PYTHON-OHJELMOINNIN SOVELLUKSIA MATRIISILASKENTAAN, TILASTOTIETEESEEN JA NUMEERISEEN MATEMATIIKKAAN LUKIOLAISILLE

Tiina Willberg-Laine

Pro gradu -tutkielma
Marraskuu 2008

MATEMATIIKAN LAITOS
TURUN YLIOPISTO

TURUN YLIOPISTO
Matematiikan laitos

WILLBERG-LAINE, TIINA: Python-ohjelmoinnin sovelluksia matriisilaskentaan, tilastotieteeseen ja numeeriseen matematiikkaan lukiolaisille
Pro gradu -tutkielma, 76 s., 0 liites.
Matematiikka
Marraskuu 2008

Tämä tutkielma on laadittu oppimateriaaliksi lukion kurssille, jossa käsitellään Python-ohjelmointikielen käyttöä matriisilaskennan, tilastotieteen ja numeerisen matematiikan osa-alueilla. Tutkielma soveltuu myös itseopiskelumateriaaliksi, sillä kaikkiin harjoitustehtäviin on esitetty ratkaisut. Lähestymistapa on valittu lukiolaiselle sopivaksi.

Ensin kussakin osiossa käsitellään teoria, jotta lukijalla olisi mahdollisuus käsittää, mitä ohjelmissa tapahtuu. Kunkin teoriaosuuden jälkeen on käsitelty kyseisen osa-alueen Python-sovelluksia. Sekä teorian että Python-sovellusten yhteydessä on runsaasti esimerkkejä.

Tämän materiaalin avulla huomataan, kuinka yksinkertainen mutta monipuolinen ohjelmointikieli Python on.

Asiasanat: Python-ohjelmointikieli, matriisilaskenta, tilastotiede, numeerinen matematiikka, koulumatematiikka.

Sisältö

1 Johdanto

Tämä tutkielman tarkoituksena on toimia oppimateriaalina lukiokurssilla, jossa käsitellään Python-ohjelmointikielen käyttöä matriisilaskennan, tilastotieteen sekä numeerisen matematiikan osa-alueilla. Kukin osa-alue käsitellään omana lukunaan. Toki materiaali soveltuu kaikille, jotka aiheesta ovat kiinnostuneet.

Python-kieli on vapaan lähdekoodin ohjelmointikieli. Yksinkertaisen syntaksinsa vuoksi se soveltuu loistavasti myös ohjelmoinnin opetteluun. Python on ladattavissa osoitteesta `www.python.org`. Sieltä löytyvät myös asennusohjeet.

Materiaalissa tärkeimpiä kirjastoja ovat NumPy-kirjasto sekä Matplotlib-kirjasto. NumPy-kirjaston asennusohjeet sekä lisätietoa kirjaston laajemmasta käytöstä löytyy osoittesta `http://numpy.scipy.org`. Matplotlib-kirjaston kotisivut löytyvät puolestaan osoittesta `http://matplotlib.souceforge.net`.

Esitietoja ohjelmoinnista lukijalta ei odoteta. Sen vuoksi alkuun on kasattu välttämättömät perustiedot Pythonilla ohjelmoimisesta. Toki materiaalin lukemista helpottaa, jos lukijalle on jo kertynyt ohjelmointikokemusta. Matematiikasta toki oletetaan tiettyjä perustietoja. Lukiokursseista olisi hyvä olla käytynä vektorilaskennan, tilastotieteen sekä numeerisen matematiikan kurssit. Tämäkään ei ole välttämätöntä, sillä jokainen luku alkaa aina teorian käsittelyllä. Toki teoria käydään hyvin pintapuolisesti, joten syvempään ymmärtämiseen suositellaan lisäopiskelua. Esimerkiksi lauseita ei tässä materiaalissa ole todistettu, koska se ei ole työn tarkoituksen kannalta olennaista. Lisäksi monet todistuksista ovat matemaattisesti liian hankalia ottaen huomioon, että materiaali on suunnattu lukio-opetukseen. Toki pieni perustelu jokaiselle esitetään.

Jokaisen kappaleen lopussa on muutama harjoitustehtävä, koska matematiikkaa tai ohjelmointia ei opi muuten kuin itse tekemällä. Lopussa on osio, johon harjoitustehtävien ratkaisut on koottu. Jos tutkielmaa käytetään kurssin oppimateriaalina, voi opettaja halutessaan jättää ratkaisut pois. Ratkaisujen kera materiaali toimii myös itseopiskelussa.

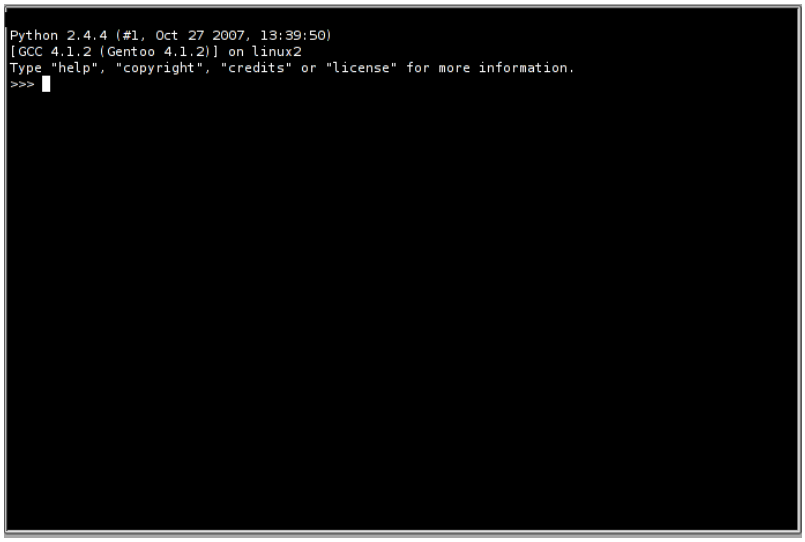
Tekijän oma osuus työssä on toki materiaalin kokoaminen ja muotoileminen oppimateriaalin muotoon. Kaikki esimerkit ja harjoitustehtävät lukuunottamatta ylioppilastehtäviä ovat tekijän itse laatimia sekä ratkaisemia.

2 Python-ohjelmoinnin perusteet

Ensiksi tutustutaan Python-ohjelmoinnin perusteisiin. Päälähteenä tässä luvussa on käytetty Kasurisen Python-ohjelmointiopasta [1] sekä Dive Into Python-opasta [2]. Näistä löytyy lisää tietoa Python-kielestä siitä kiinnostuneille.

2.1 Ohjelmoinnin aloittaminen

Pythonissa on käytössä komentorivitulkki, joka näyttää tältä.



Kuva 1: Pythonin komentorivitulkki

Komentorivitulkin saa käyntiin kirjoittamalla komentoriville käskyn `python`. Lyhyitä ohjelmia voi kirjoittaa suoraan komentorivitulkkiin. Komentorivitulkki suljetaan näpppäinyhdistelmällä `crtl+a+d`.

Esimerkki 2.1.

```
>>> print 'Hei maailma!'
>>> Hei maailma!
```

Tässä esimerkissä tulkille annettiin käsky print, joka siis tarkoittaa, että tulkki tulostaa sen, mitä lainausmerkkien sisään on kirjoitettu. Kun käsky on kirjoitettu, painetaan enteriä, jolloin tulkki siis toimii kirjoitettujen käskyjen mukaan. Tässä tapauksessa siis tulostaa ruudulle lauseen Hei maailma!.

