

Praktikum iz programiranja 3



2024/25



Regularni izrazi

- Regularni izrazi su deo svakog naprednog alata za obradu teksta
- Niz znakova koji čine obrazac koji se koristi za pronalaženje svih nizova znakova koji odgovaraju tom obrazcu
- Na primer, poznato je da `*.txt` se koristi kako bi se pronašli svi tekstualni fajlovi-regex ekvivalent toga bi bio `*\*.txt`.

`r'[\w.-]+@(?:\w+[\.])+ \w+ '`

- omogućava pretragu email adrese-bilo koje email adrese
- Obrazac opisuje niz znakova koji se sastoji od niza alfanumeričkih znakova koji mogu sadržati crticu, zatim znak @, niz alfanumeričkih znakova koji se završava tačkom (s tim što se ta sekvenca može ponavljati više puta) i na kraju konačan niz alfanumeričkih znakova.

Regularni izrazi

```
import re
s="""Elektronske adrese je potrebno ostaviti na papiru. Prva od njih je ana@gmail.com,
zatim adresu je ostavio rukovodilac: pet-ar.p@yahoo.com a na kraju i profesor
mar_pe.me@pmf.kg.ac.rs"""

print(re.findall(r'[\w.-]+@(?:\w+[\w.])+\w+',s))
```

```
['ana@gmail.com', 'pet-ar.p@yahoo.com', 'mar_pe.me@pmf.kg.ac.rs']
```

Regularni izrazi

Osnovni pojmovi:

- `.` predstavlja bilo koji znak osim znaka prelaska u novi red
- `[]` koriste se za definisanje skupa znakova
- `^` znak za početak niza znakova
- `$` znak za kraj niza znakova

```
print(re.findall(r'a.', 'anafora'))
```

```
['an', 'af']
```

```
print(re.findall(r'[aeiou].', 'anafora'))
```

```
['an', 'af', 'or']
```

Svako pojavljivanje 2 znaka, od kojih je prvi znak "a" a drugi bilo koji znak osim prelaska u novi red

Svako pojavljivanje 2 znaka, od kojih prvi znak mora biti neki od znakova (a, e, i, o ili u a drugi bilo koji znak osim prelaska u novi red

Regularni izrazi

```
print(re.findall(r'^..', 'anafora'))
```

Bilo koja dva znaka na početku niza znakova

```
['an']
```

```
print(re.findall(r'...$', 'anafora'))
```

Bilo koja tri znaka koja se nalaze na kraju niza znakova

```
['ora']
```

Regularni izrazi

Drugi skup posebnih znakova čine kvantifikatori. Kvantifikatori su simboli koji označavaju koliko puta se neki znak, skup ili grupa znakova može, odnosno mora pojaviti:

- **+** kvantifikator odgovara jednoj ili više pojava
- ***** kvantifikator odgovara nula, jednoj ili više pojava
- **?** kvantifikator odgovara nula ili jednoj pojavi
- **{m,n}** kvantifikator odgovara od m do n pojava

```
print(re.findall(r'[aeiou]+', 'neaktivan'))
```

```
['ea', 'i', 'a']
```

```
print(re.findall(r'.[aeiou]*', 'neaktivan'))
```

```
['nea', 'k', 'ti', 'va', 'n']
```

Jedan ili više samoglasnika, odnosno najduža sekvenca samoglasnika u nizu znakova

Bilo koji znak iza kog sledi nula, jedan ili više samoglasnika

Regularni izrazi

```
re.findall(r'ea?', 'neaktivne')
```

```
['ea', 'e']
```

Traži se pojava znaka “e” iza kog može ali ne mora da sledi znak “a”

```
re.findall(r'[aeiou]{2,5}', 'neaktivaaaan')
```

```
['ea', 'aaaa']
```

Sekvenca samoglasnika dužine od 2 do 5

Regularni izrazi

U Python-ovoj implementaciji regularnih izraza postoji i niz predefinisanih skupovnih znakova

