Lokaskil 3

Dæmi 1

Útskýring á reglulegri segð fyrir kennitölu einstaklinga:

Reglulega segðin fyrir kennitölu einstaklinga er: `\b(0[1-9]|[12][0-9]|3[01])(0[1-9]|1[0-2])(\d{2})-([2-9][0-9])(\d{1})([890])\b`

skilyrðin fyrir kennitölu einstaklinga eru:

- fyrstu tveir tölustafirnir eru á bilinu 01-31:
  Reglulega segðin `0[1-9]|[12][0-9]|3[01]`: ef að 1. talan er 0 getur 2. talan verið 1-9, ef að 1. talan er 1 eða 2 getur 2. talan verið 0-9, og ef að 1. talan er 3 getur seinni talan verið 0-1
- næstu tveir eru á bilinu 01-12
  Regluklega segðin `0[1-9]|1[0-2]`: ef að fyrri talan er 0 getur seinni verið á bilinu 1-9, en ef að hún er 1 getur seinni verið 1 eða tveir.
- síðustu tveir geta verið hvaða tölustafir sem er:
  Reglulega segðin `\d{2}`: hvaða tveir tölustafir á bilinu 0-9
- tölustafir 7 og 8 er tala frá og með 20
  Reglulega segðin `[2-9][0-9]`: fyrri stafurinn getur verið á bilinu 2-9 og seinni á bilinu 0-9
- tölustafur 9 er hvaða tölustafur sem er á bilinu 0-9 Reglulega segðin `\d{1}`
- Síðasti stafurinn getur verið 8, 9 eða 0 Reglulega segðin `[890]`

Síðan settum við hverja reglulegu segð inn í sviga og settum segðina saman með bandstriki eftir fyrstu 6 tölustafina eins og kennitölurnar voru settar upp í textaskránni sem að við notuðum. settum `\b` fyrir framan og aftan segðina til að skilgreina orðamörkin

Útskýring á reglulegri segð fyrir kennitölu fyrirtækja

Reglulega segðin sem að við notuðum fyrir kennitölur fyrirtækja er: `\b[4-9]\d{5}-\d{4}\b`

Fyrir kennitölur fyrirtækja er reglulega segðin töluvert einfaldari þar sem að þau fá úthlutað kennitölu þar sem að eina skilyrðið er að fyrsti stafurinn er 4 eða hærri. Til þess að passa að fyrsti stafurinn yrði ekki minni en 4 byrjaði reglulega segðin á `[4-9]` sem passar við hvaða staf á bilinu 4-9 þar á eftir koma 5 tölustafir á bilinu 0-9 með reglulegu segðinni `d{5}` Svo táknum við bandstrik þar sem að kennitölurnar eru skrifaðar þannig í textanum sem unnið var með í lokin koma svo aftur 4 handahófskenndar tölur á bilinu 0-9 með reglulegu segðinni `\d{4}` Við settum þetta allt saman og `\b` fyrir framan og aftan til þessað skilgreina orðamörkin.

Kóði Útskýring

Kóðinn er í regex_kt.py skránni. Þar skilgreindum reglulegu segðina fyrir einstaklinga og fyrirtæki og nota svo re.findall(mynstur_einstaklingar, lina) til þess að leita af öllum kennitölum einstaklings, eða fyrirtækja í hverri línu sem að passa við reglulegu segðina. Þar sem að reglulega segðin hjá kennitölum einstaklinnga var skipt niður í búta setti ég þá ala saman í streng til þess að kennitalan prentist á réttu formi.

Dæmi 2

Regluleg segð fyrir email

Reglulega segðin sem að við notuðum fyrir email er: `\b[A-Za-z0-9.]+@(?!.*\.\.)[A-Za-z0-9.-]+\.[A-Za-z]{2,3}\b`

