Regulární výrazy, práce s textem a daty

IB111 Úvod do programování skrze Python

2015

Výhled

- dnes: práce s textem
- příště: práce s obrázky

- příkazy pro práci s textem/obrázky
- ukázky na konkrétních příkladech
- použití dříve probíraných konstrukcí, datových struktur (seznam, slovník)
- uvažování o problémech, volba přístupu, kladení otázek

Práce s textem

Příklady:

- statistiky o studentech
- frekvenční analýza textu
- zpracování dotazníku, ankety

Čeština

- v tomto kurzu pro zjednodušení pracujeme jen s anglickou abecedou
- resp. s texty bez hacku a carek
- pro zájemce viz např.
  `http://www.py.cz/PythonUnicodeCestina`
- základ: specifikovat kódování na začátku souboru
  `# -*- coding: utf-8 -*-`

Regulární výrazy: motivace

- vyhledání e-mailových adres v textu
- vyhledání odkazů v HTML dokumentu
- náhrada „jméno příjmení“ za „příjmení jméno“
- změna formátu datumů
- odstranění bílých znaků

Regulární výrazy: použití

- programování
- textové editory
- příkazová řádka: např. `grep`
- teorie: formální jazyky, konečné automaty

http://xkcd.com/208/

Regulární výrazy

- obecně používaný nástroj
- základní syntax stejná ve většině jazyků, prostředí
- následuje:
  - základní syntax regulárních výrazů
  - použití v Pythonu
- nerozebíráme všechny technické detaily (podrobněji viz dokumentace)

Ukázka

```
import re
f = open("testovaci-soubor.txt")
for radek in f.readlines():
    if re.search(r'[a-z]+@[a-z]+\.cz', radek):
        print radek
f.close()
```

Znaky a speciální znaky

- základní znak „vyhoví“ právě sám sobě
- např. „cz“ v předchozím příkladě
- speciální znaky: `. ^ $ * + ? { } [ ] \ | ( )`
  - umožňují konstrukci složitějších výrazů
  - chceme, aby odpovídaly příslušnému symbolu $\Rightarrow$ prefix \

Výběr ze skupiny znaků

[]

- [abc] – jeden ze znaků a, b, c
- [a-z] – výběr z intervalu (malé písmeno anglické abecedy)
- ^ na začátku výběru = negace:
  - [^abc] cokoliv jiného než a, b, c

Často používané skupiny znaků

`\d`	Čísla: `[0-9]`
`\D`	Cokoliv kromě čísel: `[^0-9]`
`\s`	Bílé znaky: `[ \t\n\r\f\v]`
`\S`	Cokoliv kromě bílých znaků: `[^ \t\n\r\f\v]`
`\w`	Alfanumerické znaky: `[a-zA-Z0-9_]`
`\W`	Nealfanumerické znaky: `[^a-zA-Z0-9_]`

Speciální symboly

.	libovolný znak
^	začátek řetězce
$	konec řetězce
|	alternativa – výběr jedné ze dvou možností

Příklady

Jaký je význam následujících výrazů?

- `kocka|pes`
- `^[Pp]rase$`
- `\d[A-Z]\d \d\d\d\d`

Opakování

`*`	nula a více opakování
`+`	jedno a více opakování
`?`	nula nebo jeden výskyt
`{m,n}`	m až n opakování

Pozn. `*`, `+` jsou „hladové“, pro co nejmenší počet opakování `*?`, `+?`

Příklady

Jaký je význam následujících výrazů?

- `^\s*Nadpis`
- `^a.+a$`
- `\d{3}\s?\d{3}\s?\d{3}`
- `[a-z]+@[a-z]+\.cz`
- `^To:\s*(fi|kit)(-int)?@fi\.muni\.cz`

Příklady

Která z následujících slov vyhoví jednotlivým výrazům?

	`p[ars]e`	`p[ars]*e`	`p[^ars]e`
`ps`			
`pes`			
`pse`			
`poe`			
`prase`			
`poklice`			

Příklady

Která z následujících slov vyhoví jednotlivým výrazům?

	`p[ars]e`	`p[ars]*e`	`p[^ars]e`
`ps`	×	×	×
`pes`	×	✓	×
`pse`	✓	✓	×
`poe`	×	×	✓
`prase`	×	✓	×
`poklice`	×	×	×

Kontrola tabulky v Pythonu

```
texty = ["ps","pes","pse","poe","prase","poklice"]
vyrazy = [ r'p[ars]e', r'p[ars]*e', r'p[^ars]e' ]
for text in texty:
    print text,
    for vyraz in vyrazy:
        if re.search(vyraz, text): print 1,
        else: print 0,
    print
```

Backreference

- „odkaz“ na předchozí část textu
- pomocí `\1`, `\2`, `\3`, ...
- příklad: slova, která obsahují opakovaně stejnou trojici písmen (jinde než) na začátku a konci
  - p**eri**f**eri**e
  - **sta**ro**sta**
- jak zapsat regulární výraz?
- jaká další slova splňují?

Backreference a trojice písmen

- seznam slov, např.

