4.1 Sõned. Sisend ja väljund

SÕNED

Andmetüüpide juures rääkisime põgusalt ka andmete tekstilisest esitamisest ehk **sõnedest**. Vaatasime, et sõne esitatakse jutumärkide vahel ja sõne ingliskeelne vaste on *string* ning lühend *str*.

Tegelikult võib sõne panna ka ülakomade vahele. Näiteks saab sõne esitada nii "Sisenesid pangaautomaati!" kui ka 'Sisenesid pangaautomaati!'. Küll aga ei saa sõne ees olla üks märk (nt jutumärk) ja järel teine (nt ülakoma). Tuleks panna tähele, et jutumärgid ja ülakomad ise ei kuulu sõne sisu juurde. Seda saame proovida *print* käsuga:

- **Esimene võimalus**

```
print("Sisenesid pangaautomaati!")
```

- **Teine võimalus**

```
print('Sisenesid pangaautomaati!')
```

Mõlema võimaluse puhul väljastatakse ekraanile samasugune tekst:

```
>>>
Sisenesid pangaautomaati!
```

Milleks need jutumärgid ja ülakomad on vajalikud? Neid on vaja selleks, et muidu võib Python sõned segamini ajada näiteks muutujanimede või arvudega. Proovige näiteks eelmist käsku ilma ülakomadeta. Mis juhtub?

AGA KUI TEKST SISALDAB JUBA JUTUMÄRKE VÕI ÜLAKOMASID?

Asi läheb veidi keerulisemaks, kui sõne sees on vaja kasutada jutumärke, ülakomasid või muid erisümboleid. Järgnevalt demonstreerime erinevaid viise selle probleemi lahendamiseks.

- **1. võimalus.** Kui tekstis on ülakomasid, siis kõige lihtsam on kasutada piiritlejaks jutumärke ja vastupidi:

```
print("Rock 'n' roll")
print('Jim ütles vaid: "Siin see on."')
```

Ülesanne

Mis väljastatakse ekraanile?

```
print("Jim ütles vaid: "Siin see on."")
```

Vali Jim ütles vaid: "Siin see on."

Vali Jim ütles vaid:

Vali Veateade

- **2. võimalus.** Kui tekstis on vaja kasutada nii jutumärke kui ka ülakomasid, siis pole eelmisest soovitusest abi. Sellisel juhul tuleb üks neist (nt jutumärgid) ikkagi valida piiritlejaks, aga nende kasutamisel tekstis tuleb need spetsiaalselt märgistada langkriipsuga (sellist tegevust nimetatakse inglise keeles *escaping*) – see annab Pythonile märku, et tegemist pole veel teksti lõpuga, vaid sooviti kirja panna piiritlejaks valitud sümbolit ennast:

```
print("Jack vastas: \"Rock 'n' roll\".")
print('Jack vastas: "Rock \'n\' roll".')
```

Mõlemal juhul väljastatakse ekraanile tekst:

```
Jack vastas: "Rock 'n' roll".
>>>
```

(Juhime tähelepanu sellele, et langkriips \ ja kaldkriips / on erinevad märgid.)

Langkriipsu saab kasutada ka muul otstarbel, nt reavahetusi saab esitada kombinatsiooniga *n*.

```
print("Seda kuupaistet!\nOh muutuksin sündides\nmänniks mäetipul!\n-
Ryota")
```

Tulemus:

```
>>>
Seda kuupaistet!
Oh muutuksin sündides
männiks mäetipul!
-Ryota
```

Nagu näha, on langkriips spetsiaalse tähendusega. Kuidas aga esitada langkriipsu ennast? Lihtne, see tuleb ära märgistada ... langkriipsuga!:

```
print("C:\\kaustanimi\\failinimi.txt")
```

Tulemus:

```
>>>
C:\kaustanimi\failinimi.txt
```

Ülesanne

Mis väljastatakse ekraanile? (Väljatrükk on nupu "Vali" all.)

```
print("Minu lemmikraamatud on \n\"Kevade\" ja \n\"Rehepapp\"")
```

Vali
Minu lemmikraamatud on \n\"Kevade\" ja \n\"Rehepapp\"

Vali
Minu lemmikraamatud on
"Kevade" ja
"Rehepapp"

Vali
Minu lemmikraamatud on n"Kevade" ja n"Rehepapp"

Vali
Minu lemmikraamatud on
Kevade ja
Rehepapp

Vali Veateade

- **3. võimalus**. Kui tekstis on vaja kasutada palju reavahetusi, ülakomasid või jutumärke, siis võib tulemus muutuda kõigi nende \n, \' või \" tõttu väga kirjuks. Seetõttu on Pythonis veel üks sõne kirjapaneku viis – kolmekordsete ülakomade või jutumärkide vahel saab vabalt kasutada tavalisi reavahetusi, ülakomasid ja jutumärke. Katsetage iseseisvalt:

```
print("""Jack vastas: "Rock 'n' roll".""")
print('''Jack vastas: "Rock 'n' roll".''')
print("""Seda kuupaistet!
Oh muutuksin sündides
männiks mäetipul!
--Ryota""")
print("""
   _____
  / ____|
 | |  __   __ _ _ __ ___    ___    ______   ______ _ __
 | | |_ |/ _` | '_ ` _ \  / _ \  / _ \ \ / / _ \ '__|
 | |__| | (_| | | | | | | |  __/ | (_) \ V /  __/ |
  \_____|\__,_|_| |_| |_|\___|  \___/ \_/ \___|_|
""")
```

Ülesanne

Mis väljastatakse ekraanile?

```
print("""Rock 'n' rolli Jim ütles vaid: "Siin see on." """)
```

Vali Rock 'n' rolli Jim ütles vaid: "Siin see on."

