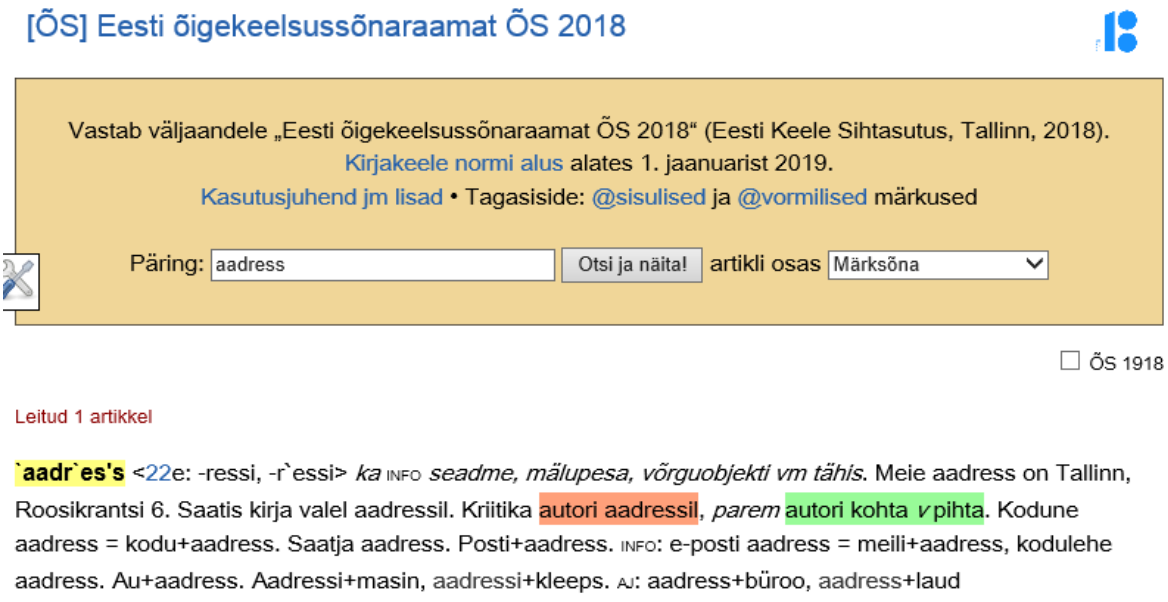
6.1 Regulaaravaldis

Eelnevates osades oleme vaadanud teemasid, meetodeid ja konstruktsioone, mida iga programmeerija oma igapäevases töös kasutab. Selles osas käsitleme vahendeid, mis ei pruugi olla igale programmeerijale kohustuslikud, kuid on siiski väga huvitavad ja kasulikud ka igapäevaelus.

Päris paljud programmeerijatel ette tulevad ülesanded on seotud sõnade ja teksti analüüsiga. Seesugusest analüüsist on sageli abi ka ülesannete juures, kus seda esmapilgul ehk ei ootakski.

ÕIGEKEELSUSSÕNARAAMAT. METAMÄRK

Kui soovime teada saada mingi eestikeelse sõna õigekirja või tähendust, siis võime abi otsida “Eesti õigekeelsussõnaraamatust ÕS 2018”. Näiteks kui tahame teada, kuidas käändub sõna “aadress”, siis sisestame veebilehel http://www.eki.ee/dict/qs/ olevasse päringuvormi selle sõna ja saamegi tema kohta info kätte.



Mis saab aga siis, kui me ei tea täpselt, kuidas otsitavat sõna kirjutada? Selle peale on mõeldud! Probleemsete tähtede asemele saame panna küsimärgi. Näiteks kui me ei tea, kas õige on “kandidaat” või “kanditaat”, võime sisestada “kandi?aat”. Küsimärki nimetatakse selles kontekstis **metamärgiks** ehk metasümboliks. Metamärk ei tähista iseennast, vaid viitab mingile reeglile, mille põhjal saame meie (ja ka arvuti) aru, kuidas antud kohta sõnas mõista. Seega metamärgina tähistab küsimärk ühte suvalist sümbolit. Otsingusõna “kandi?aat” puhul leitakse selliseid sõnu sõnaraamatust ainult üks. Sobivaid vasteid võib-olla ka rohkem, näiteks "?ell" puhul on neid lausa kuus. Kui asendada sõna esimene täht (või mitu tähte) küsimärgiga, siis see annab juba päris hea meetodi, kuidas riimuvaid sõnu otsida!

Otsingute puhul on teiseks levinud metamärgiks tärn *, mis tähistab suvalist arvu mingeid märke (seejuures ka märkide puudumist). Näiteks päring "meh*aanika" leiab kõik sõnad, mis algavad tähtedega "meh" ja lõppevad tähtedega "aanika". Nüüd võib sobivate vastete hulgas olla ka

lühemaid ja pikemaid sõnu, sest tärni asemel võib olla null, üks või rohkem märki. Nii saame "ka*ak" korral vasteteks teiste hulgas "kaak", "kajak" kui ka "kaelkirjak".