Käyttökelpoiset ohjelmat saattavat olla koodiltaan hyvinkin pitkiä. Silloin ohjelmien kirjoittaminen suoraan tulkkiin käy hankalaksi. Yleensä ohjelmien kirjoittamiseen käytetään lähdekooditiedostoja. Koodi siis kirjoitetaan

omaksi tekstitiedostokseen. Näiden tekstitiedostojen kirjoittamiseen käytetään jotakin tekstieditoria. Tekstitiedosto tallennetaan päätteellä `.py`, joilloin se tunnistetaan python-koodiksi. Ohjelma ajetaan kirjoittamalla komentoriville `python tiedostonnimi.py`. Huomaa, että komentorivitulkkia ei siis käynnistetä ennen ajoa.

Ennen ohjelmoinnin aloittamista on myös hyvä tietää muutamia seikkoja Pythonin syntaksista.

Kommentit ovat ohjelmointikielissä rivejä, jotka eivät vaikuta ohjelman suoritukseen mitenkään. Niiden tarkoitus on vain selkeyttää koodia. Ne voivat olla esimerkiksi selvennyksiä jollekulle muulle, joka koodia lukee tai sitten ohjelmoijan omia muistiinpanoja. Pythonissa kommentit merkitään risuaitamerkillä (#).

Esimerkki 2.2.

```
#Tämä on kommenttirivi
Tämä rivi taas on ohjelmakoodia
```

Tärkeää on muistaa, että Pythonissa välilyönti rivin alussa on merkitsevä merkki. Siis jos paperilla annetussa esimerkissä on jokin rivi sisennettynä, pitää se sisentää myös koodia kirjoittaessa, jotta koodi toimii. Myös isot ja pienet kirjaimet ovat Pythonissa eri asia. Tämä on hyvä muistaa esimerkiksi muuttujia nimettäessä. On myös tärkeää muistaa, että käskyt kirjoitetaan Pythonissa pienellä.

Ohjelmointikielissä, myös Pythonissa, käytetään muuttujia. Muuttujat ovat tiedonsäilytyspaikkoja, joiden sisältö nimensä mukaisesti muuttuu ohjelman suorituksen aikana. Lausekkeiksi sanotaan koodirivejä, joissa muuttujille asetetaan jokin arvo. Asetus tapahtuu yhtäsuuruusmerkin avulla.

Tässä materiaalissa ja muuallakin käytetään paljon `input`-komentoa, joten se on hyvä tuntea. Kaikessa yksinkertaisuudessaan `input`-komennon avulla muuttujan sisällöksi asetetaan se, mitä käyttäjä syöttää. Myöhemmin tutustutaan Pythonin tietotyyppeihin. `Input`-komennnon yhteydessä on hyvä muistaa, että itse `input` huolii syötteenä vain kokonaislukuja. Merkkijonoja varten käytetään komentoa `raw_input`.

Esimerkki 2.3.

```
nimi=raw_input('Anna etunimesi:')
```

Tällä rivillä siis ohjelma kysyy käyttäjältä tämän etunimeä ja muuttujaan nimi tallennetaan käyttäjän antama nimi.

Edellisessä esimerkissä voisi ohjelman suorituksessa olla myöhemmin rivi `nimi=raw_input('Anna toinen nimesi:')`, jolloin muuttujan nimi sisältö vaihtuisi annetusta etunimestä toiseksi nimeksi.

Muuttujia voidaan asettaa myös useita samalla kertaa. Silloin asetuslauseessa erotellaan muuttujat ja niiden saamat arvot pilkulla.

Harjoituksia

Tehtävä 1. *Kirjoita seuraava koodi lähdekooditiedostoon ja aja se komentoriviltä. Mitä ohjelma tekee?*

```
luku1=input('Anna jokin kokonaisluku:')
luku2=input('Anna vielä kokonaisluku:')
tulos=luku1+luku2
print 'Antamiesi lukujen summa on', tulos
```

2.2 Peruslaskutoimituksia Pythonilla

Komentorivitulkkia voi käyttää laskimen tapaan peruslaskutoimituksien suorittamiseen. Tästä on hyötyä erityisesti silloin, kun lasketaan hyvin suurilla luvuilla. Laskutoimitukset kirjoitetaan suoraan komentorivitulkkiin. Alla olevassa taulukossa on esitelty käytössä olevat operaattorit.

Taulukko 1: Pythonin operaattorit peruslaskutoimituksille

Operaattori	Nimi	Selitys
+	Plus	laskee yhteen kaksi lukua
−	Miinus	palauttaa luvun vastaluvun tai vähentää luvut toisistaan
*	Tulo	kertoo kaksi lukua keskenään
**	Potenssi	laskee potenssin
/	Jako	jakaa luvut keskenään
%	Jakojäännös	palauttaa jakojäännöksen

Esimerkki 2.4. *Alla on listattu muutamia laskutoimituksia esimerkiksi operaattoreiden käytöstä*

```
>>> 5+3
8
>>> 5-3
2
>>> 5*3
15
```

```
>>> 5**3
125
>>> 10/2
5
>>> # Palauttaa, mikä on jakojäännös kun viisi jaetaan kolmella
...5%3
2
```

Pythonilla voidaan tehdä myös vertailuja. Vertailut palauttavat arvon True tai False riippuen siitä, pitääkö vertailu paikkansa. Allaolevassa taulukossa on listattu vertailuoperaattorit.

Taulukko 2: Pythonin vertailuoperaattorit

Operaattori	Nimi	Selitys
$<$	Pienempi kuin	Palauttaa tiedon, onko luku pienempi kuin toinen
$>$	Suurempi kuin	Palauttaa tiedon, onko luku suurempi kuin toinen
$<=$	Pienempi tai yhtä suuri kuin	Palauttaa tiedon, onko luku pienempi tai yhtä suuri kuin toinen
$>=$	Suurempi tai yhtä suuri kuin	Palauttaa tiedon, onko luku suurempi tai yhtä suuri kuin toinen
$==$	Yhtä suuri kuin	Palauttaa tiedon, ovatko kaksi lukua yhtä suuret
$!=$	Eri suuri kuin	Palauttaa tiedon, ovatko kaksi lukua eri suuret

Esimerkki 2.5. *Alla on listattu muutamia vertailuja esimerkiksi vertailuoperaattoreiden käytöstä*

```
>>> 5<=3
False
>>> 5>=3
True
>>> 5==3
False
>>> 5!=3
True
```

Operaatioiden suoritusjärjestys on sama kuin matematiikassakin. Kerto- ja jakolasku suoritetaan ennen yhteen- ja vähennyslaskua sekä laskuoperaattorit ennen vertailuoperaattoreita. Sulkeilla voi muuttaa laskujärjestystä ja niitä voi käyttää myös selkeyttämään koodia. Ylenmääräinen sulkeiden käyttö kuitenkin tekee koodista hankalasti luettavaa.

2.3 Tietotyypit

Pythonista löytyy neljä erilaista tietotyyppiä numeroille, kokonaisluvut, liukuluvut, pitkät kokonaisluvut sekä kompleksiluvut. Kompleksiluvut voidaan tässä yhteydessä sivuuttaa. Lisäksi on hyvä tuntea myös tietotyypit merkkijono ja lista.

Kokonaisluvut (`int`) ovat Pythonissa aivan sama käsite kuin matematiikassakin. Ne ovat siis lukuja, jotka eivät sisällä desimaaliosaa. Pitkät kokonaisluvut (`long`) ovat taas hyvin suuria kokonaislukuja. Liukuluvut (`float`) puolestaan ovat lukuja, joita matematiikassa kutsutaan desimaaliluvuiksi.

Tietotyyppien kanssa on syytä olla tarkkana laskutoimituksia tehtäessä. Esimerkiksi kun tehdään toimitus kokonaislukuina, voi tuloksena olla ainoastaan kokonaislukuja. Toisaalta kokonaislukuja, pitkiä kokonaislukuja ja desimaalilukuja voidaan käyttää samoissa laskutoimituksissa. Python antaa tuloksen automaattisesti sopivassa muodossa.