Vali "" Rock 'n' rolli Jim ütles vaid: "Siin see on." ""

Vali Veateade

SISEND JA VÄLJUND

Programmis tuleb aeg-ajalt suhelda kasutajaga ehk tegeleda sisendi ja väljundiga.

KÄSK *PRINT*

Nagu olete juba õppinud, saab programmis väärtusi ekraanile kuvada käsuga *print*. Nüüd uurime seda käsku veidi lähemalt.

Vaatame näidet:

```
print(32 * 57)
```

See, mis ekraanile kuvatakse, antakse ette sulgudes - argumendina. Eelmises näites on 32 * 57 käsu *print* argumendiks. Kui kõik läheb ilusti, siis programm kuvab ekraanile 1824 ja lõpetab töö.

Käsule *print* võib ette anda ka mitu argumenti, sel juhul trükitakse ekraanile samale reale mitu asja järjest, kusjuures need asjad on eraldatud tühikutega. Järgnev näide demonstreerib kahte samaväärset viisi, kuidas trükkida ekraanile mitu infokildu korraga. Esimene variant kombineerib komponendid kõigepealt üheks sõneks ja kasutab seda *print*-i argumendina, teine variant annab kõik komponendid eraldi argumentidena.

```
eesnimi = "Peeter"
perenimi = "Paan"
vanus = 21
print(eesnimi + " " + perenimi + " vanus: " + str(vanus))

eesnimi = "Peeter"
perenimi = "Paan"
vanus = 21
print(eesnimi, perenimi, "vanus:", vanus)
```

Mõlemad programmid väljastavad ekraanile:

```
>>>
Peeter Paan vanus: 21
```

Eraldi argumentidega variant on küll lühem kirja panna (eriti mugav on see, et arve ei pea ise *str* käsuga sõneks teisendama), aga mõnikord see siiski ei sobi, näiteks siis, kui me ei soovi väljundis argumentide vahele tühikut. Niisiis on esimene variant pisut paindlikum, kuid teine pisut lihtsam (lihtsam nii kirjutada kui ka hiljem lugeda).

Seesuguseid dilemmasid tuleb programmeerijatel sageli ette: kas valida lihtsam või paindlikum tee? Kuidas nende vahel kompromissi leida? Ühe poole püüeldes jääb teine tihti unarusse.

Ülesanne

Mis väljastatakse ekraanile? (Väljatrükk on nupu "Vali" all.)

```
nimi1 = "Otto"
nimi2 = "Triin"
print("Tere,",nimi1)
print(nimi2 + "!")
```

Vali
Tere,OttoTriin!

Vali
Tere, Otto Triin!

Vali
Tere,Otto
Triin!

Vali
Tere, Otto
Triin!

Vali Veateade

(Otto-Triin oli üheksakümnendate aastate keskel lastesaate tegelane.)

KÄSK *INPUT*

Kasutajalt andmete küsimiseks kõige lihtsam viis on käsk *input*, mis kõigepealt kuvab ekraanile teksti selle kohta, milliseid andmeid programm ootab ning seejärel võimaldab kasutajal sisestada vastavad andmed klaviatuurilt. Varem kasutasime *input*-i pigem nii:

```
print("Sisesta PIN-kood:")
sisestatud_pin = input()
```

Käsule *input* saab anda argumendiks selle sõne, mida tahame, et kasutajale näidataks:

```
sisestatud_pin = input("Sisesta PIN-kood: ")
```

Katsetage eelnevaid näiteid! Kuidas need omavahel erinevad?

Mäletatavasti annab *input* sisestatud info alati sõne kujul. Enne kui saame sisestatud andmeid kasutada numbrilistes arvutustes, tuleb sisestatud tekst teisendada arvuks (näiteks täisarvuks saime funktsiooniga *int*, ujukomaarvuks aga funktsiooniga *float*.

TEHTED SÕNEDEGA

Sõnedega saab teha erinevaid tehteid. Me oleme neid juba liitnud. See käis plussmärgi abil.

```
"Tere " + "hommikust"
```

Lisaks on olemas funktsioonid, mis "ühendatakse" sõne külge punkti abil. (Neid nimetatakse ka meetoditeks.) Nende funktsioonide nime lõppu pannakse samuti sulud, mis võivad olla tühjad või mittetühjad. Selliste funktsioonidega saab palju huvitavat ja kasulikku korda saata!

```
print("tartu".capitalize())
print("Tartu".endswith("tu"))
print("Tartu".lower())
print("Tartu".upper())
print("Tartu".startswith("tu"))
print("Kauneim linn on Eestis Tartu".title())
```

Katsetage seda programmi ja püüdke aru saada, mida need funktsioonid teevad.