Otsingus võib korraga olla mitu metamärki, nt *us?epp. Otsingusõna võib koosneda ka ainult metamärkidest, näiteks "??????????????????????".

Ülesanne

Proovige veebilehel http://www.eki.ee/dict/qs/ erinevaid päringuid, näiteks neid:

- kandi?aat
- *sepp
- *us?epp
- ??????????????????????

REGULAARAVALDIS

Metamärkide süsteeme on mitmeid ja olenevalt süsteemist võivad metamärkide tähendused ka erineda. Järgnevalt vaatleme süsteemi, mis on paljudes programmeerimiskeeltes levinud. Selles süsteemis ongi märkide ? ja * tähendused teistsugused kui ülaltoodud sõnastiku päringutes.

Käesolevas tekstis on kasutatud termineid "sõna" ja "sõne". "Sõna" on siin tavalises tähenduses, näiteks "armastus" ja "lapsepõlv" on sõnad. Mõiste "sõne" on laiem, sest sõne võib sisaldada ka tühikuid, kirjavahemärke, isegi reavahetusi jms. Sõnest võib mõelda kui suvalisest sümbolijadast. Mingi sõne alamsõneks nimetatakse sõnet, mis sisaldub esialgses sõnes. Kui sõne on näiteks “Ma tahaksin kodus olla”, siis selle alamsõned on teiste hulgas näiteks "Ma", "kodu", "sin ko" ja " ".

Eeltoodud ÕSi päringute puhulgi anti vastuseks need sõnad, mis vastasid päringus määratud mustrile. Mitmetes programmeerimiskeeltes saab mustrit kirjeldada **regulaaravaldise** abil. Iga konkreetse sõne puhul saab siis öelda, kas see sõne vastab antud regulaaravaldisele või mitte. Regulaaravaldistes on tähtis roll metamärkidel, millest siin materjalis piirdume ainult mõne olulisemaga. Need on toodud järgnevas tabelis.

Metamärk	Kirjeldus	Näide
.	üks suvaline sümbol	"a.i"
		Sümboli "a" järel on suvaline sümbol ja seejärel "i".
		Näiteks "abi", aga mitte "appi".
?	0 või 1 kord eelnevat sümbolit või regulaaravaldist	"ai?"
		Sümboli "a" järel on 0 või 1 sümbolit "i".
		Ainsad näited on "a", "ai".
*****	suvaline arv kordi (ka 0 korda) eelnevat sümbolit või regulaaravaldist	"ai*"
		Sümboli "a" järel on suvaline arv korda sümbolit "i".
		Näiteks "a", "ai", "aii", "aiiii".
{n}	n korda eelnevat sümbolit või regulaaravaldist	"ai{2}"
		Sümboli "a" järel 2 sümbolit "i".
		Ainus näide on "aii".
[ ]	võimalike sümbolite hulga märkimine	"a[ij]"
		Sümboli "a" järel on sümbol "i" või sümbol "j".
		Ainsad näited on "ai", "aj".
[-]	võimalike sümbolite vahemiku märkimine	"a[0-9]"
		Sümboli "a" järel on suvaline number.
		Näiteks "a0", "a5".
( )	rühmitamine	"a(ij)*"
		Sümboli "a" järel on suvaline arv korda sümbolipaari "ij".
		Näiteks "a", "aij", "aijij", "aijijij", aga mitte "aji".
^	sõne algus	"^ai"
		Sõned, mis algavad sümbolipaariga "ai".
		Näiteks "ai", "aine", "aiandus".
$	sõne lõpp	"ai$"
		Sõned, mis lõpevad sümbolipaariga "ai".
		Näiteks "ai", "hai", "samurai".

Metamärkide abil saame koostada lihtsamaid ja keerulisemaid regulaaravaldisi. Ühes regulaaravaldises võib olla ka mitu metamärki. Näiteks ".a*" tähistab, et suvalisele sümbolile järgneb suvaline arv korda sümbol "a". Kuna suvaline arv võib ka 0 olla, siis "a" võib ka puududa.

Sõne vastavust regulaaravaldisele saab Pythonis kontrollida funktsioonidega `re.match` ja `re.search`. Kõigepealt tuleb importida moodul *re*. Nimi *re* tuleb ingliskeelsest mõistest *regular expression*, mis tähendabki tõlkes regulaaravaldist. Seega kirjutame programmi algusesse `import re`.

FUNKTSIOON re.match

Vaatleme esialgu funktsiooni `re.match`, millega saab kontrollida, kas sõne algus vastab regulaaravaldisele. Esimeseks argumendiks tuleb sellele funktsioonile anda regulaaravaldis, teiseks argumendiks sõne. Funktsiooni `re.match` saame kasutada valikulauses.

