Arvuhulgad ja avaldised

Gümnaasiumi reaalharu

Jan Willemson

https://varamu.eu

Saatesõna

Sa hoiad käes Eesti gümnaasiumi reaalharule mõeldud eksperimentaalsesse matemaatikaõpikute sarja kuuluvat õpikut. Sari on leidnud inspiratsiooni 2023. aasta gümnaasiumi laia matemaatika õppekavast, kuid ei vasta sellele üksüheselt. Autori hinnangul vajabki Eesti gümnaasiumi matemaatika ainekava põhjalikku uuendamist. Käesolev sari kujutab endast autori nägemust sellest, milline matemaatika ainekava 21. sajandi 1. veerandi lõpul välja peaks nägema, et üldhariduskooli lõpetajad oleksid valmis jätkama õpinguid ühiskonnale olulistel erialadel.

Sarja sõsarõpikuks on sama autori „Võistlusmatemaatika põhivara", millest huvitatud lugeja leiab palju täiendavaid ülesandeid ning süvendatud materjali. Mõned jaotised ja ülesanded on kahel õppematerjalil ka ühised – aga eks matemaatika, mida neis käsitletakse, ongi ju üks ja seesama.

Sarja õpikud on hetkel mustandi staatuses – neis võib esineda lünki, vigu ning muid vajakajäämisi. Autor on tänulik kommentaaride ja tagasiside eest, mida saab saata aadressile `matemaatika@varamu.eu`.

Sisukord

PEATÜKK 1

Hulgad ja arvuhulgad

1.1 Hulga mõiste. Tähtsamad operatsioonid hulkadega

Hulga all mõistame mingite üksteisest erinevate objektide järjestamata kogumit. Hulka moodustavaid objekte nimetame hulga *elementideks*.

Matemaatikas võivad hulgad olla nii lõplikud kui ka lõpmatud, arvutis aga tekib lõpmatute hulkade esitamisega probleem, sest arvuti ressursid (näiteks mälumaht ja salvestusruum) on paratamatult piiratud. Pythonis on hulkade jaoks olemas andmestruktuur `set`, aga selle abil saab esitada ainult lõplikke hulki.

Deklareerime hulga H, mis on hakatuseks tühi (st ei sisalda elemente; selle olukorra kohta kirjutame matemaatikas $H = \varnothing$):

```
>>> H = set()
```

ja lisame sinna mõned elemendid käsuga `add`:

```
>>> H.add("Õun")
>>> H.add("Kirss")
>>> H.add("Pirn")
>>> H.add("Kirss")
```

Laseme Pythonil hulga H välja trükkida:

```
>>> print(H)
{'Kirss', 'Pirn', 'Õun'}
```

Python kasutab hulga väljatrükil matemaatikas tavalist tähistust, kus elemendid loetakse üles looksulgude vahel ning eraldatakse üksteisest komadega.

Saadud väljatrükis paneme tähele kahte asjaolu. Esiteks ei kuvata hulga elemente tingimata selles järjekorras, milles me neid lisasime (ja polegi vaja, sest hulk on järjestamata struktuur!). Teiseks kuvatakse element $Kirss$ ühel korral,

kuigi me lisasime teda kaks korda (sest hulgas esineb iga element ainult ühes koopias!).

Muidugi ei pea Pythonis elemente ühekaupa lisama, me võime nad juba hulka luues loendina ette anda:

```
K = set(["Kirss","Õun","Pirn"])
```

Juba loodud hulgale mitut elementi Pythonis korraga niisama lihtsalt lisada ei saa:

```
>>> S = set()
>>> S.add("Kirss","Õun","Pirn")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: set.add() takes exactly one argument (3 given)

>>> S.add(["Kirss","Õun","Pirn"])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Käsku `S.add(["Kirss","Õun","Pirn"])` interpreteerib Python korralduse-na lisada hulgale S loend `["Kirss","Õun","Pirn"]`. Andmestruktuur `set` nõuab samas, et kõik tema elemendid oleksid muutmatud, aga loendis saab elemente pärast loomist muuta. Niisiis antaksegi veateade. Sama moodi ei saa Pythoni `set` andmestruktuuri elementideks olla teised `set`-id, sest ka need on muudetavad (näiteks elementide lisamisega). See on peale lõplikkuse nõude teine suur erinevus Pythoni `set` andmestruktuuri ja matemaatilise hulga mõiste vahel – matemaatikas on hulga elementidena teised hulgad täiesti lubatud.

Ülesanne 1.1 Loendi asemel võib Pythonis vajadusel kasutada mittemuudetavat enniku andmestruktuuri. Kas järgmised Pythoni käsud annavad vea või mitte?

```
>>> S = set()
>>> S.add(("Kirss","Õun","Pirn"))
```

Kui viga ei tule, siis mitmeelemendiline on saadav hulk S?

Kui hulgad koosnevad samadest elementidest, loetakse nad võrdseteks:

```
>>> H = set(["Kirss","Õun","Pirn"])
>>> K = set(["Õun","Pirn","Kirss"])
>>> H==K
True
```

Objektide kohta saab uurida, kas nad kuuluvad hulka või mitte. Pythonis kasutatakse selleks operaatorit `in`:

```
>>> "Kirss" in H
True
>>> "Ploom" in H
False
>>> "Ploom" not in H
True
```

Matemaatilistes tähistustes kirjutame vastavalt $Kirss \in H$ ja $Ploom \notin H$.

Kuna hulga elemendid pole järjestatud, ei saa Pythonis hulga elemente indeksi alusel pärida:

```
>>> H[0]
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: 'set' object is not subscriptable
```

Küll aga saab `for`-tsükliga läbi hulga elementide käia (pane tähele, et sa ei tohi eeldada midagi läbikäimise järjekorra kohta!):

```
>>> H = set(["Kirss","Õun","Pirn"])
>>> for h in H:
        print(h)

Pirn
Kirss
Õun
```

Hulga elementide arvu saame sarnaselt loendile leida operaatoriga `len`:

```
>>> len(H)
3
```

Ülesanne 1.2 Kirjuta funktsioon `unikaalne`, mis võtab sisendiks loendi ja väljastab selle loendi unikaalsete elementide arvu, st töötab niimoodi:

```
>>> unikaalne(["Kirss","Ploom","Õun","Ploom","Kirss"])
3
```

Definitsioon 1.1. *Kui hulga A kõik elemendid kuuluvad ka hulka B, ütleme, et A on hulga B* alamhulk*, ja kirjutame $A \subset B$. Kui $A \subset B$ ja $A \neq B$, siis ütleme, et A on hulga B* pärisalamhulk.

Pythonis saab alamhulgaks olemist tuvastada operaatoriga `issubset`:

```
>>> H = set(["Kirss","Õun","Pirn"])
>>> K = set(["Õun","Pirn","Kirss","Ploom"])
>>> H.issubset(K)
True
>>> K.issubset(H)
False
```

Ülesanne 1.3 Mida saab öelda hulkade A ja B kohta, kui $A \subset B$ ja $B \subset A$?

Ülesanne 1.4 Kas leidub hulk, mis on iga hulga alamhulgaks?

Hulkadel saab defineerida mitmeid operatsioone. Tähtsamad neist on ühisosa ja ühendi leidmine.

Definitsioon 1.2. *Hulkade A ja B* ühisosaks *$A \cap B$ nimetame hulka, mis koosneb mõlemasse antud hulka koosnevatest elementidest. Hulkade A ja B* ühendiks *$A \cup B$ nimetame hulka, mille elemendid kuuluvad vähemalt ühte antud hulkadest.*

Pythonis saab ühisosa ja ühendit leida vastavalt operaatoritega `intersection` ja `union`:

```
>>> H = set(["Kirss","Õun"])
>>> K = set(["Kirss","Pirn","Ploom"])
>>> H.intersection(K)
{'Kirss'}
>>> H.union(K)
{'Kirss', 'Pirn', 'Õun', 'Ploom'}
```

Tehnilistel põhjustel on need operatsioonid Pythonis realiseeritud andmestruktuuri `set` meetoditena ning neid tuleb seetõttu välja kutsuda konkreetsel hulgal. Samas pole vahet, kummal hulgal väljakutse täpselt teha, tulemus jääb samaks:

```
>>> K.intersection(H)
{'Kirss'}
>>> K.union(H)
{'Kirss', 'Pirn', 'Õun', 'Ploom'}
```

Ülesanne 1.5 Mida võib öelda hulkade A ja B kohta, kui $A \cap B = A$? Aga kui $A \cup B = A$?

1.2 Naturaalarvude ja täisarvude hulk

Kõige väiksem lõplik hulk on tühi hulk[1] ja tema elementide arvu tähistame sümboliga 0 (see ongi arvu 0 definitsioon!).

Suuruselt järgmised on hulgad, mille ainsaks pärisalamhulgaks on tühi hulk (näiteks hulgad $\{Kirss\}$, $\{Ploom\}$ ja $\{Pirn\}$). Osutub, et neis kõigis on sama palju elemente ning seda elementide arvu tähistame sümboliga 1. (Arvasid ära, see ongi arvu 1 definitsioon.)

Nõnda jätkates võtame kasutusele sümbolid järjest suuremate hulkade elementide arvude tähistamiseks. Traditsiooniliselt on nendeks sümboliteks 2,3,4 jne, milledest saame moodustada omaette hulga.

Definitsioon 1.3. Naturaalarvud *on arvud, mis saadakse lõplike hulkade elementide loendamisel. Nende hulka tähistame*

$$\mathbb{N} = \{0,1,2,3,4,\ldots\}.$$

Naturaalarvudega saab mugavalt loendada objekte, mis *on olemas*. Praktikas aga tulevad sageli ette ka kogused, mis *jäävad puudu*. Nii näiteks on majanduses igapäevasteks mõisteteks võlg ja kahjum.

[1] Rangelt võttes me alles defineerime siinkohal arve, niisiis ei saa me objekte võrrelda nende arvulise mõõdu alusel. Küll aga saame hulki võrrelda alamhulgaks olemise mõttes. „Kõige väiksem hulk" tähendab niisiis hulka, mis sisaldub alamhulgana kõigis teistes; vt ülesannet 1.4.

Teisest küljest võime naturaalarvudega väljendada füüsikalisi suurusi, näiteks mingist hetkest möödunud aega tundides. Kuidas aga rääkida ajahetkedest, mis esinesid minevikus?

Siinkohal tulevad meile appi negatiivsed arvud. Igale naturaalarvule a seame vastavusse tema *vastandarvu* $-a$ ja loeme, et

$$a + (-a) = (-a) + a = 0\,.$$

Arvu 0 vastandarvuks loeme teda ennast. Naturaalarvude ja nende vastandarvude hulka nimetame kokku täisarvude hulgaks.

Definitsioon 1.4. Täisarvude *hulk on*

$$\mathbb{Z} = \{\ldots, -4, -3, -2, -1{,}0{,}1{,}2{,}3{,}4, \ldots\}\,.$$

Ilmselt kehtib seos $\mathbb{N} \subset \mathbb{Z}$.

Pythonis esitatakse naturaal- ja täisarve nende standardsel, matemaatikast tuttaval moel ning nendega toimetamine vastab reeglitele, millega me juba harjunud oleme:

```
>>> 2+3
5
>>> 2+(-3)
-1
>>> (-2)*(-3)
6
```

Erinevalt paljudest teistest programmeerimiskeeltest lubab Python opereerida väga pikkade naturaal- ja täisarvudega. Palume Pythonil näiteks arvutada 2^{1000} (astendamisoperaatoriks on Pythonis `**`):

```
>>> 2**1000
107150860718626732094842504906000181056140481170553360744375038837035105112493612249319837881569585812759467291755314682518714528569231404359845775746985748039345677748242309854210746050623711418779541821530464749835819412673987675591655439460770629145711964776865421676604298316526243868372056680693760
```

Sisuliselt on esitatavate arvude suuruse piiriks arvuti mälumaht.

1.3 Ratsionaalarvud

Peale tervete objektide on praktikas kasulik uurida ka osalisi. Koogi või maalapi saab osadeks jagada, tunnist võib mööduda ainult pool ning maratonijooks katkeda kolmandiku peal. Nende sündmuste kirjeldamiseks on kasutusele võetud ratsionaalarvud, mis väljendavad täisarvuliste suuruste jagamist täisarvuliseks hulgaks võrdseteks osadeks (arvestades muidugi, et 0-ks osaks jagada ei saa).

Definitsioon 1.5. Ratsionaalarvudeks *nimetame arve kujul* $\frac{a}{b}$*, kus* $a, b \in \mathbb{Z}$ *ja* $b \neq 0$*. Ratsionaalarvude hulka tähistame* $\mathbb{Q}$.

Ülesanne 1.6 Tõesta, et iga täisarv on ratsionaalarv, st $\mathbb{Z} \subset \mathbb{Q}$.

Kui naturaal- ja täisarvude esitused on üheselt määratud, siis ratsionaalarvude omad mitte. Kui jagada kaks kooki võrdselt kolme lapse vahel, siis saab igaüks neist sama palju, kui igaüks kuuest lapsest, kelle vahel jagataks ära neli kooki. Murdude keeles väljendatuna kirjutame, et $\frac{2}{3} = \frac{4}{6}$. Siit saame tingimuse ratsionaalarvude võrdsuseks.

Definitsioon 1.6. *Ratsionaalarvud $\frac{a}{b}$ ja $\frac{c}{d}$ loeme võrdseteks, kui kehtib võrdus $a \cdot d = b \cdot c$.*

Näeme, et ratsionaalarvud $\frac{a}{b}$ ja $\frac{ka}{kb}$ (kus $k \neq 0$) on võrdsed, sest $a \cdot kb = b \cdot ka$.

See tähendab, et kui mingi ratsionaalarvu (murru) $\frac{m}{n}$ lugejal m ja nimetajal n on ühine tegur d, siis saame selle murru ühise teguriga d taandada, nii et tema väärtus ei muutu. Kui nimetaja n on negatiivne, võime lugeja ja nimetaja läbi korrutada arvuga -1 ja muru väärtus jääb ikka samaks. Kokkuvõttes saame ratsionaalarvu alati viia kujule, kus tema murru lugeja ja nimetaja on ühistegurita ning nimetaja on positiivne. See kuju (mida nimetame *taandatud* esituseks) on iga ratsionaalarvu jaoks nüüd üheselt määratud.

Pythonis on ratsionaalarvude jaoks olemas moodul `fractions`, milles on defineeritud ratsionaalarvude loomise käsk (konstruktor) `Fraction`. Konstruktorile `Fraction` saab ette anda erinevaid sisendeid ning konstruktor annab endast parima, et leida sisendile vastav (taandatud!) ratsionaalarv. Kahe sisendi puhul loeb `Fraction` esimese neist murru lugejaks ja teise nimetajaks.

```
>>> from fractions import Fraction
>>> Fraction(16, -10)
Fraction(-8, 5)
>>> Fraction(123)
Fraction(123, 1)
>>> Fraction()
Fraction(0, 1)
>>> Fraction('3/7')
Fraction(3, 7)
```

Defineeritud ratsionaalarvudega saab arvutada nii nagu arvudega ikka, kusjuures vastus antakse jälle taandatud kujul:

```
>>> a = Fraction(1,6)
>>> b = Fraction(3/8)
>>> a+b
Fraction(13, 24)
>>> a-b
Fraction(-5, 24)
>>> a/b
Fraction(4, 9)
>>> a*b
Fraction(1, 16)
```

Need tulemused väljendavad vastavalt võrdusi

$$\frac{1}{6} + \frac{3}{8} = \frac{13}{24} \quad \frac{1}{6} - \frac{3}{8} = \frac{-5}{24} \quad \frac{1}{6} : \frac{3}{8} = \frac{4}{9} \quad \text{ja} \quad \frac{1}{6} \cdot \frac{3}{8} = \frac{1}{16}.$$

Teine võimalus ratsionaalarvu esitamiseks on lugeja nimetajaga läbi jagada ning leida jagatisele vastav kümnendmurd. Seejuures võib saadav kümnendmurd tulla kas täpne