- `\A` predstavlja poklapanje na početku stringa
- `\b` predstavlja poklapanje na početku ili kraju, dok `\B` poklapanje koje nije na početku ili kraju
- `\d` predstavlja cifru, odnosno `[0-9]`, dok `\D` predstavlja komplement tog skupa, `[^0-9]`
- `\s` predstavlja skup svih praznina, dok `\S` je komplement tog skupa
- `\w` predstavlja skup alfanumeričkih znakova, dok je `\W` je komplement tog skupa
- `{m,n}` kvantifikator odgovara od m do n pojava
- `\Z` predstavlja poklapanje na kraju stringa

Funkcija `findall()` može imati i treći argument:

- `re.DOTALL` čini da tačka predstavlja svaki znak, pa i prelazak u novi red
- `re.MULTILINE` čini da znakovi `^` i `$` predstavljaju početak, odnosno kraj svakog reda
- `re.UNICODE` čini da skup alfanumeričkih znakova sadrži sve Unicode alfanumeričke znakove

Primeri

Pronaći dva znaka od kojih je prvi znak "s" a drugi bilo koji znak pa i znak za prelazak u novi red:

```
print(re.findall(r's.', 'danas\nsutra', re.DOTALL))  
['s\n', 'su']
```

Pronaći prve znakove na početku svake linije:

```
s="" "Elektronske adrese je potrebno ostaviti na papiru. Prva od njih je  
ana@gmail.com,  
zatim adresu je ostavio rukovodila: pet-ar.p@yahoo.com a na kraju i  
profesor  
mar_pe.me@pmf.kg.ac.rs""  
print(re.findall(r'^.', 'danas\nsutra', re.MULTILINE))  
print(re.findall(r'^.', s, re.MULTILINE))  
['d', 's']  
['E', 'z', 'm']
```

Regularni izrazi - funkcije

- `findall()` – vraća listu poklapanja, ukoliko nema poklapanja vraća praznu listu
- `search()` – vraća string koji zadovoljava obrazac, ukoliko ima više takvih stringova, vraća prvi, ukoliko ne pronađe varća `None`

```
x = re.search("\s", s)
print("Prva praznina se nalazi na poziciji:", x.start())
      Prva praznina se nalazi na poziciji: 11
```

- `sub()` – menja sva poklapanja sa stringom po izboru

```
txt = "Danas je lep dan"
x = re.sub("\s", "XXX", txt)      DanasXXXjeXXXlepXXXdan
print(x)
```

Dodatno kao argument se može navesti broj zamena

```
x = re.sub("\s", "XXX", txt, 2)   DanasXXXjeXXXlep dan
print(x)
```

Regularni izrazi - funkcije

- Kada funkcija `search()` vrati ponuđen string postoje i dodatne opcije

```
x = re.search(r"\badres\w+", s)
print("Pocetak: ", x.start())
print("Kraj: ", x.end())
print("Pocetak-kraj: ", x.span())
print("Ceo string: ", x.string)
print("String preklapanja: ", x.group())
```

```
Pocetak: 12
Kraj: 18
Pocetak-kraj: (12, 18)
Ceo string: Elektronske adrese je potrebno ostaviti na papiru. Prva od njih je
ana@gmail.com,
zatim adresu je ostavio rukovodila: pet-ar.p@yahoo.com a na kraju i profesor
mar_pe.me@pmf.kg.ac.rs
Pocetak: adrese
```

Primer

- Napisati program koji učitava tekst i pronalazi pojavljivanja novčanih iznosa u okviru teksta. Novčanim iznosom smatra se svaki uzastopni niz cifara praćen nizom rsd (bez razmaka između poslednje cifre i niza rsd). Program treba da ispiše ukupnu sumu svih novčanih iznosa pomenutih u tekstu. Izlaz sadrži ukupnu sumu praćenu oznakom rsd (bez razmaka između poslednje cifre i niza rsd).