Järgnev programm kontrollib, kas sõne "abi" algab regulaaravaldisele "a.i" vastavalt. Punkt (".") tähistab üht suvalist sümbolit ning antud hetkel sobib suvalise sümboli rolli täht "b".

```
import re
if re.match("a.i", "abi"):
    print("vastab")
else:
    print("ei vasta")
```

Funktsioon `re.match` otsib vastavust sõne algusest. Seega `re.match("a.i", "Nabi")` puhul vastavust pole. Küll aga ei pea regulaaravaldisele vastava osaga sõne ära lõppema, nt `re.match("a.i", "abiks")` näitab vastavust.

Mitme metamärgiga regulaaravaldised avavad uusi (ka keerulisi) võimalusi. Näiteks ".*" tähendab, et suvalisi sümboleid on suvaline arv kordi (võib olla ka null korda). Regulaaravaldisega ".*a.i" on seega vastavuses “Nabi”, “esmaabi”, “aabits”, “aminohape on oluline” ja veel lõpmatu arv teisi sõnesid. Metamärgiga $ on võimalik kontrollida, kas regulaaravaldisele vastav alamsõne lõpetab sõne. Näiteks regulaaravaldisega “a.i$” on vastavuses “abi”, aga “abiks” ei ole.

Ülesanne

Selle ja mitmete edasiste ülesannete puhul antakse *Kontrolli*-nuppu vajutades reaktsioon iga vastusevariandi jaoks eraldi. Reaktsioon oleneb sellest, kas vastusevariant on õigesti märgitud/märkimata või mitte.

Millised järgmistest sõnedest on `re.match` mõttes vastavuses regulaaravaldisega `".a?t"`?

- [ ] `"ot"`
- [ ] `"aat"`
- [ ] `"2at"`
- [ ] `"matt"`
- [ ] `".a?t"`

Järgmises ülesandes on regulaaravaldistes kasutusel mitmed erinevad metamärgid.

Ülesanne

Millistega järgmistest regulaaravaldistest on sõna `"ABBA"` funktsiooni `re.match` mõttes vastavuses?

- [ ] `"AB{2}A"`
- [ ] `"AB{2}$"`
- [ ] `"AB{2}"`
- [ ] `"AB{2}[AB]$"`
- [ ] `"[AB]{3}A"`
- [ ] `"B{2}A"`

FUNKTSIOON re.search

Funktsiooni re.match kõrval kasutatakse ka üsna sarnaselt toimivat funktsiooni re.search, mis otsib vastavust mitte ainult sõne algusest, vaid üle kogu sõne. Nii saame näiteks leida, kas sõnes leidub kolm kõrvuti olevat numbrit. Seda, kas tegemist on numbriga saab kontrollida "[0-9]" abil (vt metamärkide tabelit). Loogelistes sulgudes saame ette anda, mitu korda eelnevat sümbolit või avaldist peab olema.

```
import re
if re.search("[0-9]{3}", "Auto registreerimisnumbriga 007PYT
lähenes"):
    print("sisaldub")
else:
    print("ei sisaldu")
```

Tegelikult võiksime "[0-9]" asemel kirjutada "[0123456789]", sest seda me ju otsime: kas sõnes leidub selline koht, kus on järjest kirjutatud kolm numbrit ehk sümbolit, millest igaüks võib olla kas 0, 1, 2, 3, 4, 5, 6, 7, 8 või 9.

Ülesanne

Millised järgnevatest variantidest tuvastavad vastavuse?

- [ ] `re.search("^b", "abcd")`
- [ ] `re.match("b", "abcd")`
- [ ] `re.search("^a", "abcd")`
- [ ] `re.match("a", "abcd")`

Järgmises näites püüame leida, kas teates sisaldub korvpalli seis. Esialgu eeldame lihtsustatult, et kummalgi võistkonnal on kahekohaline skoor.

```
import re

korvpalli_seis = "[0-9]{2}:[0-9]{2}"

teade = "Esimene korvpallimäng olümpiamängudel toimus 1. augustil
1936. aastal. Eesti võitis Prantsusmaa 34:29."

if re.search(korvpalli_seis, teade):
    print("sisaldab ilmselt korvpalliseisu")
else:
    print("ei sisalda ilmselt korvpalliseisu")
```

Tegelikult võib korvpallis skoor olla ka kolmekohaline ja mängu algusjärgus muidugi ka ühekohaline. Seda kirjeldab paremini regulaaravaldis `"[0-9][0-9]?[0-9]?:[0-9][0-9]?[0-9]?"`.

VEEL (JA PÄRIS ELULISI) VARIANTE

Nüüd vaatame elulist, kuid pisut keerulisemat näidet geneetika valdkonnast. See on eelkõige mõeldud just huvitavaks ja harivaks lugemiseks - ülesannetes neid teadmisi ei nõuta.