  `https://ucnk.ff.cuni.cz/srovnani10.php`

- regulární výrazy:
  - `^(...).*\1$`
  - `.(...).*\1.`
- výstupy např.:
  - b**ram**bo**rami**
  - m**ult**ik**ult**urní
  - **ostrost**
  - **ska**li**ska**

Bez regulárních výrazů

Varianta „uvnitř slova“:

```
def shoda(slovo):
    for i in range(1, len(slovo)-3):
        for j in range(i+1, len(slovo)-3):
            if slovo[i:i+3] == slovo[j:j+3]:
                return True
    return False
```

Je to ekvivalentní?

Detekce opakovaných trojic

- regulární výraz: `.(...).*\1.`
- kontrola podmínky: `slovo[i:i+3] == slovo[j:j+3]`
- rozdílné chování: skl**enene**ho, k**ovovo**u, ko**cicic**h

Co je správně?
Nejasná specifikace problému...

Regulární výrazy: xkcd

http://xkcd.com/1313/
http://www.explainxkcd.com/wiki/index.php/1313:_Regex_Golf
https://regex.alf.nu/

Regulární výrazy v Pythonu

- knihovna `re` (`import re`)
- `re.match` – hledá shodu na začátku řetězce
- `re.search` – hledá shodu kdekoliv v řetězci
- (`re.compile` – pro větší efektivitu)
- „raw string“ – r'vyraz' – nedochází k interpretaci speciálních znaků jako u běžných řetězců v Pythonu

Regulární výrazy v Pythonu: práce s výsledkem

- match/search vrací „MatchObject“ pomocí kterého můžeme s výsledkem pracovat
- pomocí kulatých závorek `()` označíme, co nás zajímá
- `Alice\s+(\w+)`

Regulární výrazy v Pythonu: práce s výsledkem

```
>>> m = re.match(r"(\w+) (\w+)", \
                 "Isaac Newton, fyzik")
>>> m.group(0)
'Isaac Newton'
>>> m.group(1)
'Isaac'
>>> m.group(2)
'Newton'
```

Substituce

- nahrazení řetězce jiným výrazem
- nejen statické řetězce, ale i regulární výrazy
- `re.sub`

Rozdělení řetězce

- `split` – rozdělí řetězec podle zadaného podřetězce, vrací seznam částí
- `join` – spojení seznamu řetězců do jednoho

```
>>> retezec = "Holka modrooka   nesedavej u potoka"
>>> retezec.split()
['Holka', 'modrooka', 'nesedavej', 'u', 'potoka']
>>> retezec.split('o')
['H', 'lka m', 'dr', '', 'ka   nesedavej u p', 't', 'ka
>>> retezec.split('ka')
['Hol', ' modroo', '   nesedavej u poto', '']
```

Řetězce: další funkce

- `find`, `count` – vyhledávání a počítání podřetězců
- `lower`, `upper` – převod na malá/velká písmena
- `ljust`, `rjust`, `center` – zarovnání textu
- `lstrip`, `rstrip` – ořezání bílých znaků na začátku/konci

Práce se soubory – připomenutí

Otevírání a zavírání:

- `f = open("mujsoubor.txt")` – otevření pro čtení
- `f = open("mujsoubor.txt","w")` – otevření pro zápis
- `f.close()` – uzavření souboru
- zápis pomocí `with` – lepší praxe (ale pokročilejší, souvisí s výjimkami)

Čtení a zápis:

- `f.readline()` – vrátí další řádek ze souboru
- `f.readlines()` – vrátí seznam všech zbývajících řádků
- `f.write(retezec)` – zapíše do souboru

Příklad: Zpracování HTML

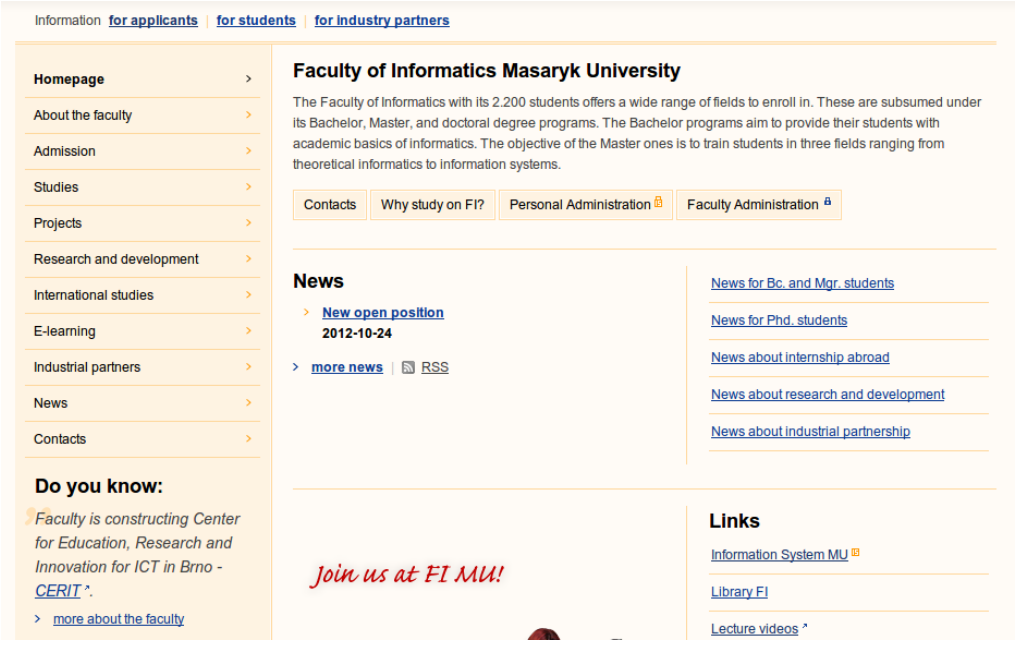
- vstup: HTML soubor
- cíl: vybrat odkazy a nadpisy
- ukážeme naivní řešení se soubory, reg. výrazy
- systémovější řešení: využití knihoven pro práci s URL zdroji, parsování HTML, "web scraping"