Funktsioonid töötavad ka muutujatega, mille väärtuseks on sõne.

```
linn = "Tartu"
print(linn.lower())
```

Kõikide sõnemeetoditega saab tutvuda
aadressil http://docs.python.org/3/library/stdtypes.html#string-methods.

4.2 Kilpkonnagraafika

Seni oleme saanud programmide töö tulemused tekstilisel kujul ekraanile väljastatuna. Samuti oleme sisendi andnud tekstilisel kujul. Reaalselt kasutatavad programmid on sageli hoopis graafilise kasutajaliidesega. Selliseid me selle kursuse raames teha ei jõua, aga natuke graafikat toome sisse siiski. Selles peatükis vaatame lähemalt *Pythoni* moodulit *turtle* (tõlkes kilpkonn). Mooduliks nimetatakse teatud kogumit, kust saab vajalikke käske võtta. Joonistamiseks võtame appi tegelase - kilpkonna, kes oma virtuaalmaailmas ringi liikudes võib jätta maha jälje. See jälg võib teinekord päris huvitava ja isegi kunstilise kujuga olla.

AJALUGU

Selline virtuaalne kilpkonn ilmus esmakordselt hariduslikus programmeerimiskeeles LOGO, mille lõid aastal 1967 Wally Feurzeig ja Seymour Papert. See keel oli mõeldud just lastele algoritmilise mõtlemise arendamiseks. Praeguseks on kilpkonnagraafikat võimalik kasutada mitmetes programmeerimiskeeltes, ka *Pythonis*. Viimasel ajal eriti populaarses hariduslikus programmeerimiskeeles *Scratch* on samuti tuntavaid kilpkonnagraafika mõjutusi. LOGO programmeerimist võib proovida siin. Kokkuleppeliselt nimetatakse kilpkonnaks väikest kolmnurka joonistamisväljal. Võib-olla on huvitav teada, et üks esimesi interneti kaudu toimunud kursusi (aastal 1996, tol ajal e-posti kaudu) Eestis oli just programmeerimiskeele Logo kursus.

ALUSTAMINE JA LIIKUMINE

Järgnevalt uurime, kuidas kilpkonna *Pythonis* kasutama hakata ning vaatame, kuidas kujundeid joonistada.

Videot võite vaadata kohe või pärast mõningat katsetamist: https://youtu.be/r5L9BM6vlng

Selleks, et joonistamisega alustada, tuleb esmalt importida kilpkonna moodul. Selleks kirjutame programmi esimesele reale:

```
from turtle import *
```

Järgnevalt proovime kirjutada programmi, mille abil joonistataks ruut. Selleks kasutame sobivaid kilpkonnakäske:

- `forward(n)` - liigu edasi (ingl. *forward*) *n* sammu võrra (ühe sammu pikkus on 1 piksel ehk üks täpike ekraanil);
- `back(n)` - liigu tagasi (ingl. *back*) *n* sammu võrra;
- `left(d)` - pööra vasakule (ingl. *left*) *d* kraadi;
- `right(d)` - pööra paremale (ingl. *right*) *d* kraadi.

Eelnevaid käske sobivas järjekorras rakendades saame järgneva programmi.

Näiteprogramm. Ruut

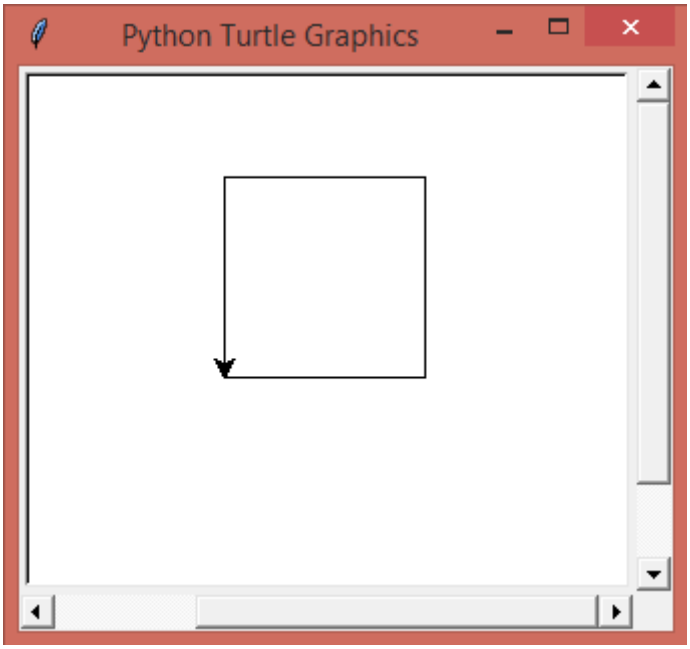
```
from turtle import *                # * lisamisel imporditakse kõik
kilpkonna käsud

forward(100)                        # Kilpkonn liigub edasi 100
pikslit
left(90)                            # Kilpkonn pöörab 90° vasakule
forward(100)                        # Kordame eelnevaid käske, sest
ruudul on neli külge
left(90)
forward(100)
left(90)
forward(100)

exitonclick()                       # Saame akna sulgeda
hiireklõpsuga
```

NB! Ärge pange oma *Pythoni* programmi nimeks turtle.py. See ajab *Pythoni* segadusse.