```
>>> 37/20
1.85
>>> 3/8
0.375
```

või ligikaudne

```
>>> 47/30
1.5666666666666667
>>> 2/7
0.2857142857142857
```

Python ümardab ligikaudsed vastused küll ära, kuid sellegipoolest paneme tähele midagi huvitavat: jagamisel tekib periood. Perioodi tähistatakse sulgude abil; nii võime kirjutada, et $\frac{47}{30} = 1{,}5(6)$ ja $\frac{2}{7} = 0,(285714)$. Tegelikult võime ju ka täpse vastuse lugeda perioodiliseks nulliga perioodis: $\frac{3}{8} = 0{,}375(0)$.

Ülesanne 1.7 Uuri arvude $\frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \frac{4}{7}, \frac{5}{7}, \frac{6}{7}$ perioode. Kas märkad midagi huvitavat?

Põhjus, miks murdarvude kümnendmurdudeks teisendamisel periood tekib, on tegelikult lihtne. Vaatleme näiteks jagamistehet $2 : 7$:

$$
\begin{array}{l}
2,\,000000 \;:7 \;=\; 0,(285714) \\
\underline{1,\,4} \\
\quad 60 \\
\quad \underline{56} \\
\quad\;\; 40 \\
\quad\;\; \underline{35} \\
\quad\;\;\;\; 50 \\
\quad\;\;\;\; \underline{49} \\
\quad\;\;\;\;\;\; 10 \\
\quad\;\;\;\;\;\; \underline{7} \\
\quad\;\;\;\;\;\;\;\; 30 \\
\quad\;\;\;\;\;\;\;\; \underline{28} \\
\quad\;\;\;\;\;\;\;\;\;\; 2
\end{array}
$$

Näeme, et lõpuks tekib jagamisel jääk 2, mistõttu kogu protsess hakkab kordama. Üldisemalt on selge, et arvutades pika jagamise abil murru $\frac{m}{n}$ väärtust, on võimalikke jääke kokku ülimalt n. Seepärast peab hiljemalt $n + 1$ sammu järel tekkima jääk, mis on korra juba esinenud, ning sellest kohast algabki jagatises periood.

Osutub, et kehtib ka vastupidine väide: iga perioodilise kümnendmurru saab teisendada harilikuks murruks. Vaatleme näiteks perioodilist kümnendmurdu $0{,}363636\ldots = 0,(36)$ ja tähistame tema väärtust muutujaga x. Siis $100x =$

$36{,}363636\ldots$ ning järelikult

$$\begin{aligned} 100x - x &= 36\,, \\ 99x &= 36\,, \\ x &= \frac{36}{99} = \frac{4}{11}\,. \end{aligned}$$

On selge, et sama võtet saame kasutada iga perioodilise kümnendmurru korral. Kokkuvõttes kehtib järgmine väide.

Teoreem 1.1 Ratsionaalarvud vastavad parajasti perioodilistele kümnendmurdudele.

Ülesanne 1.8 Avalda hariliku murruna ratsionaalarvud

(a) $1{,}(5)$;

(b) $0{,}(027)$;

(c) $0{,}(230769)$;

(d)

Esita murrud taandatud kujul!

1.4 Irratsionaalarvud ja reaalarvud

Ratsionaalarvude hulk $\mathbb{Q}$ on juba piisavalt rikas, et paljude praktiliste rakenduste jaoks piisata. Siiski selgub, et leidub lihtsaid suurusi, millel pole ratsionaalarvudes täpset väärtust.

Üheks esimeseks niisuguseks väärtuseks, mis matemaatika ajaloos leiti, on ruudu diagonaali ja küljepikkuse suhe. Pythagorase teoreemi põhjal teame, et kui ruudu küljepikkus on a, siis tema diagonaali d saame leida võrdusest $d^2 = a^2 + a^2$ ehk $d^2 = 2a^2$, millest järeldub, et $\frac{d^2}{a^2} = 2$ ja $\frac{d}{a} = \sqrt{2}$.

Teoreem 1.2 $\sqrt{2}$ ei ole ratsionaalarv.

Tõestus. Oletame vastuväiteliselt, et $\sqrt{2}$ on ratsionaalarv. Siis peavad leiduma ühistegurita täisarvud m ja n nii, et

$$\begin{aligned} \sqrt{2} &= \frac{m}{n}\,, \\ 2 &= \frac{m^2}{n^2}\,. \end{aligned}$$

Viimasest võrdusest järeldub, et m peab olema paarisarv. Muuhulgas tähendab see, et m^2 jagub 4-ga. Kuna aga $\frac{m^2}{n^2} = 2$, peab ka n jaguma 2-ga. Oleme saanud vastuolu eeldusega, et m ja n on ühistegurita. Vastuolu sai aga tekkida ainult eeldusest, et $\sqrt{2}$ on ratsionaalarv. □

Ülesanne 1.9 Kas $\sqrt{3}$ on ratsionaalarv? Aga $\sqrt{4}$?

Kuigi $\sqrt{2}$ täpset esitust pole võimalik leida ei hariliku ega kümnendmurruna, saame tema väärtust hinnata ligikaudselt. Paneme tähele järgmisi võrratusi:

$$1 < \sqrt{2} < 2\,, \quad \text{sest} \quad 1^2 = 1 < 2 < 2^2 = 4\,,$$
$$1{,}4 < \sqrt{2} < 1{,}5\,, \quad \text{sest} \quad 1{,}4^2 = 1{,}96 < 2 < 1{,}5^2 = 2{,}25\,,$$
$$1{,}41 < \sqrt{2} < 1{,}42\,, \quad \text{sest} \quad 1{,}41^2 = 1{,}9881 < 2 < 1{,}42^2 = 2{,}0164\,.$$

Niimoodi jätkates saame $\sqrt{2}$ väärtuse leida järjest täpsemini ja täpsemini. Python võimaldab ruutjuuri arvutada mooduli `math` meetodiga `sqrt`:

```
>>> from math import sqrt
>>> sqrt(2)
1.4142135623730951
```

Teoreemidest 1.1 ja 1.2 järeldub, et $\sqrt{2}$ esitus kümnendmurruna peab olema mitteperioodiline.

Definitsioon 1.7. *Lõpmatuid mitteperioodilisi kümnendmurde nimetame* irratsionaalarvudeks. *Irratsionaalarvude hulka tähistame* $\mathbb{I}$.

Irratsionaalarve on mingis mõttes oluliselt rohkem kui ratsionaalarve. Saab tõestada, et irratsionaalne on näiteks ka ringjoone ümbermõõdu ja diameetri suhe π.

Definitsioon 1.8. *Kümnendmurdudena esinevaid arve nimetame* reaalarvudeks. *Nende hulka tähistame* $\mathbb{R}$.

Kuna iga kümnendmurd on kas perioodiline või mitteperioodiline, aga mitte mõlemat korraga, kehtivad seosed $\mathbb{R} = \mathbb{Q} \cup \mathbb{I}$ ja $\mathbb{Q} \cap \mathbb{I} = \varnothing$.

Reaalarvude kasutamisel Pythonis (ja ka paljudes teistes programmeerimiskeeltes) tuleb olla ettevaatlik, sest reaalarvude sisemine esitus arvutis on optimeeritud ligikaudsete, mitte täpsete arvutuste jaoks. Näiteks ei pruugi kehtida lihtsad võrdused nagu $1{,}1^2 = 1{,}21$:

```
>>> 1.1**2 == 1.21
False
```

Segadusse toob selgust see, kui uurida, millega $1{,}1^2$ Pythoni arust õigupoolest võrdub:

```
>>> 1.1**2
1.2100000000000002
```

Kahe reaalarvu vahelise võrduse asemel on Pythonis mõtet uurida, kas nende arvude vahe on piisavalt lähedal nullile. Seda saab teha kontrollides näiteks võrratuste ahelat

```
>>> -0.0000001 < 1.1**2 - 1.21 < 0.0000001
True
```

Seda võrratuste ahelat saab kirjutada ka kompaktsemalt, kasutades absoluutväärtust.

Definitsioon 1.9. *Reaalarvu* x absoluutväärtuseks *nimetatame suurust*

$$|x| = \begin{cases} -x, & \textit{kui} \quad x < 0 \\ x, & \textit{kui} \quad x \geqslant 0 \end{cases}.$$

Sisuliselt kustutab absoluutväärtus reaalarvu eest miinuse juhul kui see seal olema juhtub. Absoluutväärtus on garanteeritult mittenegatiivne ja Pythonis saab teda leida funktsiooniga `abs`:

```
>>> abs(2.5)
2.5
>>> abs(-3.14)
3.14
>>> abs(0)
0
```

Eespool kirja pandud võrratuste ahela saame niisiis kirjutada kompaktsemalt kujul

```
>>> abs(1.1**2 - 1.21) < 0.0000001
True
```

Ülesanne 1.10 Tõesta, et suvaliste reaalarvude x ja y korral kehtivad seosed

(a) $|x \cdot y| = |x| \cdot |y|$,

(b) $|x + y| \leqslant |x| + |y|$.

Tuletame meelde, et väga suuri ja väga väikeseid reaalarve on mugavam esitada standardkujul.

Definitsioon 1.10. *Reaalarvu* x standardkujuks *nimetame esitust*

$$x = \pm a \cdot 10^b,$$

kus $1 \leqslant a < 10$ *ja* $b \in \mathbb{Z}$. *Arvu* a *nimetame arvu* x tüveks *ning arvu* b *tema* eksponendiks.

Pythonis kirjutatakse arvude a ja b vahele täht `e` (sõnast 'eksponent'). Kui võrratused $1 \leqslant a < 10$ ei kehti, teisendab Python vastavaid arve automaatselt.

```
>>> 1e10
10000000000.0
>>> 150e50
1.5e+52
>>> 0.003e-5
3e-08
```

1.4.1 Projektülesanne: $\mathbb{Q}[\sqrt{2}]$

Teoreemis 1.2 nägime, et ratsionaalarvude hulgas ei leidu elementi, mille ruut oleks 2. Kui meil on niisugust arvu (mida kokkuleppeliselt tähistatasime $\sqrt{2}$) siiski vaja täpselt esitada, saame ühe võimalusena laiendada ratsionaalarvude hulka kõigi mitteperioodiliste kümnendmurdudega ja minna üle reaalarvude valda.

Kas aga kõigi irratsionaalarvude lisamine on tingimata tarvilik? Osutub, et mitte. Selle projektülesande eesmärk ongi uurida, kui palju tuleb ratsionaalarvude hulka minimaalselt täiendada, et saadavas hulgas esineks ühest küljest $\sqrt{2}$, aga et teisest küljest saaks seal kasutada aritmeetika põhitehteid liitmist, lahutamist, korrutamist ja jagamist.

Lisame hulgale $\mathbb{Q}$ kõigepealt elemendi $\sqrt{2}$, st moodustame hulga $\mathbb{Q} \cup \{\sqrt{2}\}$. Kui me tahame, et oleks olemas kõik selle hulga elementide vahelised korrutised, peame lisama elemendid kujul $b \cdot \sqrt{2}$, kus $b \in \mathbb{Q}$. Kuna ka liitmine peaks alati võimalik olema, tuleb täiendavalt juurde võtta vähemalt elemendid $a + b \cdot \sqrt{2}$, kus $a, b \in \mathbb{Q}$. Selgub, et nendest elementidest piisab.

Definitsioon 1.11. *Tähistame* $\mathbb{Q}[\sqrt{2}] = \{a + b \cdot \sqrt{2} : a, b \in \mathbb{Q}\}$.

Ülesanne 1.11 Tõesta, et hulk $\mathbb{Q}[\sqrt{2}]$ on kinnine liitmise, lahutamise ja korrutamise suhtes, st kui $x, y \in \mathbb{Q}[\sqrt{2}]$, siis ka $x + y, x - y, x \cdot y \in \mathbb{Q}[\sqrt{2}]$.

Ülesanne 1.12 Tõesta, et hulk $\mathbb{Q}[\sqrt{2}]$ on kinnine jagamise suhtes, st kui $x, y \in \mathbb{Q}[\sqrt{2}]$ (ja $y \neq 0$), siis ka $\frac{x}{y} \in \mathbb{Q}[\sqrt{2}]$. Kuna $\frac{x}{y} = x \cdot \frac{1}{y}$ ja korrutamise suhtes kinnisuse näitasime ülesandes 1.11, piisab tõestada, et kui $y \in \mathbb{Q}[\sqrt{2}]$ (ja $y \neq 0$), siis $\frac{1}{y} \in \mathbb{Q}[\sqrt{2}]$.

Ülesanne 1.13 Tõesta, et $\sqrt{3} \notin \mathbb{Q}[\sqrt{2}]$.

Ülesanne 1.14 Tähistagu $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$ vähimat hulka, mis sisaldab kõik ratsionaalarvud, arvud $\sqrt{2}$ ja $\sqrt{3}$ ning on kinnine liitmise, lahutamise ja korrutamise suhtes. Leia hulga $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$ elementide üldkuju.

Ülesanne 1.15* Näita, et arvuhulk $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$ on kinnine ka jagamise suhtes.

Ülesanne 1.16* Kirjuta Pythonis klass `Q2`, mis realiseerib aritmeetilised operatsioonid arvuhulgas $\mathbb{Q}[\sqrt{2}]$.

Arvuhulka, milles saab liita, lahutada, korrutada ja jagada (ning kus kehtivad tavalised assotsiatiivsus-, kommutatiivsus- ja distributiivsusseadused) nimetatakse *korpuseks*. Nii näiteks on korpused ratsionaalarvude hulk $\mathbb{Q}$ ja reaalarvude hulk $\mathbb{R}$. Selles projektülesandes nägime aga, et nende vahele mahub veel (tegelikult isegi lõpmata palju) korpusi

$$\mathbb{Q} \subset \mathbb{Q}[\sqrt{2}] \subset \mathbb{Q}[\sqrt{2}, \sqrt{3}] \subset \mathbb{Q}[\sqrt{2}, \sqrt{3}, \sqrt{5}] \subset \ldots \subset \mathbb{R}\,.$$

1.5 Kompleksarvud

Kas reaalarvude hulk on nüüd piisavalt rikas kõikvõimalike ülesannetega hakkama saamiseks? Ei, osutub, et mitte! Näiteks pole reaalarvudes võimalik lahendada lihtsat võrrandit

$$x^2 = -1\,.$$

Mida siis teha? Selgub, et me võime reaalarvude hulgale $\mathbb{R}$ selle võrrandi lahendi lihtsalt lisada! Tähistagu i niisugust suurust, et $i^2 = -1$; seda suurust hakkame nimetama *imaginaarühikuks*.

Suhtume imaginaarühikusse nagu tavalisse arvu. Muuhulgas võime me teda korrutada suvalise reaalarvuga b ja tulemusena saame uue arvu $b \cdot i$. Sellele arvule võime omakorda liita mingi reaalarvu a ning tulemuseks on $a + b \cdot i$. Niisiis, lisades reaalarvudele uut tüüpi arvu i ning lubades sellega tavalisi tehteid, peame me lisama vähemalt kõik avaldised kujul $a + b \cdot i$.

Osutub, et nendest avaldistest piisab ning saadavas arvuhulgas saab liita, lahutada, korrutada, jagada ja isegi juurida.

Definitsioon 1.12. *Arve kujul*

$$a + b \cdot i \quad (a, b \in \mathbb{R},\ i^2 = -1)$$

nimetatame kompleksarvudeks *ning nende hulka tähistatame* $\mathbb{C}$. *Kompleksarvus* $a+b\cdot i$ *nimetame liidetavat* a reaalosaks *ning liidetavat* $b\cdot i$ imaginaarosaks.

Nagu on kombeks ka reaalarvuliste muutujate korrutamisel, jäetakse korrutamismärk imaginaarosas sageli ära ja kirjutatakse kompleksarve lihtsalt kujul $a + bi$.

Pythonis tähistatakse imaginaarühikut mitte tähega `i`, vaid tähega `j`. Selle kirjaviisi tagamaad ulatuvad inseneriteadustesse, kus täht I on reserveeritud voolutugevuse märkimiseks. Samuti leidis Pythoni autor Guido van Rossum, et `i` võib programmikoodis liiga kergesti teiste sümbolitega sassi minna.

Kompleksarvude liitmine ja lahutamine toimub loomulikul moel, reaal- ja imaginaarosad eraldi:

```
>>> (1+2j)+(2+3j)
(3+5j)
>>> (1+2j)-(2+3j)
(-1-1j)
>>> (2+3j)+(2-3j)
(4+0j)
>>> (2+3j)-(2-3j)
6j
>>> 0 == 0j
True
```

Loomulikult $0 = 0 \cdot i$ ja $4 + 0 \cdot i = 4$; kirjaviisiga `4+0j` hoiab Python lihtsalt meeles arvu tüüpi. Objekti tüüpi saab Pythonis pärida käsuga `type`:

```
>>> type(4)
<class 'int'>
>>> type(4+0j)
<class 'complex'>
```

Kui on vaja kasutada imaginaarühikut kordajaga 1, tuleb see kordaja Pythonis ilmutatult välja kirjutada:

```
>>> 2+j
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'j' is not defined
>>> 2+1j
(2+1j)
```

Proovime nüüd kompleksarvude korrutamist:

```
>>> (1+2j)*(2+3j)
(-4+7j)
```

Oot-oot, mis siin toimub? Ei midagi müstilist, kasutame tavalist kaksliikmete korrutamise reeglit ja võtame arvesse, et $i^2 = -1$:

$$(1 + 2i) \cdot (2 + 3i) = 2 + 3i + 4i + 6i^2 = 2 + 7i - 6 = -4 + 7i\,.$$

Üldjuhul saame kahe kompleksarvu liitmisel ja korrutamisel

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

ja

$$(a + bi) \cdot (c + di) = ac + adi + bci + bdi^2 = (ac - bd) + (ad + bc)i\,.$$

Kuidas on lugu jagamisega? Selgub, et suvalist kahte kompleksarvu (milledest jagaja pole 0) saab omavahel jagada ja tulemuseks on jälle kompleksarv.

Selleks piisab, kui suudame näidata, et suvalise kompleksarvu $a + bi \neq 0$ pöördarv $\frac{1}{a+bi}$ avaldub samuti kompleksarvuna.

Vaatleme arvu $a + bi$ *kaaskomplekarvu* $a - bi$. Osutub, et nende korrutis on reaalarv:

$$(a + bi) \cdot (a - bi) = a^2 - (bi)^2 = a^2 + b^2\,.$$

Nii saame murdu $\frac{1}{a+bi}$ laiendada nimetaja kaaskomplekarvuga ja vabaneda imaginaarsusest nimetajas:

$$\frac{1}{a + bi} = \frac{a - bi}{(a + bi) \cdot (a - bi)} = \frac{a - bi}{a^2 + b^2} = \frac{a}{a^2 + b^2} - \frac{b}{a^2 + b^2}i\,.$$

Seejuures paneme tähele, et kuna $a+bi \neq 0$, peab kehtima ka $a^2+b^2 \neq 0$, niisiis on tulemuseks saadav kompleksarv alati korrektselt määratud. Kokkuvõtteks saame

$$\frac{c + di}{a + bi} = (c + di) \cdot \left(\frac{a}{a^2 + b^2} - \frac{b}{a^2 + b^2}i\right) = \frac{ac + bd}{a^2 + b^2} + \frac{ad - bc}{a^2 + b^2}i\,,$$

mis on kompleksarv. Seda viimast valemit pole vaja pähe tuupida, sest praktikas teeme vastavaid arvutusi nagunii arvutiga.

Ülesanne 1.17 Arvuta

(a) $\frac{2+3i}{1+2j}$,

(b) $\frac{5-10i}{3+4i}$,

(c)

Aga kuidas on astmevõrrandite lahendamisega? Võrrandil $x^2 = -1$ on kompleksarvudes lahend olemas, aga mis saab teistsugustest võrranditest?

Ülesanne 1.18 Tõesta, et võrrandi

$$x^2 = i$$

lahendiks sobib kompleksarv $\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i$. Kas sellel võrrandil leidub veel mõni lahend?

Ülesanne 1.19 Tõesta, et võrrandi

$$x^3 = i$$

lahendiks sobib kompleksarv $\frac{\sqrt{3}}{2} + \frac{1}{2}i$.

Ülesanne 1.20 Leia võrrandile

$$x^4 = 1$$

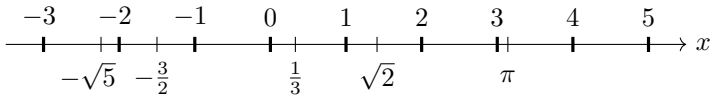
neli lahendit kompleksarvudes.

Ülesannetes 1.18–1.20 tõime küll välja mõned lahendid, aga ei tõestanud rangelt, et rohkem lahendeid pole. Üldise meetodi nende võrrandite täieliku lahendamise jaoks anname kursuses „Trigonomeetria".

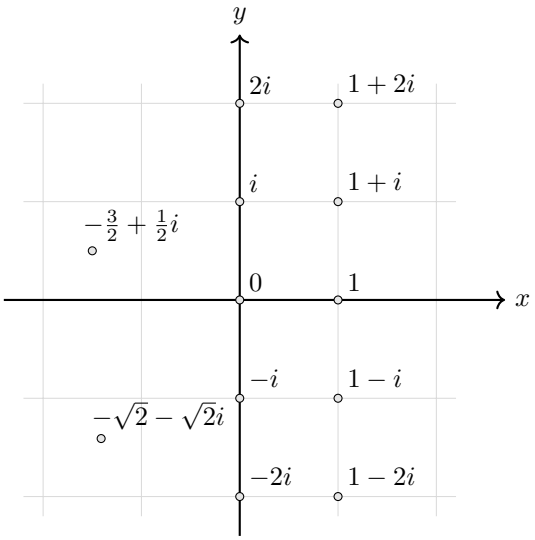
Üldiselt on võimalik tõestada, et kõiki polünoomvõrrandeid saab kompleksarvudes lahendada, seejuures on n. astme võrrandil n lahendit. Niisiis oleme arvuhulkade laiendamisega mingis mõttes lõpuni jõudnud.

1.6 Reaalarvude ja kompleksarvude geomeetriline tõlgendus

Reaalarvud saab seada üksühesesse vastavusse sirge punktidega. Sisuliselt viime me sirgel sisse ühemõõtmelise koordinaatsüsteemi, kus iga punkti saab määrata talle vastava arvu (koordinaadi) abil. Kuna reaalarvud on loomulikul moel suuruse järgi järjestatud, annab see sirgele soovi korral ka suuna.



Kompleksarvu $a+bi$ määrab reealarvupaar $(a; b)$. Seda paari saame vaadelda kui punkti koordinaate tasandil. Niisiis saab kompleksarvud seada üksühesesse vastavusse tasandi punktidega!



Reaalarvudele vastavad punktid asuvad parajasti selle tasandi x-teljel.

Ülesanne 1.21 Kirjuta programm, mis saab sisendiks kompleksarvude loendi ning kuvab igale arvule vastava punkti koordinaattasandil.

Reaalarvu absoluutväärtusest võib mõelda kui selle arvu kaugusest reaaltelje nullpunktist. Seda ideed saab üldistada ka kompleksarvudele.

Definitsioon 1.13. *Kompleksarvule z vastava punkti kaugust koordinaattelgede nullpunktist nimetatakse selle kompleksarvu* mooduliks *ning tähistatakse* $|z|$.

Pythagorase teoreemi põhjal on selge, et kompleksarvu $z = a + b \cdot i$ moodul avaldub valemiga

$$|z| = \sqrt{a^2 + b^2}\,.$$

1.6.1 Projektülesanne: Mandelbroti hulk

Olgu antud kompleksarv c. Moodustame temast lähtudes jada

$$\begin{aligned} z_0 &= 0\,, \\ z_1 &= z_0^2 + c\,, \\ z_2 &= z_1^2 + c\,, \\ &\dots \\ z_n &= z_{n-1}^2 + c\,, \\ &\dots \end{aligned}$$

Näiteks kui $c = i$, siis

$$\begin{aligned}
z_0 &= 0\,,\\
z_1 &= 0^2 + i = i\,,\\
z_2 &= i^2 + i = -1 + i\,,\\
z_3 &= (-1+i)^2 + i = 1 - 2i - 1 + i = -i\,,\\
z_4 &= (-i)^2 + i = -1 + i\,,\\
z_5 &= (-1+i)^2 + i = 1 - 2i - 1 + i = -i\,,\\
&\dots
\end{aligned}$$

Näeme, et selles jadas jäävad korduma elemendid $-1+i$ ja $-i$. Muuhulgas jääb kogu jada mooduli suuruse mõttes nullpunktile üsna lähedale.

Osutub, et see pole nii sugugi iga algse kompleksarvu korral.

Ülesanne 1.22 Kirjuta programm, mis lähtudes antud kompleksarvust c arvutab välja jada z_n esimesed mõnikümmend elementi koos nende moodulitega. (Kompleksarvu moodulit saab leida käsuga `abs`.) Kas moodulite jada on tõkestatud või kasvab tõkestamatult, kui

(a) $c = 1 + i$,

(b) $c = 0{,}6i$,

(c) $c = 0{,}7i$,

(d) $c = -0{,}04 + 0{,}64i$,

(e) $c = -0{,}04 + 0{,}66i$,

(f) $c = -0{,}04 + 0{,}68i$?

Ülesanne 1.23 Kirjuta programm, mis vaatab läbi hulga komplekstasandi punkte c ning joonistab täpi iga punkti jaoks, mille korral on vastav jada z_n tõkestatud. Saab tõestada, et tõkestatud jada moodustavate punktide c korral peab kehtima $|c| \leqslant 2$, niisiis piisab vaadelda punkte $c = a + b \cdot i$, kus $a, b \in [-2; 2]$.

Jada tõkestatust pole rangelt võttes väga lihtne tõestada, sest jada võib piiridest väljuda alles väga suurte indeksite korral. Kasutame tõkestatutse tõestuse asemel heuristilist lähenemist – arvutame jada algusest välja näiteks 100 elementi ja kui nende kõigi moodul jääb alla 2, siis loeme, et jada on tõkestatud.

Pildile tekkivat punktihulka nimetatakse Mandelbroti[a] hulgaks.

[a] Benoit B. Mandelbrot (1924 – 2010) oli Poola päritolu Prantsuse-Ameerika matemaatik.

1.7* Positsioonilised arvusüsteemid. Kahendsüsteem

Kuigi naturaalarvud loendavad *lõplike* hulkade elementide arve, on neid endid *lõpmatult* palju. Siit tekib praktiline probleem, et naturaalarvude tähistamiseks oleks vaja ka lõpmatut kogust sümboleid.

Seda probleemi saab lahendada *positsioonilise arvusüsteemi* sisseviimisega. Valime süsteemi aluseks mingi arvu a (näiteks inimese kahe käe sõrmede koguse) ning valitud arvu jagu numbrisümboleid, mis vastavad esimestele naturaalarvudele (näiteks 0,1,2,3,4,5,6,7,8,9).

Suuremate naturaalarvude väljendamiseks kirjutame numbreid kõrvuti, kusjuures kõige parempoolse numbri korrutame arvuga $a^0 = 1$, paremalt teise numbri arvuga $a^1 = a$, paremalt kolmanda numbri arvuga a^2 jne ning liidame saadud tulemused. Arvusüsteemi alus ise vastab siis numbrikombinatsioonile $10 = 1 \cdot a + 0$.

Positsioonilist arvusüsteemi, kus alus a on valitud inimese sõrmede koguse järgi, nimetatakse *kümnendsüsteemiks*. Nii näiteks saame kümnendsüsteemis esitada

$$2024 = \mathbf{2} \cdot 10^3 + \mathbf{0} \cdot 10^2 + \mathbf{2} \cdot 10^1 + \mathbf{4} \cdot 10^0 \,.$$

Inimese sõrmede arvus pole matemaatika seisukohast midagi erilist. Täpselt sama moodi võime arvusüsteemi aluseks valida mõne teise (1-st suurema) naturaalarvu, näiteks 2,3 või 16. Arvusüsteemi alusega 2 nimetatakse *kahendsüsteemiks* ning selles on kasutusel ainult kaks numbrisümbolit: 0 ja 1. Kuna arvutiriistvaras on mugav eristada signaali kahte taset (signaal on / signaali ei ole), seataksegi neile vastavusse numbrid 1 ja 0 ning esitatakse kõik väärtused riistvara tasemel kahendsüsteemis.

Inimesed on rohkem harjunud kümnendsüsteemiga, mistõttu peab arvuti inimeselt saadud kümnendsüsteemi arvud kõigepealt kahendsüsteemi teisendama ning pärast arvutuse lõppu tulemuse jälle kümnendkujule tagasi viima.

Koostame kõigepealt algoritmi kahendarvu kümnendkujule teisendamiseks. Tuletame meelde, et kahendarvu numbrite positsioonid vastavad arvu 2 astmetele, nii näiteks

$$\begin{aligned}
10_2 &= \mathbf{1} \cdot 2^1 + \mathbf{0} \cdot 2^0 = 2 + 0 = 2_{10} \,, \\
11_2 &= \mathbf{1} \cdot 2^1 + \mathbf{1} \cdot 2^0 = 2 + 1 = 3_{10} \,, \\
100_2 &= \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{0} \cdot 2^0 = 4 + 0 + 0 = 4_{10} \,, \\
101_2 &= \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0 = 4 + 0 + 1 = 5_{10} \,, \\
10101_2 &= \mathbf{1} \cdot 2^4 + \mathbf{0} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0 = 16 + 4 + 0 + 1 = 21_{10} \,.
\end{aligned}$$

Siin oleme vastava alaindeksiga eristanud kahend- ja kümnendarve (mis on segaduse vältimiseks vajalik, sest 10,11,100,101 ja 10101 võivad olla ka kümnendarvud).

Ülesanne 1.24 Esita kümnendkujul kahendarvud

(a) 111_2

(b) 1000_2

(c) $1\underbrace{00\ldots0}_{n}{}_2$

(d) $\underbrace{11\ldots1}_{n}{}_2$

(e) 11111101000_2

(f)

Hakkame nüüd koostama funktsiooni, mis teisendab kahendarve kümnendarvudeks. Eeldame, et funktsioonile antakse sisendiks kahendarv numbrite loendina, kusjuures nii, et loendi element indeksiga 0 vastab järgule 2^0, element indeksiga 1 vastab järgule 2^1 jne. Võrreldes tavalise kirjaviisiga, kus arvujärke kirjutatakse paremalt vasakule, aga loendeid vasakult paremale, on funktsiooni sisend siis tagurpidi pööratud, nt arvu 11001_2 esitame loendiga `[1,0,0,1,1]`.

See kokkulepe võimaldab teisendusfunktsiooni lihtsustada, sest kahendnumbri indeks i vastab otse järgule 2^i. Jääb veel lahendada küsimus, kuidas järkude väärtusi arvutada. Üks võimalus on kasutada Pythoni sisseehitatud funktsiooni `**`, mis leiab arvude astmeid (st `a**b` väljastab a^b). Kokkuvõttes võib esimene versioon kahendarve kümnendarvudeks teisendavast funktsioonist välja näha selline:

```
def bin2dec1(numbrid):
    tulem = 0
    for i in range(len(numbrid)):
        num = numbrid[i]
        tulem += num * 2**i
    return(tulem)
```

See funktsioon töötab ja annab õige tulemuse (kontrolli järele!), aga teda saab natuke optimeerida. Peamine probleem seisneb selles, et me arvutame 2^0,2^1,2^2,2^3 ... iga kord uuesti, aga seda pole tegelikult vaja. Selle asemel võime järgmise järgu leida eelmist lihtsalt 2-ga korrutades. Tulemuseks on efektiivsem versioon teisendusfunktsioonist:

```
def bin2dec2(numbrid):
    tulem = 0
    jark = 1
    for num in numbrid:
        tulem += num*jark
        jark *= 2
    return(tulem)
```

Ülesanne 1.25 Milline õieti on naturaalarvude a ja b korral a^b arvutamise keerukus? Definitsiooni $a^b = \underbrace{a \cdot a \cdot \ldots \cdot a}_{b}$ järgi arvutades oleks vaja teha $b-1$ korrutamist, aga kas see on alati kõige efektiivsem? Milline on kõige väiksem arv korrutamisi, millega saab arvutada näiteks a^{16}?

Kuidas teisendada kümnendarve kahendarvudeks?

1.8 Lahendused

1.1 Viga ei tule, aga saadav hulk pole mitte kolme-, vaid üheelemendiline.

```
>>> len(S)
1
>>> S
{("Kirss", 'Õun', 'Pirn')}
```

Asi seisneb selles, et ennik `(1, 'Õun', 'Pirn')` on ise üks objekt, mitte kolm objekti.

1.2 Üks võimalus on moodustada antud loendi baasil hulk. Hulka moodustades võetakse iga elementi arvesse ainult üks kord, niisiis võime väljastada selle hulga elementide arvu:

```
def unikaalne(loend):
    S = set(loend)
    return(len(S))
```

või lühemalt kirja panduna

```
def unikaalne(loend):
    return(len(set(loend)))
```

1.3 Kui $A \subset B$ ja $B \subset A$, siis järelikult $A = B$.

1.4 Jah, tühja hulka loetakse iga hulga (sealhulgas tema enda!) alamhulgaks.

1.5 Kui $A \cap B = A$, siis on hulga A elemendid ühtlasi ka hulkade A ja B ühised elemendid. See omakorda tähendab, et hulga A iga element kuulub hulka B, st $A \subset B$.

Kui aga $A \cup B = A$, siis pole hulgas B ühtegi elementi, mis ei kuuluks hulka A. Niisiis $B \subset A$.

1.6 Täisarvu a saame esitada kujul $\frac{a}{1}$.

1.7 Laseme arvutil arvutada:

```
>>> 1/7
0.14285714285714285
>>> 2/7
0.2857142857142857
>>> 3/7
0.42857142857142855
>>> 4/7
0.5714285714285714
>>> 5/7
0.7142857142857143
>>> 6/7
0.8571428571428571
```

Näeme, et kõigi nende kümnendmurdude tsüklid koosnevad samadest numbritest 142857 tsükliliselt ümber järjestatuna!

1.8 (a) $\frac{14}{9}$; (b) $\frac{1}{37}$; (c) $\frac{3}{13}$.

Murdude taandamiseks saab kasutada mooduli `fractions` konstruktorit `Fraction`, mis esitab murru taandatud kujul. Näiteks ülesande (c)-osa la-

hendamisel võime arutleda järgmiselt. Olgu $x = 0,(230769)$, siis

$$\begin{aligned} 1000000x &= 230769,(230769)\,, \\ 1000000x - x &= 230769\,, \\ 999999x &= 230769\,, \\ x &= \frac{230769}{999999} \end{aligned}$$

ja nüüd leiame Pythoni abil

```
from fractions import Fraction
>>> Fraction(230769,999999)
Fraction(3, 13)
```

seega $\frac{230769}{999999} = \frac{3}{13}$. Teine võimalus on leida arvude 230769 ja 999999 suurim ühistegur ning nad mõlemad sellega läbi jagada. Suurima ühisteguri saame leida mooduli `math` funktsiooniga `gcd`.[2]