4427

You can't parse [X]HTML with regex. Because HTML can't be parsed by regex. Regex is not a tool that can be used to correctly parse HTML. As I have answered in HTML-and-regex questions here so many times before, the use of regex will not allow you to consume HTML. Regular expressions are a tool that is insufficiently sophisticated to understand the constructs employed by HTML. HTML is not a regular language and hence cannot be parsed by regular expressions. Regex queries are not equipped to break down HTML into its meaningful parts. so many times but it is not getting to me. Even enhanced irregular regular expressions as used by Perl are not up to the task of parsing HTML. You will never make me crack. HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing HTML with regex summons tainted souls into the realm of the living. HTML and regex go together like love, marriage, and ritual infanticide. The <center> cannot hold it is too late. The force of regex and HTML together in the same conceptual space will destroy your mind like so much watery putty. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he comes. HTML-plus-regexp will liquify the nerves of the sentient whilst you observe, your psyche withering in the onslaught of horror. Regex-based HTML parsers are the cancer that is killing StackOverflow *it is too late it is too late we cannot be saved* the trangession of a child ensures regex will consume all living tissue (except for HTML which it cannot, as previously prophesied) *dear lord help us how can anyone survive this scourge* using regex to parse HTML has doomed humanity to an eternity of dread torture and security holes *using regex* as a tool to process HTML establishes a breach *between this world* and the dread realm of corrupt entities (like SGML entities, but *more corrupt) a mere glimpse* of the world of reg**ex parsers for HTML will ins**tantly transport a p*rogrammer's consciousness* into a world of ceaseless screaming, he comes~~, the pestilent sl~~ithy regex-infection wi**ll devour your HT**ML parser, application and existence for all time like Visual Basic only worse *he comes he comes do not fi*ght h**e comes, hi**s unholy radiance destro*ying all enlightenment, HTML tags* ***leaking from your eyes like liq****uid* pain, the song of regular expre~~ssion parsing~~ will exti*nguish the voices of mor****tal man from the sp****here I can see it can you see it it is beautiful t*he f `inal snuf` fing *of the lie****s of Man ALL IS LOST*** **ALL I**S LOST th*e pony he comes he* ~~comes he comes~~ ***the ich***or permeates al*l MY FACE MY FACE ᵒh god n***o NO NOOOO N**O stop t*he an*gles are not real ZALGO IS TONY THE PONY HE COMES

```
http://stackoverflow.com/questions/1732348/
regex-match-open-tags-except-xhtml-self-contained-tags/1732454#1732454
```

Příklad: Zpracování HTML




```
<div id="header">
    <div id="logo">
        <h1>Faculty of Informatics Masaryk University</h1><span></span>
    </div>
</div>

<div id="altNav">
    <a href="#wrapper">go to content</a>
    <a href="#mm">go to menu</a>
    <a href="#search">go to search</a>
</div>
```

Hledání nadpisů

```
def najdi_nadpisy(jmeno_souboru):
    soubor = open(jmeno_souboru)
    for radek in soubor.readlines():
        m = re.search(r'<h(\d)>(.*?)</h\d>',radek)
        if m:
            print m.group(1), "\t", m.group(2)
    soubor.close()
```

Kdy nebude fungovat korektně?

Hledání odkazů

- stejná základní kostra, jen jiný regulární výraz
- pokus č. 1:
- `<a href="(.*)">(.*)</a>`
- nedostatky?

Hledání odkazů

- stejná základní kostra, jen jiný regulární výraz
- pokus č. 1:
- `<a href="(.*)">(.*)</a>`
- nedostatky?

```
<a href="www.seznam.cz">Seznam</a>, <a href="www.google.com">Google</a>

<a href="www.seznam.cz" target="_blank">Seznam</a>
```

Hledání odkazů

- rozšíření (stále nedostatečná):
  - `<a href="(.*?)">(.*?)</a>`
  - `<a href="(.*?)".*?>(.*?)</a>`
- `*` – „hladové“ hledání („co nejvíc“)
- `*?` – „co nejmíň“

Příklad: Jak vykrást banku?

Přesněji: Jak převzít kurzy ze stránky ČNB?