Tulemuseks saame



Tegelikult võib kilpkonn järjekindlalt pöörata ka 90° paremale, tulemuseks on ikka ruut.

Näiteprogrammis märkame, et kilpkonn peab täitma korduvalt samu käske: minema 100 pikslit edasi ja pöörata seejärel 90° vasakule. Seega on siin tegemist tsüklilise tegevusega ja saame kasutada tsüklit. Tsüklist aga räägime edasistes osades.

Selleks, et vähendada kirjutamisvaeva, saab kilpkonna liikuma panna ka lühemate käskudega, kasutades pikemate esituste esimesi ja viimaseid tähti.

- `fd(n)` - võrdne käsuga `forward(n)`;
- `bk(n)` - võrdne käsuga `back(n)`;
- `lt(d)` - võrdne käsuga `left(d)`;
- `rt(d)` - võrdne käsuga `right(d)`.

See robot-kilpkonn oskab teha veel palju muudki. Täpsemalt võib vaadata Pythoni dokumentatsioonist.

VÄRVIMINE

Kilpkonn oskab ka joonistatud kujundeid värvida ja muuta taustavärvi. Selleks, et värvima hakata, tuleb esmalt valida sobiv värv. Seda saab teha käsuga `color("värvi_nimetus")`, kus argumendiks tuleb kasutada sõne, mille väärtuseks on värvi ingliskeelne vaste või värvi kuueteistkümnendkoodis esitus. Näiteks musta värvi esitus kuueteistkümnedkoodis on *#000000*. Värvikoode saab vaadata siit.

Näiteks

```
color("red")                          # Kilpkonn muutub punaseks
```

või

```
color("#008000")                      # Kilpkonn muutub tumeroheliseks
```

Kui värv on valitud, siis tuleb kilpkonnale käsuga `begin_fill()` teada anda, et nüüd lähebki värvimiseks. Seda peab tegema vahetult enne kujundi joonistamist, mida värvida soovime. Värvimise lõpetamiseks tuleb peale kujundi lõppu kirjutada käsk `end_fill()`. Järgmiseks kirjutame programmi, mis joonistab punase ringi. Kilpkonn oskab joonistada ringi (ingl. *circle*) käsuga `circle(r)`, kus *r* on ringi raadius pikslites.

Näiteprogramm. Jaapani lipp

```
from turtle import *

color("red")                          # Kilpkonn muutub punaseks
begin_fill()                          # Kilpkonn alustab ringi
värvimist
circle(100)                           # Kilpkonn joonistab ringi
raadiusega 100 pikslit
end_fill()                            # Kilpkonn lõpetab ringi
värvimise

exitonclick()
```

Kui eelmisel programmi puhul oli täpselt teada, mis toimub, siis täiendame seda programmi juhuslikkusega. Selles programmis imporditakse lausa kahest moodulist.

Näiteprogramm. Sinine, must või valge

```
from turtle import *
from random import randint

värv = randint(1, 3)
if värv == 1:
    color("blue")                   # Sinine
if värv == 2:
    color("black")                  # Must
if värv == 3:
    color("white")                  # Valge valgel taustal
begin_fill()                        # Kilpkonn alustab ringi
värvimist
circle(100)                         # Kilpkonn joonistab ringi
raadiusega 100 pikslit
end_fill()                          # Kilpkonn lõpetab ringi
värvimise

exitonclick()
```

Praegu tõime kilpkonna mängu selleks, et edasisi natuke keerulisemaid teemasid paremini illustreerida. Aga mitte ainult - hakkame ikka ilusaid pilte ka tegema.

4.3 Silmaring. Keeletehnoloogia I

KEELETEHNOLOOGIA

Keeletehnoloogia on valdkond, mis tegeleb programmide ja rakenduste loomisega, mis võimaldavad arvuti abil inimkeelt mõista ja töödelda. Selles peatükis saad lugeda paarist huvitavast keeletehnoloogia suunast, millega oled ehk enda teadmata juba kokku puutunud.

KIRJALIKU TEKSTI TUVASTUS

Tekstituvastus (ingl *Optical Character Recognition* ehk *OCR*) on arvutiteaduse osa, mille ülesandeks on tarkvara abil tuvastada pildilt teksti. Tekstituvastustarkvara tunneb ära tähed, numbrid ja kirjavahemärgid ning muudab need arvutile arusaadavaks. Edaspidi on tuvastatud teksti võimalik tavapäraselt arvutis redigeerida ning ühtlasi saab sellest tekstist näiteks otsisõnade abil vajalikku infot leida.

