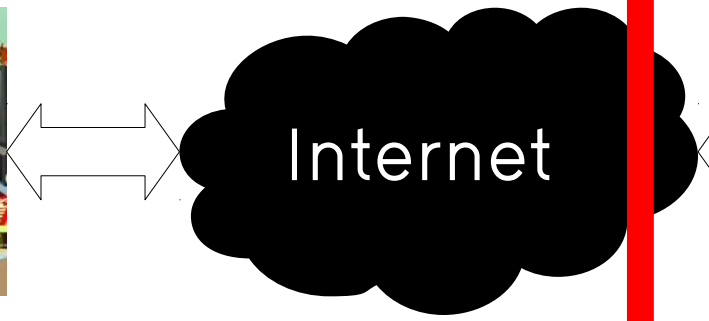
Architektura systemu

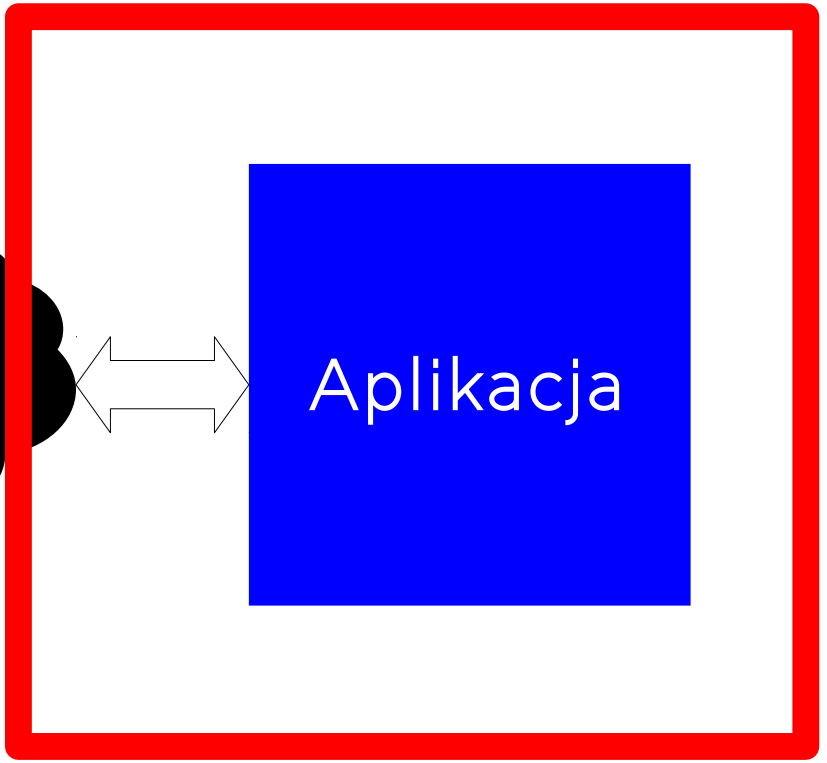
Użytkownik



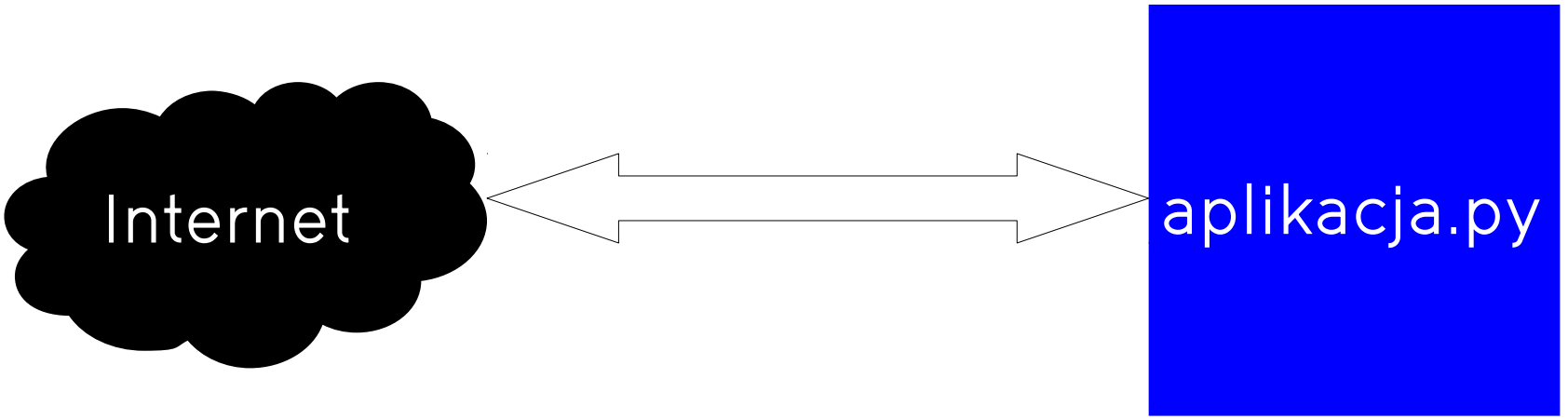
Architektura systemu

Użytkownik

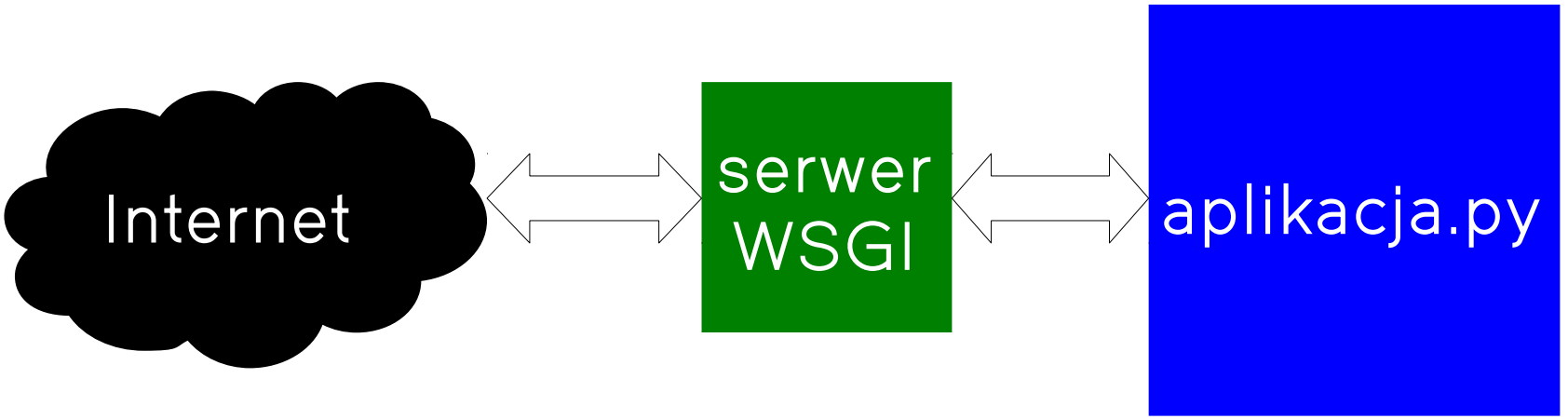




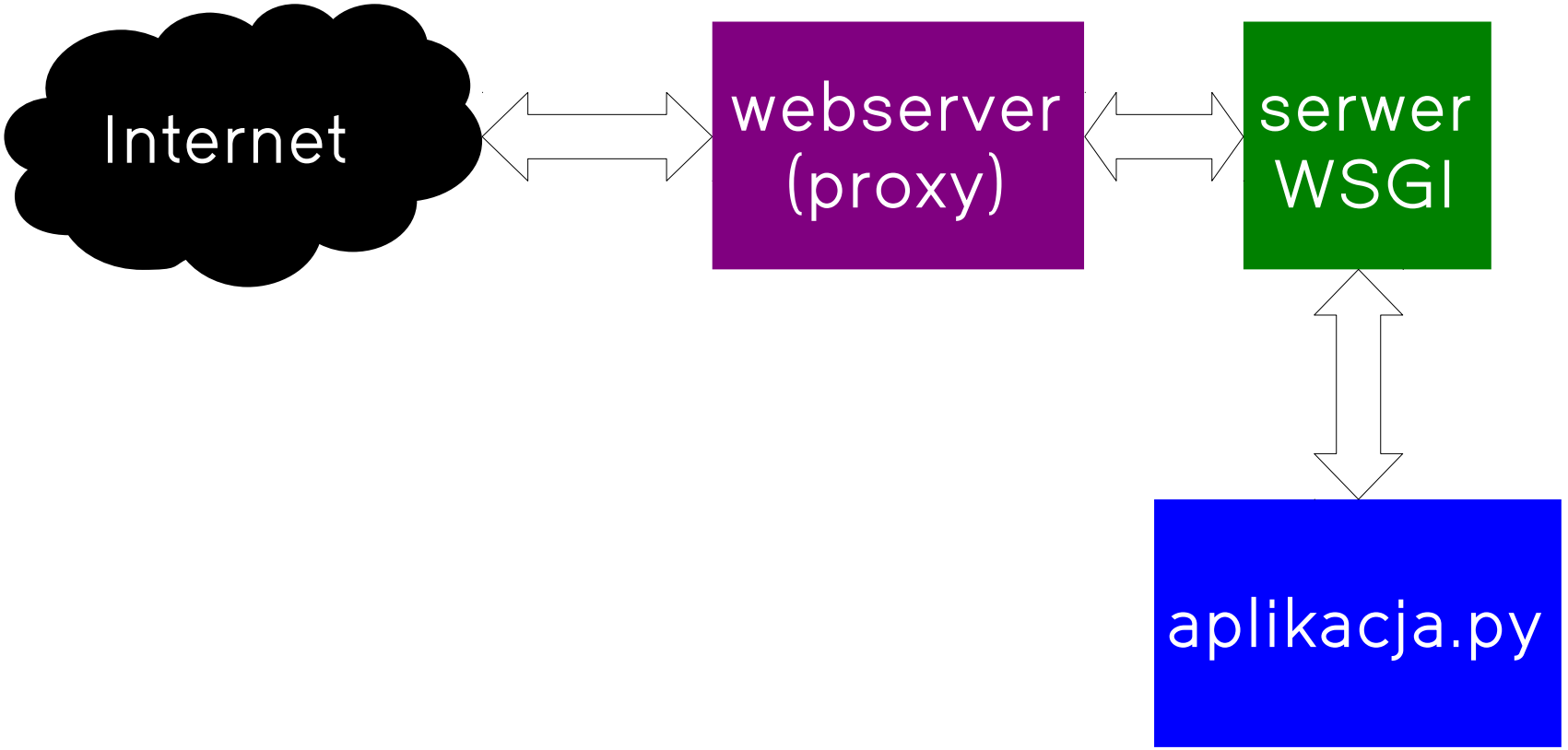
Architektura systemu



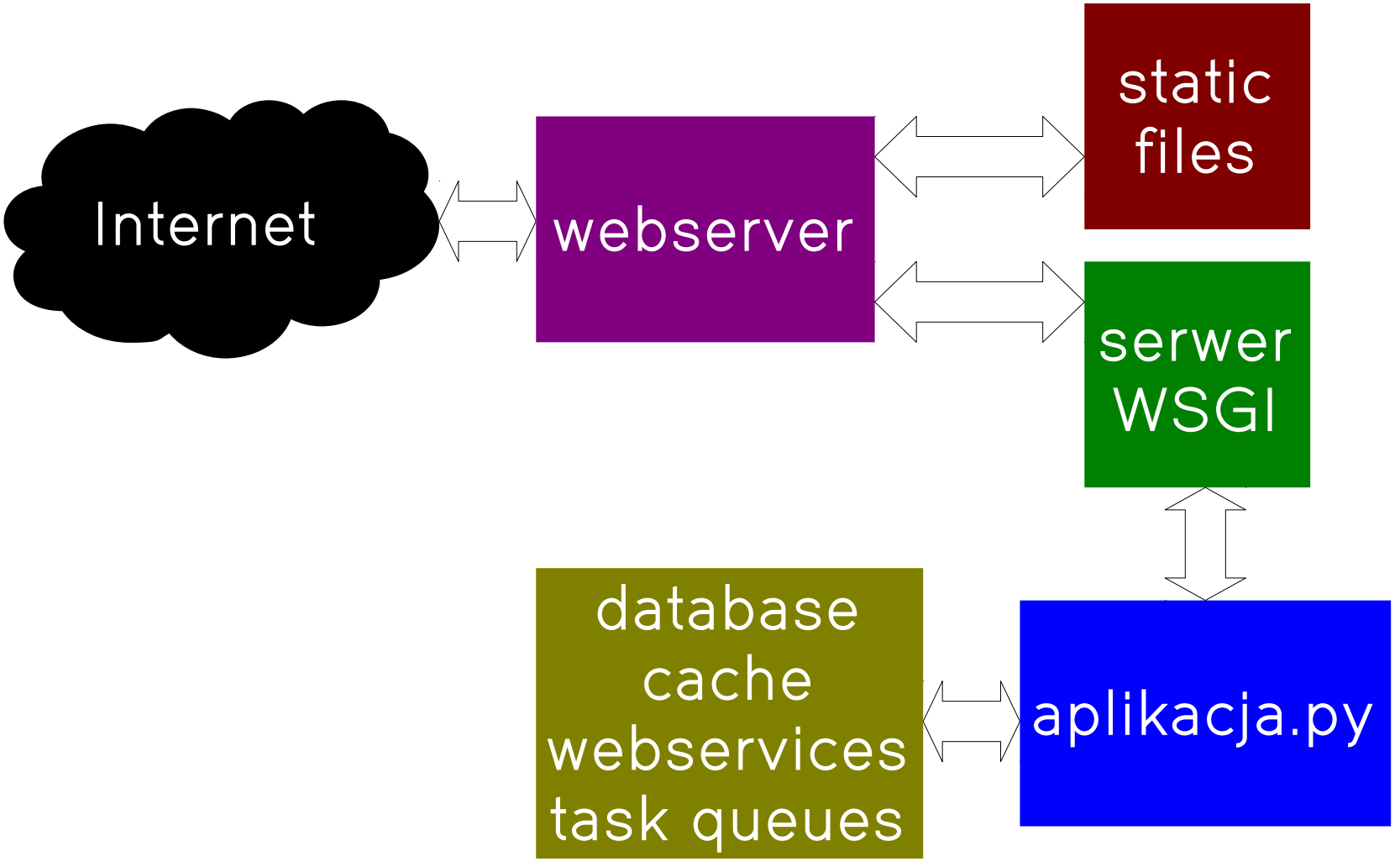
Architektura systemu



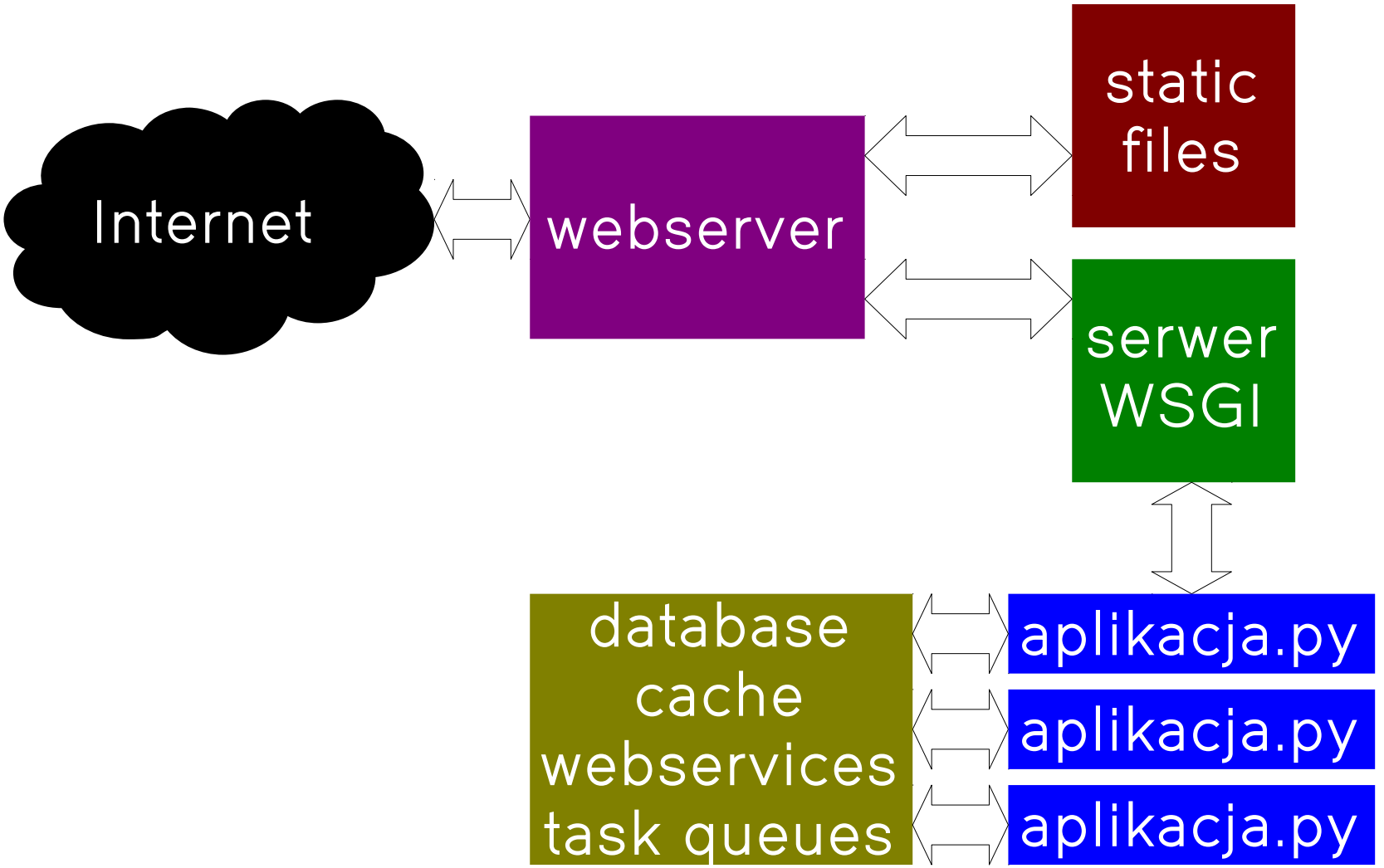
Architektura systemu



