Regularni izrazi, Izuzeci

Univerzitet u Beogradu
Elektrotehnički fakultet

Sadržaj

- *Raw* stringovi
- Funkcije, metode i *Match* objekti
- Specijalni karakteri i sekvence
  - Predstavljanje skupova
  - Predstavljanje ponavljanja
  - Grupisanje i grupe
  - Pozicija unutar stringa
  - Izbor
- Izuzeci

Raw stringovi

- Stringovi mogu sadržati specijalne karaktere kao što su \n, \t, \r, \\, \"...
- *Raw* stringovi tretiraju ovakve karaktere kao bilo koji drugi karakter (nemaju specijalno značenje)

```
string = "Fića\tTea\nSaša\rJela"
print(string)
```

```
Fića	Tea
Jela
```

```
raw_string = r"Fića\tTea\nSaša\rJela"
print(raw_string)
```

```
Fića\tTea\nSaša\rJela
```

Funkcije i metode

- Za kreiranje izraza koristi se funkcija **compile()**

```
import re

sablon_rec = re.compile(r"rec")
sablon_od = re.compile(r"od")
tekst = "recenica od cetiri reci"
```

- Za pretragu stringa koriste se metode **match()**, **search()**, **finditer()**

```
print(sablon_rec.match(tekst))                # <re.Match object; span=(0, 3), match='rec'>
print(sablon_od.match(tekst))                 # None
```

```
print(sablon_rec.search(tekst))               # <re.Match object; span=(0, 3), match='rec'>
print(sablon_od.search(tekst))                # <re.Match object; span=(9, 11), match='od'>
```

```
for pogodak in sablon_rec.finditer(tekst):    # <re.Match object; span=(0, 3), match='rec'>
    print(pogodak)                            # <re.Match object; span=(19, 22), match='rec'>
```

Match objekti

- Sadrže informacije o rezultatima pretrage
- Ukoliko nema pogotka, umesto *Match* objekta povratna vrednost je **None**
- Metode **start()**, **end()** i **span()** vraćaju informacije o poziciji pogotka unutar stringa

```
import re

sablon = re.compile(r"ovu")
tekst = "pronadji ovu rec"

pogodak = sablon.search(tekst)      # <re.Match object; span=(9, 12), match='ovu'>
if pogodak:
    pocetak = pogodak.start()       # 9
    kraj = pogodak.end()            # 12
    print(tekst[pocetak:kraj])      # ovu
```

1. zadatak

- Na programskom jeziku Python sastaviti program koji ispisuje pozicije pojavljivanja reči u tekstu.

```
Unesite tekst: mis uz pusku mis niz pusku
Unesite reč za pretragu: pusku
Reč se javlja na poziciji 7
Reč se javlja na poziciji 21
```

```
import re

text = input("Unesite tekst: ")
pattern = re.compile(input("Unesite reč za pretragu: "))
for match in pattern.finditer(text):
    print("Reč se javlja na poziciji {}".format(match.start()))
```

Specijalni karakteri i sekvence

- Specijalni karakteri:

\	.	?	*	+	^	$
|	(	)	[	]	{	}

- Specijalne sekvence:

\d	\D	\w	\W	\s	\S
\A	\Z	\B	\b		

Predstavljanje skupova (1)

- Kreiranje skupa karaktera:
  - Za kreiranje se koriste specijalni karakteri **[ ]** unutar kojih se navode elementi skupa

Primeri	Opis
[abc]	Karakteri se mogu nabrajati pojedinačno, a navode se jedan za drugim bez razmaka
[a-c]	Može se zadati opseg karaktera korišćenjem znaka **–**
[^abc]	Mogu se izabrati samo karakteri koji se nalaze van skupa korišćenjem znaka **^** na početku skupa
[.?*+(}]	Specijalni karakteri nemaju specijalno značenje unutar skupa

Predstavljanje skupova (2)

- Korišćenje postojećih skupova:

Specijalni karakter ili sekvenca	Opis	*Flag*	Opis
.	Bilo koji karakter sem novog reda	**DOTALL**	Može i novi red
\d	Bilo koja decimalna cifra	**ASCII**	[0-9]
\D	Sve što nije decimalna cifra		[^0-9]
\w	Bilo koji *word* karakter		[a-zA-Z0-9_]
\W	Sve što nije *word* karakter		[^a-zA-Z0-9_]
\s	Bilo koji *whitespace* karakter		[\t\n\r\f\v]
\S	Sve što nije *whitespace* karakter		[^ \t\n\r\f\v]