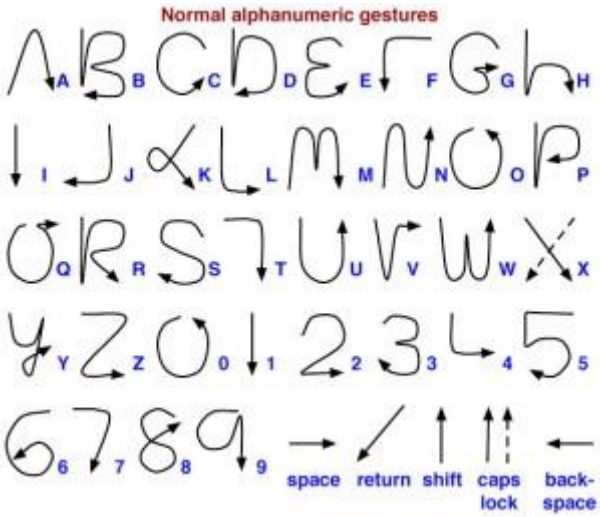
Arenenumad tekstituvastussüsteemid suudavad tuvastada väga paljudes erinevates fontides teksti. Siiski pole tulemus alati täpne - seda mõjutavad nii pildi kvaliteet kui ka näiteks teksti kujunduselemendid.

Tekstituvastuse abil on digiteeritud väga paljusid arhiive. Muuhulgas võimaldab see ka näiteks kaameratel autode numbrimärke tuvastada. Samuti leiab see kasutust postkontorites ümbrike sorteerimise automatiseerimisel. Üks tekstituvastuse algseid ja edukalt täidetud eesmärke oli aga pimedatele tekstide ettelugemise võimaldamine.

Üks kiiduväärt tekstituvastuse rakendus on Gutenbergi projekt. See on 1971. aastal Michael S. Harti poolt algatatud vanim digitaalraamatukogu. Sinna on kogutud ja digiteeritud üle 46 000 avalikult kasutatava raamatu. Raamatud on saadaval nii lihttekstina kui ka näiteks HTML, PDF ja EPUB formaatides. Enamik raamatuid on inglise keeles, kuid korralikult on esindatud ka saksa-, prantsuse-, itaalia- ja portugalikeelsed raamatud. Kuni 1989. aastani sisestati teosed käsitsi, pärast seda võeti kasutusele tekstituvastustarkvara. Kõik tekstituvastustarkvara abil lisatud raamatud on ka inimsilma poolt üle toimetatud.

Tänapäeval suudetakse tuvastada ka käsitsi kirjutatud tekste, kuigi selle puhul on tuvastusprotsent tunduvalt väiksem, sest inimeste käekirjad on erinevad ning tihti ei ole isegi ühe inimese käekiri alati samasugune. Käekirja tuvastamisega tegelev tarkvara on tavaliselt õppimisvõimeline - tarkvara suudab end kohandada uute käekirjadega.

Selleks, et suurendada tuvastusprotsenti ning vähendada erinevatest käekirjadest tulevaid erisusi, on proovitud seadmetes kasutada tavalise kirjutamise asemel ka erilisi tähestikke. Näiteks kunagistes Palm pihuarvutites kasutati kirjutamiseks tähestikku nimega Graffiti (vt pilt 1), mis sisuliselt kujutas ennast lihtsustatud varianti trükitähtedest. See kindlustas, et enamik inimesi kirjutavad oma teksti sarnaselt ning arvutil pole seda nii raske tuvastada.



Allikas: http://www.crossbrowser.net/whats-wrong-with-touch-typing/

Lisaülesanne: Rahvusraamatukogu on digiteerinud eesti ajalehed ning need internetis kättesaadavaks teinud. Valimõni enne 1945. aastat välja antud ning mõni käesoleva aasta väljaanne. Vajutades pildil ajalehe leheküljest paremat hiireklahvi ning valides avanevas menüüs "Selle lk. tekst", saad vastava lehekülje automaatselt tuvastatud teksti. Võrdle, kuidas erineb tekstituvastuse täpsus vana ja uue ajalehe puhul.

NB! Võimalik on kasutada ka sealset otsingut, et leida ülesande lahendamiseks just endale huvipakkuvat teemat käsitlevad kaks ajalehte.

MASINTÕLGE

Keeletehnoloogia üks valdkondi on masintõlge. Masintõlge seisneb arvuti abil teksti või kõne tõlkimises ühest keelest teise.

Kõige lihtsam masintõlke variant on sõnastikupõhine. See tähendab, et tõlgitavas tekstis asendatakse sõnastiku abil iga algne sõna vastava sõnaga soovitud keeles. Sõnastikupõhine masintõlge võib olla täiesti piisav, kui tõlgitavaks materjaliks on lihtsad fraasid nagu näiteks "ilus poiss". Küll aga on see võrdlemisi kasutu pikemate lausete tõlkimisel, sest ei võeta arvesse vastavate keelte semantikat ega süntaksit.