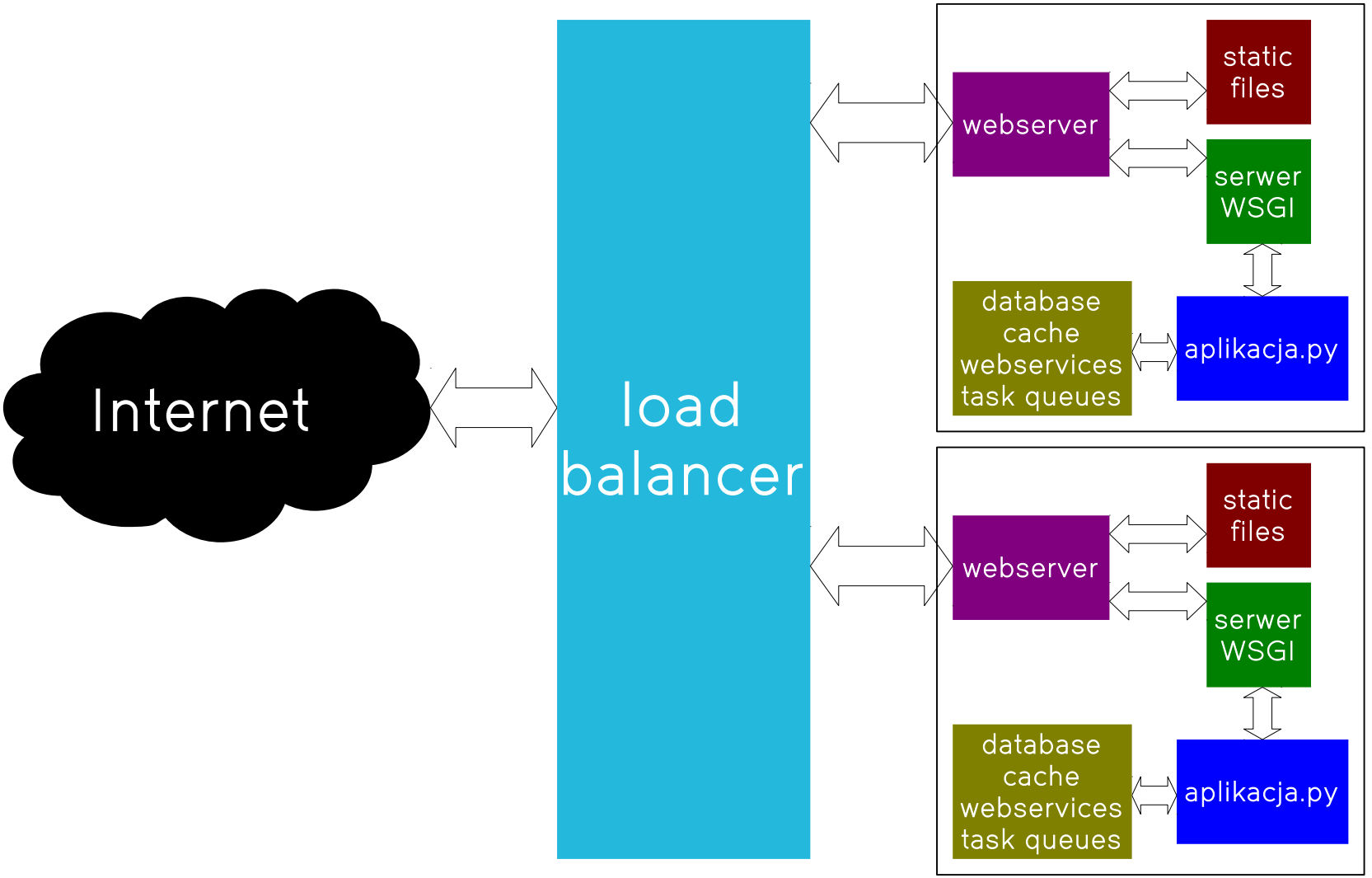
Architektura systemu



Architektura systemu



Architektura systemu



WSGI

„The Web Server Gateway Interface, or WSGI for short, is defined in PEP 333 and is currently the best way to do Python web programming."

https://docs.python.org/3.4/howto/webservers.html

PEP 3333

```
def app(environ, start_response):
    status = '200 OK'
    response_headers = [
        ('Content-type', 'text/html')]
    start_response(status, response_headers)
