Kurs rozszerzony języka Python

Wykład 7.

Marcin Młotkowski

17 listopada 2023

Plan wykładu

1. Korzystanie z usług sieciowych
   - Warstwa transportowa
   - Warstwa aplikacji
2. Komunikacja z serwerami
   - Zwykłe pobieranie strony i webscaping
   - Lecimy w kosmos:)
   - Komunikacja protokołem SOAP
   - REST API + json
   - Ciekawostka
3. Asynchroniczne pobieranie danych

Plan wykładu

Fragment dokumentacji

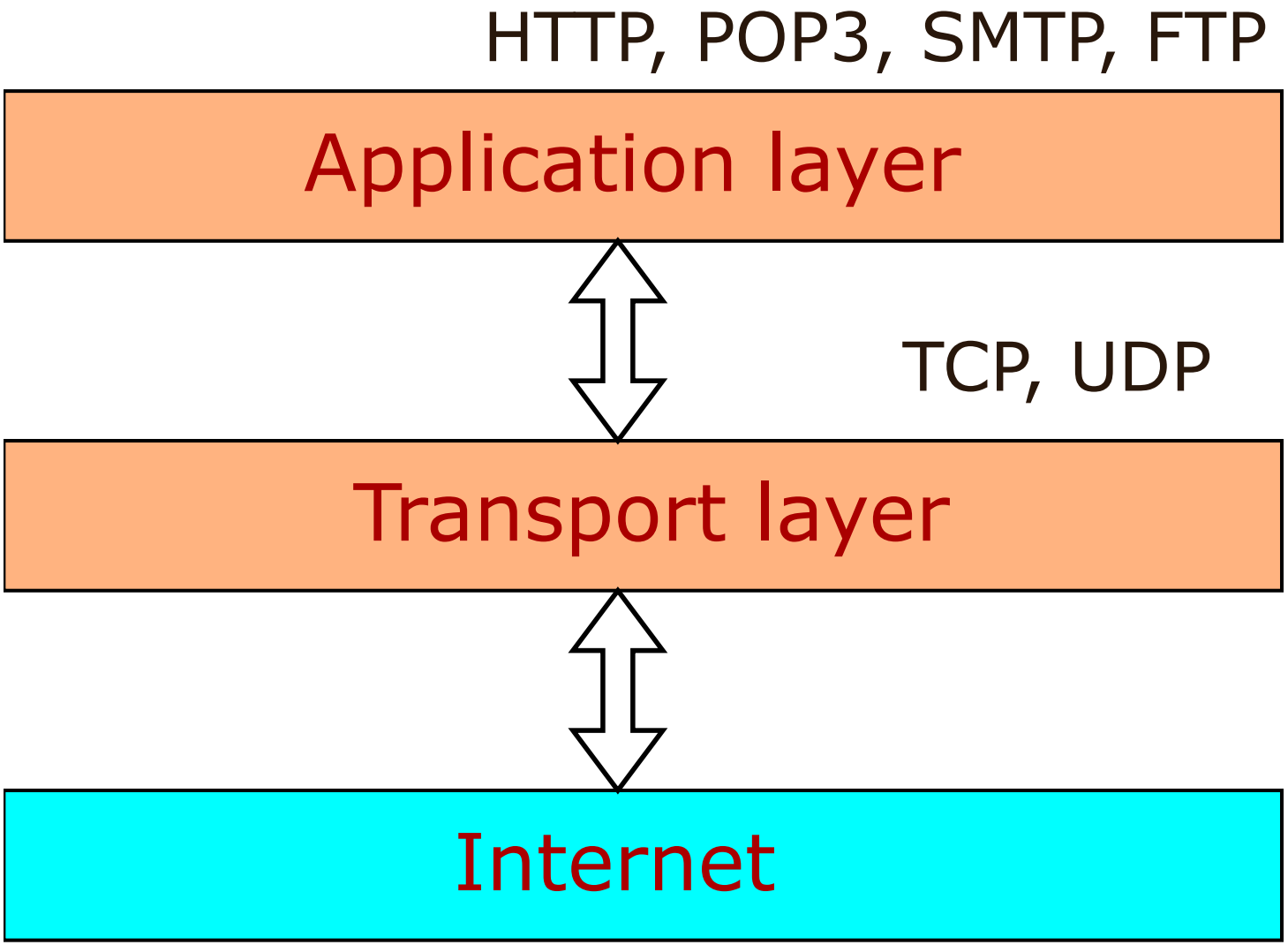
```
POST v1/customer/dispu...owledge-return-item HTTP/1.1
Content-Type: application/json
Authorization: Bearer A101.OLQiCxMmoVwigKQQDu3CYlamZ1KTKQm
               hrbAZK85RIy4IiWh9d_up_Nliuq_lfZdU.P3gvkY3PO
               28akjKYaDorm12QdfK

note=I%2Bhave%2Breceived%2Bthe%2Bitem%2Bback.
&acknowledgement_type=ITEM_RECEIVED
```