DNA koosneb nukleotiididest adeniin (A), guaniin (G), tsütosiin (C) ja tümiin (T). Lühendite abil saame DNA ahela lõigu kirja panna näiteks sõnena, mis koosneb tähtedest A, G, C ja T. DNA ahelas paiknevad geenid, millest transkripteeritakse RNA molekule ning millest omakorda moodustatakse aminohapetest koosnevaid proteiine, mis on vajalikud meie elutegevuseks ja panevad paika, kuidas meie keha töötab. Mitte kogu DNA ahel ei koosne geenidest, vaid geenid asuvad teatud positsioonidel DNA ahelas. Selleks, et teada saada, millal mõni geen algab või lõppeb, on olemas algus- ja lõpukoodonid - kindlad järjestused, mis annavad teada, et selle koha pealt tuleb geeni lugema hakata või lugemine lõpetada. Alguskoodoniks on järjestus ATG ning lõpukoodoneid on lausa mitu: TGA, TAG ja TAA.

Kui tahame näiteks leida, kas mõnel DNA lõigul asub potentsiaalne geen, peaksime kontrollima, kas leidub selline järjestus, mis algab kolmikuga ATG ning lõppeb kolmikuga TGA, TAG või TAA. Lisaks peab nende kahe kolmiku vahele jääma kolmega jaguv arv nukleotiide. Kolmega jaguvust on vaja kontrollida seetõttu, et ühele aminohappele vastab nukleotiidide kolmik ning kui nukleotiidide arv ei jagu kolmega, siis ei saa sellisest järjestusest teha proteiini. Järgmine näide demonstreerib, kuidas teha kindlaks, kas antud DNA lõigus on mõni potentsiaalne geen:

```
import re

dna_lõik
= "AAGGCTAGACTGGCTTAGCCTCCCAGCCTACATCTTTCTCCCATGCTGGATGCTTCCTGCCCTCA
AACATCAGGCTCCATGTTTTTCAGCTTTGGGACTCACAGTGGCTTCCTTGCTCCTCAGTTTACAGACA
GCCTATTGTGGGACCTTGTGATTGTGTGAGTTAATACTACTTAATAAACTCCCATATGTAGGTGTGTG
TATATGTCTTATTAGTTCTATCCCTCTAGAGAACCCTGACTGATACACCTGCTCTAACAGGCTGTTGA
ATGCCCATGGCTTTTCCAGGTGCAGGGCATAAGCTGCCAGTGGATATACTATTATTGGGTCTACAAGG
TGATGGCCCACTTCTCACAGCTCCATTAGGCAGTGTTCCGGTGAAGGCTCTGTGTGGGGACTTGAACC
CTTCATTTTCCCTTGATACTACCCTAGAAGTTATCTGTGGGGGTTCTGTTTCTGAAGTAGGCTTCTGC
CTGGACACCCAGGCTTTTCATACATCCTTGGAAATCTACGCAGAAGGTGCCAAGACTTATTCATTCTT
ACACTCTGTGCACTTGCAGGCTTAACAGTACATAGAAGACACCAAGGCTTATGGCAGTTTGTACTTTC
CTGAGCAGCAACATGAAGAGTATGTGGGAGGTTTTGAGCCAAGGCTGGAGCCGGAGTTGCCTAGACGT
GGGGAACAGTATTTCAAGACTACACAGGGCAGCAAAGCCCTGGGCCTGGCCATGGAAACCATTGTTTC
CTACTAGGCCTCAGGACTTGTCATGGAAGGGGCTGCCATGAAGTCTCTGAAATGCTTTCCAAGCATCC
TCCTCATTGTCTTGGCTATCAGCACTTGGCTCCCTTTTTAGTCATGCATACTTCTCTAGCAAGTCGTT
TCTTCACAGCCTGCCTGAATTCCTCTCCCCAGAAAGGTTTTTTTCTTCCTCTGCCACATGGCCAGGCT
GCAAATTTTGAAAACTTTTGTTGTCTGCTTCCCTTTTAAATATAGGTTCTAACTTTAAGTCATTCCTT
TGTTCCTAAATCTGAGCATAGGTTGTTAGAAGCTGCCAGGCCACATTTTGAATGCTTTTCTGCTTAGA
AATTTCCTCCACCAGGTACCCTAAATCATCCCCATGAAGTTCAAACTTCCAAAGATCCTCAGGGCATG
AACAGAATGCATCCAGGGTTTTTACTAAGGTATAACACACATGACTTTTGCTCCAGCTCTCAATAAGT
TATTTCTATCTGAGACCTCAGCAGCCTGGAATTCGGTCTCCGTATCACTATAAGTATTTTGATCACAA
CCATTTAACCAGTCTCTAAGACATTTCTAACTTTTCCTCATCTTCCTGTCTTCTTCAAAGCTCTCCAA
ACTCTTTCAAACATAGCTCATTACCCAGTTCCAAAGCTGCTTCCACATTTTCAGATATCTTTATAGCA
AAGGCTCGATCCTCAATACCAATTTTCTTTACTAGACTGCTCTTGTATTTCTATGAAGAAATATCTGA
GAGAGGGTAATTTATAAAGAAAAGAGGTTTAATTGGCTCCCAGTACTGCAGGATTTACAGAAATAATG
ATACTGGCATCTGCCTGGCTTCTTGGGAGGTCTCAGGCAACTTACAATCACGGTAGAAGGTGAAGAGA
GAGCAGGCATGCCACTTGGTGAAAGCAGGAGCGTGAGAGAGACAGTGGGTG"

leidub = re.search("ATG(...)+T((GA)|(AG)|(AA))", dna_lõik)

if leidub:
    print("Antud DNA lõigus asub vähemalt üks potentsiaalne geen.")
else:
    print("Antud DNA lõigus ei asu ühtegi potentsiaalset geeni.")
```