Platnost od 05.11.2012 Pořadí: 215

země	měna	množství	kód	kurz
Austrálie	dolar	1	AUD	20,454
Brazílie	real	1	BRL	9,706
Bulharsko	lev	1	BGN	12,902
Čína	renminbi	1	CNY	3,162
Dánsko	koruna	1	DKK	3,383
EMU	euro	1	EUR	25,235
Filipíny	peso	100	PHP	47,847

```
477 <h3 class="kurzy_tisk">Platnost od 05.11.2012 Pořadí: 215</h3>
478 <table class="kurzy_tisk">
479 <tr><th>země</th><th>měna</th><th>množství</th><th>kód</th><th>kurz</th></tr>
480 <tr><td>Austrálie</td><td>dolar</td><td align="right">1</td><td>AUD</td><td align="right">20,454</td></tr>
    align="right">1</td><td>BRL</td><td align="right">9,706</td></tr><tr><td>Bulharsko</td><td>lev</td><td ali
    align="right">12,902</td></tr><tr><td>Čína</td><td>renminbi</td><td align="right">1</td><td>CNY</td><td
    align="right">3,162</td></tr><tr><td>Dánsko</td><td>koruna</td><td align="right">1</td><td>DKK</td><td
    align="right">3,383</td></tr><tr><td>EMU</td><td>euro</td><td align="right">1</td><td>EUR</td><td
    align="right">25,235</td></tr><tr><td>Filipíny</td><td>peso</td><td align="right">100</td><td>PHP</td><td
```

Příklad: Kurzy ze stránky ČNB

Základní řešení:

- najít řádek, na kterém jsou kurzy (začíná `<tr><td>Aust`)
- rozdělit na řádky tabulky (podle `</tr><tr>`)
- hledat trojice velkých písmen a za nimi čísla
- převést na typ float, uložit do slovníku

Nevýhody?

Příklad: Kurzy ze stránky ČNB

```
def zjisti_kurzy(jmeno_souboru):
  kurzy = {}
  soubor = open(jmeno_souboru)
  for radek in soubor.readlines():
    if re.match(r'<tr><td>Aust',radek):
      for radek_tab in radek.split('</tr><tr>'):
        m = re.search(
              r'<td>([A-Z]{3}).*right">([\d,]+)</td>
              radek_tab)
        kurzy[m.group(1)] =
              float(re.sub(',','.',m.group(2)))
  soubor.close()
  return kurzy
```

Příklad: informace o studentech

Export informací z ISu (CSV soubor):

```
1.;50668;"Sukany, Martin";zk;"FI B-AP BcAP [sem 1, roc
2.;421714;"Veznik, Ondrej";zk;"FI B-AP SOCI [sem 2, roc
3.;564138;"Machala, David";zk;"FI B-AP BcAP [sem 1, roc
4.;43583;"Mikes, Martina";zk;"FF B-FI PLIN [sem 5, cyk
5.;81908;"Sulc, Tomas";zk;"FF B-FI PLIN [sem 5, cyk 1]"
6.;844632;"Novak, Karel";zk;"FI B-IN PSK [sem 1, roc 1]
7.;798639;"Dunickova, Dagmar";zk;"FI B-AP SOCI [sem 1,
8.;195660;"Stipsky, Tomas";zk;"FI B-AP BcAP [sem 1, roc
9.;278740;"Fojt, Roman";zk;"FI B-AP INVS [sem 3, roc 2]
10.;236293;"Zachar, Samuel";zk;"FI B-IN UMI [sem 1, roc
```

Pozn. Příklad je „mutovaný“ z důvodu ochrany osobních údajů.

Statistiky o studentech

- výpis křestních jmen (abecedně seřazený)
- statistika studovaných oborů

Výpis křestních jmen

```
def krestni_jmena(jmeno_souboru):
    f = open(jmeno_souboru)
    jmena = []
    for radek in f.readlines():
        m = re.match(r'\d+\.;\d+;"\w+, (\w+)', radek)
        if m: jmena.append(m.group(1))
    jmena.sort()
    print " ".join(jmena)
    f.close()
```

Jiné řešení: použití split

Statistiky oborů

```
def obory(jmeno_souboru):
    f = open(jmeno_souboru)
    vyskyty_oboru = {}
    for radek in f.readlines():
        m = re.search(r'\s(\w+) \[sem', radek)
        if m:
            obor = m.group(1)
            vyskyty_oboru[obor] = \
                    vyskyty_oboru.get(obor, 0) + 1
    f.close()
    for obor in vyskyty_oboru.keys():
        print obor, vyskyty_oboru[obor]
```

Nedostatky: např. studenti studující více oborů.

Regulární výrazy: Rekapitulace

často používané:

`.`	libovolný znak
`\d`	čísla
`\w`	alfanumerické znaky
`\s`	bílé znaky
`+ *`	opakování
`^ $`	začátek, konec řádku
`[ ]`	výběr z možností

Procvičení regulárních výrazů

- `http://tutor.fi.muni.cz`
- úloha Regulární výrazy
- 40 příkladů
- vyhledávání, nahrazování

Analýza textu

statistiky délky slov a vět:

- $\bar{x}$ – průměr
- s – směrodatná odchylka (míra variability)

	slova		**věty**	
	$\bar{x}$	s	$\bar{x}$	s
Starý zákon	4.3	2.3	14.9	7.8
Čapek	4.5	2.5	14.9	13.3
Pelánek	5.9	3.6	13.5	6.9
Wikipedie	5.6	3.0	14.8	8.3

Analýza textu – postup

1. text $\rightarrow$ seznam délek slov (vět)
2. seznam délek $\rightarrow$ statistiky

přímočaré řešení 1. kroku:

- procházet vstup po znacích
- pamatovat si délku aktuálního slova, věty
- speciální znak (mezera, tečka a podobně) $\rightarrow$ aktualizace seznamu

Imitace textu

- vstup: rozsáhlý text
- výstup: náhodně generovaný text, který má „podobné charakteristiky“ jako vstupní text
- imitace na úrovni písmen nebo slov

Náhodnostní imitace vstupního textu

I špiské to pole kavodali pamas ne nebo kdy v Dejný Odm sem uvalini se zabijí s Pan stěží ře, a silobe lo v ne řečekovících blova v nadrá těly jakvěmutelaji rohnutkohonebout anej Fravinci V A pěk finé houty. zal Jírakočítencej ské žil, kdDo jak a to Lorskříže si tomůžu schno mí, kto.