    return [b'<h1>Hello, World!</h1>']
```

https://www.python.org/dev/peps/pep-3333/

uWSGI

application.py:

```
def app(environ, start_response):
    status = '200 OK'
    response_headers = [
        ('Content-type', 'text/html')]
    start_response(status, response_headers)
    return [b'<h1>Hello, World!</h1>']
```

```
$ uwsgi -w application:app -http 127.0.0.1:3000
```

Dokumentacja:
https://uwsgi-docs.readthedocs.org/en/latest/

Nie tylko uWSGI

- gunicorn
- mod_wsgi
- wsgiref (Python 3)
- ...

http://wsgi.readthedocs.org/en/latest/servers.html

Web framework?

- URL dispatching
- request/response classes
- middlewares
- views
- forms
- templates

Flask

- http://flask.pocoo.org
- http://flask.pocoo.org/docs/0.10/
- microframework
- development server + debugger
- URL dispatching
- templates (Jinja2)
- sessions
- http://flask.pocoo.org/extensions/

Przykładowa aplikacja

- na wzór pastebin.com
- przeglądanie listy wklejonych tekstów
- wklejanie tekstu
- oglądanie wklejonego tekstu
- baza danych: Redis

Django

- http://djangoproject.com/
- https://docs.djangoproject.com/en/1.8/
- URL dispatching
- ORM
- forms
- templates
- django.contrib
- ...

Przykładowa aplikacja

- na wzór pastebin.com
- przeglądanie listy wklejonych tekstów
- wklejanie tekstu
- oglądanie wklejonego tekstu
- baza danych: SQLite
- panel administratora

Literatura - Python

- www.python.org/dev/peps/pep-0008/
- wiki.python.org/moin/GlobalInterpreterLock
- docs.python.org/3/reference/index.html
- docs.python.org/3/library/index.html
- shop.oreilly.com/product/0636920028154.do

Literatura – web development

- wtforms.readthedocs.org/en/latest/
- docs.python-requests.org/en/latest/
- djangopackages.com
- django-rest-framework.org
- tornadoweb.org/en/stable/

Pytania

Dziękuję za uwagę