```
>>> from math import gcd
>>> d = gcd(230769,999999)
>>> d
76923
>>> 230769//d
3
>>> 999999//d
13
```

1.9 $\sqrt{3}$ ei ole ratsionaalarv ja selle väite tõestus on analoogiline teoreemi 1.2 tõestusega. $\sqrt{4} = 2$ on ratsionaalarv, isegi täisarv. Üldiselt kehtib väide, et täisarvu ruutjuur on kas täisarv või irratsionaalarv.

1.10 (a) Vaatame läbi neli võimalikku juhtu.

Kui $x, y \geqslant 0$, siis ka $x \cdot y \geqslant 0$. Järelikult

$$|x \cdot y| = x \cdot y = |x| \cdot |y|\,.$$

Kui $x, y \leqslant 0$, siis $x \cdot y \geqslant 0$. Järelikult

$$|x \cdot y| = x \cdot y = (-x) \cdot (-y) = |x| \cdot |y|\,.$$

Kui $x \geqslant 0$ ja $y \leqslant 0$, siis $x \cdot y \leqslant 0$. Järelikult

$$|x \cdot y| = -(x \cdot y) = x \cdot (-y) = |x| \cdot |y|\,.$$

Sama moodi arutleme ka juhul, kui $x \leqslant 0$ ja $y \geqslant 0$.

(b) Vaatame jälle läbi neli võimalikku juhtu.

Kui $x, y \geqslant 0$, siis ka $x + y \geqslant 0$. Järelikult

$$|x + y| = x + y = |x| + |y|\,.$$

[2] Inglise keeles tähendab *greatest common divisor* suurimat ühsitegurit.

Kui $x, y \leqslant 0$, siis $x + y \leqslant 0$. Järelikult

$$|x + y| = -(x + y) = (-x) + (-y) = |x| + |y| .$$

Niisiis kehtib neil juhtudel vajalik võrratus võrdusena.

Juhul $x \geqslant 0$ ja $y < 0$ on omakorda kaks võimalust: kas $x + y \geqslant 0$ või $x + y < 0$.

Kui $x + y \geqslant 0$, siis

$$|x + y| = x + y < x = |x| < |x| + |y| ,$$

sest $y < 0$.

Kui aga $x + y < 0$, siis

$$|x + y| = -(x + y) = (-x) + (-y) = (-x) + |y| \leqslant |y| \leqslant |x| + |y| ,$$

sest $-x \leqslant 0$.

Analoogiliselt arutleme ka juhul $x < 0$ ja $y \geqslant 0$.

1.11 Olgu $x = a + b\sqrt{2}$ ja $y = c + d\sqrt{2}$, kus $a, b, c, d \in \mathbb{Q}$. Siis

$$\begin{aligned}
x + y &= (a + b\sqrt{2}) + (c + d\sqrt{2}) = (a + c) + (b + d)\sqrt{2} \in \mathbb{Q}[\sqrt{2}] , \\
x - y &= (a + b\sqrt{2}) - (c + d\sqrt{2}) = (a - c) + (b - d)\sqrt{2} \in \mathbb{Q}[\sqrt{2}] , \\
x \cdot y &= (a + b\sqrt{2}) \cdot (c + d\sqrt{2}) = ac + ad\sqrt{2} + bc\sqrt{2} + bd(\sqrt{2})^2 = \\
&= (ac + 2bd) + (ad + bc)\sqrt{2} \in \mathbb{Q}[\sqrt{2}] .
\end{aligned}$$

1.12 Olgu $y = c + d\sqrt{2}$, kus $c, d \in \mathbb{Q}$. Siis saame murru $\frac{1}{y}$ lugejat ja nimetajat laiendada arvuga $c - d\sqrt{2}$ (paneme tähele, et tänu $\sqrt{2}$ irratsionaalsusele $c - d\sqrt{2} \neq 0$). Lõpetuseks kasutame ruutude vahe valemit ja teisendame:

$$\begin{aligned}
\frac{1}{y} &= \frac{1}{c + d\sqrt{2}} = \frac{c - d\sqrt{2}}{(c + d\sqrt{2})(c - d\sqrt{2})} = \frac{c - d\sqrt{2}}{c^2 - (d\sqrt{2})^2} = \frac{c - d\sqrt{2}}{c^2 - 2d^2} = \\
&= \frac{c}{c^2 - 2d^2} - \frac{d}{c^2 - 2d^2}\sqrt{2} \in \mathbb{Q}[\sqrt{2}] .
\end{aligned}$$

1.13 Kui $\sqrt{3}$ kuuluks hulka $\mathbb{Q}[\sqrt{2}]$, siis peaksid leiduma niisugused ratsionaalarvud a ja b, et $\sqrt{3} = a + b\sqrt{2}$. Järelikult peaksid kehtima ka võrdused

$$\begin{aligned}
(\sqrt{3})^2 &= (a + b\sqrt{2})^2 , \\
3 &= a^2 + 2ab\sqrt{2} + (b\sqrt{2})^2 , \\
3 &= a^2 + 2ab\sqrt{2} + 2b^2 , \\
3 - a^2 - 2b^2 &= 2ab\sqrt{2} .
\end{aligned}$$

Paneme tähele, et $b \neq 0$, sest muidu peaks kehtima $\sqrt{3} = a + b\sqrt{2} = a \in \mathbb{Q}$, mis pole võimalik, sest $\sqrt{3}$ on irratsionaalarv.

Samuti näitame, et $a \neq 0$. Kui kehtiks $a = 0$, järelduks viimati tõestatud võrdusest, et $3 = 2b^2$ ehk $b^2 = \frac{3}{2}$. Edasi saame kasutada teoreemi 1.2 tõestuse sarnast arutelu. Kuna b on ratsionaalarv, peab ta avalduma mingite ühistegurita täisarvude m ja n jagatisena, st

$$\begin{aligned} \left(\frac{m}{n}\right)^2 &= \frac{3}{2}\,, \\ \frac{m^2}{n^2} &= \frac{3}{2}\,, \\ 2m^2 &= 3n^2\,. \end{aligned}$$

Viimasest võrdusest järeldub, et n peab olema paarisarv. Muuhulgas tähendab see, et n^2 jagub 4-ga. Niisiis jaguvad võrduse mõlemad pooled 4-ga, millest järeldub omakorda, et ka m peab olema paaris. Oleme saanud vastuolu eeldusega, et m ja n on ühistegurita.

Kokkuvõttes näitasime, et $a, b \neq 0$, niisiis võime võrduse $3 - a^2 - 2b^2 = 2ab\sqrt{2}$ mõlemaid pooli jagada arvuga $2ab$. Järelikult

$$\sqrt{2} = \frac{3 - a^2 - 2b^2}{2ab} \in \mathbb{Q}\,,$$

mis aga ei saa kehtida, sest $\sqrt{2}$ on irratsionaalarv. Saime vastuolu eeldusega $\sqrt{3} \in \mathbb{Q}[\sqrt{2}]$.

1.14 $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}$, kus $a, b, c, d \in \mathbb{Q}$.

1.15 Olgu $y = a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6} \in \mathbb{Q}[\sqrt{2}, \sqrt{3}]$, kus $a, b, c, d \in \mathbb{Q}$. Sama moodi nagu ülesandes 1.12 piisab tõestada, et $\frac{1}{y} \in \mathbb{Q}[\sqrt{2}, \sqrt{3}]$. Selleks peame näitama, et arv $\frac{1}{y}$ esitub hulga $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$ elemendi üldkujul. Kasutame jälle laiendamisvõtet sobiva laiendajaga, mis võimaldab nimetajas vabaneda tegurist $\sqrt{3}$:

$$\begin{aligned} \frac{1}{y} &= \frac{1}{a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}} = \frac{1}{a + b\sqrt{2} + \sqrt{3}(c + d\sqrt{2})} = \\ &= \frac{a + b\sqrt{2} - \sqrt{3}(c + d\sqrt{2})}{(a + b\sqrt{2} + \sqrt{3}(c + d\sqrt{2}))(a + b\sqrt{2} - \sqrt{3}(c + d\sqrt{2}))} = \\ &= \frac{a + b\sqrt{2} - \sqrt{3}(c + d\sqrt{2})}{(a + b\sqrt{2})^2 + 3(c + d\sqrt{2})^2} = \\ &= \frac{a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6}}{a^2 + 2b^2 + 3c^2 + 6d^2 + (2ab + 6cd)\sqrt{2}}\,. \end{aligned}$$

Saadud murru lugeja kuulub hulka $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$, nimetaja aga hulka $\mathbb{Q}[\sqrt{2}]$. Ülesandes 1.12 näitasime, et nimetaja pöördväärtus peab seega esituma kujul $e + f\sqrt{2}$, kus $e, f \in \mathbb{Q}$. Järelikult

$$\frac{1}{y} = (a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6}) \cdot (e + f\sqrt{2})\,,$$

millest lahti korrutades saame hulga $\mathbb{Q}[\sqrt{2}, \sqrt{3}]$ elemendi üldkujul esituva väärtuse.

1.16 Aritmeetilise klassi realiseerimiseks tuleb defineerida meetodid `__add__`, `__sub__`, `__mul__` ja `__truediv__`. Selleks, et klassi elemente ilusti välja trükkida, kulub ära ka meetod `__str__`. Meetodite realiseerimisel tuginеme ülesannetele 1.11 ja 1.12.

```
from fractions import Fraction

class Q2:
    def __init__(self, a, b):
        self.a = Fraction(a)
        self.b = Fraction(b)

    def __str__(self):
        return f"{self.a} + {self.b}*√2"

    def __add__(self, other):
        return Q2(self.a + other.a, self.b + other.b)

    def __sub__(self, other):
        return Q2(self.a - other.a, self.b - other.b)

    def __mul__(self, other):
        a, b = self.a, self.b
        c, d = other.a, other.b
        return Q2(a*c + 2*b*d, a*d + b*c)

    def __truediv__(self, other):
        c, d = other.a, other.b
        z = Q2(c/(c*c-2*d*d),-d/(c*c-2*d*d))  # z=1/y
        return self*z
```

Koodi tööd saame katsetada näiteks nii:

```
x = Q2(3, 4)     # x = 3 + 4*√2
y = Q2(1, 2)     # y = 1 + 2*√2

print(x+y)
print(x-y)
print(x*y)
print(x/y)
print((x/y)*y)
```

ja väljund tuleb

```
4 + 6*√2
2 + 2*√2
19 + 10*√2
13/7 + 2/7*√2
3 + 4*√2
```

1.17 Kasutame Pythonit:

```
>>> (2+3j)/(1+2j)
(1.6-0.2j)
```

```
>>> (5-10j)/(3+4j)
(-1-2j)
```

1.18 Arvutame x^2 väärtuse $x = \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i$ korral välja:

$$\left(\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i\right)^2 = \frac{2}{4} + 2 \cdot \frac{2}{4}i + \frac{2}{4}i^2 = \frac{1}{2} + i - \frac{1}{2} = i\,.$$

Leidub veel teinegi kompleksarv, mille ruut on i, nimelt ülesandes antud arvu vastandarv $-\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i$.

1.19 Arvutame x^3 väärtuse $x = \frac{\sqrt{3}}{2} + \frac{1}{2}i$ korral välja:

$$\begin{aligned}
\left(\frac{\sqrt{3}}{2} + \frac{1}{2}i\right)^3 &= \left(\frac{\sqrt{3}}{2} + \frac{1}{2}i\right)^2 \cdot \left(\frac{\sqrt{3}}{2} + \frac{1}{2}i\right) = \\
&= \left(\frac{3}{4} + 2 \cdot \frac{\sqrt{3}}{2} \cdot \frac{1}{2}i + \frac{1}{4}i^2\right) \cdot \left(\frac{\sqrt{3}}{2} + \frac{1}{2}i\right) = \\
&= \left(\frac{3}{4} + \frac{\sqrt{3}}{2}i - \frac{1}{4}\right) \cdot \left(\frac{\sqrt{3}}{2} + \frac{1}{2}i\right) = \\
&= \left(\frac{1}{2} + \frac{\sqrt{3}}{2}i\right) \cdot \left(\frac{\sqrt{3}}{2} + \frac{1}{2}i\right) = \\
&= \frac{1}{2} \cdot \frac{\sqrt{3}}{2} + \frac{1}{2} \cdot \frac{1}{2}i + \frac{\sqrt{3}}{2} \cdot \frac{\sqrt{3}}{2}i + \frac{1}{2} \cdot \frac{\sqrt{3}}{2}i^2 = \\
&= \frac{\sqrt{3}}{4} + \frac{1}{4}i + \frac{3}{4}i - \frac{\sqrt{3}}{4} = i\,.
\end{aligned}$$

1.20 Võrrandi lahenditeks sobivad $1, -1, i, -i$.

1.21 Pythonil on olemas palju graafikateeke, mille abil koordinaattasandit ja sellele punkte joonistada saab. Lahendaja võib omale internetiotsingu abil meelepärase välja valida. Õpikus kasutame siinkohal teegi `matplotlib` alamteeki `pyplot`, mille laadime käsuga

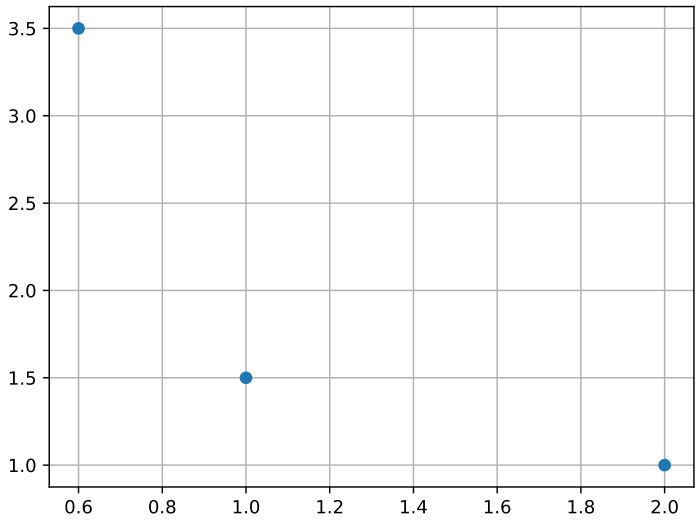
```
import matplotlib.pyplot as plt
```

Edaspidi saame alamteegi `pyplot` käskudele viidata lühinime `plt` järgi. Kui kasutad Thonnyt ja Thonny kurdab, et teeki `matplotlib` pole, tuleb see kõigepealt installeerida menüüvalikust „Tools → Manage packages”.

Joonistame kõigepealt mõned punktid; olgu nende koordinaadid näiteks $(1;1{,}5)$, $(0{,}6;3{,}5)$ ja $(2;1)$. Mooduli `pyplot` käsk `plot` võtab sisendiks kaks loendit: esimeses on kuvatavate punktide x-koordinaadid ja teises y-koordinaadid. Kolmandaks argumendiks saame anda joonise kirjelduse; näiteks `"o"` tähendab, et punkte kujutatakse ringikeste abil. Koordinaattelgedele vastavat ruudustikku näitab käsk `plt.grid(True)`. Lõpuks kasutame käsku `plt.show()`, et joonis ekraanile kuvada. Kokkuvõtteks on lihtne punkte joonistav programm selline:

```
import matplotlib.pyplot as plt

plt.plot([1,0.6,2], [1.5,3.5,1], "o")
plt.grid(True)
plt.show()
```

ja tulemuseks saame niisuguse pildi:



Olgu programmile ette antud väärtused kogutud loendisse `kompleksarvud`. Me peame loendi elementidest eraldama reaal- ja imaginaarosad; seda saab teha meetoditega `real` ja `imag`. Jääb veel reaal- ja imaginaarosad `for`-tsükli abil eraldi loenditesse koguda ning ülejäänud lahendus on samasugune nagu eelpool kirja pandud.

```
import matplotlib.pyplot as plt

kompleksarvud = [1+2j,0,3-1j,1j] # Muuda neid arve

reaalosad = []
imaginaarosad = []

for kompleksarv in kompleksarvud:
    reaalosad.append(kompleksarv.real)
    imaginaarosad.append(kompleksarv.imag)

plt.plot(reaalosad, imaginaarosad, "o")
plt.grid(True)
plt.show()
```

1.22 (a) tõkestamata; (b) tõkestatud; (c) tõkestamata; (d) tõkestatud; (e) tõkestamata; (f) tõkestatud.

Kood võib välja näha näiteks selline:

```
c = 1+1j       # Muuda seda arvu

z = 0
for n in range(30):
    print(abs(z))
    z = z**2 + c
```

Paneme tähele, et kui arvu z moodul läheb üle 2, hakkab jada väga kiiresti kasvama. (Kui Python trükib väärtuseks `nan`, siis see tähendab „not a number", st arv läks liiga suureks, et seda ilma erivõteteta esitada.)

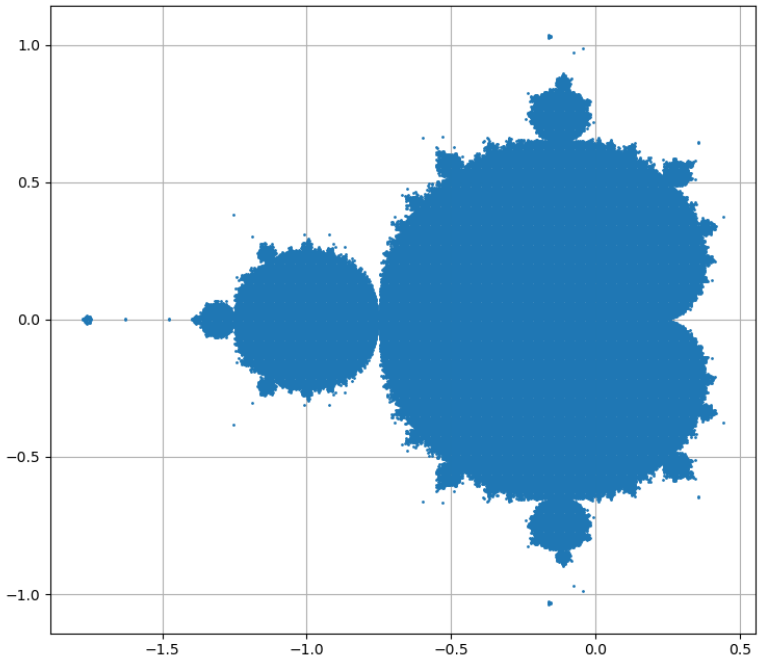
Samuti näeme, alamülesannetest (d)–(f), et jada tõkestatuse käitumine on mingis mõttes üsna kaootiline ja võib hüppeliselt edasi-tagasi muutuda sisendi väga väikese muutuse korral.

1.23 Kasutame joonise tegemisel jälle `matplotlib.pyplot` teeki. Punktide c koordinaatide genereerimiseks võime lõigu $[-2; 2]$ jagada m võrdseks osaks. Üks võimalus selleks on paketi `numpy` meetod `linspace`, millele anname ette lõigu otspunktid ning soovitavate jaotuspunktide arvu. Defineerime funktsioon `tokestatud`, mis väljastab `False`, kui etteantud iteratsioonide arvu (nt 100) jooksul läheb $|z|$ suuremaks kui 2, ning `True` vastasel juhul. Seda funktsioni rahuldavate punktide korral jätame meelde nende x- ja y-koordinaadid ning joonistame lõpuks iga punkti kohta täpi. Tulemuseks võib olla näiteks selline kood:

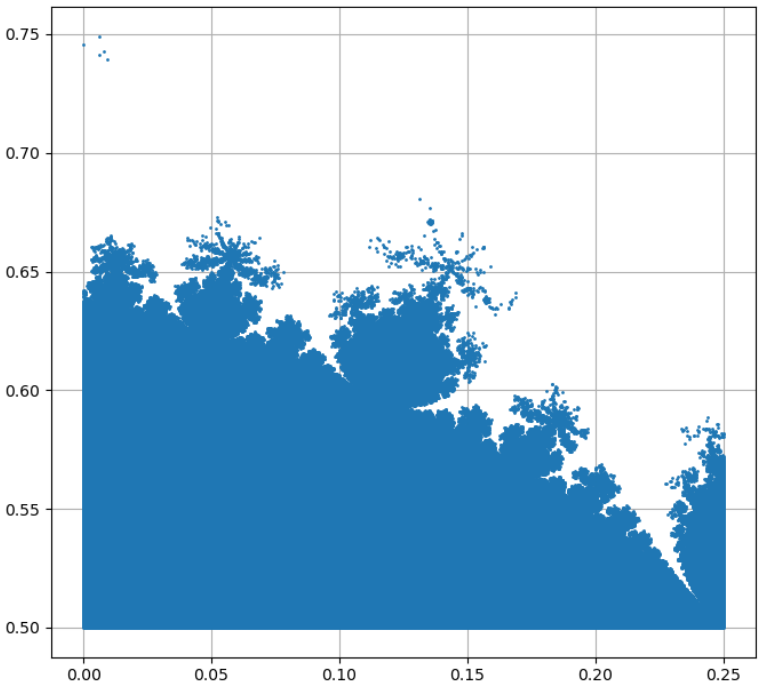
```
import matplotlib.pyplot as plt
import numpy as np

xmin = -2
xmax = 2
ymin = -2
ymax = 2

nmax = 100     # Jada elementide max arv
m = 1000       # Punktide arv lõigul

def tokestatud(c):
    z = 0
    for n in range(nmax):
        z = z**2+c
        if abs(z) > 2:
            return(False)
    return(True)

A = np.linspace(xmin,xmax,m)
B = np.linspace(ymin,ymax,m)