Kterak král kočku kupoval V zemi Taškářů panoval král a zapřisáhl se velikou přísahou že bude pochválena První pán si jí ani nevšimnul zato druhý se rychle shýbl a Jůru pohladil Aha řekl sultán a bohatě obdaroval pana Lustiga koupil od něho telegram z Bombaje v Indii není o nic horší člověk nežli někdo z mých hraček Kdepak mávl Vašek rukou

Základní přístup

1. vstupní text $\Rightarrow$ statistiky textu
2. statistiky $\Rightarrow$ generování náhodného textu

Co jsou vhodné statistiky?

Statistiky textu

- základ: frekvence písmen (slov)
- rozšíření: korelace mezi písmeny (slovy)

příklad: pokud poslední písmeno bylo `a`:

- `e` velmi nepravděpodobné (méně než obvykle)
- `l`, `k` hodně pravděpodobná (více než obvykle)

Implementace

- základní frekvenční analýza – datová struktura slovník
  písmeno $\Rightarrow$ frekvence
- rozšířená analýza – slovník slovníků
  písmeno $\Rightarrow$ { písmeno $\Rightarrow$ frekvence }

generování

- podle aktuálního písmene získám frekvence
- vyberu náhodné písmeno podle těchto frekvencí – „vážená ruleta“

Imitace sofistikovaněji

- Recurrent Neural Networks – dokáží postihnout i složitější aspekty jazyka
- básně, recepty, Wikipedia články, zdrojové kódy, ...

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

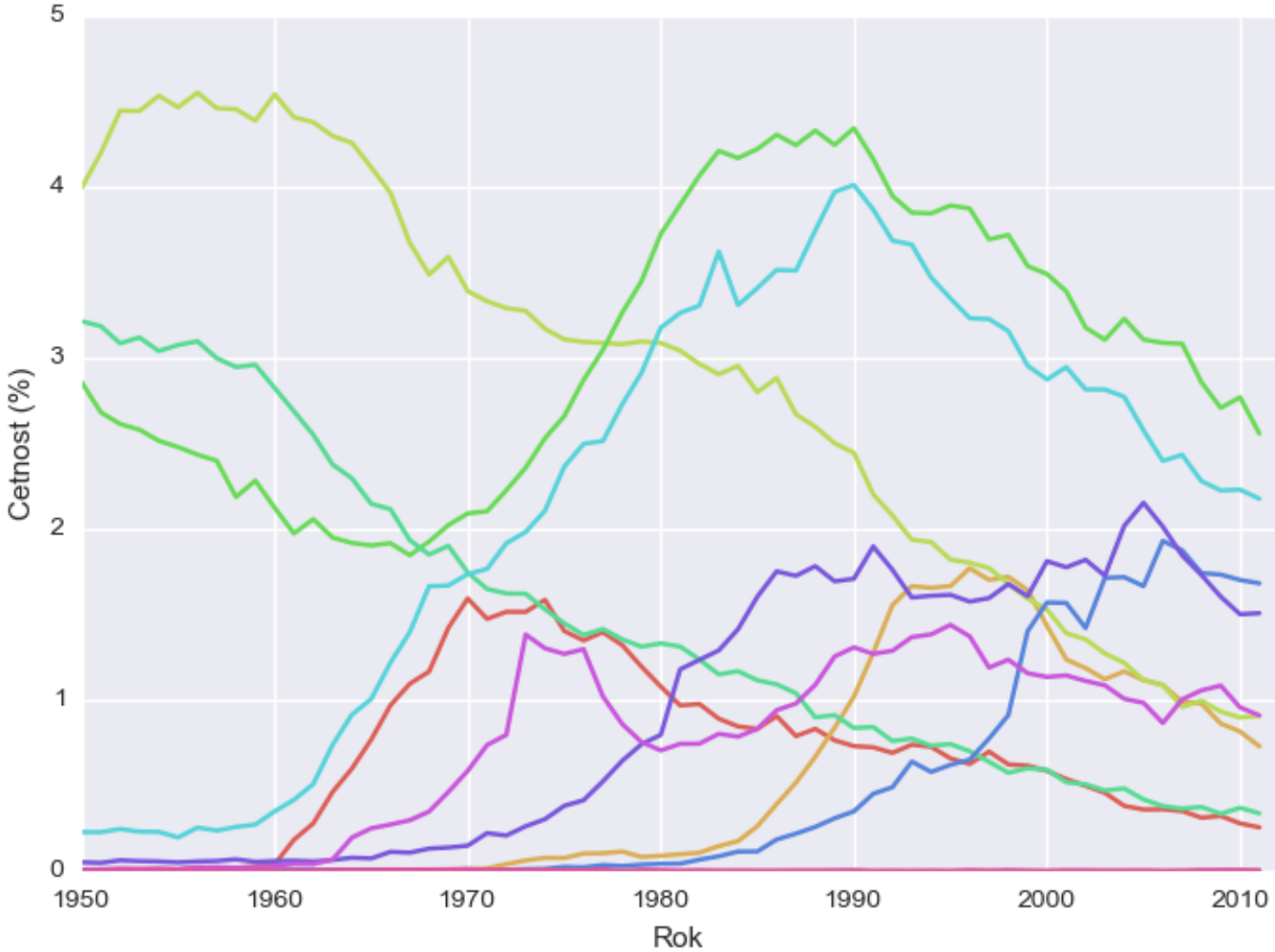
```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.
```