Esimerkki 2.6. *Lasketaan $\frac{6}{8}$ Pythonilla.*

```
>>> 6/8
0
```

Koska luvut on annettu kokonaislukuina, Python antaa tuloksen kokonaislukuna. Tarkemmin sanoen tulkki palauttaa osamäärän kokonaisosan eli tässä tapauksessa nollan. Kun laitetaan luvut 6 ja 8 desimaaliluvuiksi, saadaan oikea tulos, koska silloin Python antaa tuloksen desimaalilukuina.

```
>>> 6.0/8.0
0.75
```

Toisaalta riittää antaa vain toinen luku desimaalimuodossa, koska Python muuttaa tuloksen automaattisesti sopivaan muotoon.

```
>>> 6.0/8
0.75
```

Merkkijonot (`str`) ovat nimensä mukaisesti jono merkkejä. Merkkijonoja voidaan käyttää esimerkiksi sanojen tallentamiseen. Merkkijonoissa voidaan käyttää kaikkia merkkejä, isoja ja pieniä kirjaimia, numeroita, skandinaavisia merkkejä ja jopa erikoismerkkejä. Merkkijono kirjoitetaan aina sitaattien sisään. Sitaatit voivat olla joko yksinkertaisia (') tai kaksinkertaisia (“). Erikoismerkkejä käytettäessä on siis huomiotava, että mikäli merkkijonossa esiintyy sitaatti, pitää sen eteen laittaa kenoviiva (\). Mikäli merkkijonon alkamista ja päättymistä merkitään yksinkertaisilla sitaateilla, kaksinkertaisten eteen ei tarvitse laittaa kenoviivaa ja toisinpäin.

Lista on indeksoitu jono muuttujia. Jokaisella listan alkiolla on oma tietotyyppinsä, joten samassa listassa voi olla niin kokonaislukuja kuin merkkijonojakin. Lista luodaan hakasulkeiden sisään ja alkiot erotetaan toisistaan pilkulla.

Sekä merkkijonon että listan pituuden saa selville komennolla `len` ja yksittäisiin alkioihin päästään käsiksi hakasulkeilla. Alkioihin viitattaessa pitää muistaa, että indeksointi alkaa nollasta.

Esimerkki 2.7.

```
>>> #luodaan kolmen alkion lista a
>>> a=[5,"kissa",2.5]
>>> print a
[5, 'kissa', 2.5]
>>> #selvitetään listan pituus
>>> len(a)
3
>>> a[1]
'kissa'
```

Tässä esimerkissä on siis luotu lista a. Huomionarvoista on, että listan alkiot voivat olla mitä tietotyyppiä tahansa eikä niiden tarvitse edes olla keskenään samaa tietotyyppiä. Lisäksi on haettu listan pituus sekä palautettu toinen alkio.

2.4 Ehto- ja toistorakenteet

Useimmissa ohjelmointikielissä, myös Pythonissa, on mahdollisuus haarauttaa koodia. Jokin ohjelmanpätkä suoritetaan siis vain jos annettu ehto täyttyy.

Pythonissa käytetään if-else-rakennetta. If-osiossa annetaan jokin väittämä. Jos väittämä on tosi, suoritetaan koodista se osa, joka on kirjoitettu if-osioon. Muuten jatketaan else-osioon. Else-osiota ei kuitenkaan ole pakko kirjoittaa. Siis jos if-rakenteessa ei ole else-osiota, jatketaan väittämän jäätyä toteutumatta koodin suorittamista heti if-osion jälkeen.

Pythonissa on käytössä myös elif-osio. Niitä voidaan laittaa if-osion jälkeen vaikka kuinka monta. Elif-osiossa annetaan uusi ehto, joka määrittelee suoritetaanko kyseinen osio. Elif-osioon mennään siis vain jos if-osion tai minkään edellisen elif-osion väittämä ei toteudu. Elif-osion nimen voi ajatella olevan lyhenne sanoista else ja if. Voi siis ajatella, että else osiossa tuleekin nyt uusi if-osio. Else-osion tavoin elif-osiota ei ole pakko laittaa if-osion jälkeen.

Esimerkki 2.8.

```
luku=input('Anna jokin kokonaisluku:')
if luku%2==0:
        print 'Antamasi luku on parillinen'
elif luku%3==0:
```

```
        print 'Antamasi luku on pariton, mutta jaollinen kolmella'
else:
        print 'Antamasi luku on pariton'
```

Ohjelma siis kysyy käyttäjältä jonkin kokonaisluvun. If-lauseella on nyt väittämä, joka testaa, mikä on muuttujan luku jakojäännös kahdella jaettaessa. Mikäli se on nolla eli luku on jaollinen kahdella, tulostetaan teksti 'Antamasi luku on parillinen'. Elif-osiossa testataan samoin, onko luku jaollinen kolmella. Huomaa, että elif-osioon ei mennä, mikäli if-osion väittämä toteutuu. Jos kumpikaan väittämistä ei toteudu, tulostetaan ruudulle teksti 'Antamasi luku on pariton'. Else-osioonkaan ei mennä, jos jompi kumpi edellisistä osioista suoritetaan.

Ohjelmointikielissä on rakenteiteta, joiden avulla tiettyä osaa ohjelmaa voidaan toistaa tarpeellinen määrä kertoja. Toistettavaa ohjelmaosaa sanotaa silmukaksi. Pythonissa on kaksi erilaista toistorakennetta, while- ja for-rakenteet.

While-rakenteessa tiettyä osaa ohjelmasta toistetaan niin kauan kun annettu ehto on voimassa. While-rakenteeseen voidaan liittää if-rakenteen tavoin else-haara. Tämä haara kertoo, mitä tehdään, jos while-silmukkaa varten annettu ehto ei täytykään ja silmukkaan ei mennä laisinkaan. While-rakenteen heikkous on, että etukäteen ei tiedetä, kuinka monta kertaa silmukka tullaan käymään läpi. Ei voida olla edes varmoja loppuuko silmukka koskaan eli loppuuko ehdon voimassaolo.

For-rakenteessa määritellään etukäteen kuinka monta kertaa silmukka toistetaan. Useimmiten for-rakennetta käytetään esimerkiksi listojen läpikäymiseen. Myös for-rakenteeseen on mahdollista liittää else-haara if- ja while-rakenteiden tavoin. Toistojen määrä määritellään range-funktion avulla.

Range-funktio on hyvin yksinkertainen Pythonin sisäänrakennettu funktio. Range-funktiolle riittää antaa parametriksi vain käytettävän lukujonon alkioiden määrä. Tällöin oletuksena lukujono alkaa nollasta ja jatkuu niin monta kokonaislukua eteenpäin kun arvoksi on annettu. Jos ei haluta aloittaa nollasta, voidaan range-funktiolle antaa argumentiksi myös aloitusluku. Silloin mennään annetusta luvusta niin monta kokonaislukua eteenpäin kun on annettu. Jos taas halutaan mennä enemmän kuin yksi luku kerrallaan, sekin voidaan antaa argumentiksi range-funktiolle. Silloin mennään annetusta aloitusluvusta annettuja askeileita niin monta kuin on määrätty annettuun ylärajaan asti.

Esimerkki 2.9.

```
>>>range(5) #tulostaa kokonaisluvut nollasta neljään
[0, 1, 2, 3, 4]
>>>range(3,11) #tulostaa kokonaisluvut kolmesta kymmeneen
[3, 4, 5, 6, 7, 8, 9, 10]
>>>#tulostaa joka kolmannen kokonaisluvun miinus viidestä
```

```
>>>#viiteentoista
>>>range(-5, 16, 3)
[-5, -2, 1, 4, 7, 10, 13]
```

Seuraavassa esimerkissä käydään läpi toistorakenteiden käyttöä.