2. zadatak (1/2)

- Na programskom jeziku Python sastaviti funkciju koja pronalazi godine izbora u zvanje saradnika u nastavi i ispisuje ih na standardnom izlazu sortirane u opadajućem poretku.

v06z02.csv

```
Milana, 2015
Aleksa, 2019
Jovan, 2017
Vladimir, 2016
```

```
Unesite naziv datoteke: v06z02.csv
Godine izbora u zvanje: 2019, 2017, 2016, 2015
```

2. zadatak (2/2)

```
import re

def find_and_sort(filename):
    with open(filename) as file:
        text = file.read()
    pattern = re.compile(r"\d\d\d\d")
    years = pattern.findall(text)
    years.sort(reverse=True)
    return years

filename = input("Unesite naziv datoteke: ")
years = find_and_sort(filename)
print("Godine izbora u zvanje:", ", ".join(years))
```

- Da li se može napisati kompaktniji regularni izraz?

Predstavljanje ponavljanja (1)

- Kreiranje ponavljanja karaktera:
  - Za kreiranje se koriste specijalni karakteri **{ }** unutar kojih se navodi broj ponavljanja

Primer	Opis
{m}	Prethodno navedeni karakter se ponavlja tačno **m** puta
{m,n}	Prethodno navedeni karakter se ponavlja najmanje **m** a najviše **n** puta
{m,}	Prethodno navedeni karakter se ponavlja najmanje **m** puta, ne postoji gornja granica
{,n}	Prethodno navedeni karakter se ponavlja najviše **n** puta, ne postoji donja granica

Predstavljanje ponavljanja (2)

- Korišćenje postojećih ponavljanja:
  - Koriste se specijalni karakteri **?**, ***** i **+**

Specijalni karakter	Ekvivalentno sa	Opis
?	**{0,1}**	Prethodno navedeni karakter se pojavljuje jednom ili nijednom
*	**{0,}**	Prethodno navedeni karakter se ponavlja proizvoljan broj puta, ne postoje ni gornja ni donja granica
+	**{1,}**	Prethodno navedeni karakter se ponavlja najmanje jedanput, ne postoji gornja granica

Grupisanje i grupe

- Za grupisanje se koriste specijalni karakteri **( )**

```
import re

tekst = "abababab"
sablon = re.compile(r"(ab)*")
print(sablon.match(tekst))          # <re.Match object; span=(0, 8), match='abababab'>
```

```
sablon = re.compile(r"(a)(b)")
pogodak = sablon.match(tekst)
print(pogodak.group())              # ab
print(pogodak.group(0))             # ab
print(pogodak.group(1))             # a
print(pogodak.group(2))             # b
print(pogodak.group(1, 2))          # ('a', 'b')
print(pogodak.groups())             # ('a', 'b')
```

```
sablon = re.compile(r"(a(b))")
pogodak = sablon.match(tekst)
print(pogodak.group(1))             # ab
print(pogodak.group(2))             # b
```

3. zadatak (1/2)

- Na programskom jeziku Python sastaviti funkciju koja prosleđenoj datoteci menja format u kom su zapisani podaci o studentu.

v06z03.csv

```
162/2018, Dorotea Pavlović
28/2019, Filip Ristić
413/2019, Jelena Petrović
```

v06z03.csv

```
18/162, Pavlović Dorotea
19/28, Ristić Filip
19/413, Petrović Jelena
```

3. zadatak (2/2)

```
import re

def rewrite(filename):
    with open(filename, "r+", encoding="UTF-8") as file:
        text = file.read()
        pattern = re.compile(
            r"(\d{1,4})/\d{2}(\d{2}),\s+(\w+)\s+(\w+)"
        )
        new_text = pattern.sub(r"\2/\1, \4 \3", text)
        file.seek(0)
        file.truncate()
        file.write(new_text)

filename = input("Unesite naziv datoteke: ")
rewrite(filename)
```

Pozicija unutar stringa

- Može se eksplicitno tražiti određena pozicija unutar stringa:

Specijalni karakter ili sekvenca	Opis
^	Početak stringa (ili reda ako je posavljen flag **MULTILINE**)
$	Kraj stringa (ili reda ako je postavljen flag **MULTILINE**)
\A	Početak stringa
\Z	Kraj stringa
\b	Prazan string na početku ili na kraju reči
\B	Prazan string svuda osim na početku i na kraju reči

Izbor

- Za izbor se koristi specijalni karakter |

```
import re

sablon = re.compile(r"a|b")
print(sablon.match("a"))              # <re.Match object; span=(0, 1), match='a'>
print(sablon.match("b"))              # <re.Match object; span=(0, 1), match='b'>
print(sablon.match("c"))              # None
```

```
sablon = re.compile(r"a|b|c")
print(sablon.match("a"))              # <re.Match object; span=(0, 1), match='a'>
print(sablon.match("b"))              # <re.Match object; span=(0, 1), match='b'>
print(sablon.match("c"))              # <re.Match object; span=(0, 1), match='c'>
```

```
sablon = re.compile(r"(a|b)c")
print(sablon.match("ac"))             # <re.Match object; span=(0, 2), match='ac'>
print(sablon.match("bc"))             # <re.Match object; span=(0, 2), match='bc'>
print(sablon.match("abc"))            # None
print(sablon.match("ab"))             # None
print(sablon.match("c"))              # None
```

4. zadatak (1/3)

- Na programskom jeziku Python sastaviti program koji pronalazi i ispisuje sve nekorektne e-mail adrese iz zadatate datoteke. Smatrati da korektna adresa ima oblik **local@domain** pri čemu važi sledeće:
  - Oba dela adrese (local i domain) smeju da sadrže slova, brojeve, tačke, crtice i pluseve
  - Oba dela adrese moraju da počinju slovom
  - Prvi deo (local) ne sme da ima više od 64 karaktera
  - Drugi deo (domain) mora da se završi tačkom praćenom sa dva ili tri slova

4. zadatak (2/3)

v06z04.txt

```
fica@student.etf.rs
dora.gmail.com
5a5a@yahoo.com
pavlovic.tea@student.etf.bg.ac.rs
i++@yahoo.com
jela_petrovic@gmail.com
a"b(c)d,e:f;g<h>i[j\k]l@etf.rs
d@j.edu
f121p-h0m3@dva+2.mkv
aleksandar@mail.mp3
a12345678901234567890123456789012345678901234567890123436++@etf.bg.ac.rs
```

4. zadatak (3/3)

```
dora.gmail.com
5a5a@yahoo.com
a"b(c)d,e:f;g<h>i[j\k]l@etf.rs
aleksandar@mail.mp3
a12345678901234567890123456789012345678901234567890123456789
23436++@etf.bg.ac.rs
```

```
import re

pattern = re.compile(
    r"^[a-zA-Z][\w.+-]{0,63}@[a-zA-Z][\w.+-]*.[a-zA-Z]{2,3}$"
)
with open("v06z04.txt") as mails:
    for mail in mails:
        if not pattern.match(mail):
            print(mail.strip())
```

5. zadatak (1/2)

- Na programskom jeziku Python sastaviti program koji izdvaja različite reči iz HTML datoteke. Ignorisati sadržaj zadat između oznaka <>.

v06z05.html

```
<html>
   <head>
      <title>Jednostavan HTML dokument</title>
   </head>
   <body>
         <p> Prvi paragraf.</p>
         <p>Drugi paragraf!</p>
   </body>
</html>
```

5. zadatak (2/2)

```
Prvi Jednostavan HTML dokument paragraf Drugi
```

```
import re

with open("v06z05.html") as html:
    content = html.read()
pattern = re.compile(r"<[^<>]+>|[\s,.!?]+")
words = pattern.split(content)
words = set(words)
words.remove("")
print(*words, sep=" ")
```

- Zašto reči nisu uređene po redosledu pojavljivanja unutar datoteke?