Fragment dokumentacji

```
POST v1/customer/dispu...owledge-return-item HTTP/1.1
Content-Type: application/json
Authorization: Bearer A101.OLQiCxMmoVwigKQQDu3CYlamZ1KTKQm
               hrbAZK85RIy4IiWh9d_up_Nliuq_lfZdU.P3gvkY3PO
               28akjKYaDorm12QdfK

note=I%2Bhave%2Breceived%2Bthe%2Bitem%2Bback.
&acknowledgement_type=ITEM_RECEIVED
```

Dokumentacja integracji z PayPal.

Schemat sieci



Obsługa gniazd

```
import socket

port = 8081
host = "localhost"
s = socket.socket(socket.AF_INET,
                  socket.SOCK_DGRAM)
s.sendto("Hello, Python!!!", (host, port))
```

Dla porównania wersja w C I

```c
#define _XOPEN_SOURCE 700

#include <assert.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <arpa/inet.h>
#include <netdb.h> /* getprotobyname */
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>

int main(int argc, char **argv) {
    char buffer[BUFSIZ];
    char protoname[] = "tcp";
    struct protoent *protoent;
    char *server_hostname = "127.0.0.1";
    char *user_input = NULL;
    in_addr_t in_addr;
    in_addr_t server_addr;
    int sockfd;
    size_t getline_buffer = 0;
    ssize_t nbytes_read, i, user_input_len;
    struct hostent *hostent;
    /* This is the struct used by INet addresses. */
    struct sockaddr_in sockaddr_in;
    unsigned short server_port = 12345;

    if (argc > 1) {
        server_hostname = argv[1];
```

Dla porównania wersja w C II

```
    if (argc > 2) {
        server_port = strtol(argv[2], NULL, 10);
    }
}

/* Get socket. */
protoent = getprotobyname(protoname);
if (protoent == NULL) {
    perror("getprotobyname");
    exit(EXIT_FAILURE);
}
sockfd = socket(AF_INET, SOCK_STREAM, protoent->p_proto);
if (sockfd == -1) {
    perror("socket");
    exit(EXIT_FAILURE);
}

/* Prepare sockaddr_in. */
hostent = gethostbyname(server_hostname);
if (hostent == NULL) {
    fprintf(stderr, "error: gethostbyname(\"%s\")\n", server_hostname);
    exit(EXIT_FAILURE);
}
in_addr = inet_addr(inet_ntoa(*(struct in_addr*)*(hostent->h_addr_list)));
if (in_addr == (in_addr_t)-1) {
    fprintf(stderr, "error: inet_addr(\"%s\")\n", *(hostent->h_addr_list));
    exit(EXIT_FAILURE);
}
sockaddr_in.sin_addr.s_addr = in_addr;
sockaddr_in.sin_family = AF_INET;
sockaddr_in.sin_port = htons(server_port);
```

Dla porównania wersja w C III

```c
/* Do the actual connection. */
if (connect(sockfd, (struct sockaddr*)&sockaddr_in, sizeof(sockaddr_in)) == -1) {
    perror("connect");
    return EXIT_FAILURE;
}
while (1) {
    fprintf(stderr, "enter string (empty to quit):\n");
    user_input_len = getline(&user_input, &getline_buffer, stdin);
    if (user_input_len == -1) {
        perror("getline");
        exit(EXIT_FAILURE);
    }
    if (user_input_len == 1) {
        close(sockfd);
        break;
    }
    if (write(sockfd, user_input, user_input_len) == -1) {
        perror("write");
        exit(EXIT_FAILURE);
    }
    while ((nbytes_read = read(sockfd, buffer, BUFSIZ)) > 0) {
        write(STDOUT_FILENO, buffer, nbytes_read);
        if (buffer[nbytes_read - 1] == '\n') {
            fflush(stdout);
            break;
        }
    }
}
free(user_input);
```

Dla porównania wersja w C IV

```
    exit(EXIT_SUCCESS);
}
```

Sposoby komunikacji poprzez API

Formaty danych:

- XML
- JSON
- ...

