MTAT.03.311
Loomuliku keele töötlus *Python*is
(6 EAP)

Karl-Oskar Masing

22.09.2015

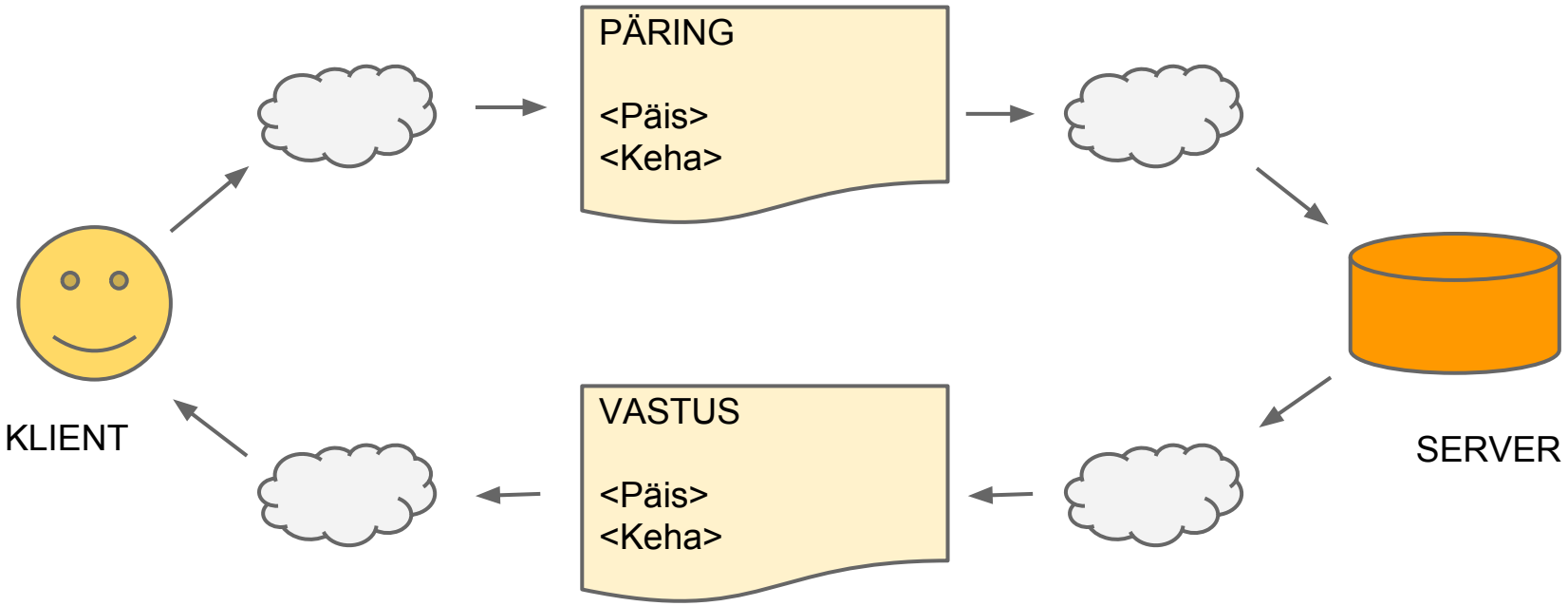
Uhked keeletöötluse meetodid selged.

Aga kust saada andmeid?

Keeleandmete kogumine veebist

Veeb

Klient-server arhitektuur



Klient saadab päringu. Server saadab vastuse.