- `\b` í byrjun skilgreinir orðamörkin þannig að tölvupóstfangið er ekki hluti af lengri streng.
- `[A-Za-z0-9.]` þar á eftir merkir að allt fyrir fyrri hluti netfangsins má innihalda alla bókstafi, stóra: `A-Z` og litla: `a-z`, alla tölustafi frá 0-9:`0-9` og `.`
- `+` þar á eftir það sé a.m.k. eitt eða fleiri af táknunnum í hornklofanum á undan
- `@` þá kemur "@"
- `(?!.*\.\.)` þetta passar að það komi ekki tvöfaldur punktur neins staðar fyrir eftir núverandi stöðu í strengnum
  - `(?! mynstur)` er negative lookahead og þýðir að eitthvað mynstur sem er skilgreint inni í sviganum má ekki koma fram seinna í strengnum.
  - `.` táknar hvaða staf sem er og `*` táknar "0 eða fleiri skipti", þ.e. mynstrið má hvorki koma strax eða eftir einnhverja stafi
  - `\.\.` merkir tvöfaldan punkt eða ..
- `[A-Za-z0-9.-]+` þar á eftir þýðir að næsti hluti netfangsins má innihalda alla bókstafi, stóra: `A-Z` og litla: `a-z`, alla tölustafi frá 0-9:`0-9`, `.` og `.-`
- `+` þar á eftir það sé a.m.k. eitt eða fleiri af táknunnum í hornklofanum á undan
- þar á eftir kemur punktur táknaður með `\.`
- `[A-Za-z]{2,3}`: fyrir aftan punktinn mega vera hvaða bókstafir sem er stórir: `A-Z` og litlir: `a-z` og fjöldi þeirra má vera 2 eða 3: `{2,3}`
- `\b` í lokin til að skilgreina orðamörkin

Kóði

Við settum þessa segð inn í skránna regex_email.py og notuðum `re.findall(email_regex, full_text)` til þess að leita af netföngum í textanum sem að uppfylla skilyrði segðarinnar.

Dæmi 3

3. Endurraða skrá

Markmið dæmisins er að endurraða línum með hjálp reglulega segða. Reglulegar segðir eru notaðar til að leita að, breyta eða staðfesta texta. Þær eru byggðar á mynstrum sem saman- standa af táknum sem tákna sérstakar tegundir texta eða aðgerðir. Eru þær mikið nýttar í gagngagreiningu og textaúrvinnslu. Í þessu dæmi verður textinn úr skjali nafn_heimilisfang_simanumer.csv breytt eftir fyrirmælum. En stendur til að breyta uppröðuninni á línunni. Í stað þess að röðin sé nafn, heimilisfang, póstnúmer og símanúmer. Verður hún heimilisfang, póstnúmer, símanúmer, kenninafn og að lokin eiginafn og millinafn. en einnig viljum við taka kommu út og setja `\t` inn í staðinn (nema á eftir kenniafni).

Upprunalegu línurnar líta svona út:

Jón Jónsson, Litla-Saurbæ, 816 Ölfusi, 555-1234 Guðrún Helgadóttir, Fiskislóð 15, 101 Reykjavík, 510-7000 Jón Oddur Guðmundsson, Úthlíð 6, 450 Patreksfirði, 897-1234

Reglulega segðin sem notað er:

```
([^,]+)\s([^,]+),\s([^,]+),\s([^,]+),\s(.+)
```

Reglulega segðin hér er notuð til að finna og hópa saman upplýsingarnar í hverri línu fyrir sig.

Brjótum þetta niður:

`([^,]+)` = táknar fyrsta hópinn. `[^,]` þýðir hvað sem er nema komma hérna er ss komman tekin af. Hornklofarnir `[]` lýsa hvað má koma fram í þessum hóp. `+` merkir að það þarf að vera eitt eða fleiri af þessum táknum. Eða einn eða fleiri stafir hér sem eru ekki komma. Svigarnir `()` afmarka hópinn til að hægt sé að vísa til þeirra seinna. `\s` = táknar bil. Síðan birtast hóparnir eins koll ap kolli. Í lokinn kemur seinasti hópurinn = `(.+)` en hann tekur það sem eftir er á línunni, sem er hér símanúmer. Reglulega segðin skiptir textanum í hópa eftir:

1. eiginnafn og millinafn ef á við
2. kenninafn
3. Heimilisfang
4. póstnúmer
5. símanúmer

Næst kemur substitution sem er notað til að breyta uppsetningu tecxtans út frá því sem reglulega segðin hefur greint út.