Tänapäeval kasutatakse valdavalt statistilist masintõlget. Niisuguse tõlke puhul tõlgitakse tekst vastavalt statistilistele mudelitele, mis saadakse paralleeltekstide analüüsimisel. Paralleeltekstid on sama sisuga mitmes keeles esinevad tekstid, näiteks Euroopa Liidu seadused. Tõlge saadakse tuginedes sellele, kuidas sarnaseid tekste on varasemalt tõlgitud.

Lähemalt masintõlke ajaloo ja meetodite kohta võite lugeda Heiki-Jaan Kaalepi ja Mare Koidu artiklist "Kuidas masin tõlgib".

Üks tuntuimaid statistilise masintõlke rakendusi on *Google Translate*. Kindlasti tasub vaadata seda *Google*'i poolt tehtud videot, mis seletab lihtsasti, kuidas see tõlkimisprotsess täpselt töötab. Video on inglise keeles, kuid saab valida eestikeelsed subtiitrid. Selleks klikkida videoakna all paremas

nurgas olevale  (hammasratta) nupule ning avanenud menüüst valida subtriiti keeleks eesti keel. Subtiitreid (CC) saab sisse ja välja lülitada ka nupule  vajutades: https://youtu.be/Rq1dow1vTHY

Google Translate võimaldab tõlkida nii kasutaja enda poolt sisestatud teksti kui ka terveid veebilehekülgi. Lisavõimalustena on saadaval

- kasutatavate keelte virtuaalsed klaviatuurid,
- võimalus sisestatud keel automaatselt tuvastada,
- võimalus vasteid foneetilise kirjapildiga esitada (kui tõlge pole ladina tähestikus),
- võimalus sisestatud sõnade häädlust kuulata.

Osade keelte tõlkega on *Google Translate* jõudnud arvestatavale tasemele, näiteks inglise-prantsuse tõlge on küllaltki täpne. Sama ei saa väita inglise-eesti tõlke kohta. Tingitud on see nii inglise ja prantsuse keelte suuremast omavahelisest sarnasusest kui ka sellest, et vajalikke paralleeltekste on rohkem.

Google Translate toetab hetkel 80 erinevat keelt, kuid tasub arvestada, et soovides tõlkida näiteks läti keelest mongoolia keelde, kasutatakse vaheastmena inglise keelt. See tähendab, et rakendus tõlgib kõigepealt teksti lätikeelest inglise keelde ja seejärel saadud tulemuse inglise keelest mongoolia keelde.

Tõlkimisel mängib suurt rolli ka teksti sisu. Instruktsioone ja pealiskaudseid dialooge arusaadavalt tõlkida on lihtne, aga ilukirjanduslike teoste või suisa luuletuste tõlkimisel võib saada väga kummalisi tulemusi.

Seda kõige arvestades pole ka imestada, et tõlked tihti peale soovida jätavad.

Lisaülesanne: Järgnevalt on esitatud Lewis Carrolli raamatust "Alice Imedemaal" originaalkeeles ja selle eesti- ning venekeelne tõlge. Eesti keelde on teose tõlkinud J. Kross ja vene keelde N. Demurova.

"Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversations?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her."

"Alice oli juba õige tüdinenud istumast tegevusetult õe kõrval jõepervel. Korra või teise oli ta kiiganud raamatusse, mida õde luges, kuid selles polnud ei pilte ega tegelaste kõnelusi, ja Alice mõtles: "Mis kasu on raamatust, kus pole ei pilte ega kõnelusi?"

Nüüd ta siis arutas (nii hästi-halvasti kui suutis, sest päeva kuumus tegi ta kangesti uniseks ja roiuks), kas lõbu, mida saaks karikakrapärja punumisest, kaaluks üles püstitõusmise ja karikakarde noppimise vaeva - kui äkitselt jooksis üsna ta juurest mööda valge roosasilmaline küülik."

"Алисе наскучило сидеть с сестрой без дела на берегу реки; разок-другой она заглянула в книжку, которую читала сестра, но там не было ни картинок, ни разговоров. -- Что толку в книжке, -- подумала Алиса, -- если в ней нет ни картинок, ни разговоров?

Она сидела и размышляла, не встать ли ей и не нарвать ли цветов для венка; мысли ее текли медленно и несвязно -- от жары ее клонило в сон. Конечно, сплести венок было бы очень приятно, но стоит ли ради этого подыматься? Вдруг мимо пробежал кролик с красными глазами."

Proovige *Google Translate*'i abil tõlkida antud teksti ingliskeelset varianti eesti ja vene keelde. Võrdle saadud tulemust päris tõlgetega. Proovi ka näiteks saadud tulemust *Google Translate*'iga tagasi inglise keelde tõlkida. Mida märkad?

Google translate'i abil eesti keele tõlkimisel tekkivatest probleemidest saate lähemalt lugeda Toomas Koitmäe ja Merli Mänduli artiklist "Masintõlge - kas emakeele päästerõngas?"