X = []
Y = []

for a in A:
    for b in B:
```

```
        if tokestatud(a+b*(1j)):
            X.append(a)
            Y.append(b)

plt.plot(X,Y,"o", markersize=1)
plt.grid(True)
plt.show()
```

Tulemusena näeme sellist pilti:



Uuri seda kujundit lähemalt, muutes „akent” väiksemaks. Näiteks kui $xmin = 0; xmax = 0{,}25; ymin = 0{,}5; ymax = 0{,}75$, saame niisuguse pildi:



Mandelbroti hulk on näide enesesarnasest fraktaalstruktuurist.

1.24

1.25 Arvu a^{16} saab arvutada nelja korrutamisega: kõigepealt leiame $a^2 = a \cdot a$, siis $a^4 = a^2 \cdot a^2$, $a^8 = a^4 \cdot a^4$ ja lõpuks $a^{16} = a^8 \cdot a^8$. Võrreldes 15 korrutamisega on see märkimisväärne ajakokkuhoid!

PEATÜKK 2

Astmed ja juured

2.1 Astme mõiste üldistamine

Tuletame kõigepealt meelde astme definitsiooni positiivse täisarvulise astendaja korral.

Definitsioon 2.1. *Kui n on positiivne täisarv ja a suvaline reaalarv, tähistab a^n arvu a n-kordset korrutist iseendaga, st*

$$a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n} .$$

Sellest definitsioonist saame tuletada astendamise põhireeglid positiivsete täisarvude m ja n jaoks:

$$a^{m+n} = \underbrace{a \cdot a \cdot \ldots \cdot a}_{m+n} = \underbrace{a \cdot a \cdot \ldots \cdot a}_{m} \cdot \underbrace{a \cdot a \cdot \ldots \cdot a}_{n} = a^m \cdot a^n ,$$

$$a^{m \cdot n} = \underbrace{a \cdot a \cdot \ldots \cdot a}_{m \cdot n} = \underbrace{\underbrace{a \cdot a \cdot \ldots \cdot a}_{m} \cdot \ldots \cdot \underbrace{a \cdot a \cdot \ldots \cdot a}_{m}}_{n} =$$

$$= \underbrace{a^m \cdot \ldots \cdot a^m}_{n} = (a^m)^n .$$

Osutub, et astme mõistet saab üldistada nii, et võrdused $a^{m+n} = a^m \cdot a^n$ ja $a^{m \cdot n} = (a^m)^n$ jäävad kehtima ka juhul, kui m ja n pole positiivsed täisarvud.

Olgu kõigepealt $m = 0$. Kuidas tuleks defineerida a^0, et võrdus $a^{m+n} = a^m \cdot a^n$ endiselt kehtiks? Seda võrdust eeldades saame

$$a^n = a^{0+n} = a^0 \cdot a^n .$$

Eeldusel $a \neq 0$ saame viimase võrduse mõlemad pooled a^n-ga läbi jagada, mis annab

$$a^0 = 1 \quad (a \neq 0)\,.$$

Järgmiseks uurime, mis juhtub negatiivse täisarvulise astendaja korral. Püüame a^{-n} defineerida nii, et võrdus $a^{m+n} = a^m \cdot a^n$ endiselt kehtima jääks. Võtame selles võrduses $m = -n$ ja saame

$$1 = a^0 = a^{(-n)+n} = a^{-n} \cdot a^n\,.$$

Eeldusel $a \neq 0$ saame võrduse mõlemad pooled a^n-ga läbi jagada, mis annab

$$a^{-n} = \frac{1}{a^n} \quad (a \neq 0)\,.$$

Uurime järgmiseks võrdust $a^{m \cdot n} = (a^m)^n$ ning küsime, mis juhtub, kui $m = \frac{1}{n}$? Kuna $a^1 = a$, saame

$$a = a^1 = a^{\frac{1}{n} \cdot n} = (a^{\frac{1}{n}})^n\,.$$

Näeme, et $a^{\frac{1}{n}}$ on arv, mida n. astmele tõstes saame arvu a. Juhul $n = 2$ on seega näiteks tegu meile tuttava *ruutjuure* mõistega, niisiis võime kirjutada

$$a^{\frac{1}{2}} = \sqrt{a}\,.$$

Ruutjuute arvutamiseks saame Pythonis kasutada käsku `sqrt` moodulist `math` või astendamist astmesse $\frac{1}{2}$:

```
>>> from math import sqrt
>>> sqrt(3)
1.7320508075688772
>>> 3**(1/2)
1.7320508075688772
```

Kontrollime tulemust ruutu tõstes:

```
>>> 1.7320508075688772**2
2.9999999999999996
```

Tuletame meelde, et reaalarvude esitus Pythonis on ligikaudne ja lisaks on $\sqrt{3}$ irratsionaalne (vt ülesannet 1.9). Niisiis polegi õige oodata, et $(\sqrt{3})^2$ tuleks täpselt 3. Küll aga saime väärtuse, mis on 3-le väga lähedal.

Mis juhtub siis, kui püüda leida ruutjuurt negatiivsest arvust? Funktsioon `sqrt` (ingl. k. *square root*) moodulist `math` annab veateate, sest ta otsib reaalarvulisi juuri:

```
>>> from math import sqrt
>>> sqrt(-4)
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
ValueError: math domain error
```

Astendamine astendajaga $\frac{1}{2}$ leiab aga kompleksarvulise väärtuse:

```
>>> (-4)**(1/2)
(1.2246467991473532e-16+2j)
```

Tulemus näeb kole välja, kuid tuletagem meelde, et reaalarve koheldakse Pythonis ligikaudsetena. Saadud kompleksarvu reaalosa `1.2246467991473532e-16` väljendab reaalarvu standardkujul $1{,}2246467991473532 \cdot 10^{-16}$. See arv on väga-väga väike, ümardamise täpsusega sisuliselt 0. Niisiis jääb järele kompleksarv `2j` ehk $2i$, mille kohta on lihtne näha, et tegemist on korrektse vastusega, sest

$$(2i)^2 = 4 \cdot i^2 = -4\,.$$

Juhul $n = 3$ küsime, millist arvu kolmadale astmele (ehk kuupi) tõstes saame etteantud reaalarvu a. Seda arvu nimetatakse arvu a *kuupjuureks* ja tähistatakse

$$a^{\frac{1}{3}} = \sqrt[3]{a}\,.$$

Kuupjuure leidmiseks on moodulis `math` olemas funktsioon `cbrt` (ingl. k. *cubic root*):

```
>>> from math import cbrt
>>> cbrt(3)
1.4422495703074083
>>> cbrt(8)
2.0
>>> cbrt(31)
3.141380652391393
```

Näeme, et kuupjuur väljendatakse (potentsiaalselt ligikaudse) reaalarvuna, isegi siis, kui on olemas täpne täisarvuline väärtus (nt $\sqrt[3]{8} = 2$, sest $2^3 = 8$). Ja kas panid tähele, kui lähedal on $\sqrt[3]{31}$ arvule π?

Mis saab negatiivsete arvude kuupjuurtest?

```
>>> cbrt(-8)
-2.0
```

Tõepoolest, $(-2)^3 = -8$, seega $\sqrt[3]{-8} = -2$. Üldiselt võib öelda, et reaalarvulised kuupjuured on olemas kõigil reaalarvudel.

Nagu eespool nägime, saab kuupjuurt väljendada ka arvu astendamisega astmele $\frac{1}{3}$. Uurime, mida Python selle peale teeb:

```
>>> (-8)**(1/3)
(1.0000000000000002+1.7320508075688772j)
```

See vastus on väga ootamatu. Mis siin toimub? Miks tuleb vastuseks kompleksarv? On see vastus üldse õige? Vastuse reaalosa `1.0000000000000002` on ilmselt ümardamise täpsusega 1, imaginaarosa kordajas `1.7320508075688772` tunneme aga ära $\sqrt{3}$ ligikaudse väärtuse.

Ülesanne 2.1 Tõesta, et

$$(1 + \sqrt{3}i)^3 = -8\,.$$

Nagu tõdesime jaotises 1.5, on n. astme võrrandil komplekarvudes n lahendit. Muuhulgas kehtib see ka võrrrandi

$$x^3 = -8$$

kohta, mille jaoks -2 on ainult üks võimalik lahend ja $1 + \sqrt{3}i$ teine.

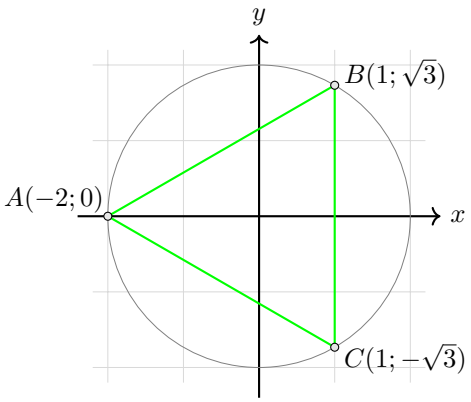
Ülesanne 2.2 Milline on võrrandi $x^3 = -8$ kolmas lahend?

Üks võimalus otsitava lahendini jõuda on mängida märkidega teadaolevas lahendis $1 + \sqrt{3}$. Loomulikult oskab Python leida ka võrrandi kõiki lahendeid, kui tema käest õigesti küsida. Üks võimalus on kasutada mooduli `sympy` (ingl.k. *symbolic Python*) funktsiooni `solve`.

```
>>> from sympy import solve
>>> from sympy.abc import x
>>> solve(x**3+8)
[-2, 1 - sqrt(3)*I, 1 + sqrt(3)*I]
```

Funktsioon `solve` võtab argumendiks avaldise, mille ta võrrandi saamiseks võrdsustab 0-ga (niisiis peame võrrandi $x^3 = -8$ esitama kujul $x^3 + 8 = 0$). Teiseks tuleb talle ilmutatult ette öelda, millised sümbolid selles avaldises vastavad muutujatele. Käsk `from sympy.abc import x` deklareerib sümboli `x` muutujaks, mille suhtes võrrandit lahendada. Lõpetuseks paneme tähele, et `sympy` kasutab imaginaarühiku tähistamiseks sümbolit `I`, mitte `j`.

Kanname võrrandi $x^3 = -8$ lahendid punktidena komplekstasandile (vt jaotist 1.6).

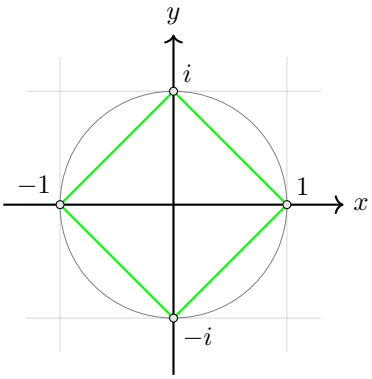


Ülesanne 2.3

(a) Tõesta, et punktid A, B ja C asuvad ringjoonel keskpunktiga koordinaattelgede lõikepunktis ning raadiusega 2.

(b) Tõesta, et ABC on võrdkülgne kolmnurk.

Võrdkülgse kolmnurga tekkimine pole kokkusattumus. Ülesandes 1.20 nägime, et võrrandil $x^4 = 1$ on neli kompleksarvulist lahendit: $1, -1, i$ ja $-i$. Nendele arvudele vastavad punktid moodustavad komplekstasandil korrapärase nelinurga ehk ruudu:



Üldisemalt kehtib teoreem

Teoreem 2.1 Olgu $n \in \mathbb{N}$ ($n \geqslant 1$) ja $c \in \mathbb{C}$ ($c \neq 0$). Võrrandi

$$z^n = c$$

lahendid moodustavad komplekstasandil korrapärase n-nurga tipud, mis asuvad ringjoonel keskpunktiga koordinaattelgede lõikepunktis.

Korrapäraseks 1-nurgaks nimetame siinkohal punkti ja 2-nurgaks lõiku. Teoreemi 2.1 tõestuse anname kursuses „Trigonomeetria".

2.2 Reaalarvu n. juur

Definitsioon 2.2. *Olgu antud naturaalarv $n \geqslant 1$. Reaalarvu (kompleksarvu) x n.* juureks *nimetame niisugust reaalarvu (kompleksarvu) y, et*

$$y^n = x\,.$$

Arvu x n. juurt tähistame $\sqrt[n]{x} = x^{\frac{1}{n}}$.

Uurime, millistel reaalarvudel on olemas reaalarvulised n. juured. Selleks peame küsima, millised arvud saavad esineda n. astmetena.

Mittenegatiivse arvu n. aste on kindlasti alati mittenegatiivne, negatiivse arvu korral tuleb aga eristada kahte juhtu vastavalt juurija n paarsusele.

Kui $x < 0$ ja n on paaritu, on x^n positiivne (näiteks $(-2)^4 = 16$). Kui aga n on paaritu, tuleb x^n negatiivne (näiteks $(-5)^3 = -125$).

Kokkuvõtteks saame järgmised reeglid.

- Kui n on paaritu arv, leidub igal reaalarvul täpselt üks reaalarvuline n. juur.
- Kui n on paaritu arv, leiduvad n. juured parajasti mittenegatiivsetel reaalarvudel. Seejuures on positiivsetel reaalarvudel täpselt kaks n. juurt, mis on teineteise vastandarvud.

Ülesanne 2.4 Kirjuta funktsioon `juur(n,x)`, mis saab sisendiks naturaalarvu $n \geqslant 1$ ja reaalarvu x ning trükib $\sqrt[n]{x}$ väärtuse (või veateate, kui juurida ei saa). Kui juurel on kaks reaalarvulist väärtust, väljasta neist positiivne.

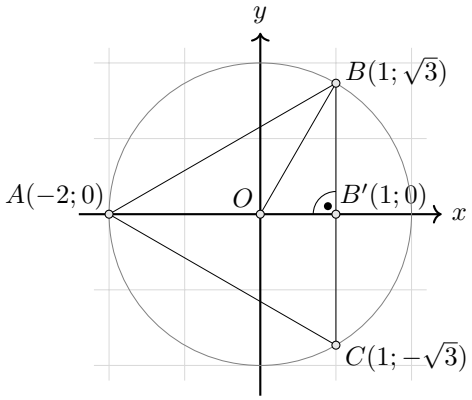
2.3 Lahendused

2.1 Teisendame ülesande avaldist:

$$\begin{aligned}(1+\sqrt{3}i)^3 &= (1+\sqrt{3}i)^2 \cdot (1+\sqrt{3}i) = (1+2\sqrt{3}i+3i^2)\cdot(1+\sqrt{3}i) = \\ &= (-2+2\sqrt{3}i)\cdot(1+\sqrt{3}i) = -2-2\sqrt{3}i+2\sqrt{3}i+2\cdot 3\cdot i^2 = \\ &= -2-6=-8\,.\end{aligned}$$

2.2 $1-\sqrt{3}i$.

2.3 (a) Ringjoon on niisuguste punktide hulk, mis asuvad etteantud keskpunktist fikseeritud raadiuse kaugusel. Tähistades koordinaatide algupunkti $O(0;0)$, peame tõestama, et $|OA| = |OB| = |OC| = 2$. Lõigu OA jaoks on see väide ilmne. Lõikude OB ja OC pikkuse leidmiseks saame kasutada Pythagorase teoreemi. Olgu B' punkti B projektsioon x-teljele.



Siis $|OB| = \sqrt{3}$, $|OB'| = 1$ ja $\angle OB'B = 90°$. Pythagorase teoreemist saame

$$|OB|^2 = |OB'|^2 + |BB'|^2 = 1^2 + (\sqrt{3})^2 = 1+3 = 4\,,$$

järelikult $|OB| = 2$, nagu oligi vaja. Tõestus, et $|OC| = 2$, on analoogiline.

(b) Paneme tähele, et $|AB'| = 3$ ja $\angle AB'B = 90°$. Kasutame Pythagorase teoreemi kolmnurgas $AB'B$:

$$|AB|^2 = |AB'|^2 + |BB'|^2 = 3^2 + (\sqrt{3})^2 = 9+3 = 12\,,$$

järelikult $|AB| = \sqrt{12}$. Sama moodi tõestame võrduse $|AC| = \sqrt{12}$. Teisest küljest saame lõigu BC pikkuse jaoks

$$|BC| = |BB'| + |B'C| = \sqrt{3}+\sqrt{3} = 2\sqrt{3}\,.$$

Jääb veel tähele panna, et

$$|BC|^2 = (2\sqrt{3})^2 = 2\sqrt{3} \cdot 2\sqrt{3} = 4 \cdot 3 = 12\,,$$

niisiis $|AB|^2 = |AC|^2 = |BC|^2$ ja järelikult $|AB| = |BC| = |AC|$.

Selles ülesandes kasutatud meetod lõikude OB ja AB pikkuse arvutamiseks on üldine. Kui punktid A ja B on antud oma koordinaatidega $A(x_A; y_A)$ ja $B(x_B; y_B)$, siis nendevaheline kaugus (ehk lõigu AB pikkus) avaldub Pythagorase teoreemi põhjal kujul

$$d(A,B) = |AB| = \sqrt{(x_A - x_b)^2 + (y_a - y_B)^2}\,.$$

Teiseks üldistub ka $(2\sqrt{3})^2$ arvutamisel kasutatud võte suvaliste tegurite ja (naturaalarvuliste) astendajate korral:

$$(ab)^n = \underbrace{(ab) \cdot (ab) \cdot \ldots \cdot (ab)}_{n} = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n} \cdot \underbrace{b \cdot b \cdot \ldots \cdot b}_{n} = a^n b^n\,.$$

Juhul kui n. juure võtmine on lubatud (näiteks kui reaalarvuliste väärtuste korral on astmete alused mittenegatiivsed või kui n on paaritu), saame

$$\sqrt[n]{a^n} \cdot \sqrt[n]{b^n} = ab = \sqrt[n]{(ab)^n} = \sqrt[n]{a^n b^n}\,,$$

millest (tähistades a^n ümber a-ga ja b^n ümber b-ga), saame

$$\sqrt[n]{ab} = \sqrt[n]{a} \cdot \sqrt[n]{b}\,.$$

Viimast reeglit rakendades saame näiteks kohe arvutada, et

$$\sqrt{12} = \sqrt{4 \cdot 3} = \sqrt{4} \cdot \sqrt{3} = 2\sqrt{3}\,.$$

2.4 Mittenegatiivse x korral võime trükkida `x**(1/n)`. Negatiivse x väärtuse puhul peame andma veateate, kui n on paaris, ning negatiivse reaalarvulise väärtuse, kui n on paaritu. Me teame juba, et `x**(1/n)` peale väljastab Python kompleksarvulise vastuse, niisiis peame $\sqrt[n]{x}$ sel juhul arvutama kui $-\sqrt[n]{-x}$.