Esimerkki 2.10. *Alla on kahdella tavalla kirjoitettu ohjelma, joka tulostaa annetun luvun kertoman. Ensiksi toistorakenteena on käytetty while-rakennetta.*

```
#pyydetään käyttäjältä positiivinen kokonaisluku
luku=input('Anna jokin positiivinen kokonaisluku:')
#alustetaan apumuuttuja arvoksi 1
apu=1
#alustetaan muuttuja, johon tulos tallennetaan
#arvoksi 1
tulos=1
#niin kauan kun muuttujan apu arvo pienempää tai yhtäsuurta
#kuin annettu luku kerrotaan muuttuja tulos ja apu keskenään
# ja tallennetaan tulos muuttujaan tulo
while apu <= luku:
        tulos=tulos*apu
#kasvatetaan lopuksi muuttujaa apu yhdellä
         apu=apu+1
#tulostetaan saatu tulos
print tulos
```

Alla on sama ohjelma kirjoitettuna for-rakenteen avulla.

```
#pyydetään käyttäjältä positiivinen kokonaisluku
 luku=input('Anna jokin positiivinen kokonaisluku:')
#alustetaan muuttuja, johon tulos tallennetaan
tulos=1
#kerrotaan keskenään apumuuttuja ja tulosmuuttuja apumuuttujan
#arvolla [1,luku] ja tallenetaan tulos muuttujaan tulos
for apu in range(1,luku+1):
        tulos=tulos*apu
#tulostetaan saatu tulos
print tulos
```

Harjoituksia

Tehtävä 2. *Kirjoita seuraava koodi tekstieditoriin. Mitä ohjelma tekee ? Kiinnitä erityistä huomiota for-rakenteen käyttöön.*

```
kerrottava=input('Anna jokin kokonaisluku väliltä [1,10]:')
for i in range(1,11):
        tulos=kerrottava*i
        print kerrottava, '*', i, '=', tulos
```

Tehtävä 3. *Kirjoita edellisen tehtävän koodi while-rakenteen avulla.*

Tehtävä 4. *Kirjoita ohjelma, joka kysyy käyttäjältä lukua ja kertoo onko luku jaollinen kuudella.*

Tehtävä 5. *Tulosta range-funktion avulla parilliset positiiviset luvut, jotka ovat pienempiä tai yhtä suuria kuin 40.*

2.5 Funktiot ja kirjastot

Ohjelmoijan työtä helpottaa suuresti niin sanottu uudelleenkäyttettävyys. Uudelleenkäytettävyydellä tarkoitetaan sitä, että kun kerran kirjoitetaan jokin ohjelmanpätkä, niin sitä voidaan käyttää aina uudelleen ja uudelleen. Siis joka kerta kun kyseisen ohjelmanpätkän suorittamaa toimintoa tarvitaan ei tarvitse kirjoittaa koodia uudestaan, vaan voi käyttää jo kirjoittamaansa ohjelmaa.

Pythonissa uudelleenkäytettävyys tulee esille funktioiden avulla. Funktiot ovat ohjelmia, jotka tekevät aina jonkin asian. Funktioita voidaan kutsua muista ohjelmista, joten samaa funktioita voidaan käyttää useita kertoja.

Koodissa funktio aloitetaan avainsanalla `def`. Näin komentotulkki tietää, että kyseessä on funktio. Funktiota kutsutaan yksikertaisesti kirjoittamalla sen nimi.

Funktiolle voidaan antaa kutsun yhteydessä eri parametrejä. Parametrit ovat muuttujia, jotka käyttäjä voi funktiokutsun yhteydessä määrätä ja näin saada ohjelman toimimaan haluamallaan tavalla. Toistorakenteiden yhteydessä käsitteltiin range-funktiota. Kutsun yhteydessä annettiin parametriksi muun muassa aloitusluku, jolloin ohjelma aloitti lukujonon tekemisen halutusta kohdasta.

Esimerkki 2.11.

```
#testaa onko annettu luku parillinen
def onkoParillinen(luku):
        #muuttujaan parillinen tallennetaan
        #totuusarvo sen mukaan onko luku
        #parillinen, oletuksena arvo False
        parillinen=False
```

```
    if luku%2==0 :
        parillinen=True
    return parillinen
```

Tässä on tehty funktio, joka yksinkertaisesti testaa, onko luku parillinen vai ei. Joka kerta kun tarvitaan tietoa luvun pariteetista, niin siihen voidaan käyttää tätä funktiota. Funktion parametrina on siis testattava luku. Funktio on tallennettu nyt lähdekooditiedostoon nimeltä parillinen.py. Testataan funktion toimintaa.

```
>>> import parillinen
>>> parillinen.onkoParillinen(33)
False
>>> parillinen.onkoParillinen(102)
True
```

Kun funktiolle antaa argumentiksi luvun 33, se palauttaa totuusarvon False. Funktion mukaan siis luku 33 ei ole parillinen, mikä on totta. Samoin kun funktion argumenttina on luku 102, se palauttaa totuusarvon True. Funktion mukaan luku 102 on parillinen, mikä siis pitää paikkansa.

Tässä myös nähtiin, kuinka funktiota kutsutaan ja kuinka parametrin arvoa vaihdellaan.

Funktioiden lisäksi uudelleenkäytettävyyttä Pythonissa tuovat esille kirjastot. Kirjastot ovat paketteja, joista löytyy valmiita funktioita, joita voi hyödyntää koodia kirjoitettaessa. Esimerkkinä mainittakoon math-kirjasto, joka sisältää kaikenlaisia matemaattisia operaatioita kuten neliöjuuren ottamisen.

Kirjasto otetaan käyttöön kirjoittamalla lähdekooditiedoston alkuun `import kirjastonnimi`. Koodissa voi sitten kutsua normaalisti mitä tahansa kirjaston funktiota. Itse asiassa tässä materiaalissa tutustutaan kahteen Pythonin kirjastoista, NumPy-kirjastoon sekä Matplotlib-kirjastoon.

Esimerkki 2.12. *Lasketaan luvun neljä neliöjuuri math-kirjaston avulla.*

```
>>>import math
>>>math.sqrt(4)
2.0
```

3 Matriisilaskentaa

Tässä luvussa käydään lyhyesti läpi matriisilaskennan perusteita. Asiaa johdatellaan tutun asian, yhtälöryhmien, kautta. Lopuksi katsotaan miten Python-ohjelmointia voidaan hyödyntää matriisilaskennassa. Päälähteinä on käytetty Fraleighin ja Beauregardin kirjaa Linear Algebra [3] sekä Metsänkylän luentomonistetta lineaarialgebran kurssille [4].

3.1 Lineaariset yhtälöryhmät

Tässä kappaleessa tutustaan lineaarisiin yhtälöryhmiin ja palautetaan mieleen niiden ratkaiseminen eliminointimenetelmällä. Uutena asiana tulee lineaarisen yhtälöryhmän kerroinmatriisi.

Määritelmä 3.1. Lineaarinen yhtälöryhmä on sellainen yhtälöryhmä, jonka kaikki muuttujat ovat ensimmäistä astetta. Lineaarinen yhtälöryhmä on siis muotoa

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\cdots\cdots\cdots\cdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m
\end{aligned}$$

Kuten määritelmästä huomataan yhtälöitä ja muuttujia voi olla kuinka paljon hyvänsä eikä niitä ole pakko olla edes samaa määrää. Yksinkertaisuuden vuoksi tässä materiaalissa rajoitutaan kuitenkin vain yhtälöryhmiin, joissa on yhtä monta muuttujaa kuin yhtälöä.