Sposób komunikacji:

- Web Services (SOAP/WSDL): XML + HTTP (zwykle)
- RPC (oparty na HTTP)
- REST (oparty na HTTP)
- ...

Publiczne serwisy

- Google
- Amazon
- Allegro
- GUS
- Geodezja
- NASA
- Zapisy
- ...

Plan wykładu

Logowanie i sesja

Logowanie do Systemu Zapisy i pobranie wiadomości.

Protokół http/https, żądanie GET

```
GET / HTTP/1.1
Host: zapisy.ii.uni.wroc.pl
User-Agent: Mozilla/5.0
```

Protokół http, odpowiedź serwera

```
HTTP/1.1 200 OK
Date: Mon, 21 Nov 2022 09:14:01 GMT
Server: Apache/2.0.54 (Debian GNU/Linux)
Content-Length: 37402
dane
```

Protokół http/https, żądanie POST

```
POST /login.php HTTP/1.1
Host: zapisy.ii.uni.wroc.pl
User-Agent: Mozilla/5.0

username=ja&password=hasło
```

Podstawowe narzędzie

```
from urllib.request import urlopen
with urlopen('http://python.org/') as resp:
    html = resp.read()
```

Podstawowe narzędzie

```
from urllib.request import urlopen
with urlopen('http://python.org/') as resp:
    html = resp.read()
```

Lepszy niestandardowy

```
import requests
```

Jak doinstalowywać pakiety

- skorzystanie z jakiegoś instalatora w swoim SO;
- wykorzystanie instalatora `pip3`.

Środowisko lokalne

```
python3 -m venv /tmp/.venv
source /tmp/.venv/bin/activate
pip3 install requests
```

Sesja

Za obsługę sesji (ciasteczek etc) odpowiada obiekt `requests.Session()`.

Kod

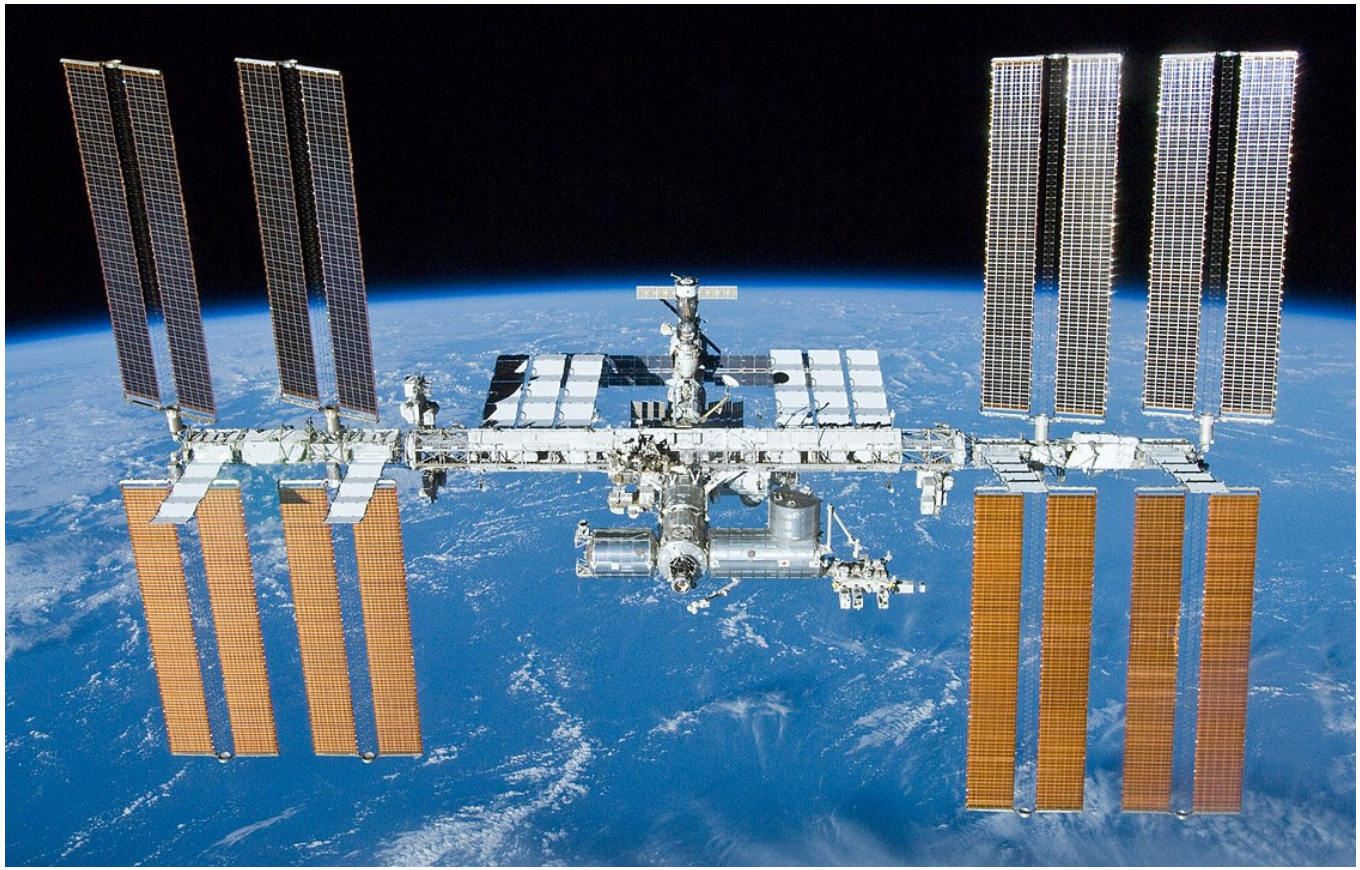
Logowanie do systemu Zapisy

```
import requests

url = "https://zapisy.ii.uni.wroc.pl/"

cred = {"username": "usrname", "password": "kdfjaskd"}
with requests.Session() as s:
    s.get(url)
    s.post(url + "users/login", data=cred)
    odp = s.get(url + "news/")
    print(odp.text)
```