```
def juur(n,x):
    if x>=0:
        print(x**(1/n))
    else:
        if n%2 == 0:
            print("Reaalarvulist juurt pole!")
        else:
            print(-(-x)**(1/n))
```

PEATÜKK 3

Avaldised ja nende teisendamine

Tuletame meelde, et jaotises 2.1 defineerisime avaldise x^n kui arvu, mille saame arvu x iseendaga n korda korrutamisel. Seejuures käsitleme x-i muutujana, st me võime talle omistada erinevaid reaalarvulisi (või lausa kompleksarvulisi) väärtusi ning x^n annab meile tema n. astme.

Definitsioon 3.1. *Olukorda, kus igale sisendväärtusele vastab üks kindel väljundväärtus, nimetame* funktsionaalseks sõltuvuseks*, ja eeskirja, mille järgi sisendi põhjal väljund leitakse, nimetame* funktsiooniks*.*

Funktsionaalset vastavust võib tähistada mitmel moel, näiteks noolekesega $\mapsto$. Nõnda kirjeldab vastavus $x \mapsto x^n$ astmefunktsiooni, vastavus $x \mapsto \sqrt[n]{x}$ aga juurfunktsiooni. Kui funktsioonile on vaja anda eraldi nimi, võime kasutada tähistust

$$f : x \mapsto x^n \quad \text{või} \quad f(x) = x^n \,.$$

Funktsioone saab üldiselt esitada paljudel erinevatel viisidel. Nii näiteks võime sisendi iseendale liitmise tulemust kirjeldada vastavusena $x \mapsto x + x$, aga ka $x \mapsto x + x + x - x$ ja $x \mapsto 2x$. Kuigi avaldised $x + x$, $x + x + x - x$ ja $2x$ on vormilt erinevad, väljendavad nad siiski kõik ühte ja sama suurust ning neid võib selles mõttes samaväärseteks lugeda. Küll aga on mõni neist avaldistest arvutuseeskirjana lihtsam kui mõni teine. Nii näiteks nõuab $x + x$ ühte liitmist, $x + x + x - x$ kahte liitmist ja ühte lahutamist, $2x$ aga hoopis ühte korrutamist.

Selles peatükis uurimegi avaldiste samaväärsuse kindlakstegemise reegleid ning võimalusi teisendada avaldisi nii, et nende väärtusi oleks lihtsam arvutada.

3.1 Polünoomid

Kuigi arvuteid kasutatakse tänapäeval väga keerukate arvutusülesannete lahendamiseks, oskavad nad madalal riistvaralisel tasemel sooritada vaid üsna

piiratud hulka tehteid, näiteks liitmist, lahutamist ja korrutamist ning veidi suurema vaevaga ka jagamist. Programmeerimine tähendabki ju tegelikult keeruka ülesande jagamist väiksemateks sammudeks, mida arvuti vahetult sooritada suudab.

Uurime kõigepealt lähemalt, milliseid arvutusi saab teha, kui meil on kasutada ainult liitmine, lahtutamine ja korrutamine.

Võttes sisendiks arvu x, saame nende tehete ühekordsel rakendamisel arvud

$$x+x, \quad x-x \quad \text{ja} \quad x \cdot x\,.$$

Me teame juba, et arvu, mis tekib arvu x liitmisel iseendale, saame leida ka x-i korrutamisel 2-ga. Seda asjaolu kirjutame võrdusmärgi abil

$$x+x=2x\,.$$

On väga oluline mõista, mida see võrdus meile sisuliselt ütleb. Ta ütleb, et vastavused (funktsioonid) $x \mapsto x+x$ ja $x \mapsto 2x$ annavad *iga* reaalarvu x korral sama tulemuse.

Ülesanne 3.1 Pane Pythoni abil käima kood

```
for x in [1, -2.2, 3.14, -0.17]:
    print(x+x,2*x)
```

ja veendu, et iga kord väljastatakse sama väärtus kaks korda.

Muuda sisendloendis väärtusi. Kas samad väärtused väljastatakse ka siis, kui sisenditeks on kompleksarvud?

Mis juhtub lahutamise korral? Me teame juba, et

$$x-x=0\,,$$

aga jällegi on oluline aru saada, et see võrdus kehtib *iga* reaalarvu (või lausa kompleksarvu) korral. Seda võrdust võime mõista ka funktsioonide võrdusena, kus võrdusmärgi vasakul pool asub funktsioon $x \mapsto x-x$, paremal pool aga $x \mapsto 0$, st konstantne funktsioon, mis annab iga sisendi puhul väljundiks sama arvu 0.

Sama moodi tähendab kirjutus

$$x \cdot x = x^2$$

funktsioonide $x \mapsto x \cdot x$ ja $x \mapsto x^2$ võrdust, st nende väljundväärtuste võrdust kõigil sisenditel.

Ülesanne 3.2 Muuda ülesande 3.1 koodi nii, et see kontrolliks võrduste $x-x=0$ ja $x \cdot x = x^2$ kehtivust etteantud sisendloendi elementidel. Muuda loendi elemente seni, kuni sa neid võrdusi uskuma jääd.

Peale sisendväärtuse enda saame tehetes muidugi kasutada ka arvulisi konstante ja moodustada avaldisi stiilis

$$x+1, \quad 5-x \quad \text{ja} \quad (-3{,}14) \cdot x\,.$$

Jällegi on oluline mõista, et need avaldised esitavad teatavaid funktsioone, näiteks $x+1$ on funktsioon, mis seab igale sisendväärtusele vastavusse ühe võrra suurema väljundi.

Nende avaldistega saame edasi arvutada. Näiteks võime omavahel korrutada $x+1$ ja $5-x$ ning kasutada sulgude avamise ja sarnaste liikmete koondamise reegleid:

$$(x+1)\cdot(5-x) = 5x - x^2 + 5 - x = -x^2 + 4x + 5\,.$$

Taaskord kehtivad need võrdused suvalise sisendväärtuse x korral. Avaldisi $(x+1)\cdot(5-x)$ ja $-x^2+4x+5$ võime mõista kui sama väärtuse kahte erinevat arvutuseeskirja. Esimese puhul peame avaldise väärtustamiseks sooritama ühe liitmise, ühe lahutamise ja ühe korrutamise, teise puhul kaks korrutamist (x^2 ja $4x$), kaks liitmist ja ühe vastandelemendi leidmise (mis on sisuliselt lahutamine). Niisiis võime öelda, et esimene esitus on mingis mõttes lihtsam kui teine. Selle teema juurde tuleme tagasi jaotises .

Sama moodi saame jätkata avaldiste liitmist, lahutamist ja korrutamist:

$$\begin{aligned} -x^2+4x+5+(-3{,}14)x &= -x^2+0{,}86x+5\,,\\ (-x^2+4x+5)\cdot\left(-\frac{1}{2}x+1\right) &= \frac{1}{2}x^3-2x^2-\frac{5}{2}x-x^2+4x+5 = \\ &= \frac{1}{2}x^3-3x^2+\frac{3}{2}x+5 \end{aligned}$$

jne.

Pythonis saame selliseid arvutusi mugavalt sooritada mooduli `sympy` abil. Avaldiste liitmisel-lahutamisel lihtsustab `sympy` ise vastuse ära, korrutamise jaoks peame importima käsu `expand`. Muidugi tuleb ka sümbol `x` muutujaks deklareerida:

```
>>> from sympy import expand
>>> from sympy.abc import x
>>> -x**2+4*x+5+(-3.14)*x
-x**2 + 0.86*x + 5
>>> expand((-x**2+4*x+5)*(-1/2*x+1))
0.5*x**3 - 3.0*x**2 + 1.5*x + 5
```

Ülesanne 3.3 Arvuta kästitsi ja kontrolli tulemus `sympy` abil üle:

(a) $(x+1)(x-2)+(x+2)$;

(b) $(x^2-x+1)(x^2+x+1)$;

(c)

Näeme, et kõik tekkivad avaldised saab sulgude avamise reeglite abil teisendada kujule, mis koosneb muutuja x astmetest, mis on korrutatud teatud kordajatega ning siis liidetud-lahutatud. (Kuivõrd lahutamine on sama, mis vastandarvu liitmine, piisab tegelikult käsitleda ainult liitmist.)

Kuna niisugused avaldised on matemaatikas tähtsal kohal, on neile antud eraldi nimed.

Definitsioon 3.2. *Avaldist kujul*

$$a \cdot x^n \,,$$

kus a on arvuline konstant ja n on naturaalarv, nimetatame üksliikmeks. *Üksliikmete summat kujul*

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0 \,,$$

kus $a_n, a_{n-1}, \ldots, a_2, a_1, a_0$ on mingid arvulised konstandid nimetatame (ühe muutuja) polünoomiks. *Konstante a_i nimetame polünoomi* kordajateks.

Definitsioonis 3.2 võivad kordajad a_i pärineda erinevatest arvuvaldadest, täiteks täis-, reaal- või kompleksarvude hulgast. Kui pole eraldi täpsustatud, vaatleme selles peatükis vaikimisi reaalarvuliste kordajatega polünoome.

Kokkuvõttes näeme, et liitmise, lahutamise ja korrutamise abil saame ühest sisendväärtusest ja konstantidest moodustada parajasti polünomiaalseid avaldisi.

Polünoomavaldises muutujale väärusi andes ja kõiki tehteid läbi tehes saame iga kord ühe konkreetse väljundväärtuse. Seega esitab iga polünoom mingit funktsiooni.

Vastupidine pole õige – sugugi mitte iga funktsioon ei avaldu polünoomina.

Ülesanne 3.4* Näita, et funktsioon $x \mapsto \frac{1}{x}$ pole polünoom.

On võimalik tõestada, et polünoomidena ei avaldu näiteks ka juurfunktsioonid $x \mapsto \sqrt[n]{x}$ (kus $n \geqslant 2$) ega trigonomeetrilised funktsioonid. Küll aga osutub, et polünoomide abil saab kõiki neid funktsioone *lähendada*.

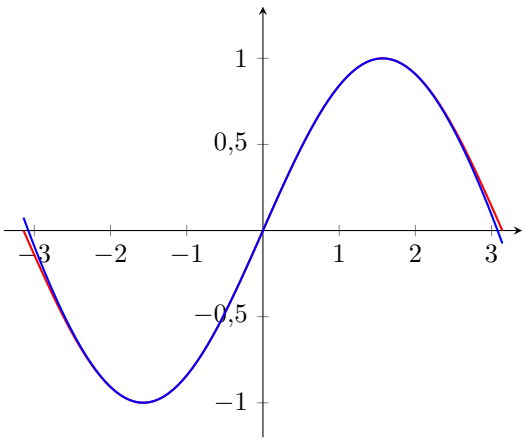
Tegelikult arvutataksegi arvutis peaaegu kõiki funktsioone polünoomidega lähendamise abil. Mõtleme näiteks siinuse peale. Tema formaalne definitsioon täisnurkse kolmnurga vastaskaateti ja hüpotenuusi suhte kaudu on vahetuks arvutamiseks enamasti kasutu. Arvutiprotsessor ei hakka ju $\sin 34^\circ$ väärtuse leidmiseks joonestama täisnurkset kolmnurka teravnurgaga 34° ning siis kuidagimoodi küljepikkuseid mõõtma.

Selle asemel on palju mugavam kasutada hulkliiget

$$P(x) = x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 \,.$$

Vahemikus $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ lähendab polünoom p siinusfunktsiooni nii hästi, et viga jääb alla $0{,}02\%$.

Vahemiku $[-\pi, \pi]$ otspunktide läheduses on viga juba silmaga nähtav, aga ikka veel väga väike. Joonisel tähistab punane joon siinuse tõelist väärtust ja sinine joon polünoomi P graafikut.



Küsimus, miks just polünoom P siinusfunktsiooni nii hästi lähendab ja kuidas leida polünoome, mis sobiksid teiste funktsioonidega, jääb selle õpiku raamidest välja. Oluline on aga meelde jätta, et polünoomid võimaldavad taandada keerukate funktsioonide arvutamise aritmeetika põhitehetele liitmisele, lahutamisele ja korrutamisele.

Ülesanne 3.5 Võta mõni programm, mis oskab joonistada funktsioonide graafikuid (näiteks GeoGebra, `https://www.geogebra.org/`). Uuri polünoomi P laiemas piirkonnas, näiteks $[-2\pi; 2\pi]$. Kui hästi P siinusfunktsiooni seal lähendab?

Ülesanne 3.6 Polünoomi P avaldist uurides torkab silma, et $1 = 1!$, $6 = 3! = 1 \cdot 2 \cdot 3$, $120 = 5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$ ja $5040 = 7! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7$. Selline kokkusattumus ei saa olla juhuslik ja ega ta ei olegi. Mis juhtub siis, kui polünoomile sama reegli järgi liikmeid juurde lisada? Mitu liiget tuleks lisada, et *absoluutne viga* (st lähendi ja õige väärtuse vahe absoluutväärtus) vahemikus $[-2\pi; 2\pi]$ oleks väiksem kui $0{,}011$?

Ülesanne 3.7 Joonista `matplotlib`-teegi abil 1.-6. astme polünoomide graafikuid. Varieeri kordajaid ja nende märke. Mida huvitavat tähele paned?

3.1.1 Projektülesanne: ruutjuure lähendamine

Sind palgatakse rahvusvahelisse kiibitootmisfirmasse ja saadetakse arenduses oleva kiibi aritmeetikaprotsessori väljatöötamise meeskonda. Protsessori jaoks on juba realiseeritud liitmine, lahutamine, korrutamine, täisarvude jäägiga jagamine, reaalarvude jagamine konstantidega ning lihtsamad loogikaoperatsioonid (nt arvude võrdlemine ja tingimuslaused `if ... then`). Sinu ülesanne on luua nende vahenditega ruutjuure arvutamise algoritm.

Reaalarvud on kiibis esitatud standardkujul $x = a \cdot 10^b$, kus $1 \leqslant a < 10$ ja $b \in \mathbb{Z}$ (arvu ees võib põhimõtteliselt olla ka miinusmärk, aga ruutjuure võtmiseks eeldame, et sisend on mittenegatiivne). Niisiis on algoritmi sisendiks arvud a ja b ning ka vastus tuleb anda standardkujul.

Paksust kõrgkooliõpikust loed välja, et ruutjuurt saab arvu 1 ümbruses lä-

hendada polünoomiga

$$p(x) = 1 + \frac{1}{2}(x-1) - \frac{1}{8}(x-1)^2 + \frac{1}{16}(x-1)^3\,.$$

Ülesanne 3.8 Esita polünoom $p(x)$ definitsioonile 3.2 vastaval kujul, st korruta sulud lahti ja koonda sarnased liikmmed.

Ülesanne 3.9 Uuri, millises (võimalikult suures) piirkonnas polünoom $p(x)$ ruutjuurt piisavalt hästi lähendab, st et suhteline viga jääks alla 1%. Lähendi *suhteliseks veaks* nimetame absoluutse vea ja funktsiooni väärtuse suhet, st antud juhul väärtust $\frac{|p(x)-\sqrt{x}|}{\sqrt{x}}$.

Ülesanne 3.10 Paneme tähele, et

$$\sqrt{x} = \sqrt{a \cdot 10^b} = \sqrt{x} \cdot \sqrt{10^b}\,,$$

niisiis võime eraldi arvutada $\sqrt{a}$ ja $\sqrt{10^b}$ ning seejärel tulemused korrutada.

(a) Mõtle välja algoritm $\sqrt{a}$ arvutamiseks, kus $1 \leqslant a < 10$.

(b) Mõtle välja algoritm $\sqrt{10^b}$ arvutamiseks, kus $b \in \mathbb{Z}$. Väljasta tulemus standardkujul, st arvupaarina (a_1, b_1), kus $1 \leqslant a_1 < 10$, $b_1 \in \mathbb{Z}$ ja $a_1 \cdot 10^{b_1} = \sqrt{10^b}$.

Ülesanne 3.11 Katseta, kui hästi ülesande 3.10 (a)-osa programm töötab, st uuri tema väljundi suhtelist viga erinevate sisendite $a \in [1; 10)$ korral.

Ülesanne 3.12 Pane ülesande 3.10 (a)- ja (b)-osa lahenduste põhjal kokku funktsioon `ruutjuur_ab`, mis saab sisendiks standarkujul esitatud reaalarvu tüve ja eksponendi ning väljastab vastuse samuti standardkujul, st tüve ja eksponendi paarina.

Ülesanne 3.13 Ülesannetes 3.9 ja 3.11 tuli hinnata erinevate funktsioonide suhtelisi vigu. Kui sa seda juba teinud ei ole, kirjuta funktsioon `suhtelineviga` ümber nii, et ta võtaks sisse kaks argumenti: koha x, millel suhtelist viga leida, ja lähendifunktsiooni f, mille suhtelist viga funktsiooni $x \mapsto \sqrt{x}$ lähendamisel arvutatakse.

Ülesanne 3.14* Hinda ülesandes 3.12 loodud funktsiooni `ruutjuur_ab` suhtelist viga ülesandes 3.13 kirjutatud funktsiooni `suhtelineviga` abil.

3.1.2 Projektülesanne: polünoomide efektiivne arvutamine

Sinu poolt jaotises 3.1.1 välja töötatud algoritm realiseeritakse kiibi prototüübil ära ja testitakse läbi. Hindamiskomitee kiidab algoritmi põhimõttelise disaini heaks, aga leiab, et suhteline viga 1% on natuke liiga palju. Kaevud uuesti paksu kõrgkooliõpikusse ja leiad, et paremate lähendite leidmiseks tuleb tõsta polünoomi astet. Näiteks ruutjuurt lähendab arvu 1 ümbruses paremini polü-

noom

$$1+\frac{1}{2}(x-1)-\frac{1}{8}(x-1)^2+\frac{1}{16}(x-1)^3-\frac{5}{128}(x-1)^4+\frac{7}{256}(x-1)^5=$$
$$=\frac{7}{256}x^5-\frac{45}{256}x^4+\frac{63}{128}x^3-\frac{105}{128}x^3+\frac{315}{256}x+\frac{63}{256}\,.$$

Kas see lähend on nüüd piisavalt hea või peab veel kõrgemate astmeteni minema? (Tuletame meelde, et näiteks ülesandes 3.6 läks vaja 17. astme polünoomi.) Hindamiskomitee kõhkleb. Kõrgemad astmed võimaldavad paremat täpsust, aga nõuavad ka rohkem arvutusaega ja kiibi protsessor pole ülearu kiire.

Kuniks komitee mõtleb, millise astme polünoom valida, antakse sulle ülesandeks optimeerida polünoomide väärtuste arvutamist üleüldiselt.

Esitame polünoomi $a_nx^n+a_{n-1}x^{n-1}+\ldots+a_2x^2+a_1x+a_0$ kordajate loendina `a`. Paneme tähele, et traditsiooni kohaselt kirjutatakse polünoomi puhul üksliikmed astmete kahanemise järjekorras vasakult paremale, loendi puhul aga on vasakul pool väiksema indeksiga elemendid. Nii on polünoom ja teda esitav kordajate loend visuaalselt vastupidises järjekorras. Näiteks polünoomile x^3+8x^2-3x+5 vastab loend `a=[5,-3,8,1]`; siis polünoomi vabaliiget $a_0=5$ esitab loendi element `a[0]` väärusega `5` jne.

Esimene mõte, mis pähe tuleb, on arvutada välja üksliikmete väärtused (nt alates madalaimast astmest) ja siis kokku liita. Muutuja x astmete väärtused saame leida definitsiooni järgi korrutades, st $x^0=1$, $x^1=x$, $x^2=x\cdot x$, $x^3=x\cdot x\cdot x$ jne. Tulemusena saadav kood võiks välja näha umbes nii:

```
def eval1(a,x):    # a on polünoomi kordajate loend
                   # x on väärtustuskoht
    tulem = 0
    for i in range(len(a)):
        xaste = 1
        for j in range(i):
            xaste *= x
        tulem += a[i]*xaste
    return(tulem)
```

Loendi `a` moodustavad elemendid $a_0,a_1,\ldots,a_{n-1},a_n$, mida on kokku $n+1$ tükki, st `len(a)` väärtus on $n+1$. Niisiis võtab muutuja `i` parajasti väärtused $0,1,\ldots,n-1,n$. Arvu x astme väärtuse arvutamiseks võtame muutuja `xaste` algväärtusega 1 ning korrutame talle `i` korda juurde arvu x. (Muuhulgas kui `i` väärtus on 0, saame kätte õige tulemuse 1.) Jääb üle suurendada muutujat `tulem` suuruse $a_i\cdot x^i$ võrra ja lõpuks tulemus tagastada.

Ülesanne 3.15 Mitu liitmist ja korrutamist tuleb teha 3. astme polünoomi väärtustamisel funktsiooni `eval1` abil? Aga 5. astme polünoomi korral? Kas oskad vajalike aritmeetiliste tehete arvu üldistada ka n. astme polünoomile?

3.2 Polünoomide jäägiga jagamine

Jaotises 3.1 nägime, et polünoome saab liita, lahutada ja korrutada. Kas polünoome saab omavahel ka jagada?

Osutub, et saab küll, kusjuures polünoomide korral on olemas midagi üsna sarnast täisarvude vallast tuttavale jäägiga jagamisele. Seejuures järkude kaupa jagamise asemel jagame astmete kaupa. Asi saab selgemaks näite varal.

Jagame polünoomi $x^4 + 2x^3 - x + 3$ polünoomiga $x^2 - x - 3$, st otsime niisuguseid polünoome $Q(x)$ ja $R(x)$, et

$$x^4 + 2x^3 - x + 3 = (x^2 - x - 3) \cdot Q(x) + R(x)\,.$$

Kirjutame otsitava $Q(x)$ kohale tühjad sulud, jätame jagatavasse puuduva liikme x^2 kohale veidi ruumi ja saame järgmise üleskirjutuse.

$$x^4 + 2x^3 \qquad - x \;\; + 3 = (x^2 - x - 3)(\qquad\qquad)$$

Jagatava kõrgeima astme liige on x^4, jagajal aga x^2. Nende jagatis on x^2, mis annabki jagatispolünoomi $Q(x)$ esimese liikme.

$$x^4 + 2x^3 \qquad - x \;\; + 3 = (x^2 - x - 3)(x^2 \qquad\qquad)$$

Sarnaselt täisarvude pika jagamisega leiame osakorrutise $x^2 \cdot (x^2 - x - 3) = x^4 - x^3 - 3x^2$ ja lahutame selle jagatavast (st liidame vastandpolünoomi $-x^4 + x^3 + 3x^2$) kuni astmeni x^2.

$$\begin{array}{rl} x^4 + 2x^3 & - x \;\; + 3 = (x^2 - x - 3)(x^2 \qquad\qquad) \\ \underline{- x^4 \;\; + x^3 + 3x^2} & \\ 3x^3 + 3x^2 & - x \end{array}$$

Tuues jagatavast juurde järgmise astme liikme $-x$, tuleb järgmiseks jagada polünoomi $3x^3 + 3x^2 - x$. Jagades selle pealiikme $3x^3$ jälle jagaja pealiikmega x^2 on tulemuseks $3x$, mis annab meile jagatise $Q(x)$ teise liikme.

$$\begin{array}{rl} x^4 + 2x^3 & - x \;\; + 3 = (x^2 - x - 3)(x^2 + 3x \qquad) \\ \underline{- x^4 \;\; + x^3 + 3x^2} & \\ 3x^3 + 3x^2 & - x \end{array}$$

Lahutame $3x \cdot (x^2 - x - 3) = 3x^3 - 3x^2 - 9x$ vahetulemusest ja toome juurde jagatava viimase liikme 3.

$$\begin{array}{rl} x^4 + 2x^3 & - x \;\; + 3 = (x^2 - x - 3)(x^2 + 3x \qquad) \\ \underline{- x^4 \;\; + x^3 + 3x^2} & \\ 3x^3 + 3x^2 & - x \\ \underline{- 3x^3 + 3x^2 \;\; + 9x} & \\ 6x^2 \;\; + 8x & + 3 \end{array}$$

Viimase vahetulemuse pealiikme $6x^2$ ja jagaja pealiikme x^2 jagatis on 6, mis annabki otsitava jagatise viimase liikme. Lahutame $6 \cdot (x^2 - x - 3) = 6x^2 - 6x - 18$ viimasest vahetulemusest.

$$\begin{array}{rl} x^4 + 2x^3 & - x \;\; + 3 = (x^2 - x - 3)(x^2 + 3x + 6) \\ \underline{- x^4 \;\; + x^3 + 3x^2} & \\ 3x^3 + 3x^2 & - x \\ \underline{- 3x^3 + 3x^2 + 9x} & \\ 6x^2 + 8x & + 3 \\ \underline{- 6x^2 + 6x + 18} & \\ 14x + 21 & \end{array}$$

Järele jääb lineaarpolünoom $14x + 21$, mille pealiikme aste on väiksem kui jagaja aste. See tähendab, et jagamine on lõppenud ja $14x + 21$ on jagamisel tekkinud jääk.

$$\begin{array}{rl} x^4 + 2x^3 & - x \ + 3 = (x^2 - x - 3)(x^2 + 3x + 6) + 14x + 21 \\ \underline{- x^4 \ + x^3 + 3x^2} & \\ 3x^3 + 3x^2 & - x \\ \underline{- 3x^3 + 3x^2 + 9x} & \\ 6x^2 + 8x & + 3 \\ \underline{- 6x^2 + 6x + 18} & \\ 14x + 21 & \end{array}$$

Ülesanne 3.16 Kontrolli lahti korrutades, et

$$(x^2 - x - 3)(x^2 + 3x + 6) + 14x + 21 = x^4 + 2x^3 - x + 3\,.$$

Ülesanne 3.17 (Talvine lahtine võistlus 2010, noorem rühm) Olgu x selline reaalarv, et $x^3 + 2x + 2 = 0$. Leia avaldise $x^5 + 2x^2 - 4x + 2010$ väärtus.

3.3 Polünoomide tegurdamine

Nagu ka täisarvude puhul, pakub polünoomide jäägiga jagamise juures erilist huvi olukord, kus jääk on null.

Definitsioon 3.3. *Ütleme, et polünoom $P(x)$* jagub *polünoomiga $Q(x)$ ja kirjutame $P(x) \vdots Q(x)$, kui nende jagamisel tekkiv jääk on 0, st leidub selline polünoom $S(x)$, et*

$$P(x) = S(x) \cdot Q(x)\,.$$

Sama seose kohta ütleme ka, et polünoom $Q(x)$ jagab *polünoomi $P(x)$, ja kirjutame $Q(x) \mid P(x)$.*

Täisarvude vallast tuttavale algarvu mõistele vastab polünoomide puhul taandumatuse omadus. Kui algarvu p korral on tema ainus tegurdus triviaalne $1 \cdot p$ (ja negatiivseid tegureid arvestades ka $(-1) \cdot (-p)$), siis polünoomidel on triviaalseid tegurdusi rohkem. Nii näiteks võime hulkliikme $2x^2 + 4x - 6$ esitada kujul

$$2x^2 + 4x - 6 = 1 \cdot (2x^2 + 4x - 6) = 2 \cdot (x^2 + 2x - 3) = 4 \cdot \left(\frac{1}{2}x^2 + x - \frac{3}{2}\right)$$

jne, aga need esitused ei anna meile tema omaduste kohta uut informatsiooni. Huvitavaks läheb asi alles siis, kui tegurite aste on madalam kui tegurdatava aste ja suurem kui 0.

Definitsioon 3.4. *Ütleme, et n. astme polünoom $P(x)$ on* taanduv, *kui leiduvad polünoomid $R(x)$ ja $S(x)$ nii, et nende aste on vähemalt 1 ja*

$$P(x) = R(x) \cdot S(x)\,.$$

Kui niisuguseid polünoome $R(x)$ ja $S(x)$ ei leidu, nimetame polünoomi $P(x)$ taandumatuks.

Ülesanne 3.18 Kas polünoom $x^2 - x + 1$ on taanduv või taandumatu? (Otsime reaalarvuliste kordajatega tegureid.)

Lahendus. Kui polünoom $x^2 - x + 1$ oleks taanduv, peaksid leiduma esimese astme reaalpolünomid $ax + b$ ja $cx + d$ nii, et

$$x^2 - x + 1 = (ax + b)(cx + d)\,.$$

Paneme tähele, et sel juhul peaksid antud polünoomil leiduma reaalarvulised nullkohad $-\frac{b}{a}$ ja $-\frac{d}{c}$. Tõepoolest, $a \neq 0$ ja $c \neq 0$, sest muidu poleks $ax + b$ ja $cx + d$ esimese astme polünoomid, ja lisaks

$$a \cdot \left(-\frac{b}{a}\right) + b = -b + b = 0 \quad \text{ning} \quad c \cdot \left(-\frac{d}{c}\right) + d = -d + d = 0\,.$$

Ruutvõrrandi

$$x^2 - x + 1 = 0$$

diskriminant on aga $-1 - 4 \cdot 1 = -5 < 0$, seega polünoomil $x^2 - x + 1$ ei leidu reaalarvulisi nullkohti. Järelikult peab ta olema taandumatu. □

Definitsioon 3.5. *Väärtust x, mille korral $P(x) = 0$, nimetatakse polünoomi* nullkohaks *ehk* juureks.

Ülesandes 3.18 tundis tähelepanelik lugeja loodetavasti ära järgmise põhikoolist pärineva tulemuse, mis annab üldise eeskirja ruutkolmliikmete tegurdamiseks.

Teoreem 3.1 Kui ruutvõrrandil $ax^2 + bx + c = 0$ on lahendid x_1 ja x_2, siis kehtib võrdus

$$ax^2 + bx + c = a(x - x_1)(x - x_2)\,.$$

Ülesanne 3.19 (Lõppvoor 1995, 9. klass) Leia kõik täisarvud n, mille korral $4n^2 + 16n - 65$ on algarv.

Harjutuses 3.18 nägime, et reaalarvuliste kordajatega polünoomi juure leidumine on tihedalt seotud lineaarteguri leidumisega. Osutub, et need omadused ongi samaväärsed. Järgmist teoreemi tuntakse ka Bézout' väikese teoreemi[1] nime all.

Teoreem 3.2 Olgu $P(x)$ reaalarvuliste kordajatega polünoom ja $a \in \mathbb{R}$. Polünoomi $P(x)$ jagamisel vahega $x - a$ tekkiv jääk on $P(a)$. Muuhulgas on a polünoomi $P(x)$ juureks parajasti siis, kui $(x - a) \mid P(x)$.

Tõestus. Jagame polünoomi $P(x)$ jäägiga polünoomiga $x - a$. Kuna $x - a$ on 1. astme polünoom ja jääk on jagajast madalama astmega, saab jääk olla ainult

[1] Étienne Bézout [bezu:] (1730 – 1783) oli tuntud Prantsuse matemaatik.

0. astme ehk konstantne polünoom; olgu ta näiteks c. Tähistades jagatise $Q(x)$ saame

$$P(x) = (x - a) \cdot Q(x) + c\,.$$

See võrdus kehtib muutuja x iga väärtuse korral, muuhulgas ka siis, kui $x = a$. See asendus annab

$$P(a) = (a - a) \cdot Q(a) + c = 0 \cdot Q(a) + c = c;$$

teisisõnu – polünoomi $P(x)$ jagamisel vahega $x - a$ tekkiv jääk ongi täpselt $P(a)$.

Nüüd on lihtne näha, et a on polünoomi $P(x)$ juureks (st $P(a) = 0$) parajasti siis, kui $P(x)$ avaldub kujul $P(x) = (x-a) \cdot Q(x)$ mingi polünoomi $Q(x)$ korral, st $(x - a) \mid P(x)$.

□

Teoreemi 3.2 saab kasutada polünoomi lineaartegurite otsimiseks. Lihtne võimalus selleks on proovida läbi näiteks $a = 1$, $a = 2$, $a = -1$ jne ning kui mõni neist osutub $P(x)$ juureks, olemegi leidnud lineaarteguri $x - a$.

Eriti lihtne on kontrollida, kas $a = 1$ on juur või mitte. Selle juures osutub kasulikuks järgmine teoreem.

Teoreem 3.3 Polünoomi kordajate summa võrdub tema väärtusega kohal 1.

Tõestus. Olgu

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0\,.$$

Kuna iga i korral $1^i = 1$, saame

$$P(1) = a_n \cdot 1^n + a_{n-1} \cdot 1^{n-1} + \ldots + a_1 \cdot 1^1 + a_0 = a_n + a_{n-1} + \ldots + a_1 + a_0\,.$$

□

Teoreemist 3.3 saamegi lihtsa kriteeriumi otsustamaks, kas 1 on vaadeldava polünoomi juur.

Teoreem 3.4 Arv 1 on polünoomi $P(x)$ juureks (ja seega ka $P(x) \vdots (x - 1)$) parajasti siis, kui selle polünoomi kordajate summa on 0.

Ülesanne 3.20 Polünoomi kordajate põhjal saab lihtsasti otsustada ka seda, kas -1 on antud polünoomi juureks või ei. Kuidas?

3.4 Lahendused

3.1 Jah, ka kompleksarvude puhul kehtib võrdus $x + x = 2x$.

3.2 Töötavad näiteks sellised koodilõigud:

```
for x in [1, -2.2, 3.14, -0.17]:
    print(x-x,0)
```

```
for x in [1, -2.2, 3.14, -0.17]:
    print(x*x,x**2)
```

Pane tähele, et reaalarve esitab Python üldjuhul ligikaudselt ja nii ei pruugi näiteks $(-0{,}17)^2$ olla täpselt see, mida sa eeldad.

3.3 (a) x^2; (b) $x^4 + x^2 + 1$.

3.4 Polünoomid on määratud kõigil reaalarvudel, aga funktsiooni $x \mapsto \frac{1}{x}$ määramispiirkonda ei kuulu arv 0.

Ka siis, kui funktsiooni $x \mapsto \frac{1}{x}$ definitsiooni täiendada nii, et ta kohal 0 kuidagi lisaks määrata, ei saa me ikkagi polünoomi. Selles võime veenduda näiteks pidevuseargumendiga (kõik polünoomid on pidevad, aga funktsiooni $x \mapsto \frac{1}{x}$ graafikul on kohal $x = 0$ hüpe $-\infty$-st ∞-sse).

Kui me ei soovi kasutada pidevuseargumenti (mida koolis tegelikult rangelt ei käsitleta), saame uurida funktsiooni $x \mapsto \frac{1}{x}$ käitumist piirkonnas $(0; 1]$. Kui x võtab järjest väiksemaid positiivseid väärtusi, kasvab $\frac{1}{x}$ tõkestamatult. Samas kui $x \in (0; 1]$, kehtib iga astendaja $n \geqslant 1$ korral võrratus $x^n \leqslant 1$. Seega ülesandes 1.10 tõestatud reeglite põhjal

$$\begin{aligned}
&|a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0| \leqslant \\
\leqslant &|a_n x^n| + |a_{n-1}x^{n-1}| + \ldots + |a_2 x^2| + |a_1 x| + |a_0| \leqslant \\
\leqslant &|a_n| \cdot |x^n| + |a_{n-1}| \cdot |x^{n-1}| + \ldots + |a_2| \cdot |x^2| + |a_1| \cdot |x| + |a_0| \leqslant \\
\leqslant &|a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|\,,
\end{aligned}$$

st polünoomi väärtused selles piirkonnas on tõkestatud.

3.5 Mitte eriti hästi. $P(2\pi) \approx -30$, sellal kui $\sin 2\pi = 0$.

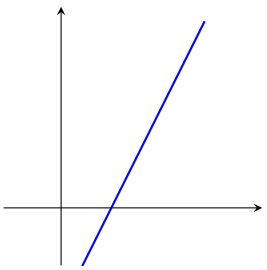
3.6 Koostame Pythoni skripti, mis arvutab lähendpolünoomi P väärtuse sisendil x, arvestades liikmeid astmetega $1{,}3{,}5, \ldots, n$. Seejuures kirjutame eraldi abifunktsiooni avaldise $1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ väärtustamiseks (seda suurust nimetatakse arvu n *faktoriaaliks* ja tähistatakse $n!$). Arvestame ka, et liikmeid kujul $\frac{1}{i!}x^i$ tuleb vaheldumisi liita ja lahutada sõltuvalt sellest, kas i jääk jagamisel 4-ga on 1 või 3. Me küll pole seda formaalselt tõestanud, aga ülesande 3.5 eeskujul GeoGebraga mängides näeme, et viga on kõige suurem lõigu $[-2\pi; 2\pi]$ otspunktides (kusjuures mõlemas otspunktis sama suur). Niisiis võime vea hindamiseks arvutada polünoomi väärtuse kohal $2\pi \approx 6{,}283185307$ ja võrrelda seda arvuga $\sin 2\pi = 0$.

