Regulární výrazy, práce s textem

IB111 Úvod do programování skrze Python

2012

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: použití

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

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?

- `\d[A-Z]\d \d\d\d\d`
- `^a.*a$`
- `kocka|pes`

Opakování

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

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
```

Příklady

Jaký je význam následujících výrazů?

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

Regulární výrazy v Pythonu

- knihovna `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

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)

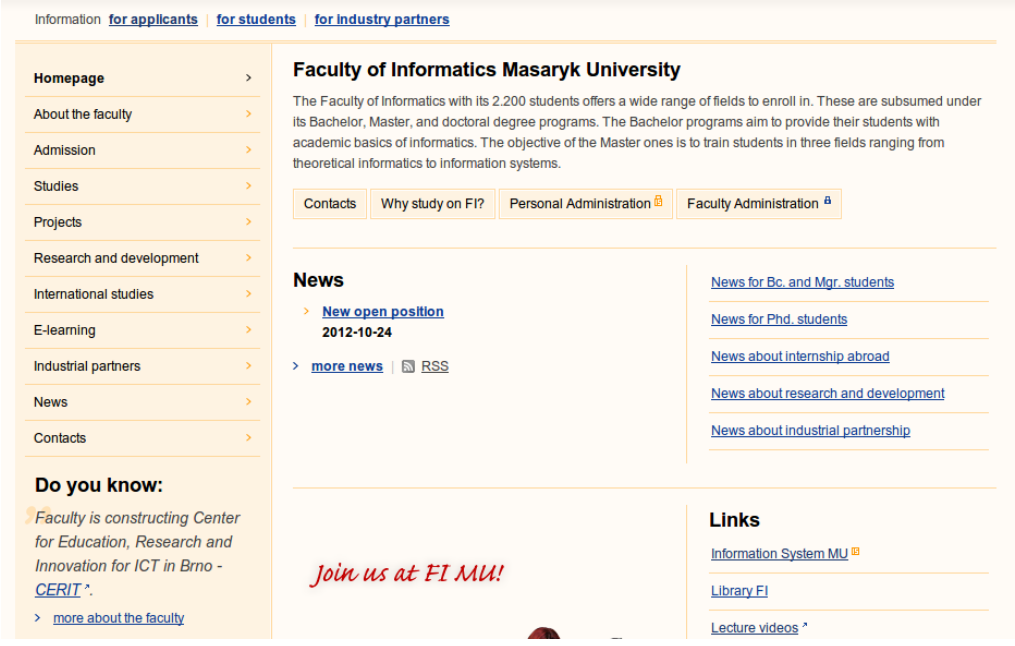
Čtení a zápis:

- `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

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
- `<a href="(.*?)">(.*?)</a>`
- `<a href="(.*?)".*?>(.*?)</a>`
- v čem je rozdíl mezi uvedenými výrazy?
- proč otazníky?

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.

Náhodnostní imitace vstupního 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

Náhodnostní imitace vstupního textu

- základ: vypočítat frekvence písmen (slov) a generovat podle těchto frekvencí
- rozšíření: uvažovat korelace mezi písmeny (slovy)
- příklad: pokud poslední písmeno bylo `a`:
  - `e` velmi nepravděpodobné (méně než obvykle)
  - `l`, `k` hodně pravděpodobná (více než obvykle)

Shrnutí

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