PS Zadanie 2
2020/21

5. mája 2021

0.1 Formálne aspekty

- Odovzdávať sa bude prostredníctvom githubu, sprístupnením repozitára menom `httpclient` užívateľovi `gjenca`
- Repozitár bude obsahovať iba jeden súbor so zdrojovým kódom, nič iné.
- Termín odovzdania projektu je 12.6.2021 23:59:59.
- Po tomto termíne budú projekty vyhodnotené do troch pracovných dní, známka bude zapísaná do AIS a študentom budú oznámené prípadné požadované zmeny pre zvýšenie počtu bodov.
- V prípade, že niekto bude mať záujem o zvýšenie počtu bodov, bude vypísaný ďalší termín dokedy budú môcť záujemcovia odovzdať druhú verziu programu.

1 HTTP klient

Napíšte v Pythone skript `http_get`, ktorý očakáva ako jediný povinný argument URL v a na štandardný výstup vráti web stránku (alebo obrázok alebo niečo iné) ktoré je na danom URL. Nesmiete používať Pythonovské moduly, ktoré implementujú HTTP (t.j. `requests`, `urllib2` a iné). Máte to spraviť cez normálne TCP sockety, s prípadnou podporou šifrovania SSL, ak je v URL uvedený protokol *https:*.

1.1 Bližšia špecifikácia

- Obsah máte dávať na výstup ak je status `200 OK`, inak nie.
- Na presmerovacie statusy (301,302,303,307,308) reagujte tak, že pôjdete na to URL, ktoré vám bolo poslané v `Location` headri.
- Musíte implementovať aj `Content-length` aj `Transfer-encoding: chunked`[1].

[1] Toto vysvetlím ešte na prednáške

- Statusy, ktoré nie sú implementované vypíšte na `sys.stderr` a dajte `sys.exit(1)`.
- Ak je uvedené `http:` URL, pripájate sa na port 80, ak je uvedené `https:` URL, pripájate sa na port 443 a obalíte pripojený socket pomocou funkcie `ssl.wrap_socket`, takto:

```
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
if is_ssl:
    s.connect((hostname,443))
    s=ssl.wrap_socket(s)
else:
    s.connect((hostname,80))
```

a potom môžete ten socket používať ako ste zvyknutí.

1.2 Na čo si treba dať pozor

- HTTP je zmiešaný textovo-binárny protokol – headre sú text (ASCII), obsah je binárny. Z toho vyplýva, že máte použiť `f=socket.makefile("rwb")`, pre čítanie a zapisovanie v binárnom móde.
- Dáta ktoré budete potom čítať zo socketu prostredníctvom `f` budú typu `bytes` nie `str`. Napriek tomu môžete a máte pri čítaní HTTP statusu a headrov používať `f.readline`. Ja som riadky ihneď prevádzal na typ `str` cez `bytes.decode('ASCII')`, aby som mal pre interné spracovanie už `str`.
- Po ukončení headrov už nesmiete predpokladať, že to čo prúdi zo socketu je text v akomkoľvek kódovaní.
- POZOR: pre vypisovanie obsahu (typ `bytes`) je nutné použiť `sys.stdout.buffer`, ten slúži ako štandardný výstup pre *binárne* dáta. Typ `bytes` vôbec na `sys.stdout` nejde zapisovať, tam môžete zapisovať iba typ `str`.

2 Ako som to robil ja

- Zistil som, či je protokol v URL `http` alebo `https`.
- Z URL som vytiahol *hostname* a *path* pomocou `re.match`.
- Pripojil som sa (viď vyššie) a poslal `GET` request, včítane povinného `Host:` headra a poslal som aj `Accept-charset: UTF-8`.

- Prečítal som prvý riadok odpovede, dekódoval ako ASCII a potom pomocou `re.match` vytiahol status a jeho popis.
- Prečítal som všetky headre, dekódoval ako ASCII a uložil si ich do slovníka. POZOR veľkosť písiem nie je v HTTP protokole určená, môžete dostať povedzme `CoNTent-LeNGtH` a je to legálne. To som vyriešil tak, že som na každý názov headra dával pred jeho uložením do slovníka `str.lower()`, čím sa všetky písmená hodia na malé.
- Ak bol status presmerovací, vytiahol som URL z location a pokračoval tam (mám to ako nekonečný cyklus, vyskakuje sa z neho pri statuse 200).
- Prečítal som obsah odpovede (dve rôzne vetvy: bolo `content-length` alebo máme `transfer-encoding: chunked`) a ak bol status 200, zapísal som obsah na sys.stdout.buffer. Pri presmerovacích statusoch treba obsah ignorovať a zavrieť socket.