Regulaaravaldiste abil saab näiteks üles leida kõik potentsiaalsed geenid ja nende asukohad DNA ahelal ning edasi saavad geenitehnoloogid ja bioinformaatikud uurida, millise funktsiooni eest erinevad geenid vastutavad.

Püüame nüüd ka kasutatud regulaaravaldisest `ATG(...)+T((GA)|(AG)|(AA))` detailsemalt aru saada.

- "ATG" regulaaravaldise alguses näitab, et otsitakse alamsõne, mis algab tähtedega "ATG".
- (...) tähistab kolme suvalist märki. Pluss ("+") pärast seda ütleb, et eelnevat (antud juhul siis kolme märki) peab olema 1 või rohkem korda. See tagab, et algus- ja lõpukoodoni vahele jääb kolmega jaguv arv nukleotiide.
- Otsitav alamsõne peab lõppema kolmikuga "TGA", "TAG" või "TAA". See tagatakse regulaaravaldise lõpuga `T((GA)|(AG)|(AA))`, kus märk | tähendab "või". Seega võib lõpukoodon olla kas "TGA", "TAG" või "TAA".

Ülesanne

Millistele regulaaravaldistele vastavad alamsõnad on sõnes "Ta jõudis vaevu-vaevu kohale."? Teisiti sõnastades, mida saab kirjutada lünka _____, et ekraanile väljastataks “sisaldub”.

```
import re
if re.search(_____, "Ta jõudis vaevu-vaevu kohale."):
    print("sisaldub")
else:
    print("ei sisaldu")
```

- ☐ `"(vaevu){2}"`
- ☐ `"(vaevu.){2}"`
- ☐ `"ko[a-z]al"`
- ☐ `"j[a-zõ]u"`
- ☐ `"..."`

Lõpetuseks võib veel öelda seda, et regulaaravaldist algkursustes tavaliselt ei käsitleta, aga loodetavasti olete nõus, et tegemist on huvitava ja kasuliku teemaga.

6.2 Keeletehnoloogia II

KÕNETUVASTUS (seadmete juhtimine häälega)

Üks olulistest keeletehnoloogia valdkondadest on kõnetuvastus. Kõnetuvastus tegeleb inimkõne sisu tuvastamisega ja tekstiks teisendamisega. Ülesanne tundub esialgu keeruline ja eks ta seda olegi, kuid siiski leiduvad meetodid, kuidas inimkõne tekstiks teisendada suhteliselt väikese veaprotsendiga. Kõigepealt tuleb kõne salvestada digitaalsel kujul ja edastada arvutile, nii nagu ikka helifaile arvutis esitatakse. Helifaili saab seejärel ette anda programmile, mis leiab vastava teksti. Selleks kasutavad algoritmid on päris keerulised ja põhinevad masinõppel. Masinõpe siinkohal tähendab, et programmile tutvustatakse sõnade (helifailide kujul) ja õigete tekstide paare ning programm püüab ära õppida, et kuidas helifailile vastavat teksti leida.

Kõnetuvastust kasutatakse paljudes erinevates valdkondades. Võib olla olete kasutanud kõnetuvastust oma mobiiltelefoniga suhtlemiseks. Näiteks on populaarsel mobiiltelefonil *iPhone* kõnetuvastusprogramm Siri, millele saab anda käsklusi nagu näiteks “Saada mu naisele sõnum, et jään hiljaks” või “Leia lähedalasuvad bensiinijaamad”. Siri töötab päris suure täpsusega, aga kahjuks eesti keeles seda kasutada pole võimalik. Lisaks on kõnetuvastus üha enam olemas autodes, et juhtidel oleks lihtne sõidu ajal kõnesid teha ja samal ajal sõitmisele keskenduda. Kindlasti on üheks oluliseks kasutuseks ka kurtidega suhtlemise võimaldamine, kus räägitud jutu saab kohe tekstiks teisendada.

