PS Zadanie 2
2024/25

5. mája 2025

1 Formálne aspekty

Vaše repozitáre musia byť súkromné (private), nie verejné (public). Predídeme tým nepríjemným nedorozumeniam ohľadom autorstva.

- Odovzdávať sa bude prostredníctvom githubu, sprístupnením repozitára menom `httpclient` užívateľovi `gjenca`
- Repozitár bude obsahovať iba **jeden súbor so zdrojovým kódom, nič iné.**
- Normálny termín odovzdania projektu je 21.6.2025 23:59:59.
- Predtermín odovzdania projektu je 17.5.2025 23:59:59 (piatok, nie sobota!)– pre tých, čo chcú mať predmet z krku čím skôr.
- Po normálnom termíne budú projekty vyhodnotené, známka z celého predmetu 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. Tento cyklus bude pokračovať ďalej, podľa situácie.

2 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:`.

2.1 Bližšia špecifikácia

- Obsah máte dávať na štandardný výstup ak je status `200 OK`, inak nie.
- Na štandardný výstup máte dávať iba obsah odpovede, alebo nič (v prípade chyby).
- 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`.
- Statusy, ktoré nie sú implementované vypíšte zrozumiteľne na `sys.stderr` a dajte `sys.exit(1)`. (napr. `404 Not found`)
- Ak je uvedené `http:` URL, pripájate sa na port 80, ak je uvedené `https:` URL, pripájate sa na port 443 a „obalíte" pripojený socket šifrovacou vrstvou. Robí sa to v Pythone ľahko. Najskôr sa vytvorí objekt typu `SSLContext`, ktorý sa dá vytvoriť všelijako ale existuje jeho štandardná verzia, ktorú vráti funkcia `create_default_context` v module `ssl`. Tento objekt má metódu `wrap_socket`, ktorá už vie obaliť pripojený socket šifrovacou vrstvou.

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

  a potom môžete ten socket používať tak, ako ste zvyknutí. Pozor! Na webe sa nachádza mnoho zastaralých informácií o SSL v Pythone, toto je ale spôsob, ktorým to funguje v roku 2025.
- Ak dostanete iný status ako `200 OK` alebo presmerovanie, vypíšte ho na `sys.stderr` a dajte `sys.exit(1)`

2.2 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`.
- Testujte to nielen na HTML, ale aj na obrázkoch.
- Váš program by mal vracať skoro to isté ako `curl -sL`. Obrázky majú byť úplne rovnaké. Na porovnávanie textových súborov v UNIXoch sa používa program `diff`.

```
$ ./http_get http://www.stuba.sk/ > out1.txt
$ curl -sL http://www.stuba.sk/ > out2.txt
$ diff out1.txt out2.txt
1437c1437
<         <input type="hidden" name="_token" value="vvlcAlZletKPR74aIKeEAD9J3KbPWpLtGPSuLMny">
---
>         <input type="hidden" name="_token" value="hQ3dx8PsGuEhiDQlYK1XMkh5WSXNds8wAvZOWuIi">
$
```

Vidíme, že sa výstupy odlišujú iba v nejakom malom dynamicky generovanom reťazci (prevencia útoku typu CSRF).

Naproti tomu obrázok alebo iný statický obsah je vždy rovnaký, teda diff nevypíše nič.

3 Ako som to robil ja

- Zistil som, či je protokol v URL `http` alebo `https`. Podľa toho som určil číslo portu, pripojil socket a prípadne ho obalil šifrovaním.
- 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 *reťazec*`.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 prečítať ale nevypisovať, potom nezabudnúť po sebe slušne zavrieť socket.
- Správne riešenie presmerovania je cyklus, ,,pokiaľ dostávam presmerovací status, idem na ďalšie URL", pretože presmerovanie nemusí byť iba jedno.