substitution = `r'\3\t\4\t\5\t\2, \1'`

Hér merkir hvert \tala tiltekin hóp sem var afmarkaður í reglulegu segðinni að ofan. `\t` setur inn bil. Hér prentast því út heimilisfang -bil- póstnr. -bil- símanr. -bil- kenninafn, eignnafn

Ath. komman sem birtist í úrtakinu á eftir kenninafni kemur fyrir tilstilli hennar úr substitution setningunni. Þannig ég þurfti að bæta henni við hér til að hún birtist. Því að í reglulegu segðinni að ofan eru kommurnar aðeins til staðar til að setja textann upp á skipulagðan hátt en `([^,]+)` sér til þess að komman sé ekki tekin með í úttakið. Substitution notar aðeins „fangaða" hópa og þar eru engar kommur nema það sé tekið fram.

Hér fyrir neðan má sjá þetta útfært í python kóða. Restin af kóðanum má finna inni á skráni: regex_reorder.py

`def endurraða_skra(linur)` """ Tekur línur á formattinu Jón Jónsson, Litla-Saurbæ, 816 Ölfusi, 555-1234 Guðrún Helgadóttir, Fiskislóð 15, 101 Reykjavík, 510-7000 Jón Oddur Guðmundsson, Úthlíð 6, 450 Patreksfirði, 897-1234 og skilar endurröðuðum línum á formattinu Litla-Saurbæ 816 Ölfusi 555-1234 Jónsson, Jón Fiskislóð 15 101 Reykjavík 510-7000 Helgadóttir, Guðrún Úthlíð 6 450 Patreksfirði 897-1234 Guðmundsson, Jón Oddur. Þannig Orðunum er enduraðað Kommur fara út nema á milli eftirnafns og fornafns og bil myndast.

```
    substitution = r'\3\t\4\t\5\t\2\t\1'
    """
    result = []
    pattern = r'([^,]+)\s([^,]+),\s([^,]+),\s([^,]+),\s(.+)'  # reges pattern notað
    til að bera kennsl á og hópa saman miðað við formattið
    substitution = r'\3\t\4\t\5\t\2, \1'  # Skiptimunstrið skilgreint til þess að
    enduraða orðunum, kommum skipt út fyrir bil

    for lina in linur:  # ítra yfir hverja línu i input listanum
        if re.match(pattern, lina):  # athuga hvort línurnar passi við regex patternið
            newline = re.sub(pattern, substitution, lina)  # enduraða línunum eftir
    substitution-inu
```

```
        result.append(newline)  # bæta enduraðaðri línunni við útkomu listann

return result ```
```

Niðurstaðan birtist úr skjalinu heimilisfang_simanumer_nafn.tsv

Litla-Saurbæ 816 Ölfusi 555-1234 Jónsson, Jón Fiskislóð 15 101 Reykjavík 510-7000 Helgadóttir, Guðrún Úthlíð 6 450 Patreksfirði 897-1234 Guðmundsson, Jón Oddur regex_reorder.py

Dæmi 4

Regluleg segðir fyrir timataka.py

Regluleg segð til að skoða URL-slóðina

Reglulega segðin sem við notum til að skoða URL: `pattern = re.compile( r'<tr[^>]*>\s*<td[^>]*>(\d+)</td>\s*<td[^>]*>(3845)</td>\s*<td[^>]*>Finnur Björnsson</td>\s*<td[^>]*>(\d{4})</td>\s*<td[^>]*>(.*?)</td>\s*<td[^>]*>(\d{2}:\d{2}:\d{2})</td>\s*<td[^>]*>(\+\d{2}:\d{2})</td>\s*<td[^>]*>(\d{2}:\d{2}:\d{2})</td>', re.DOTALL )`