Kõnetuvastusel on oluline koht automaattõlkimise juures. Näiteks ingliskeelselt räägitud jutt tuvastatakse ja teisendatakse tekstiks, siis tõlgitakse arvuti abil ning seejärel esitatakse see näiteks eesti keeles. Väga hästi töötavaid lahendusi sellele ülesandele veel ei leidu, aga kõnetuvastus mängiks seal olulist rolli. Kas suudate ka ise kõnetuvastuse rakendamise ja vajalikkuse näiteid välja mõelda?

Eriti raskeks teeb kõnetuvastuse see, et inimestel on erinevad aktsendid ja sõnu hääldatakse erinevalt ning ka erineva kiiruse ja rõhuasetusega. Lisaks on sageli kuulda taustamüra või -teksti, mis võib samuti tuvastamist segada.

Ka Eesti Foneetika ja Kõnetehnoloogia Laboratoorium tegeleb kõnetuvastuse probleemiga (https://phon.ioc.ee/dokuwiki/doku.php?id=projects:tuvastus:veebituvastus.et). Proovige nende rakenduse abil oma kõnet tuvastada lasta: http://bark.phon.ioc.ee/webtrans/

Kui oskate natuke inglise keelt, võite proovida allolevaid rakendusi, et oma juttu tõlkida. (Siin ei pea seda eelnevalt salvestama, vaid saab kohe rääkida. Peab ainult lubama veebilehitsejal mikrofoni sisendit kuulata. Selleks ilmub veebilehtitseja aadressiriba alla märge, et antud veebisait soovib kasutada teie mikrofoni. Klikkida "Luba" (ingl. k "Allow") nupul.

NB! Järgmised programmid töötavad kõige paremini *Google Chrome*' veebilehitsejaga .

- https://dictation.io/
- https://talktyper.com

TEHISINTELLEKT JA KEELETEHNOLOOGIA

Tehisintellektika on informaatika haru, mis tegeleb intellekti simuleerimisega arvutitel. Intellektiks nimetatakse inimolendite kõrgemaid intellektuaalseid protsesse (mõistuslikkus, tähenduse väljauurimine, üldistamine, kogemustest õppimine). Näiteks kuuluvad siia programmid, mis mängivad kasutajaga malet või trips-traps-trulli, programmid, mis tegelevad erinevate probleemide lahendamisega või programmid, mis inimesega suhtlevad. Viimase headuse mõõtmiseks kasutatakse näiteks Turingi testi. Seda testi mainiti ka silmaringi teemas "Katkeid programmeerimise ajaloost". Programm läbib selle testi, kui inimene, kes arvuti vahendusel temaga suhtleb, ei suuda vahet teha, kas ta suhtleb inimese või programmiga. Selliste suhtlusprogrammide loomine haakub ka keeletehnoloogiaga. Selleks, et inimesega suhelda, peaks programm aru saama, mida inimene mõtleb, ehk peaks aru saama lause ja jutu kontekstist. Näiteks, et kui inimene küsib, et "Mis on sinu lemmiktegevus?", siis programm peaks aru saama, et see on küsimus ja vastuseks võiks olla mingi tegevus, mis oleks samas loogiline jätk jutule. Selle saavutamiseks on erinevaid viise. Tihti kasutatakse siin jällegi masinõpet. Programmile tutvustakse erinevaid päris vestlusi ning programm õpib, millised laused kokku sobivad.

Üheks selliseks programmiks, mis inimesega suhtleb, on juturobot Cleverbot. Proovige temaga rääkida siin: http://www.cleverbot.com/.

See juturobot oskab ka eesti keeles vastata, aga sisukamaid vestlusi saab teha inglise keeles. Seda seetõttu, et Cleverbot õpib vastuseid temaga peetud vestlustest (temaga on räägitud üle 150 miljoni vestluse!), mis on enamasti toimunud inglise keeles. Cleverbot võib olla õppinud ütlema ka rumalaid sõnu, nii et ärge ehmatage, kui ta kogemata ebaviisakas on!

Keeletehnoloogiat kasutatakse ka robootika valdkonnas, näiteks humanoid (kujult inimest meenutav robot) NAO peab aru saama talle antud käsklusest (kõnetuvastus) ja siis sellele mõistlikult reageerima. Võite vaadata videot NAO toimetamisest siit:
https://www.youtube.com/watch?v=_ouYfFHvbC8

Tihti muretsetakse, et kas arvutid suudavad ka päriselt ise mõtlema hakata ja omada teadvust. Kui programmeerimisega natukene tuttavamaks saada, siis on lihtne mõista, et arvutid teevad siiski vaid seda, mida neid on õpetatud tegema. Ehk täidavad inimese poolt programmeeritud käske. Siiski keerulisemate algoritmide abil (masinõpe) on võimalik panna neid ise keskkonnast uusi asju juurde õppima ja selle abil oma otsuseid muutma. Arvutid küll ise teadvust ei oma (ning ei suuda oma tegevust ise mõtestada), aga neid on võimalik programmeerida nii, et tunduks, nagu nad teaksid, mida nad teevad. Mõelge, kas Teie arvates on see probleem ning mida positiivset või negatiivset selles näete?