Lineaaristen yhtälöryhmien ratkaisemiseen on useita tapoja. Yksinkertaisin niistä on Gaussin eliminointimenetelmä. Siinä yhtälöitä kerrotaan sopivilla luvuilla ja vähennetään toisistaan siten, että alimmassa yhtälöissä vain viimeisen termin kerroin on nollasta poikkeava, toiseksi alimmassa korkeintaan kahden viimeisen termin kerroin on nollasta poikkeava jne. Lopulta ylimmässä yhtälössä kaikki termit voivat olla kertoimiltaan nollasta poikkeavia. Tästä saadaan ratkaistua yksi muuttuja ja sijoittamalla ratkaisu muihin yhtälöihin, saadaan myös muille muuttujille ratkaisut. Tavoitteena on siis saada yhtälöryhmä muotoon

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
0 \cdot x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
0 \cdot x_1 + 0 \cdot x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
&\cdots\cdots\cdots\cdots \\
0 \cdot x_1 + 0 \cdot x_2 + \cdots + 0 \cdot x_{n-1} + a_{nn}x_n &= b_n
\end{aligned}$$

Sallittuja operaatioita ovat yhtälöiden paikan vaihtaminen, yhtälöiden kertominen nollasta eroavalla vakiolla sekä yhtälöiden lisääminen toisiinsa.

Ratkaistaessa yhtälöryhmää eliminointimenetelmällä ratkaisutapa ei ole yksikäsitteinen. Saman ratkaisun voi saada myös jollain muulla tavalla eli kertomalla joillain muilla luvuilla tai eliminoimalla eri järjestyksessä.

Esimerkki 3.1. *Ratkaistaan alla oleva yhtälöryhmä eliminointimenetelmällä.*

$$\begin{aligned} x_1 - 5x_2 + 3x_3 &= 12 \\ 2x_1 - x_2 - x_3 &= 6 \\ -2x_1 + 10x_2 + 2x_3 &= 24 \end{aligned}$$

Lisätään ensin keskimmäinen yhtälö alimmaiseen yhtälöön. Näin saadaan yhtälöryhmä muotoon

$$\begin{aligned} x_1 - 5x_2 + 3x_3 &= 12 \\ 2x_1 - x_2 - x_3 &= 6 \\ 0 \cdot x_1 + 9x_2 + x_3 &= 30. \end{aligned}$$

Kerrotaan sitten ylin yhtälö luvulla 2 ja keskimmäinen yhtälö luvulla -1. Nyt yhtälöryhmä on muodossa

$$\begin{aligned} 2x_1 - 10x_2 + 6x_3 &= 24 \\ -2x_1 + x_2 + x_3 &= -6 \\ 0 \cdot x_1 + 9x_2 + x_3 &= 30 \end{aligned}$$

Kun lisätään keskimmäinen yhtälö ylimpään yhtälöön, saadaan yhtälöryhmä muotoon

$$\begin{aligned} 0 \cdot x_1 - 9x_2 + 7x_3 &= 18 \\ -2x_1 + x_2 + x_3 &= -6 \\ 0 \cdot x_1 + 9x_2 + x_3 &= 30 \end{aligned}$$

Vaihtamalla kahden ylimmäisen yhtälön paikkaa yhtälöryhmä saadaan muotoon

$$\begin{aligned} -2x_1 + x_2 + x_3 &= -6 \\ 0 \cdot x_1 - 9x_2 + 7x_3 &= 18 \\ 0 \cdot x_1 + 9x_2 + x_3 &= 30. \end{aligned}$$

Kun vielä lisätään keskimmäinen yhtälö alimpaan yhtälöön, saadaan yhtälöryhmä toivottuun muotoon.

$$\begin{aligned} -2x_1 + x_2 + x_3 &= -6 \\ 0 \cdot x_1 - 9x_2 + 7x_3 &= 18 \\ 0 \cdot x_1 + 0 \cdot x_2 + 8x_3 &= 48. \end{aligned}$$

Alimmasta yhtälöstä voidaan nyt ratkaista muuttuja x_3. Ratkaisuksi saadaan $x_3 = 6$. Kun tämä sijoitetaan keskimmäiseen yhtälöön, saadaan se muotoon $-9x_2 + 42 = 18$. Tästä saadaan ratkaisuksi $x_2 = 2\frac{2}{3}$. Muuttuja x_1 saadaan ratkaistua kun sijoitetaan muuttujien x_2 ja x_3 ratkaisut ylimpään yhtälöön. Sijoituksen jälkeen yhtälö on muodossa $-2x_1 + 2\frac{2}{3} + 6 = -6$. Tästä saadaan ratkaisuksi $x_1 = 7\frac{1}{3}$. Yhtälöryhmän ratkaisu on siis

$$\begin{cases} x_1 = 7\frac{1}{3} \\ x_2 = 1\frac{2}{3} \\ x_3 = 6 \end{cases}$$

Yhtälöryhmästä voidaan muodostaa ns. kerroinmatriisi. Kerroinmatriisi saadaan kun vasemmalla puolella olevista lausekkeista otetaan pelkät yhtälön kertoimet. Määritelmässä 3.1 käytettyjen merkintöjen mukaan kerroinmatriisi on siis matriisi

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Matriiseista kerrotaan lisää seuraavissa kappaleissa. Myöhemmin käy myös ilmi, mitä hyötyä matriiseista on lineaaristen yhtälöryhmien kannalta.

Esimerkki 3.2. *Yhtälöryhmän*

$$\begin{aligned} x_1 + x_2 - 4x_3 &= 10 \\ 2x_1 - 2x_2 + 3x_3 &= 4 \\ -4x_1 + 3x_2 - x_3 &= -1 \end{aligned}$$

kerroinmatriisi on matriisi $\begin{pmatrix} 1 & 1 & -4 \\ 2 & -2 & 3 \\ -4 & 3 & -1 \end{pmatrix}$.

Harjoituksia

Tarkastellaan yhtälöryhmää

$$\begin{aligned} 2x_1 + x_2 &= 5 \\ 4x_1 - 4x_2 &= 4 \end{aligned}$$

Tehtävä 6. *Ratkaise yhtälöryhmä eliminointimenetelmällä.*

Tehtävä 7. *Muodosta yhtälöryhmän kerroinmatriisi A.*

3.2 Perusasioita matriiseista

Aivan ensiksi määritellään, mikä matriisi on. Sen jälkeen tutustutaan peruslaskutoimituksiin, joita matriiseille voidaan tehdä.

Määritelmä 3.2. Matriisi on tietynlainen taulukko lukuja. Matriisissa on jokaisella vaakarivillä n alkiota ja jokaisella pystyrivillä m alkiota. Alla oleva matriisi A on $m \times n$-matriisi.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Alkioita a_{ij} kutsutaan matriisialkioiksi. Matriisialkiot voivat olla mitä tahansa lukuja tai jopa esimerkiksi polynomeja. Yksinkertaisuuden vuoksi tässä materiaalissa matriisialkiot rajoitetaan reaalilukuihin. Matriisialkioita a_{ij}, joille pätee $i = j$, sanotaan matriisin diagonaalialkioksi.

Matriisin käsite hahmottuu paremmin vertaamalla niitä vektoreihin, sillä $1 \times n$-matriisi $X = (x_1, x_2, \ldots, x_n)$ on itse asiassa vektori.

Yleisen matriisin lisäksi on paljon matriiseja, joilla on joitakin erikoispiirteitä. Eräs niistä on neliömatriisi. Joihinkin muihin erikoispiirteisiin tutustutaan myöhemmin tässä materiaalissa.

Määritelmä 3.3. Neliömatriisiksi kutsutaan matriisia, jolle pätee $m = n$.