```
def f(n): # faktoriaal
    res = 1
    for i in range(1,n+1):
        res *= i
    return(res)

def P(x,n): # n on maksimaalne aste
    sum = 0
    for i in range(1,n+1,2):
        if i%4 == 1:
            sum += 1/f(i)*x**i
```

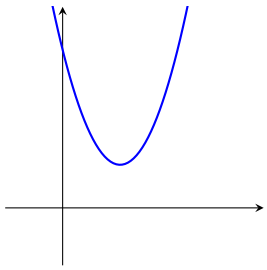
```
        elif i%4 == 3:
            sum -= 1/f(i)*x**i
    return(sum)

for n in range(1,20,2):
    print(n,P(6.283185307,n))
```

Skripti prinditud tulemusest näeme, et viga jääb alla $0\ 011$ siis, kui võtta polünoomi liikmeid vähemalt 17. astmeni.

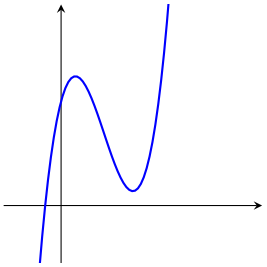
3.7 1.-6. astme polünoomde graafikud näevad välja umbkaudu sellised:



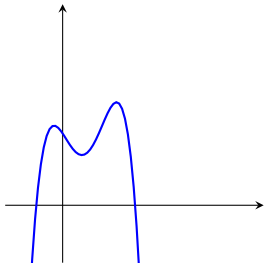
1. astme polünoom



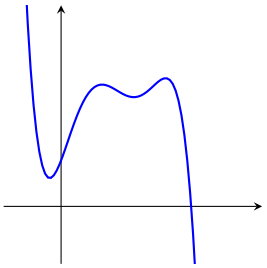
2. astme polünoom



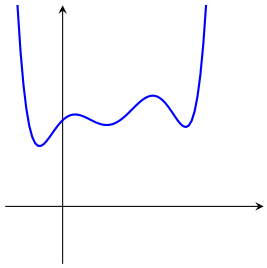
3. astme polünoom



4. astme polünoom



5. astme polünoom



6. astme polünoom

Näeme, et n. astme polünoomi graafik teeb üldjuhul $n-1$ „jõnksu".

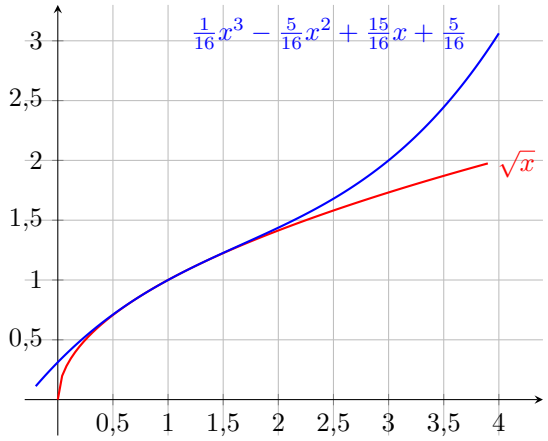
Samuti näeme, et paarisarvulise astmega polünoomi graafiku harud on suunatud ühes suunas, aga paarituarvulise polünoomi graafiku harudest on üks suunatud üles ja teine alla. Kuna polünoomi graafik on pidev, peab paarituarvulise astmega polünoomi puhul leiduma koht, kus tema graafik läbib x-telge. See tähelepanek osutub meile hiljem kasulikuks kursuses „Võrrandid ja võrrandisüsteemid". Paarisarvulise astmega polünoomi puhul pole x-telje läbimine üldjuhul garanteeritud.

Selle, kuhupoole graafiku harud suunatud on, määrab ära pealiikme kordaja märk. Kui pealiikme kordaja on positiivne, suundub graafiku parempoolne haru üles, vastasel juhul aga allapoole. Miks? x-teljel paremale liikudes kasvab muutuja x väärtus kuitahes suureks. See tähendab, et kui väike ka poleks polünoomi pealiikme $a_n x^n$ kordaja a_n absoluutväärtus, kui ta on positiivne, saab $a_n x^n$ lõpuks ikkagi suuremaks kui kogu ülejäänud polünoom. Järelikult hakkab kogu polünoomi väärtus mingist kohast alates järjest kiiremini kasvama. Juhul kui pealiikme kordaja on negatiivne, saame sama moodi arutledes, et mingist kohast alates peab polünoomi väärtus hakkama järjest kiiremini kahanema.

3.8 Tuletame meelde, et $(x-1)^2 = x^2-2x+1$ ja $(x-1)^3 = x^3-3x^2+3x-1$. Seega

$$\begin{aligned} p(x) &= 1+\frac{1}{2}(x-1)-\frac{1}{8}(x-1)^2+\frac{1}{16}(x-1)^3 = \\ &= 1+\frac{1}{2}x-\frac{1}{2}-\frac{1}{8}x^2+\frac{1}{4}x-\frac{1}{8}+\frac{1}{16}x^3-\frac{3}{16}x^2+\frac{3}{16}x-\frac{1}{16} = \\ &= \frac{1}{16}x^3-\frac{5}{16}x^2+\frac{15}{16}x+\frac{5}{16}\,. \end{aligned}$$

3.9 Uuri funktsioonide $x \mapsto \sqrt{x}$ ja $x \mapsto \frac{1}{16}x^3-\frac{5}{16}x^2+\frac{15}{16}x+\frac{5}{16}$ graafikud. Selleks võid kasutada näiteks GeoGebrat või mõnda muud programmi, mis oskab funktsioonide graafikuid joonestada. Kui oled funktsioonid õigesti sisestanud, tuleb pilt umbkaudu selline:



Silma järgi tundub, et vaadeldav polünoom lähendab ruutjuurt hästi umbkaudu vahemikus $[0{,}5;\,2]$. Kontrollime Pythoni abil, kui suur suhteline viga selle lõigu otspunktides on. (Me pole seda küll formaalselt tõestanud, aga selle lõigu sisepunktides on lähendus parem kui otspunktides, niisiis piisab otspunktide uurimisest.)

```
from math import sqrt

def p(x):     #Defineerime polünoomi
    return(1/16*x**3-5/16*x**2+15/16*x+5/16)
```

```
def suhtelineviga(x):
    return(abs((p(x)-sqrt(x))/sqrt(x)))

print(suhtelineviga(0.5))
print(suhtelineviga(1.8))
```

Kohal $x = 0{,}5$ on suhteline viga umbes $0{,}0054$, seega alla 1%, aga kohal $x = 2$ on suhteline viga umbes $0{,}0165$, st $1{,}65\%$. Sama skripti abil saame leida, et suhteline viga kohal $x = 1{,}8$ on umbes $0{,}0077$, mis jääb allapoole lubatud veapiiri. Niisiis sobib selle ülesande vastuseks näiteks vahemik $[0{,}5; 1{,}8]$. (See vastus pole päris täpne ja tegelikult sobib ka veidi suurem vahemik, aga meile piisab hetkel sellest.)

3.10 (a) Proovime kasutada polünoomi $p(x)$ poolt antud lähendit. Kahjuks ei klapi vahemik: $a \in [1; 10)$, aga ülesande 3.10 põhjal lähendab $p(x)$ ruutjuurt hästi ainult vahemikus $[0{,}5; 1{,}8]$.

Õnneks saab sisendit teisendada. Kui $a \in [1; 10)$, siis $\frac{a}{2} \in [0{,}5; 5)$. See pole veel täpselt see, mis vaja, aga sinnapoole. Kui võtta $a \in [1; 3)$, siis $\frac{a}{2} \in [0{,}5; 1{,}5)$, mis tähendab, et vahemikus $[1; 3)$ on plaan selge: jagame sisendi 2-ga ja rakendame tulemusele polünoomi p.

Mida teha, kui $a \in [3; 10)$? Paneme tähele, et sel juhul $\frac{a}{6} \in [0{,}5; 1{,}(6))$ ja kuna $1{,}(6) < 1{,}8$, mahume nüüd ära vahemikku, kus polünoom p ruutjuurt hästi lähendab.

Veel tuleb välja mõelda, kuidas polünoomi väljundist $\sqrt{a}$ kätte saada. Paneme tähele, et vahemikus $a \in [1; 3)$

$$\sqrt{a} = \sqrt{2 \cdot \frac{a}{2}} = \sqrt{2} \cdot \sqrt{\frac{a}{2}} \approx 1{,}414213562 \cdot p\left(\frac{a}{2}\right)$$

ja vahemikus $a \in [3; 10)$

$$\sqrt{a} = \sqrt{6 \cdot \frac{a}{6}} = \sqrt{6} \cdot \sqrt{\frac{a}{6}} \approx 2{,}449489743 \cdot p\left(\frac{a}{6}\right).$$

Kokkuvõttes saame järgmise programmijupi (mis kasutab funktsiooni `p(x)` definitsiooni ülesandest 3.9):

```
def ruutjuur_a(a):  # Eeldame, et 1 <= a < 10
    if a<3:
        return(1.414213562*p(a/2))
    else:
        return(2.449489743*p(a/6))
```

(b) Kui b on paarisarv, siis saame ruutjuure arvutada kujul $\sqrt{10^b} = 1 \cdot 10^{\frac{b}{2}}$.

Kui b on paaritu arv, on $b - 1$ paaris ja seega võime arvutada

$$\sqrt{10^b} = \sqrt{10 \cdot 10^{b-1}} = \sqrt{10} \cdot \sqrt{10^{b-1}} \approx 3{,}16227766 \cdot 10^{\frac{b-1}{2}}.$$

Kokkuvõttes saame koodijupi

```
def ruutjuur_b(b):   # Eeldame, et b on täisarv
    if b%2 == 0:
        return(1,b//2)
    else:
        return(3.16227766,(b-1)//2)
```

3.11 Trükime näiteks välja suhtelised vead kohtadel $1; 1{,}1; 1{,}2; \ldots; 9{,}9$:

```
def suhtelineviga(x):
    return(abs((ruutjuur_a(x)-sqrt(x))/sqrt(x)))

x=1
while x<10:
    print(x,suhtelineviga(x))
    x+=0.1
```

Selle programmi väljundist näeme, et suhteline viga jääb alla 1% (tuleta meelde, et Python esitab nullilähedasi arve standardkujul, st näiteks $0{,}0000814026729413321$ kohta kirjutab Python `8.14026729413321e-05`). Seda tulemust oligi oodata, sest me ju ise valisime lähendi nii, et viga alla 1% jääks.

Funktsiooni `ruutjuur_a` suhtelise vea kohal x saame siis näiteks käsuga `suhtelineviga(x,ruutjuur_a)`.

3.12 Olgu reaalarv x esitatud standardkujul $x = a \cdot 10^b$, kus $1 \leqslant a < 10$, $b \in \mathbb{Z}$. Funktsioon `ruutjuur_a(a)` väljastab meile $\sqrt{a}$ ligikaudse väärtuse, mis jääb ligikaudu vahemikku $[1; \sqrt{10})$. Funktsioon `ruutjuur_b(b)` väljastab paari (a_1, b_1), kus b_1 on täisarv, a_1 aga on kas 1 või $\sqrt{10}$. See tähendab, et $a_1 \cdot \sqrt{a}$ jääb (ligikaudu) vahemikku $[1; 10)$, st ta sobib vastuse tüveks. Vastuse eksponendiks sobib järelikult b_1 ning kood tuleb kokkuvõttes lihtne:

```
def ruutjuur_ab(a,b):
    a1,b1 = ruutjuur_b(b)
    return(a1*ruutjuur_a(a),b1)
```

3.13 Sobib näiteks niisugune kood:

```
def suhtelineviga(x,f):
    return(abs((f(x)-sqrt(x))/sqrt(x)))
```

3.14 Esimesel pilgul tundub, et see ülesanne ei saa ju eriti raske olla, aga siis tuleb välja, et funktsioon `suhtelineviga` ootab sisendiks ühe muutuja funktsiooni `f`, funktsioon `ruutjuur_ab` aga töötab kahel muutujal, eeldades sisendiks reaalarvu standardkuju tüve ja eksponenti eraldi.

Niisiis tuleb kõigepealt kirjutada teisendusfunktsioon `standardkuju`, mis väljastab antud reaalarvu tüve ja eksponendi. Selle leidmiseks on olemas mitu võimalust. Näiteks võime kasutada meetodit `format`, mille abil saame Pythonit sundida reaalarve standardkujul esitama[2]:

[2]Peale meetodi `format` võime kasutada ka logaritmi, millega tutvume kursuses „Eksponent- ja logaritmfunktsioon".

```
>>> "{:e}".format(2.5)
'2.500000e+00'
```

Edasi võime (stringina!) esitatud tulemuse tähe `e` kohalt kaheks tükiks jagada ning väljastada esimese poole reaalarvuna, teise poole aga täisarvuna. Kokkuvõtteks võib vastav funktsioon välja näha näiteks selline:

```
def standardkuju(x):
    stkuju = "{:e}".format(x)
    tyvi,eksponent = stkuju.split('e')
    t = float(tyvi)
    e = int(eksponent)
    return(t,e)
```

Nüüd saame defineerida funktsiooni `ruutjuur` ja seda katsetada. Ära unusta, et funktsiooni `ruutjuur` väljund tuleb standardkujult jälle üheks reaalarvuks teisendada!

```
def ruutjuur(x):
    a,b = standardkuju(x)
    a1,b1 = ruutjuur_ab(a,b)
    return(a1*10**b1)

for i in [2,3,5,7,11,13,17,19]:
    print(i,ruutjuur(i),suhtelineviga(i,ruutjuur))
```

3.15 Funktsioon `eval1` kasutab x^i arvutamiseks $i-1$ korrutamist, lisaks tuleb iga üksliikme kohta veel üks korrutamine kordajaga a_i. Üksliikmete liitmiseks läheb vaja ka $n+1$ liitmist (sest väärtustamist alustatakse väärtusest 0, millele tuleb kõigepealt liita ka 0. astme üksliige).

Niisiis kulub 3. astme polünoomi väärtustamiseks $1+2+3+4=10$ korrutamist ja 4 liitmist, 5. astme polünoomi korral aga $1+2+3+4+5+6=21$ korrutamist ja 6 liitmist.

n. astme polünoomi leidmiseks on vaja $n+1$ liitmist ja $1+2+\ldots+n+(n+1)$ korrutamist. Kas viimast avaldist saab kuidagi kompaktsemalt esitada? Üldjuhul õpime niisuguste summade arvutamist kursuses „Funktsioonid. Arvjadad", aga praegu võime tähele panna, et kui $S=1+2+\ldots+n+(n+1)$, siis

$$\begin{aligned} 2S &= 1+2+\ldots+n+(n+1)+(n+1)+n+\ldots+2+1= \\ &= (1+(n+1))+(2+n)+\ldots+(n+2)+((n+1)+1)= \\ &= (n+1)(n+2)\,, \end{aligned}$$

seega $S=\frac{(n+1)(n+2)}{2}$.

3.16

3.17 Vastus: 2014.

Jagame polünoomi $x^5+2x^2-4x+2010$ jäägiga polünoomiga x^3+2x+2:

$$\begin{array}{rrrr} x^5 & +2x^2 & -4x+2010 = (x^3+2x+2)(x^2-2)+2014 \\ \underline{-x^5} & \underline{-2x^3} & \underline{-2x^2} & \\ & -2x^3 & -4x+2010 & \\ & \underline{2x^3} & \underline{+4x} & \underline{+4} \\ & & & 2014 \end{array}$$

Kuna ülesande tingumuste põhjal $x^3+2x+2=0$, saame

$$x^5+2x^2-4x+2010 = (x^3+2x+2)(x^2-2)+2014 = 0\cdot(x^2-2)+2014 = 2014\,.$$

3.19 Vastus: sobivad $n=-7$ ja $n=3$.

Ülesande ruutkolmliikme tegurdamiseks vaatleme n-i reaalarvulise muutujana ja lahendame ruutvõrrandi $4n^2+16n-65=0$. Ruutvõrrandi lahendivalem annab

$$n_{1,2} = \frac{-16\pm\sqrt{(-16)^2-4\cdot 4\cdot(-65)}}{2\cdot 4} = \frac{-16\pm\sqrt{1296}}{8} = \frac{-16\pm 36}{8}\,.$$

Niisiis $n_1=-\frac{13}{2}$ ja $n_2=\frac{5}{2}$, mistõttu teoreemist 3.1

$$4n^2+16n-65 = 4\cdot\left(n+\frac{13}{2}\right)\left(n-\frac{5}{2}\right) = (2n+13)(2n-5)$$

iga reaalarvu, aga järelikult ka iga täisarvu n jaoks.

Korrutis $(2n+13)(2n-5)$ saab täisarvulise n korral anda algarvu ainult siis, kui $2n+13=\pm 1$ või $2n-5=\pm 1$. Võimalikud n väärtused on seega $-7, -6{,}2$ ja 3. Leiame uuritava avaldise väärtuse nende n-ide jaoks.

$$\begin{aligned} (2\cdot(-7)+13)(2\cdot(-7)-5) &= (-1)\cdot(-19) = 19\,, \\ (2\cdot(-6)+13)(2\cdot(-6)-5) &= 1\cdot(-17) = -17\,, \\ (2\cdot 2+13)(2\cdot 2-5) &= 17\cdot(-1) = -17\,, \\ (2\cdot 3+13)(2\cdot 3-5) &= 19\cdot 1 = 19\,. \end{aligned}$$

Kuna -17 pole algarv, sobivad ainult $n=-7$ ja $n=3$.

3.20 Vaatleme jälle polünoomi

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x^1 + a_0\,.$$

Olgu -1 selle polünoomi juureks, st

$$0 = P(-1) = a_n\cdot(-1)^n + a_{n-1}\cdot(-1)^{n-1} + \ldots + a_1\cdot(-1)^1 + a_0\,.$$

Paneme tähele, et jadas $(-1)^n, (-1)^{n-1}, (-1)^{n-2}, \ldots, (-1)^1, (-1)^1$ esinevad väärtused 1 ja -1 vaheldumisi, seega on avaldis

$$a_n\cdot(-1)^n + a_{n-1}\cdot(-1)^{n-1} + \ldots + a_1\cdot(-1)^1 + a_0$$

polünoomi P kordajate vahelduvate märkidega summa. Niisiis on -1 polünoomi juureks parajasti siis, kui selle polünoomi kordajate vahelduvate märkidega summa on 0.

Indeks