ALLIKAD

1. http://en.wikipedia.org/wiki/Artificial_intelligence
2. http://en.wikipedia.org/wiki/Speech_recognition
3. http://et.wikipedia.org/wiki/Keeletehnoloogia

Materjali koostasid Agnes Lepikult ja Mari-Liis Kruup. Kohendatud kursuse korraldajate poolt.

6.3 Kontrollülesanne VI

Isikukoodi korrektsus

Kirjutage programm, mis küsib kasutajalt isikukoodi ja kontrollib, kas see on Eestis võimalik. Kui on, siis väljastada ekraanile "**On Eesti isikukood**". Kui mitte, siis väljastada "**Ei ole Eesti isikukood**". Lihtsuse mõttes teeme järgmised eeldused. Tegemist on Eesti isikukoodiga, kui

- see koosneb 11 numbrist,
- esimene number on vahemikus 1-6 ja järgmised 10 numbrit on vahemikus 0-9.

Vihjed regulaaravaldiste kasutamise kohta:

- "[0-9]" - üks number
- "[0-9]{2}" - looksulguses arv näitab, mitu korda eelnevat võtta, antud juhul siis kaks numbrit
- $ abil saab märkida sõne lõppu, nt. "a$" - täht a ja lõpp
- ^ abil saab märkida sõne algust, nt. "^a" - sõne algab tähega a
    - sõne algust saab kontrollida ka funktsiooniga *match*, nt. *match("a")* - sõne algab tähega a

Näited programmi tööst:

```
>>> %Run yl6.py

  Sisesta isikukood: 48007140350
  On Eesti isikukood

>>>
>>> %Run yl6.py

  Sisesta isikukood: 509041127230
  Ei ole Eesti isikukood

>>>
```

Kontrollülesannete lahendused esitatakse *Moodle*'is.

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt. Püütud on tüüpilisemaid probleemseid kohti selgitada ja anda vihjeid. Ühtlasi võite vaadata ühe analoogse ülesande lahendust: http://www.uttv.ee/naita?id=23749

6.4 Maalähedane lugu VI

Kell kõrvaltoas lõi täistundi ja Rasmusel oli uni hoobilt kadunud. Mis see kell nüüd siis on - viis? kuus? seitse? Väljas oli juba valge, aga suvel lähebki vara valgeks. Ada on kindlasti juba õues toimetamas. Lembit pidi kalale minema. Kutsus noori kaasa, aga need ei viitsinud nii vara tõusta. Nüüd siis oli Rasmus ikkagi ärkvel, kuigi püüdis uuesti magama jääda. Ta silmitses, nagu lapsepõlves oli enne magamajäämist kümneid kordi teinud, suurt ristpistes seinavaipa. Vaiba ülemine äär oli peaaegu tapeedi ülemise äärega tasa. Sealt jäi laeni veel paarkümmend sentimeetrit valget lubjakrohviseina. Värv kippus mõne koha pealt murenema.

Vaip oli seinal seisnud juba aegade algusest peale ja nüüd oli juba küllaltki vanunud. Vaip oli väga korralikult ja hingega tehtud ning see oli kahtlemata üheks põhjuseks, miks ta ikka veel seinal oli, kuigi oli tubli tolmukoguja. Vaip oli tõesti suur ja tema muster koosnes mitmetest korduvatest elementidest. Vaiba ääres oli seitsme piste laiune tumepruun riba, millel olid sümmeetriliselt kollased ruudukesed. Sealt sissepoole olid suuremad kujundid, kõik ikka korrapäraselt. Vaip oli tõesti tehtud väga püüdlikult ja kõik tundus olevat veatu. Ometi oli Rasmus juba ammu leidnud vaibal kolm kohta, kus paistis tikkija olevat natuke eksinud. Näiteks oli üks ääreribal olev ruuduke kollase asemel hoopis punane. Nüüd ta siis püüdis neid kohti uuesti leida ja leidiski.

> VAHELEPÕIGE Rasmus ja Adagi, kes selle vaiba kunagi teinud oli, ei mõelnud muidugi, et mustrid on vägagi programmeerimisega seotud. Näiteks saab tekstist teatud mustrile vastavaid osasid otsida. Seejuures selleks tekstiks ei pruugi olla tavaline tekst, vaid näiteks hoopis DNA ahela kirjeldus.