Tartu Ülikooli Matemaatika-informaatikateaduskonna keeletehnoloogia teadusgrupi masintõlke töörühm on loonud oma väikese eesti-inglise-eesti masintõlkesüsteemi. Vahest annab see süsteem paremaid vasteid kui *Google Translate*.

Lisaülesanne: Proovi aadressil http://masintolge.ut.ee/ asuvat rakendust mõne lause tõlkimiseks ning võrdle saadud tulemust *Google Translate*'i poolt pakutuga. Näiteks võid proovida lauseid "Ära pane teda tähele", "Mind pole küll vaja aidata" või mõnda muud meelepärast.

Keeletehnoloogiast räägime ka edaspidi silmaringi materjalides.

ALLIKAD

1. http://et.wikipedia.org/wiki/Keeletehnoloogia
2. http://en.wikipedia.org/wiki/Optical_character_recognition
3. http://www.keelekoda.ee/est-ocr-ehk-tekstituvastus/
4. http://en.wikipedia.org/wiki/Project_Gutenberg
5. http://en.wikipedia.org/wiki/Handwriting_recognition
6. http://en.wikipedia.org/wiki/Google_Translate
7. http://translate.google.com/about/intl/en_ALL/

Suur tänu Kadri Varele, kes saatis järgmised lingid eesti keelega seotud võimalustele

- Masintõlge, inglise-eesti-inglise: http://tilde.ee/
- Emotsioonituvastaja. Reaalajas töötav emotsioonituvastaja beetaversioon tuvastab aktiivsuse-passiivsuse ning positiivsuse, negatiivsuse, neutraalsuse: https://github.com/EKT1/emotional
- Audiovisuaalse kõnesünteesi prototüüp: http://massy-est.phon.ioc.ee/MASSY/peamudel.php
- Kõnetuvastus. Dikteeri: reaalajaline kõnetuvastus veebibrauseris: http://bark.phon.ioc.ee/dikteeri/
- Kõnele: eestikeelne kõnetuvastus Androidil: https://play.google.com/store/apps/details?id=ee.ioc.phon.android.speak

- Arvutaja: kõnetuvastuse abil kasutatav "intelligentne abimees"
  Androidile: https://play.google.com/store/apps/details?id=ee.ioc.phon.android.arvutaja
- Diktofon: kõnesalvestaja ja salvestuste transkribeerija
  Androidile: https://play.google.com/store/apps/details?id=kaljurand_at_gmail_dot_com.diktofon
- Kõnesalvestuste brauser: automaatselt transkribeeritud raadiosaadete arhiiv: http://bark.phon.ioc.ee/tsab/p/index
- Kollokatsioonide tuvastamine https://korpused.keeleressursid.ee/clc/
- EstNLKT: Pythoni teegid eestikeelsete vabatekstide lihtsamaks töötlemiseks https://www.keeletehnoloogia.ee/et/ekt-projektid/estnltk-pythoni-teegid-eestikeelsete-vabatektside-lihtsamaks-tootlemiseks,https://github.com/tpetmanson/estnltk

---

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

4.4 Kontrollülesanne IV

SUURED TÄHED

Paljudel dokumentidel (nt ID kaart, juhiluba) on inimese nimi (ees- ja perenimi) kirjutatud läbivalt suurte tähtedega. Kirjutage programm, mis küsib kasutajalt eraldi kõigepealt eesnime ja siis perenime (just sellises järjekorras) ning seejärel väljastab kogu nime läbivalt suurte tähtedega.

Eesnime ja perenime võib kasutaja sisestada ainult väikeste tähtedega, ainult suurte tähtedega või kasutades nii suuri kui ka väikseid tähti segamini. Programm peab igal juhul nime (ees- ja perenime) väljastama läbivalt suurte tähtedega.

Näide programmi tööst:

```
>>> %Run y14.py

  Sisestage eesnimi: KATE eliZAbeTh
  Sisestage perenimi: winslet
  KATE ELIZABETH WINSLET

>>>
```

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.

4.5 Maalähedane lugu IV

ESMANE KEELEOSKUS

Ei olnud muidugi päris kindel, et väikese koera olid just hundid ära viinud, kuigi see tundus kõige tõenäolisem. Siin kandis oli tegelikult päris palju metsloomi. Tavaliselt oli kõige rohkem muret metssigadega, kes põllud pahupidi pöörasid, aga ka huntide pahateod polnud haruldased. Tegelikult Lembitule isegi meeldis, et siin metsi ja loomi oli. Mitte nagu näiteks Kesk-Euroopas, kus linnad-külad nii tihedalt on, et üks lõpeb-teine algab ja kui pole asustust, siis on põld. Lembit oli Euroopas küllalt palju ringi käinud, eriti meeldis talle Tšehhimaa. Juba noorena oli ta lugenud vahva sõduri Švejki lugusid ja pärast oli ta mitmetes sealmainitud kohtades käinud. Näiteks oli ta käinud České Budějovices, kus muide asus bussijaam kaubanduskeskuse katusel!