Statistiky jmen

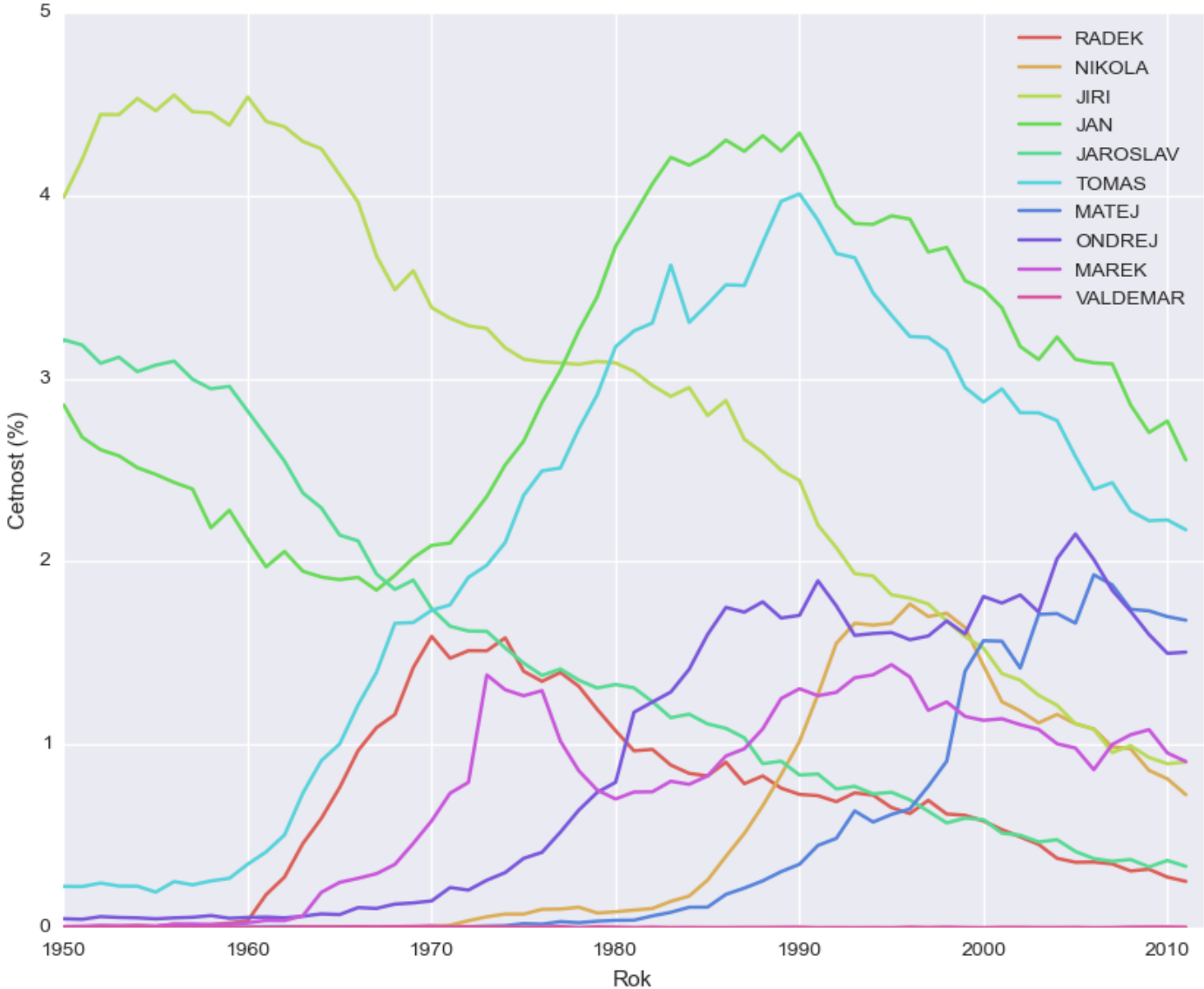
- data: četnosti jmen, příjmení podle roků, krajů, ...
- zdroj: Ministerstvo vnitra ČR
  `http://www.mvcr.cz/clanek/cetnost-jmen-a-prijmeni-722752.aspx`
- XLS – pro zpracování v Pythonu uložit jako CSV (comma-separated values)
- doporučené cvičení
  - snadno zpracovatelné
  - zajímavá data
  - cvičení na vymýšlení otázek
- následuje několik ukázek pro inspiraci ...