Ulaz	Izlaz
23.4.2016. skuplja se dobrovoljni prilog za izgradnju muzicke sobe na IMI-ju,4 profesora su dala po 500rsd, specijalan prilog stigao je od dekana 1200rsd. Broj indeksa 66/14 dao je 320rsd, pronasli so i ostavljenih 20rsd, kao i metalne kovanice od po 5rsd i 10rss.	2045rsd

s = ""23.4.2016. skuplja se dobrovoljni prilog za izgradnju muzicke sobe na IMI-ju,4 profesora su dala po 500rsd, specijalan prilog stigao je od dekana 1200rsd.Broj indeksa 66/14 dao je 320rsd, pronasli so i ostavljenih 20rsd, kao i metalne kovanice od po 5rsd i 10rss. ""

```
suma = 0
i = 0
n = len(s)
broj = 0
while i < n:
    if (s[i] >= '0') and (s[i] <= '9'):
        broj = broj*10 + ord(s[i]) - ord('0')
        k = 1    #indikator
    else:
        if k==1:
            if s[i:i+3]=='rsd':
                suma +=broj
                i +=2
            broj=0
            k=0
        i +=1
print(suma)
```

```
regex = re.compile(r'\d+(?=rsd)')
L = regex.findall(s)
brojevi= list(map(int, L))

print(sum(brojevi),"rsd")
```

Primer

- U tekstu može da se nađe izraz oblika broj1! ili izraz $\text{broj1}^{\text{broj2}}$, gde su broj1 i broj2 celi brojevi, operacija $!$ je faktoriyel, a $^$ označava stepen. Ukoliko se u tekstu pojave izrazi ovog oblika, potrebno ih je zameniti rezultatom odgovarajućih operacija.

Ulaz	Izlaz
Temperatura za ponedeljak je 4!. Vazdusni pritisak se procenjuje na 2^{10} . Sansa za pojavu kise je !50%. Ponesite kisobran!	Temperatura za ponedeljak je 24. Vazdusni pritisak se procenjuje na 1024. Sansa za pojavu kise je !50%. Ponesite kisobran!

```
import re
import math as m
s = """Temperatura za ponedeljak je 4!.
Vazdusni pritisak se procenjuje na 2^10.
Sansa za pojavu kise je !50%. Ponesite kisobran!
"""
```

```
regex = re.compile(r'\d+!|\d+\^\d+')
L = regex.findall(s)
for x in L:
    if x.find("!")!=-1:
        br = m.factorial(int(x[:-1]))
        print(br)
        s = s.replace(x,str(br))
    else:
        poz = x.find("^")
        br1 = int(x[:poz])
        br2 = int(x[poz+1:])
        print(br1,br2)
        s = s.replace(x,str(br1**br2))
print(s)
```

Primer

- Napisati funkciju Bullet koja ispituje da li niz karakera, koji prihvata kao argument, predstavlja ispravnu liniju u nabrajanju. Ispravna linija u nabrajanju je ako
 - počinje brojem (bar jedna cifara),
 - potom se nalazi bar jedan od simbola slova, cifara i SPACE karakter u proizvoljnom redosledu i
 - završava se jednim od znakova „.“, „?“, „!“, iza kojih su dozvoljeni samo SPACE karakteri, ali mogu biti izostavljeni.

Ulaz	Izlaz
1 Prva linija. 2 linija Trecu linija! 55? 41. C!	1 Prva linija. 55? 41. C!


```

import re
def Bullet(x):
    regex = re.compile(r'^(\d+)([\w\d ]+)([.\?\\!]+)([ ]?)')
    L = regex.findall(x)
    if L==[]:
        return False
    else:
        return True
s = """1Prva linija.
2 linija
Treci linija!
55?
41. C!"""
L=s.split("\n")
print(L)

nova_L=[]
for x in L:
    if Bullet(x):
        nova_L = nova_L + [x]
print(nova_L)

```