Izuzeci (1)

- Greške otkrivene u toku izvršavanja programa:
  - **TypeError**, **IndexError**, **KeyError**, ...

```
a = 10
b = input("b: ")
c = a + b
```

```
b: 20
Traceback (most recent call last):
  File "...", line 3, in <module>
    c = a + b
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
lista = [1, 2, 3]
lista[3] = 4
```

```
Traceback (most recent call last):
  File "...", line 2, in <module>
    lista[3] = 4
IndexError: list assignment index out of range
```

Izuzeci (2)

- Rezevisane reči **try**, **except**, **finally**

```
while True:
    try:
        broj = int(input("Unesite broj: "))
        break
    except ValueError:
        print("Niste uneli broj! Pokušajte ponovo.")
```

```
try:
    datoteka = open("datoteka.txt")
    datoteka.write("Datoteka nije otvorena za pisanje.")
finally:
    datoteka.close()
```

- Rezevisana reč **raise**

```
if broj < 0:
    raise Exception("Negativan broj!")
```

6. zadatak (1/5)

- Na programskom jeziku Python sastaviti program koji računa ukupan broj sedišta na svim letovima do željenog grada. Dostupne su sledeće datoteke:
  - Datoteka *v06z06d01.txt* sadrži informacije o modelima aviona. Jedan red datoteke sadrži oznaku modela aviona i broj sedišta u avionu odvojene dvotačkom.
  - Datoteka *v06z06d02.txt* sadrži informacije o odlaznim letovima sa aerodroma *Nikola Tesla*. Jedan red datoteke sadrži ime grada, broj leta i oznaku modela aviona. Ime grada i broj leta su odvojeni dvotačkom, a broj leta i oznaka modela zarezom. Može postojati više letova ka istom gradu.

6. zadatak (2/5)

v06z06d01.txt

```
Airbus A319: 156
Airbus A320: 180
ATR 72: 74
Bombardier CRJ900: 90
Embraer 195: 122
```

v06z06d02.txt

```
Podgorica: JU 172, Airbus A319
Munich: LH 1729, Bombardier CRJ900
Geneva: DS 1438, Airbus A320
Podgorica: JU 170, ATR 72
Podgorica: YM 101, Embraer 195
Munich: LH 1723, Airbus A319
```

6. zadatak (3/5)

```
Unesite grad: Tel Aviv
Nema letova za željeni grad.
Unesite grad: Podgorica
Ukupan broj sedišta: 352
```

```
import re

def models(airplanes_filename):
    models = {}
    with open(airplanes_filename) as airplanes:
        pattern = re.compile(r"\s*[:]\s*")
        for airplane in airplanes:
            airplane = airplane.strip()
            model, seats = pattern.split(airplane)
            models[model] = int(seats)
    return models
```

6. zadatak (4/5)

```
def cities(departures_filename, models):
    cities = {}
    with open(departures_filename) as departures:
        pattern = re.compile(r"\s*[:,]\s*")
        for departure in departures:
            departure = departure.strip()
            city, flight, model = pattern.split(departure)
            city = city.lower()
            seats = models[model]
            try:
                cities[city] += seats
            except KeyError:
                cities[city] = seats
    return cities

def seats(cities, city):
    return cities[city]
```

6. zadatak (5/5)

```
models = models("v06z06d01.txt")
cities = cities("v06z06d02.txt", models)
while True:
    try:
        city = input("Unesite grad: ").lower()
        if not city:
            break
        total = seats(cities, city)
        print("Ukupan broj sedišta: {}".format(total))
    except KeyError:
        print("Nema letova za željeni grad.")
```

7. zadatak (1/4)

- Na programskom jeziku Python sastaviti program koji rangira korisnike po vremenu provedenom u video razgovorima na nekoj socijalnoj mreži. Korisnici se rangiraju u nerastućem redosledu.
- Dostupna je datoteka koja sadrži podatke o video razgovorima na toj socijalnoj mreži. Svaki red sadrži imena korisnika (dva ili više), datum razgovora (**dd.mm.gggg.**) i vremenski interval trajanja razgovora (**hh:mm-hh:mm**). Podaci su međusobno odvojeni razmacima.