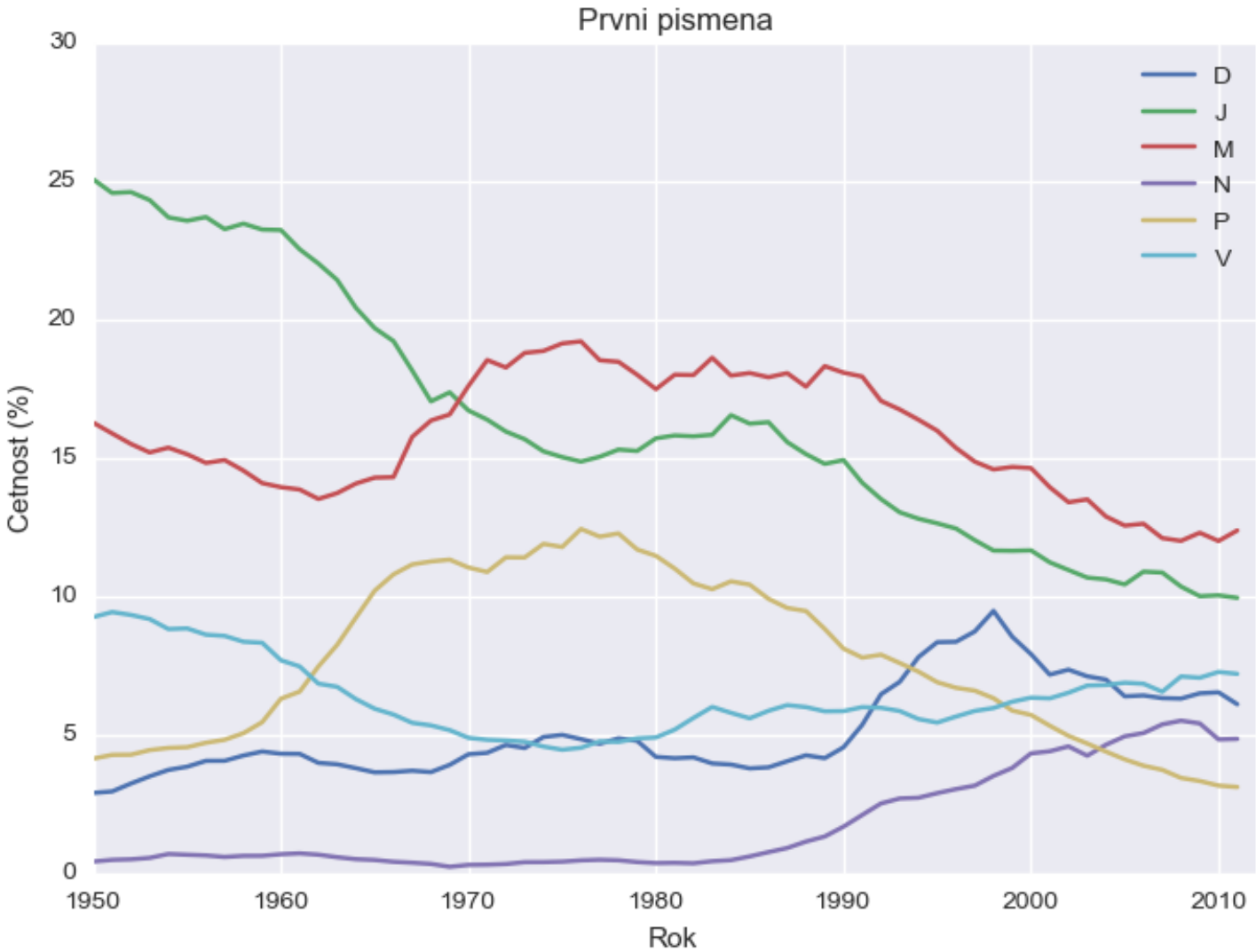
Poznámky ke zpracování

- slovník: jméno $\rightarrow$ seznam výskytů
- CSV – funkce `split` $\rightarrow$ seznam
- normalizace (relativní výskyty jmen) – podělit součtem (pro daný rok)
  - různě velké ročníky
  - neúplná data u starých ročníků



Vyučující IB111: JAN, JAROSLAV, JIŘÍ, MAREK, MATĚJ, NIKOLA, ONDŘEJ, RADEK, TOMÁŠ, VALDEMAR


RADEK
NIKOLA
JIRI
JAN
JAROSLAV
TOMAS
MATEJ
ONDREJ
MAREK
VALDEMAR
Cetnost (%)
Rok
0
1
2
3
4
5
1950
1960
1970
1980
1990
2000
2010


Prvni pismena
Cetnost (%)
Rok
0
5
10
15
20
25
30
1950
1960
1970
1980
1990
2000
2010
D
J
M
N
P
V

Otázky

Co zajímavého můžeme z dat zjistit?

Kladení otázek – důležitá dovednost hodná tréninku.

> Computers are useless. They can only give you answers. (Pablo Picasso)

Identifikace trendů

U kterých jmen nejvíce roste/klesá popularita?

- co to vlastně znamená?
- jak formalizovat?