Uzupełnienie: nagłówki żądania

```
import requests

url = "https://zapisy.ii.uni.wroc.pl/"

cred = {"username": "usrname", "password": "kdfjaskd"}

headers = {'User-Agent':
      'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) \
       AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2

with requests.Session() as s:
    s.get(url, headers=headers)
    s.post(url + "users/login", data=cred, headers=headers)
    odp = s.get(url + "news/", headers=headers)
    print(odp.text)
```

International Space Station



Źródło: Wikipedia

Zapytanie i odpowiedź

```
GET /iss-now.json
```

Zapytanie i odpowiedź

```
GET /iss-now.json
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

Gdzie jest ISS, kto tam jest

```
import requests
res = requests.get("http://api.open-notify.org/astros.json")
print(res.json())
res = requests.get("http://api.open-notify.org/iss-now.json")
print(res.json())
```

Jaka będzie pogoda

- `http://api.openweathermap.org`
- uzyskanie własnego API key

Postać żądania

```
"http://api.openweathermap.org/data/2.5/forecast?q=W
↪  roclaw&units=metrics&mode=json&APPID=..."
```

```
url = "http://api.openweathermap.org/data/2.5/forecast"
params = {"q": "Wroclaw", "mode": "json",
          "units": "metric", "APPID": private.KEY}
res = requests.get(url, params=params)

with open("prognoza.json", 'w') as fh:
    fh.write(json.dumps(res.json()))
```

Przykład: TERYT

Oficjalny rejest jednostek terytorialnych: województw, powiatów, gmin, miast, dzielnic, ulic etc. Administratorem danych jest Główny Urząd Statystyczny udostępniający dane m.in. przez API.

Wiadomości

Zła

Standardowy python nie obsługuje SOAP.

Wiadomości

Zła

Standardowy python nie obsługuje SOAP.

Dobra

Ale są biblioteki, np. zeep.

Wiadomości

Zła

Standardowy python nie obsługuje SOAP.

Dobra

Ale są biblioteki, np. zeep.