Esimerkki 3.3. *Matriisi* $A = \left(\begin{smallmatrix} 1 & 2 \\ 5 & 6 \end{smallmatrix}\right)$ *on* 2×2*-neliömatriisi.*

Matriiseja voidaan laskea yhteen ja vähentää toisistaan, mikäli kaikki matriisit ovat $m \times n$-matriiseja. Kummastakin operaatiosta on tuloksena aina $m \times n$-matriisi. Matriisien yhteenlasku tapahtuu siten, että lasketaan samalla paikalla olevat alkiot yhteen ja näin saadaan kyseiselle paikalle alkio tulosmatriisissa. Vähennnyslasku tapahtuu vastaavalla tavalla. Olkoon A ja B mielivaltaisia matriiseja ja olkoon C matriisi, joka on saatu laskemalla A ja B yhteen. Silloin matriisin C alkiot ovat $c_{ij} = a_{ij} + b_{ij}$.

Esimerkki 3.4. *Lasketaan matriisien* $A = \begin{pmatrix} 3 & 4 & -6 \\ -1 & 0 & 5 \end{pmatrix}$ *ja* $B = \begin{pmatrix} 7 & -5 & 8 \\ 10 & 3 & 9 \end{pmatrix}$ *summa (A+B) ja erotus (A-B).*

$$\begin{aligned} A + B &= \begin{pmatrix} 3 & 4 & -6 \\ -1 & 0 & 5 \end{pmatrix} + \begin{pmatrix} 7 & -5 & 8 \\ 10 & 3 & 9 \end{pmatrix} = \begin{pmatrix} 3+7 & 4+(-5) & -6+8 \\ -1+10 & 0+3 & 5+9 \end{pmatrix} \\ &= \begin{pmatrix} 10 & -1 & 2 \\ 9 & 3 & 14 \end{pmatrix} \end{aligned}$$

$$A - B = \begin{pmatrix} 3 & 4 & -6 \\ -1 & 0 & 5 \end{pmatrix} - \begin{pmatrix} 7 & -5 & 8 \\ 10 & 3 & 9 \end{pmatrix} = \begin{pmatrix} 3-7 & 4-(-5) & -6-8 \\ -1-10 & 0-3 & 5-9 \end{pmatrix}$$
$$= \begin{pmatrix} -4 & 9 & -14 \\ -11 & -3 & -4 \end{pmatrix}$$

Matriiseja voidaan myös kertoa reaaliluvuilla. Kertominen tapahtuu siten, että kerrotaan kukin alkio kerrottavalla. Olkoon siis r mielivaltainen reaaliluku ja A mielivaltainen matriisi. Kun matriisi A kerrotaan reaaliluvulla r , saadaan tulokseksi matriisi B, joka on siis samaa tyyppiä kuin A. Matriisin B alkiot saadaan kertomalla matriisin A alkiot luvulla r. Matriisissa B siis alkio $b_{ij} = r \cdot a_{ij}$.

Esimerkki 3.5. *Kerrotaan matriisi* $A = \begin{pmatrix} 4 & 0 & 7 & 1 \\ 6 & 8 & 3 & 2 \end{pmatrix}$ *luvulla 5. Kerrotaan siis kukin matriisin A alkio luvulla 5. Näin saadaan*

$$5 \cdot \begin{pmatrix} 4 & 0 & 7 & 1 \\ 6 & 8 & 3 & 2 \end{pmatrix} = \begin{pmatrix} 5 \cdot 4 & 5 \cdot 0 & 5 \cdot 7 & 5 \cdot 1 \\ 5 \cdot 6 & 5 \cdot 8 & 5 \cdot 3 & 5 \cdot 2 \end{pmatrix} = \begin{pmatrix} 20 & 0 & 35 & 5 \\ 30 & 40 & 15 & 10 \end{pmatrix}.$$

Määritelmä 3.4. Olkoon matriisi

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

Kun vaihdetaan matriisin A pystyrivit vaakariveiksi, saadaan matriisin A transponoitu matriisi A^T tai lyhyesti transpoosi

$$A^T = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}.$$

Esimerkki 3.6. *Matriisin* $A = \begin{pmatrix} 12 & 3 & 4 \\ 5 & 7 & 13 \end{pmatrix}$ *transpoosi* $A^T = \begin{pmatrix} 12 & 5 \\ 3 & 7 \\ 4 & 13 \end{pmatrix}$.

Harjoituksia

Kaikissa tehtävissä on käytetty matriiseja
$A = \begin{pmatrix} 1 & 8 & -2 \\ 1 & 5 & 0 \\ 3 & 9 & 2 \end{pmatrix}$ ja $B = \begin{pmatrix} 1 & -4 & 0 \\ 6 & 7 & 12 \\ -9 & 8 & -3 \end{pmatrix}$.

Tehtävä 8. *Laske niiden summa (A+B), erotus (A-B) sekä kerro kummatkin matriisit luvulla 3.*

Tehtävä 9. *Muodosta matriisin A transpoosi.*

3.3 Determinantti ja Cramerin sääntö

Neliömatriiseille voidaan määritellä determinantti. Matriisin determinanttia merkitään det(A) tai $|A|$. Tyyppiä 2×2 olevan matriisin determinantti sovitaan laskettavaksi seuraavasti [5].

$$\det \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right) = \left| \begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right| = ad - bc$$

Yleisesti determinantti määritellään seuraavasti.

Määritelmä 3.5. Tyyppiä $n \times n$ olevan matriisin determinantti on

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{k2} & \cdots & a_{nn} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} & \cdots & a_{2n} \\ a_{32} & a_{33} & \cdots & a_{3n} \\ . & . & . & . \\ a_{n2} & a_{n3} & \cdots & a_{nn} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{33} & \cdots & a_{3n} \\ . & . & . & . \\ a_{n1} & a_{n3} & \cdots & a_{nn} \end{vmatrix} + \ldots$$

$$\ldots + (-1)^{n+1} a_{1n} \begin{vmatrix} a_{22} & a_{23} & \cdots & a_{2(n-1)} \\ a_{31} & a_{32} & \cdots & a_{3(n-1)} \\ . & . & . & . \\ a_{n1} & a_{n2} & \cdots & a_{n(n-1)} \end{vmatrix}$$

Determinantti saadaan siis kertomalla ensimmäisen rivin alkiot sen matriisin determinantilla, joka saadaan kun poistetaan ensimmäinen vaakarivi ja se pystyrivi, jolla alkio on. Lopuksi vielä saadut luvut lasketaan yhteen tai vähennetään toisistaan. Plus- ja miinusmerkit vuorottelevat.

Tämän määritelmän avulla determinantin laskeminen palautuu n kappaleen $n-1 \times n-1$-determinantin laskemiseen. Jatkamalla laskemista eteenpäin päästään lopulta 2×2-determinanttiin, jotka siis lasketaan edellä sovitulla tavalla.

Tyyppiä 1×1 olevalle matriisille determinantti määritellään seuraavasti.

Määritelmä 3.6. Olkoon (a) 1×1-matriisi. Silloin $\det(a) = a$.

Määritelmästä huomataan, että myös 2×2-determinantille toteuttaa edellä määritellyn yleisen laskusäännön.