- `https://timataka\.net\`:
  - Passar nákvæmlega við linkinn "https://timataka.net/".
  - Ath. að við notum bakstrik (`\`) á undan punkti (`.`) til að sleppa honum og tryggja að hann passi við bókstaflega punkt (`.`) í URL-inu.
- `.+/urslit/:`
  - `.`: Passar við hvaða staf sem er (tákn fyrir einn staf).
  - `+`: Passar við einn eða fleiri stafi af hvaða gerð sem er.
  - Saman `. + /urslit/` tryggir að það sé einhver texti á undan "/urslit/".
- `\?race=\d+&cat=\w+`:
  - `\?`: Passar við spurningamerki (`?`), sem er notað í URL fyrir fyrirspurn.
  - `race=\d+`: Passar við "race=" fylgt eftir af einni eða fleiri tölustöfum (`\d+`).
  - `&cat=\w+`: Passar við "&cat=" fylgt eftir af einum eða fleiri bók- eða tölustöfum (`\w+`), þar sem `\w` táknar hvaða staf eða tölustaf sem er.
- `(&age=\d+)?`:
  - `(&age=\d+)?`: Passar við valfrjálsan hluta sem byrjar á "&age=" og fylgt er eftir með einum eða fleiri tölustöfum (`\d+`). Spurningamerkið ? gerir þetta að valfrjálsu samsvörun.

Regluleg segð til að vinna úr HTML-gögnum

Reglulega sem við notuðum til að vinna úr HTML-gögnum er: `pattern = re.compile( r'<tr[^>]*>\s*<td[^>]*>(\d+)</td>\s*<td[^>]*>(3845)</td>\s*<td[^>]*>Finnur Björnsson</td>\s*<td[^>]*>(\d{4})</td>\s*<td[^>]*>(.*?)</td>\s*<td[^>]*>(\d{2}:\d{2}:\d{2})</td>\s*<td[^>]*>(\+\d{2}:\d{2})</td>\s*<td[^>]*>(\d{2}:\d{2}:\d{2})</td>', re.DOTALL )`

- `<tr[^>]*>`:
  - Passar við hvaða `<tr>` tag sem er.
  - `[^>]*`: Passar við hvaða staf sem er nema >, svo það tekur inn hvaða eiginleika sem er innan `<tr>` tagsins (t.d.).

- `\s*`:
  - Passar við hvaða bil sem er (hvít bil, nýjar línur o.s.frv.).
  - Notað til að tryggja að bilið í HTML hafi ekki áhrif á samsvörunina.
- `<td[^>]*>(\d+)</td>`:
  - Passar við `<td>` tag með tölustaf innan þess.
  - (`\d+`): Grípur röðina (`Rank`) með einum eða fleiri tölustöfum, sem verður `match.group(1)`.
- `<td[^>]*>(3845)</td>`:
  - Passar við annað `<td>` tag með keppnisnúmerið "3845" (sérstakt fyrir "Finnur Björnsson").
  - `(3845)`: Nákvæm samsvörun við keppnisnúmerið.
- `<td[^>]*>Finnur Björnsson</td>`:
  - Passar við þriðja `<td>` tagið með nafninu "Finnur Björnsson".
- `<td[^>]*>(\d{4})</td>`:
  - Passar við fjórða `<td>` tagið með fæðingarárinu.
  - `(\d{4})`: Grípur fjóra tölustafi sem tákna árið (Year), sem verður `match.group(3)`.
- `<td[^>]*>(.*?)</td>`:
  - Passar við `<td>` tag sem inniheldur nafn klúbbsins.
  - `(.*?)`: Grípur hvaða texta sem er innan `<td>` tagsins, þar til það finnur næsta loka tag. Þetta verður `match.group(4)`.
- `<td[^>]*>(\d{2}:\d{2}:\d{2})</td>`:
  - Passar við `<td>` tag með keppnistímanum.
  - `(\d{2}:\d{2}:\d{2})`: Grípur tímamynstur "hh:mm ", sem verður `match.group(5)`.
- `<td[^>]*>(\+\d{2}:\d{2})</td>`:
  - Passar við `<td>` tag með muninum "Behind".
  - `(\+\d{2}:\d{2})`: Grípur mynstur "+mm ", sem verður `match.group(6)`.
- `<td[^>]*>(\d{2}:\d{2}:\d{2})</td>`:
  - Passar við síðasta `<td>` tag með chip-tímanum.
  - `(\d{2}:\d{2}:\d{2})`: Grípur tímamynstur "hh:mm ", sem verður `match.group(7)`.