Käsitööd oli selle küla naispere alati au sees pidanud. Eks see oli varasematel aegadel ka palju olulisem olnud, kust siis ikka riie selga sai, kui mitte ise tehtud. Praegusel ajal oli see kõik rohkem ilu jaoks ja ega külaski keegi peale Ada ja naabri-Kadri suuremalt käsitööga ei tegelenud. Just üleeile oligi Kadri jälle uute kindamustritega tulnud ja neid vaadatud ja võrreldud. Liisugi oli vaadanud ja kiitnud. Noor laps, aga on silma! Rasmus oli ka kindamustreid vaadanud, aga rohkem köitsid teda näiteks mootorratta või traktori rehvide mustrid.

Rasmus vähkres veel natuke ja kuna und ikka ei tulnud, siis ta tõusiski üles ning läks õue. Ada oligi juba aias peenraid rohimas, aga nii musterlaps Rasmus siiski polnud, et oleks talle kohe appi rutanud.

Lisamaterjale ja lisaülesandeid

Siia on kogutud selliste materjalide linke, mis avardavad teie teadmisi.

Julgesti võite linke juurde pakkuda!

- Pykkar

Lisaülesanne: Supp. (Ühtegi suppi ei sööda nii kuumalt, kui seda keedetakse)

Kausitäis paprikasuppi jahtub minuti jooksul 19% võrra supi ja ruumi temperatuuride vahest. Koostage programm, mis küsib sööjalt supi algtemperatuuri ja toatemperatuuri (just sellises järjekorras) ning väljastab, milline on supi täisarvuni ümardatud temperatuur 10 minuti pärast. Ümardamiseks saab kasutada funktsiooni *round*. Katsetage ise, kuidas funktsioon *round*töötab!

Eeldame, et supi algtemperatuur on väiksem kui 100 kraadi ja toatemperatuur on üle nulli. Lihtsalt supi keemise ja külmumise puhul ei ole see mudel õige, programm võib põhimõtteliselt töötada küll.

Näide programmi tööst:

```
>>> %Run supp.py

  Sisesta supi temperatuur: 90
  Sisesta toa temperatuur: 20
  Supi temperatuur 10 minuti pärast: 29

>>>
```

Selle ülesande lahenduse võib esitada *Moodle*'is ja saada automaatset tagasisidet, aga kohustuslik see ei ole.

Lisaülesanne: Auto registreerimisnumber Eestis

Kirjutage programm, mis küsib kasutajalt auto registreerimisnumbri ja kontrollib, kas see on Eestis võimalik. Kui on, siis väljastada ekraanile "**On Eesti registreerimisnumber**". Kui mitte, siis väljastada "**Ei ole Eesti registreerimisnumber**". Lihtsuse mõttes piirdume eeldusega, et registreerimisnumbris peab olema kolm numbrit ja siis kolm tähte (mis ei ole täpitähed). Lubame ka näiteks selliseid, kus numbriosa on 000. Ei luba aga selliseid, kus on kahenumbriline numbriosa.

Vihje:

- "[0-9]" - üks number
- "[A-Z]" - üks suur täht
- "[A-Z]{2}" - looksulguses arv näitab, mitu korda eelnevat võtta, antud juhul siis kaks tähte
- $ abil saab märkida sõne lõppu, nt. "a$" - täht a ja lõpp
- ^ abil saab märkida sõne algust, nt. "^a" - sõne algab tähega a
    - sõne algust saab kontrollida ka funktsiooniga *match*, nt. *match("a")* - sõne algab tähega a

Näited programmi tööst:

```
>>> %Run autonumber.py

  Sisesta number: 318MKF
  On Eesti registreerimisnumber

>>>
>>> %Run autonumber.py

  Sisesta number: MKF318
  Ei ole Eesti registreerimisnumber

>>>
```

Selle ülesande lahenduse võib esitada *Moodle*'is ja saada automaatset tagasisidet, aga kohustuslik see ei ole.

Lisaülesanne: Erinevate riikide autode registreerimisnumbrid

Erinevate riikide autode registreerimisnumbrid on üsna erinevad ja pakuvad väga palju toredaid võimalusi regulaaravaldise ülesanneteks. Püstitame siin väga avatud ülesande.

- Valige mingi riigi autode registreerimisnumber.
- Sõnastage nõuded registreerimisnumbrile.
- Kirjutage programm, mis kontrollib, kas sõne vastab nõuetele.
- Kuna põhirõhk on siin regulaaravaldisel, siis pole oluline, kas sõne küsitakse kasutajalt või on juba programmi sisse kirjutatud.

Paljudel juhtudel võib täieliku kontrolli tegemine olla päris keeruline. Sellisel juhul tehke nõuetes lihtsustusi.

Erinevate riikide autode registreerimisnumbrid (inglise keeles)

Seda ülesannet võib lahendada ka teiste huvitavate koodide/tähiste jms. Näiteks erinevate riikide isikukoodidega.

Selle ülesande lahendused võib saata vastavasse *Moodle*'i foorumisse. Muidugi seal saab lahenduste üle ka arutada, enda ja teiste saavutuste üle rõõmustada.