Esimerkki 3.7. *Lasketaan matriisin* $A = \begin{pmatrix} 7 & 15 & -8 \\ 6 & 4 & 9 \\ 5 & 3 & 3 \end{pmatrix}$ *determinantti. Määritelmän mukaan otetaan siis kukin ensimmäisen rivin alkio ja kerrotaan se determinantilla, joka saadaan kun poistetaan ensimmäinen vaakarivi ja se pystyrivi, jolla alkio on. Näin saadaan siis*

$$\begin{vmatrix} 7 & 15 & -8 \\ 6 & 4 & 9 \\ 5 & 3 & 3 \end{vmatrix} = 7 \begin{vmatrix} 4 & 9 \\ 3 & 3 \end{vmatrix} - 15 \begin{vmatrix} 6 & 9 \\ 5 & 3 \end{vmatrix} + (-8) \begin{vmatrix} 6 & 4 \\ 5 & 3 \end{vmatrix}.$$

Alkioiden perään tuli nyt 2×2-determinantteja ja ne lasketaan suoraan edellä sovitun mukaisesti. Determinantin laskemisessa onkin ideana pudottaa rivejä pois niin kauan kunnes päädytään 2×2-determinanttiin. Lasketaan siis 2×2-determinantit ja näin saadaan

$$7(4\cdot 3-9\cdot 3)-15(6\cdot 3-9\cdot 5)-8(6\cdot 3-4\cdot 5) = 7(12-27)-15(18-45)-8(18-20)$$

$$= 7\cdot(-15)-15\cdot(-27)-8\cdot(-2) = -105+405+16 = 316.$$

Determinantiksi saatiin siis 316.

Palautetaan mieleen kappaleessa 3.1 käsitellyt lineaariset yhtälöryhmät sekä niiden kerroinmatriisi. Lineaarinen yhtälöryhmähän on muotoa

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

Sen kerroinmatriisi taas on matriisi

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

Itse asiassa lineaariset yhtälöryhmät voidaan esittää matriisimuodossa. Kerroinmatriisi A saatiin siis ottamalla vasemmanpuolen lausekkeista kertoimet. Samoin saadaan muuttujille matriisi $X = (x_1, x_2, \ldots, x_n)^T$ ja yhtälön oikealle puolelle matriisi $B = (b_1, b_2, \ldots, b_m)^T$. Matriisimuodossa yhtälöryhmä on siis $AX = B$. Matriisimuodossa on vasemmalla puolella kaksi matriisia kerrottu keskenään. Matriisin kertolaskua ei ole vielä määritelty, mutta se määritellään myöhemmin.

Lineaarisella yhtälöryhmällä ei välttämättä ole yksikäsitteistä ratkaisua. Ratkaisun yksikäsitteisyys on helppo tarkistaa yhtälöryhmän kerroinmatriisin avulla.

Lause 3.1. *Lineaarisella yhtälöryhmällä on yksikäsitteinen ratkaisu jos ja vain jos sen kerroinmatriisin determinantti on nollasta poikkeava.*

Perustelu tälle lauseelle on ilmeinen, kun tutustutaan kohta Cramerin sääntöön.

Jos kerroinmatriisin determinantti on nolla, niin yhtälöryhmällä ei ole ratkaisua tai sitten niitä on äärettömän monta.

Esimerkki 3.8. *Tarkastellaan lineaarista yhtälöryhmää*

$$\begin{aligned} x_1 &= 2 \\ 2x_1 - x_2 - 2x_3 &= 1 \\ 3x_1 + 2x_2 + 4x_3 &= 12 \end{aligned}$$

Yhtälöryhmän kerroinmatriisin determinantti on nyt nolla. Tämän voi vaikka itse todeta. Tällä yhtälöryhmällä ei siis ole yksikäsitteistä ratkaisua. Itse asiassa ratkaisuja on äärettömän monta.

Lineaarisia yhtälöryhmiä voidaan ratkaista myös niiden kerroinmatriisien avulla. Tapoja on monia. Eräs niistä on Cramerin sääntö.

Lause 3.2. *Olkoon yhtälöryhmässä yhtä monta muuttujaa ja yhtä monta yhtälöä sekä kerroinmatriisin determinantti nollasta poikkeava. Silloin yhtälöryhmällä on tarkalleen yksi ratkaisu ja se on seuraava*

$$x_j = \frac{det(A_j)}{det(A)}$$

missä $j = 1, 2, \ldots, n$ ja A_j on se matriisi, joka saadaan korvaamalla j:s pystyrivi matriisilla B.

Esimerkki 3.9. *Ratkaistaan alla oleva yhtälöryhmä Cramerin säännöllä.*

$$\begin{aligned} 3x_1 + 2x_2 - x_3 &= 7 \\ x_1 - x_2 + 2x_3 &= 2 \\ 2x_1 + 3x_2 - 2x_3 &= 3 \end{aligned}$$

Yhtälöryhmän kerroinmatriisi on nyt $A = \begin{pmatrix} 3 & 2 & -1 \\ 1 & -1 & 2 \\ 2 & 3 & -2 \end{pmatrix}$. *Lasketaan sen determinantti.*

$$\begin{vmatrix} 3 & 2 & -1 \\ 1 & -1 & 2 \\ 2 & 3 & -2 \end{vmatrix} = 3\begin{vmatrix} -1 & 2 \\ 3 & -2 \end{vmatrix} - 2\begin{vmatrix} 1 & 2 \\ 2 & -2 \end{vmatrix} - \begin{vmatrix} 1 & -1 \\ 2 & 3 \end{vmatrix} = 3 \cdot (-4) - 2 \cdot (-6) - 5 = -5$$

Matriisi B on nyt matriisi $(7, 2, 3)^T$.

Ratkaistaan ensin muuttuja x_1. Matriisi A_1 on siis matriisi, joka on saatu korvaamalla matriisista A ensimmäinen rivi matriisilla B. Matriisiksi A_1 saadaan siis matriisi $\begin{pmatrix} 7 & 2 & -1 \\ 2 & -1 & 2 \\ 3 & 3 & -2 \end{pmatrix}$. *Kun lasketaan matriisin A_1 determinantti, tulokseksi saadaan -17. Näin ollen muuttujaksi x_1 saadaan $x_1 = \frac{-17}{-5} = 3\frac{2}{5}$.*

Ratkaistaan sitten muuttuja x_2 vastaavalla tavalla. Nyt korvataan matriisista A toinen pystyrivi matriisilla B ja saadaan siis matriisi A_2. Matriisin A_2 determinanatiksi saadaan 13. Muuttujan x_2 arvoksi saadaan siis $x_2 = \frac{13}{-5} = -2\frac{3}{5}$.

Lopuksi vielä ratkaistaan muuttuja x_3. Matriisin A_3 determinantiksi saadaan 10 ja siten $x_3 = \frac{10}{-5} = -2$.

Yhtälöryhmän ratkaisu on siis

$$\begin{cases} x_1 = 3\frac{2}{5} \\ x_2 = -2\frac{3}{5} \\ x_3 = -2 \end{cases}$$

Harjoituksia

Tehtävä 10. *Olkoon matriisi $A = \begin{pmatrix} 1 & -1 & 3 \\ 2 & 0 & 2 \\ -2 & 1 & -3 \end{pmatrix}$. Mikä on matriisin A determinantti?*

Tehtävä 11. *Ratkaise yhtälöryhmä Cramerin säännön avulla.*

$$5x_1 - x_2 = 12$$
$$x_1 - 2x_2 = 6$$

3.4 Matriisien kertolasku

Matriisien kertolasku on yhteenlaskua ja luvulla kertomista hieman monimutkaisempi operaatio. Tässä kappaleessa perehdymme matriisien kertolaskuun.

Ensinnäkin kahta mielivaltaista matriisia ei voida välttämättä kertoa keskenään. Seuraava lause kertoo, millaisia matriiseja voidaan kertoa keskenään.

Lause 3.3. *Olkoon matriisi A jokin $m \times n$-matriisi ja B jokin $l \times s$-matriisi. Matriisit A ja B voidaan kertoa keskenään jos ja vain jos $n = l$.*

Perustelu tälle lauseelle on ilmeinen kun on tutustuttu ensin siihen, kuinka matriiseja oikein kerrotaan keskenään.