Üks asi, mis talle Tšehhimaal meeldis oli see, et ta tundis, et sai mõningatest siltidest ja teadetest aru. Nimelt oskas Lembit vene keelt ja mitmed sõnad on neis keeltes sarnased - sugulaskeeled ju. Näiteks seesama hunt on vene keeles “волк” (volk), aga tšehhi keeles “vlk”. Täishäälikutega on tšehhid vahel kokkuhoidlikud! Ja tänav on vene keeles “улица” ja tšehhi keeles “ulice”. No eks eesti keeleski öeldi vanasti tänava kohta “uulits”.

Liisbeth ja Rasmus olid ka koolis natuke vene keelt õppinud - tähti juba tundsid ja lihtsamatest tekstidest said arugi. Kui Ada palus neil telekat sättida ja juhendist oli eestikeelne osa kadunud, siis asusid nad hoogsalt venekeelse juhendi kallale. Igatahes tundus see lubavam kui läti- ja leedukeelsed, mis samuti alles olid. Väga lihtsalt see tekstist aru saamine ei läinud, aga sõnaraamatu abiga said nad mure siiski lahendatud.

> VAHELEPÕIGE Rasmus, Liisbet ja Lembit muidugi ei mõelnud sellele, et programmeerimiskeeled on mõnes mõttes küllaltki sarnased loomulikele keeltele. Nagu mõned sõnad ja konstruktsioonid on mitmetes loomulikes keeltes üksteisele sarnased või lausa samasugused, nii on ka erinevates programmeerimiskeeltes sarnaseid käske ja konstruktsioone. Siiski võivad mõned loomulikud keeled üksteisest väga erinevad olla, nagu ka mitmed programmeerimiskeeled on üksteisest väga erinevad.
> Veel on sarnasus ka selles, et pärast mõningat keeleõppimist juba võib üldjoontes tekstist aru saada, aga täieliku mõistmiseni on ikka veel ruumi. Ja sealt edasi veel loomingulise tekstiloomeni...

Õhtusöögilauas räägiti keeltest ja nende oskusest mitme kandi pealt. Selgus, et Ada oskas näiteks saksa keelt. Lembit siis, nagu juba teada, oskas vene keelt ja oli koolis ka saksa keelt õppinud. Rasmus oskas inglise keelt, natuke vene ja prantsuse keelt. Liisbeth oskas inglise keelt, natuke vene ja saksa keelt. Põnev oli see, et kõigil oli keele õppimise ajast mõni luuletus või laul meelde jäänud! Või vähemalt algus!

Igatahes tõdesid nad, et isegi kui teist keelt kunagi eriti kasutada ei ole tulnud, ei kahetse nad küll selle õppimist. Kunagi ei või teada, milleks see kõik hea võib olla! Teiselt poolt olid nad muidugi rõõmsad, et kõiki asju saab ka eesti keeles öelda!

Lisaülesandeid

Lisaülesanne: Nädalapalk

Kui inimene töötab nädalas 40 tundi või vähem, siis nende tundide eest saab ta palga vastavalt oma tavalisele tunnitasule. Kui inimene töötab rohkem kui 40 tundi, siis ületundide eest on tunnitasu 50% kõrgem.

Koostage programm, mis küsib kasutajalt töötundide arvu nädalas ja tavalise tunnitasu (küsige just sellises järjekorras, esimese asjana - töötundide arvu ja teise asjana - tunnitasu) ning väljastab vastava nädalapalga arvestades ka ületundidega, kui neid on.

Nt. Kui töötundide arv nädalas on 30 ja tunnitasuks on 10, siis nädalapalgaks on 300 eurot (arvutamise käik: 30 * 10). Kui töötundide arvuks on 60 ja tunnitasuks on 8, siis nädalapalgaks on 560 eurot (arvutamise käik: 40 * 8 + (60 - 40) * (8 * 1.5) = 320 + 20 * 12 = 320 + 240 = 560).

Näited programmi tööst:

```
>>> %Run nadalapalk.py

  Sisesta töötundide arv nädalas: 30
  Sisesta töötunni tasu: 10
  Nädalapalk on 300

>>>
>>> %Run nadalapalk.py

  Sisesta töötundide arv nädalas: 60
  Sisesta töötunni tasu: 8
  Nädalapalk on 560.0

>>>
```

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

Lisaülesanne: Eurod ja sendid

Kirjutage programm, mis küsib kasutajalt arvu, mis tähistab sente ning väljastab sõne, kus rahakogus on esitatud täiseurodena ja sellest üle jäävate sentidena, või ainult eurodena või ainult sentidena, nt:

- kui kasutaja sisestab arvu 207, siis väljastatakse "2 eurot ja 7 senti"
- kui kasutaja sisestab arvu 101, siis väljastatakse "1 euro ja 1 sent" (NB! mitte "1 eurot ja 1 senti", ainsusel ja mitmusel tuleb vahet teha)
- kui kasutaja sisestab arvu 95, siis väljastatakse "95 senti" (st. ilma eurodeta)
- kui kasutaja sisestab arvu 100, siis väljastatakse "1 euro" (st. ilma sentideta)

Vihje

- // täisarvuline jagamine
- % jäägi leidmine

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