7. zadatak (2/4)

v06z07.txt

```
Fica Tea Sasa 01.11.2019. 10:11-11:02
Sasa Fica 01.11.2019. 05:00-05:15
Tea Fica Jela 10.12.2019. 12:03-15:47
```

```
Fica > Tea > Jela > Sasa
```

```
import re

times = {}
pattern = re.compile(
    r"(([a-zA-Z]+\s+){2,})"
    r"(\d{1,2}\.){2}\d{4}\.\s+"
    r"(\d{1,2}:\d{1,2})-(\d{1,2}:\d{1,2})"
)
```

7. zadatak (3/4)

```
with open("proba.txt", encoding="UTF-8") as calls:
    for call in calls:
        for i in pattern.finditer(call):
            users = i.group(1).split()
            start = [int(x) for x in i.group(4).split(":")]
            end = [int(x) for x in i.group(5).split(":")]
            time = (end[0]-start[0])*60 + end[1]-start[1]
            for user in users:
                if not user in times:
                    times[user] = 0
                times[user] += time
times = sorted(times, key=times.get, reverse=True)
print(*times, sep=" > ")
```

- Izmeniti regularni izraz da podržava č, ć, đ, š, i ž.

Pregled

Pregled – Funkcije i metode

- Funkcije iz modula re:
  - compile()
  - match()
  - search()
  - finditer()
  - findall()
  - split()
  - sub()
- Funkcije za rad sa datotekama:
  - truncate()

Pregled – Ključne reči

- Za rad sa izuzecima:
  - try, except, finally
  - raise

Pregled – Operatori

- Aritmetički operatori
- Bitski operatori
- Operatori dodele
- Relacioni operatori
- Logički operatori
- Operatori identiteta
- Operatori pripadnosti

Pregled – Tipovi podataka

- Int
- Float
- Complex
  - Numerički tipovi
- Boolean
  - Logički tip
- String
- List
- Tuple
- Set
- Dict
  - Kolekcije

Pregled – Funkcije i metode

- Funkcije iz modula re:
  - compile()
  - match()
  - search()
  - finditer()
  - findall()
  - split()
  - sub()
- Funkcije za rad sa datotekama:
  - truncate()

Pregled – Ključne reči

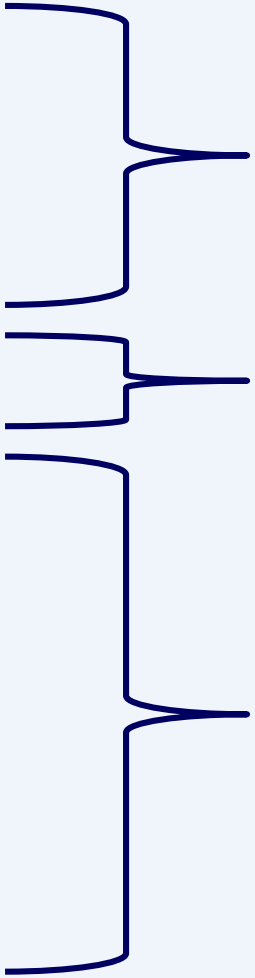
- Za rad sa izuzecima:
  - try, except, finally
  - raise

Pregled – Operatori

- Aritmetički operatori
- Bitski operatori
- Operatori dodele
- Relacioni operatori
- Logički operatori
- Operatori identiteta
- Operatori pripadnosti

Pregled – Tipovi podataka

- Int
- Float
- Complex
- Boolean
- String
- List
- Tuple
- Set
- Dict



- Numerički tipovi
- Logički tip
- Kolekcije

Literatura – Knjige

- M. Kovačević, Osnove programiranja u Pajtonu, Akademska misao, Beograd, 2017.
- M. Lutz, Learning python: Powerful object-oriented programming, 5th edition, O'Reilly Media, Inc., 2013.
- J. Zelle, Python Programming: An Introduction to Computer Science, 3rd Ed., Franklin, Beedle & Associates, 2016.
- D. Beazley, B. K. Jones, Python Cookbook, 3rd edition, O'Reilly Media, 2013.
- A. Downey, J. Elkner, C. Meyers, How To Think Like A Computer Scientist: Learning With Python, free e-book

Literatura – *Online* izvori

- Python 3 documentation, https://docs.python.org/3/index.html
- Learn Python, Basic tutorial, https://www.learnpython.org/
- TutorialsPoint, Python tutorial https://www.tutorialspoint.com/python/index.htm
- W3Schools, Python tutorial https://www.w3schools.com/python/
- GeeksforGeeks, Python programming language https://www.geeksforgeeks.org/python-programming-language/