Päringud

- Koosneb kahest osast
  - domeeni/serveri aadress
    - nt http://maailm.postimees.ee
    - DNS lahendab IP-ks (domain name system)
  - domeeni/serveri ressurss
    - nt /3336253/galerii-brasiilia-zoos-leiab-peavarju-23-orvuks-jaanud-loomabeebit
    - lahendab konkreetne server

Päringute liigid (HTTP meetodid)

- Liigid
  - GET
  - POST
  - PUT
  - DELETE

GET

- Küsib serverilt ressurssi
  - enamasti kuvatavat HTML lehte
- Kasutavad veebilehitsejad
  - iga aadressi sissetoksimisel teeb lehitseja GET päringu
- Kogu päringu info on URL-is
  - http://www.domain.ee/search?name=aet&age=29
- Omadusi
  - saab *cache*'ida, *bookmark*'ida, omab mahupiirangut, ebaturvaline

POST

- Saadab serverile andmeid
- Ei saa veebilehitsejas vahetult teha
- Päringu info talletatakse kehas
  - http://www.domain.ee/search
  - kehas sõnaraamat {‘name’:’aet’, ‘age’:29}
- Omadusi
  - ei saa *cache*’ida ega *bookmark*’ida, ei oma mahupiirangut
    - ei saa saiti sisselogitust talletada

Uus populaarne formaat - JSON

JSON (JavaScript Object Notation)

- Pythoni sõnaraamatu kujul
  - võtmed on sõned
  - väärtused on
    - sõned
    - arvud
    - sõnaraamatud
    - listid
    - tõeväärtused (**true, false**)
    - None (**null**)
- Lihtne lugeda ja kirjutada nii inimese kui arvuti jaoks

JSON-i näide

```
{“pupils”:
  [
    {
      “name”:”Liina”,
      “age”:19
    },
    {
      “name”:”Taavi”,
      “age”: null
    }
  ]
}
```

JSON Pythonis

```
>>> import json
>>> str_ = json.dumps({‘name’:’aet’, ‘age’:29})
>>> str_
'{"age": 29, "name": "aet"}'
>>> json.loads(str_)
{u'age': 29, u'name': u'aet'}
```

Veebist tõmbamine

Veebist tõmbamine

1. Ressursi aadressi leidmine
2. Ressursi allatõmbamine
3. Ressursist tarviliku informatsiooni eraldamine
4. Informatsiooni töötlemine

Ressursi allatõmbamine (iganenud)

```
>>> import urllib
>>> import urllib2
>>> “GET”
>>> fin = urllib2.urlopen(“http://www.postimees.ee”)
>>> response = f.read()
>>>
>>> “POST”
>>> values = {‘name’:’aet’,’age’:29}
>>> request = urllib2.Request(‘http://www.domain.ee/search’, urllib.urlencode(values))
>>> response = urllib2.urlopen(request).read()
```

Ressursi allatõmbamine (moodne)

```
>>> import requests
>>> “GET”
>>> response =  requests.get(“http://www.postimees.ee”)
>>> text_content = response.text
>>> loaded_json_content = response.json()
>>>
>>> “POST”
>>> response = requests.post(“http://www.domain.ee/search”,{‘name’: ’aet’, ‘age’: 29})
```

Ressursist info eraldamine

- Parser vastavalt formaadile
- Sagedaseimad
  - XMLParser
  - HTMLParser
  - json

Web Crawling

Web crawling

- Eesmärk
  - otsingumootorite indeksite loomine
    - googlebot-X
  - andmete allatõmbamise/otsimise automatiseerimine

Web crawling - terminoloogia

- *seeds*
  - lähte URL-id
- *crawl frontier*
  - külastamist vajavad URL-id
- *visit*
  - veebilehe sisu parsimine/*crawl frontier*'i täiendamine

Web crawling - algoritm

```
urls = seeds
while len(urls) > 0:
    current_url = select_url(urls)
    content, new_urls = process(current_url)
    do_sth_with(content)
    urls.extend(new_urls)
```

Web crawling - nüansid

- URL-ide valimine (*selection policy*)
- URL-ide taaskülastamine (*re-visit policy*)
- Viisakus (*politeness policy*)
- Paralleliseerimine (*parallelization policy*)