```
pip3 install zeep
```

Czemu nie lubię SOAP :(

```
<client>
    <endpoint address="https://uslugaterytws1.stat.gov.pl/T
        binding="customBinding" bindingConfiguration="custo
        contract="ServiceReference1.ITerytWs1" name="custom
</client>
<bindings>
  <customBinding>
    <binding name="custom">
    <security defaultAlgorithmSuite="Default"
        authenticationMode="UserNameOverTransport"
        requireDerivedKeys="true" includeTimestamp="true"
        messageSecurityVersion="WSSecurity11WSTrustFebruary
    <localClientSettings detectReplays="false" />
    <localServiceSettings detectReplays="false" />
    </security>
```

Kod

```
from zeep import Client
from zeep.wsse.username import UsernameToken

CREDENTIALS = {
    'wsdl': 'https://uslugaterytws1test.stat.gov.pl/wsdl/te
    'username': 'TestPubliczny',
    'password': '1234abcd'
}
token = UsernameToken(
    username=CREDENTIALS['username'],
    password=CREDENTIALS['password']
)
```

Pobranie danych o Wrocławiu

```
client = Client(wsdl=CREDENTIALS['wsdl'], wsse=token)

print(client.service.CzyZalogowany())

for jpt in client.service.WyszukajJPT(nazwa='Wrocław'):
    print(jpt)
```

Rodzaje żądań HTTP

POST	tworzenie nowego zasobu
GET	pobranie zasobu
PUT	aktualizacja zasobu
DELETE	usunięcie zasobu

Wskazanie zasobu

Zasoby wskazujemy adresem URL, np:

`student/{id}`

Pobranie informacji o studencie:

```
GET http://api.serwis.org/student/123
```

Przykład: Allegro

- Link

Klient P2P: Sieć TinyP2P

```
import sys, os, SimpleXMLRPCServer, xmlrpclib, re, hmac # (C) 2004, E.W. Felten
ar,pw,res = (sys.argv,lambda u:hmac.new(sys.argv[1],u).hexdigest(),re.search)
pxy,xs = (xmlrpclib.ServerProxy,SimpleXMLRPCServer.SimpleXMLRPCServer)
def ls(p=""):return filter(lambda n:(p=="")or res(p,n),os.listdir(os.getcwd()))
if ar[2]!="client": # license: http://creativecommons.org/licenses/by-nc-sa/2.0
  myU,prs,srv = ("http://"+ar[3]+":"+ar[4], ar[5:],lambda x:x.serve_forever())
  def pr(x=[]): return ([(y in prs) or prs.append(y) for y in x] or 1) and prs
  def c(n): return ((lambda f: (f.read(), f.close()))(file(n)))[0]
  f=lambda p,n,a:(p==pw(myU))and(((n==0)and pr(a))or((n==1)and [ls(a)])or c(a))
  def aug(u): return ((u==myU) and pr()) or pr(pxy(u).f(pw(u),0,pr([myU])))
  pr() and [aug(s) for s in aug(pr()[0])]
  (lambda sv:sv.register_function(f,"f") or srv(sv))(xs((ar[3],int(ar[4]))))
for url in pxy(ar[3]).f(pw(ar[3]),0,[]):
  for fn in filter(lambda n:not n in ls(), (pxy(url).f(pw(url),1,ar[4]))[0]):
    (lambda fi:fi.write(pxy(url).f(pw(url),2,fn)) or fi.close())(file(fn,"wc"))
```

Plan wykładu

Narzędzia

Jak pobierać asynchronicznie dane:

~~`requests`~~: nie, bo nie zwalnia GIL'a

[2] To też niestandardowy pakiet i trzeba go doinstalować

Narzędzia

Jak pobierać asynchronicznie dane:

~~`requests`~~: nie, bo nie zwalnia GIL'a

`aiohttp`[2].

[2] To też niestandardowy pakiet i trzeba go doinstalować

Współprogramy

Funkcja jest współprogramem:

```
async def wspolprogram():
    ...
```

Instrukcja jest nieblokująca *(awaitable)* i można w tym czasie wykonać inne zadanie:

```
await <instrukcja>
```

Uruchomienie całości

```
async main():
    ...

asyncio.run(main())
```

Przykład

```
import asyncio

async def nested():
    return 42

async def main():
    nested() # nic nie zrobi, tylko stworzy współprogram

    print(await nested())  # wypisze "42".

asyncio.run(main())
```

Na podstawie dokumentacji

Jednoczesne pobieranie danych z sieci

```
import asyncio
import aiohttp
from aiohttp import ClientSession

async def fetch_page(session, url):
    async with session.get(url) as result:
        res = await result.text()

    return res

urls = ['https://www.ii.uni.wroc.pl'] * 100

async def main():
  async with aiohttp.ClientSession() as session:
    requests = [fetch_page(session, url) for url in urls]
    pages = await asyncio.gather(*requests)

asyncio.run(main())
```