Nejdelší růst/pokles

Kolik let v řadě roste popularita jména:

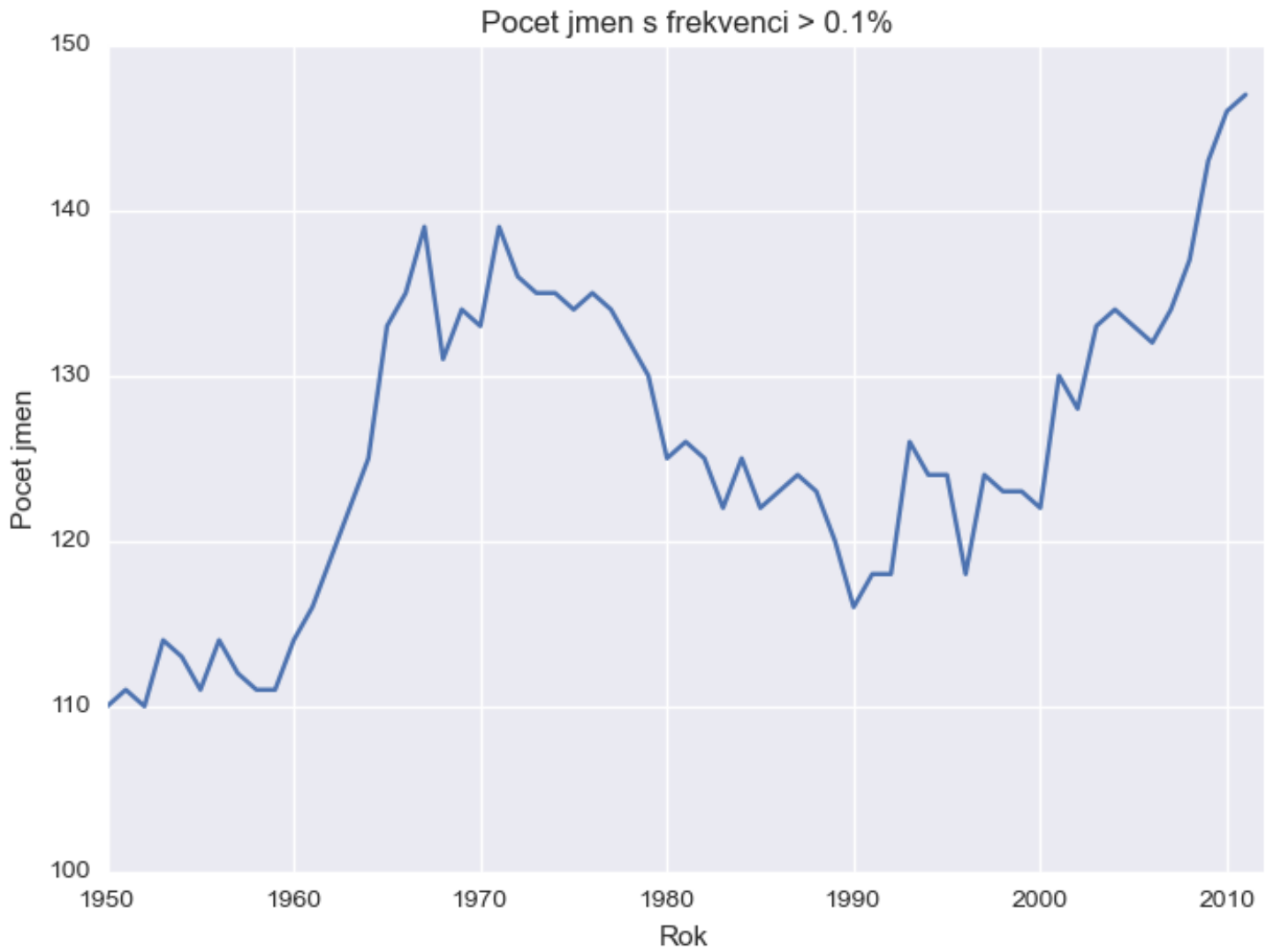
- Tobiáš – 14
- Viktorie, Ella, Sofie – 9
- Elen, Tobias – 8

Kolik let v řadě klesá popularita jména:

- Jana – 26
- Martin – 21
- Petra – 11
- Zdeněk – 9

Největší skok v popularitě za 10 let

- alespoň desetinásobný nárůst popularity:
  Sofie, Elen, Amálie, Ella, Nicol, Nella, Tobias
- pokles alespoň o 60 %:
  Petra, Pavlína, Martina


Pocet jmen s frekvenci > 0.1%
150
140
130
120
110
100
Pocet jmen
1950
1960
1970
1980
1990
2000
2010
Rok

Zdroje zajímavých dat

Otevřená data / Open data

- `http://www.opendata.cz`
- `http://www.otevrenadata.cz`
- `http://www.data.gov`
- `http://data.gov.uk`

Zpracování dat seriózněji

využití existujících knihoven:

- načítání dat ve standardních formátech: HTML, XML, JSON, CSV, ...
- operace s daty: `numpy`, `pandas`
- vizualizace: `matplotlib`

prostředí `ipython` – interaktivní prozkoumávání dat

Shrnutí

- regulární výrazy – obecně užitečný nástroj
- práce s textem, soubory, daty v Pythonu
- příklady
  - kurzovní lístek
  - zpracování seznamu studentů
  - imitace textu
  - statistiky jmen

příště: obrázky