Crawling - probleemid

- URL-ide kordumine
- Sisu kordumine
- Suur otsinguruum
- Serverite koormamine

Crawling - selection policy

- linkide piirangud (ainult .html etc)
- URL-i normeerimine
  - lowercase, . ja .. likvideerimine, lõpust / eemaldamine
- path-ascending
  - kom.ee/kala/hobune/index.html
  - kom.ee/kala/hobune/
  - kom.ee/kala/

Crawling - selection policy

- teemaspetsiifiline *crawl*'imine
  - sisult sarnaste lehtede otsimine
    - akadeemiline
      - google scholar
      - siteceerx

Crawling - re-visit policy

- Oluline otsingumootoritele
- Tüübid
  - ühtlane
  - proportsionaalne
- Andmete kogumiseks pole sageli oluline
  - v.a sisu ajalise muutuse analüüsimise korral

Crawling - politeness policy

- Crawler sirvib lehti palju kiiremini kui inimene
- Liiga kiire lehtede tõmbamine võib serveri kokku jooksutada
- Ühelt domeeniaadressilt ei tohiks tõmmata sagedamini kui iga 10 sekundi tagant
  - sagedamini võib tõmmata domeenidelt ringiratast
- Tasub järgida veebilehe juhiseid
  - *robots.txt* ja *sitemap.xml*

http://www.domain.ext/robots.txt

- Defineerib
  - milliseid lehti ei tohi *crawl*'ida

```
User-agent: *                   # kõik crawlerid
Disallow: /                     # keela kogu leht

User-agent: kommikrooler        # ainult kommikrooler
Disallow: /secret/              # keela /secret/ ja /secret/*, va /secret/notverysecret/*
Allow: /secret/notverysecret/

User-agent: *
Crawl-delay: 25                 # nõuab crawler'itelt 25-sekundilist lehtede tõmbamise vahet
```

  - *sitemap*'i asukoha

```
Sitemap: http://www.domain.ext/sitemap
```

http://www.postimees.ee/robots.txt

```
User-agent: *
Disallow: /search*
Disallow: /*/print/*
Disallow: /print/*
Disallow: /*/com/*
Disallow: /mobile/*
User-agent: Mediapartners-Google
Disallow:
Sitemap: http://www.postimees.ee/sitemap
```

Eksimine reeglite vastu

- Ebaeetiline
- Lühema/väiksema rikkumise korral üldiselt tagajärgedeta
- Pikema korral
  - võimalik ühendusevõtmine vastava lehe süsteemi administraatori poolt
  - IP aadressi bokeerimine
  - võimalik kohtuasi

Crawler’i identifitseerimine

- Päringu *User-Agent* päise abil
- Identiteedita *crawl*’imine tekitab kahtlusi
  - sageli pahavara
- Lisada kontaktandmed

```
>>> import urllib, urllib2
>>>
>>> url = ‘http://some.site.to/crawl’
>>> user_agent = “KOM crawler (kom@ut.ee) / on studying purpose”
>>> headers = { ‘User-Agent’ : user_agent }
>>> values = {}
>>> req = urllib2.Request(url, urllib.urlencode(values), headers)
>>> response = urllib2.urlopen(req).read()
```

Paralleliseerimine

- Eesmärgiks kiirendada lehtede läbimist
- Pythonis eelkõige läbi *multiprocessing* teegi
  - *Global Interpreteter Lock*'i tõttu ei saa mitmelõimeline programm mitmest protsessorist kasu
    - Ühelõimeliste programmide kiiruse huvides pole peaaegu miski *thread safe*

Web Scraping

Web Scraping - olemus

- Veebist informatsiooni eraldamine
- Võib kasutada
  - *crawl*'imist
  - fikseeritud domeeni
  - fikseeritud lehte

Web Scraping - vahendid

- grep / regulaaravaldised
- HTML parserid
- XML parserid
- Eksisteerivad scrape’imise raamistikud

Viited

https://en.wikipedia.org/wiki/Web_crawler