

Vooskeem ja Python

Plaan

- Tutvus vooskeemiga, mis peaks aitama algoritme lahti mõtestada.
- Summa, loendamine, miinimum ja maksimum
- Programmeerimiskeele Python põhilaused ja paar näiteprogrammi.

Meenutuseks mõisted

- muutuja ja andmetüübid
- omistamine
- aritmeetikaavaldis
- loogikaavaldis
- valiku- ehk tingimuslause (if-lause)
- tsüklilause (while-lause ja for-lause)

Vooskeem Flowgorithmiga

Kui Scratch kasutab arvatavasti ideid sellisest algoritmide esitusvahendist, nagu Nassi–Shneidermani diagramm (kirjeldatud 1972 a, https://en.wikipedia.org/wiki/Nassi%E2%80%93Shneiderman_diagram), siis vooskeem (pikka aega tuntud nimega plokkskeem) (<https://en.wikipedia.org/wiki/Flowchart>) on oma olemuselt tunduvalt vanem vahend.

Vooskeem (*ingl flowchart, flow diagram*) loodi algselt möödunud sajandi alguses (töö)protsesside kirjeldamiseks. 1947. a. kirjeldas John von Neumann oma teoses "*Planning and coding of problems for an electronic computing instrument, Part II Volume I*" vooskeemi võimalusi algoritmide esitamiseks. Praegu kasutatav vooskeemi komponentide kogum erineb siiski John von Neumanni pakutust.

Tölge / mõiste "vooskeem" on toodud IT terministandardi sõnastikus (<http://www.eki.ee/dict/its/>) ja Andmekaitse ja infoturbe portaalis (<https://akit.cyber.ee/term/5413-flowchart>). Sõnastikes toodud definitsioon on järgmine: "*Ülesande sammhaaval lahendamise protsessi graafiline esitus, mis protsessi või programmi projekteerimise või dokumenteerimise eesmärgil kasutab voojoontega ühendatud ja sobivalt annoteeritud geomeetrilisi kujundeid.*"

Tore vahend vooskeemide joonistamiseks on Flowgorithm (<http://www.flowgorithm.org/>). Tarkvara tuleb installeerida ja originaalis on olemas vaid Windowsi variant. Otsides võib leida ka MacOSi ja Linuxi variante, aga info nende töötavuse kohta on ebaselge.

Joonistame programmiga Flowgorithm mõned vooskeemid. Ja vaatame, mida see programm veel võimaldab (nt skeemi tõlkimist mõnda programmeerimiskeelde).

Ülesanne 1 Kahe arvu võrdlemine

Sisestatakse kaks arvu, algoritm teatab, kumb arv on suurem.

Ülesanne 2 Õnneseitse (*ingl lucky seven*)

Kui suur on lootus võita õnnemängus? Selleks võib välja arvutada tõenäosuseid, aga võib ka mängu käiku simuleerida.

Mängija veeretab kahte täringut. Kui täringutel olevate silmade summa on 7, võib mängija 4 raha. Kui ei ole 7, siis kaotab 1 raha. Mängu alguses tehakse panus ehk otsustatakse, kui palju raha mängu panna. Seejärel veeretatakse täringuid ja suurendatakse või vähendatakse mängu pandud rahasummat. Mäng lõppeb, kui raha on otsas. Täringuid aitab veeretada juhuslike arvude generaator.

Täienduseks võib mängu lõpus teatada, mitu korda täringuid veeretati enne raha otsasaamist ja kui suur oli vahepeal kõige suurem rahasumma.

Täringuveeretuste loendamiseks, rahasumma leidmiseks ja vahepealse suurima summa tuvastamiseks on oma algoritmilised võtted: summeerimine, loendamine, suurima (või vähima) väärtuse leidmine.

Ülesanne 3 Spordipäev

Koolis oli spordipäev ja lapsed viskasid palli. Õpetaja pani oma pabermärgmikusse kirja iga õpilase parima viske. Siis jooksis õpetaja juurde väike Paul ja küsis: "Õõõpetaja-õõõpetaja! Mitmenda koha ma sain?" Kuidas leiab õpetaja kõige hõlpsamalt Pauli koha?

Algoritmile mõeldes lähtu sellest, et kõigepealt sisestatakse eraldi Pauli tulemus ja seejärel ükshaaval suvalises järjekorras ükshaaval teiste õpilaste tulemused. Võimalikud vastused: Paul sai palliviskes 4. koha. Paul jagas 2. kuni 4. kohta.

String ehk sõne

Senised ülesanded olid seotud arvandmetega. Aga ka tekste on võimalik arvuti abil töödelda. Andmetüüpi **string** ehk **sõne** (*ingl string*) kasutatakse enamasti tekstist koosnevate väärtuste talletamiseks ja töötlemiseks. Kuid teinekord võib olla kasulik ka arve stringina hoida.

Stringist on võimalik indeksi abil tähti (üldisemalt sümboleid) ühe või ka mitme kaupa kätte saada. Indekseerimine algab 0-ga.

M	a	a	l	i
0	1	2	3	4

Näiteks:

```
nimi = "Maali"
```

```
nimi[1:4] -> "aal"
```

```
nimi[0] -> "M"
```

Proovime järgnevalt stringi töötlemist ja koos sellega ka arvukalt if-lauset kasutamist. Koodi kirjutame otse Pythoni keskkonnas Thonny (<https://thonny.org/>, vajab koduarvutis installeerimist).

Ülesanne 4 Isikukood

Programmi sisendiks on kasutaja nimi ja Eesti Vabariigi kodaniku isikukood (vaata veebilehel eraldi lingi all isikukoodi selgitust). Leia isikukoodist sünnipäev (koos täisaastaga!), vanus ja sugu ning trüki võimalikult viisakalt ja arusaadavalt ekraanile.

Kontrolli, kas isikukoodi kontrollsumma on õige.

Vanuse arvutamiseks on vaja tänast kuupäeva. Seda saab tuvastada moodulis `time` oleva funktsiooniga `localtime()`, näiteks nii: `ajaandmed = time.localtime()`

Programmi algusesse tuleb kindlasti lisada rida `import time`. Sellega öeldakse Pythoni interpretaatorile, et funktsioone tuleb otsida moodulist `time`.

Funktsioon `localtime()` tagastab terve hulga andmeid (Soovi korral võid teha täpsemat tutvust siin: <http://docs.python.org/3/library/time.html>).

Vanuse leidmiseks on vaja aastat - `aasta = ajaandmed[0]`, kuud - `kuu = ajaandmed[1]` ja päeva - `p2ev = ajaandmed[2]`.

Täpse vanuse määramiseks harjuta if-lauset, mis arvestab sellega, kas sünnipäev on juba olnud või mitte.

Lihtsam variant on leida vanus, mis käesoleval aastal varem või hiljem kätte jõuab.

Varasema programmeerimise kogemuse korral võib isikukoodi kontrollsumma leidmiseks kasutada tsüklit. Kui kogemus vähene, siis on lihtsam kirjutada üks väga pikk avaldis, mis vajalikud korrutamised ja liitmised ära teeb. Kui aga üldse kogemus puudub, siis võiks rõõmu tunda vaid soo, sünniaja ja vanuse tuvastamise üle :)

NB! Suure tõenäosusega me tegelikult siia lõpuni välja ei jõua ja uurime isikukoodi hoopis järgmisel kohtumisel.