Matriisien kertolasku tapahtuu siten, että otetaan kertojan ensimmäinen vaakarivi ja kerrottavasta ensimmäinen pystyrivi. Kerrotaan ensimmäiset alkiot keskenään ja lisätään siihen toisten alkioiden tulo. Tähän lisätään kolmansien alkioiden tulo ja näin jatketaan. Saatu summa on tulosmatriisin alkio a_{11}. Tulosmatriisin alkio a_{12} saadaan toistamalla operaatio kerrottavan ensimmäiselle vaakariville ja kertojan toiselle pystyriville. Näin siis jatketaan kunnes jokainen kerrottavan vaakarivi on kerrottu jokaiselle kertojan pystyrivillä. Tuloksena kertolaskusta saadaan $m \times s$-matriisi.

Olkoon siis A ja B mielivaltaisia matriiseja ja C niiden kertolaskun tuloksena saatu matriisi. Silloin alkio $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}$.

Matriisien kertolaskua kannataa verrata vektoreiden pistetuloon. Usein ajatellaan, että matriisin rivit ja sarakkeet ovat vektoreita. Tulosmatriisin C alkiota c_{ij} varten siis otetaan matriisista A i:s vaakarivi ja matriisista B j:s sarake ja lasketaan näiden kahden vektorin pistetulo.

Esimerkki 3.10. *Matriiseja* $A=\begin{pmatrix} -4 & 2 \\ 3 & 1 \end{pmatrix}$ *ja* $B=\begin{pmatrix} 5 & 6 & 1 & -3 \\ -2 & 0 & -1 & 4 \\ 7 & 5 & 8 & 2 \end{pmatrix}$ *ei pysty kertomaan keskenään, koska A on tyyppiä* 2×2 *ja B tyyppiä* 3×4.

Matriisien kertolaskusta on tärkeää huomata eräs asia. Lukujen kertolaskussa voidaan operaation suorittaa missä järjestyksessä hyvänsä. Esimerkiksi olkoon a ja b joitakin reaalilukuja. Silloin $a \cdot b = b \cdot a$. Matriiseille tämä ominaisuus ei kuitenkaan aina päde. Tämä huomataan seuraavasta esimerkistä.

Esimerkki 3.11. *Olkoon matriisi* $A=\begin{pmatrix} 1 & 3 \\ 8 & 6 \end{pmatrix}$ *ja matriisi* $B=\begin{pmatrix} 8 & 9 \\ 2 & 3 \end{pmatrix}$ *. Lasketaan ensin matriisi AB*

$$\begin{pmatrix} 1 & 3 \\ 8 & 6 \end{pmatrix}\begin{pmatrix} 8 & 9 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 1\cdot 8+3\cdot 2 & 1\cdot 9+3\cdot 3 \\ 8\cdot 8+6\cdot 2 & 8\cdot 9+6\cdot 3 \end{pmatrix}$$

$$= \begin{pmatrix} 8+6 & 9+9 \\ 64+12 & 72+18 \end{pmatrix} = \begin{pmatrix} 14 & 18 \\ 76 & 90 \end{pmatrix}$$

Lasketaan sitten matriisi BA.

$$\begin{pmatrix} 8 & 9 \\ 2 & 3 \end{pmatrix}\begin{pmatrix} 1 & 3 \\ 8 & 6 \end{pmatrix} = \begin{pmatrix} 8\cdot 1+9\cdot 8 & 8\cdot 3+9\cdot 6 \\ 2\cdot 1+3\cdot 8 & 2\cdot 3+3\cdot 6 \end{pmatrix}$$

$$= \begin{pmatrix} 8+72 & 24+54 \\ 2+24 & 6+18 \end{pmatrix} = \begin{pmatrix} 80 & 78 \\ 26 & 24 \end{pmatrix}$$

Huomataan, että tuloksena on eri matriisit.

Määritelmä 3.7. Identiteettimatriisiksi I kutsutaan neliömatriisia, jonka diagonaalialkiot ovat ykkösiä ja muut alkiot ovat nollia. Identiteettimatriisi on siis matriisi

$$I = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ . & . & . & . & . \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Identiteettimatriisilla on se ominaisuus, että sillä kerrottaessa tai sitä kerrottaessa tuloksena on aina matriisi mitä sillä kerrotaan tai millä se kerrotaan. Identiteettimatriisi on siis kertolaskussa sama kuin luvuissa kertoisi ykkösellä. Matemaattisesti ilmaistuna siis

$$IA = AI = A$$

missä A on mielivaltainen matriisi ja I on identiteettimatriisi.

Esimerkki 3.12. *Kerrotaan matriisit* $A = \begin{pmatrix} 8 & 1 & -6 \\ -4 & 7 & 2 \\ -2 & 3 & 5 \end{pmatrix}$ *ja* $I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ *keskenään.*

$$\begin{aligned} AI &= \begin{pmatrix} 8 & 1 & -6 \\ -4 & 7 & 2 \\ -2 & 3 & 5 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 8\cdot 1 + 1\cdot 0 + (-6)\cdot 0 & 8\cdot 0 + 1\cdot 1 + (-6)\cdot 0 & 8\cdot 0 + 1\cdot 0 + (-6)\cdot 1 \\ -4\cdot 1 + 7\cdot 0 + 2\cdot 0 & -4\cdot 0 + 7\cdot 1 + 2\cdot 0 & -4\cdot 0 + 7\cdot 0 + 2\cdot 1 \\ -2\cdot 1 + 3\cdot 0 + 5\cdot 0 & -2\cdot 0 + 3\cdot 1 + 5\cdot 0 & -2\cdot 0 + 3\cdot 0 + 5\cdot 1 \end{pmatrix} \\ &= \begin{pmatrix} 8+0+0 & 0+1+0 & 0+0-6 \\ -4+0+0 & 0+7+0 & 0+0+2 \\ -2+0+0 & 0+3+0 & 0+0+5 \end{pmatrix} = \begin{pmatrix} 8 & 1 & -6 \\ -4 & 7 & 2 \\ -2 & 3 & 5 \end{pmatrix} = A \end{aligned}$$

Jos suorittaa kertolaskun toisinpäin, on myös tuloksena matriisi A. Kertolaskun yksityiskohtainen suorittaminen jätetään harjoitustehtäväksi.

Harjoituksia

Tehtävä 12. *Laske, mikäli mahdollista, seuraavien matriisien tulo (AB).*

a)

$$A = \begin{pmatrix} 5 & 6 \\ 7 & 4 \\ 8 & 3 \end{pmatrix}, B = \begin{pmatrix} 1 & 2 & -4 \\ 1 & 2 & 0 \\ 3 & -4 & 3 \end{pmatrix}$$

b)

$$A = \begin{pmatrix} 3 & 2 \\ 4 & 6 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & -1 & 2 \\ -2 & 3 & 1 & 2 \end{pmatrix}$$

c)

$$A = \begin{pmatrix} -1 & 5 & 2 \\ -4 & 10 & 8 \\ 0 & -2 & 3 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

d)

$$A = \begin{pmatrix} 4 & -3 \\ 5 & 4 \end{pmatrix}, B = \begin{pmatrix} 2 & 1 \\ 2 & -4 \end{pmatrix}$$

Tehtävä 13. *Laske, mikäli mahdollista edellisen tehtävän matriisien tulo (BA).*

3.5 Käänteismatriisi

Tässä kappaleessa tutustutaan käänteismatriisin käsitteeseen. Lisäksi katsotaan millaisilla matriiseilla käänteismatriisi on ja lopuksi vielä lyhyesti käänteismatriisin laskemista [5].

Määritelmä 3.8. Matriisin A käänteismatriisi on sellainen matriisi A^{-1}, jolla pätee $AA^{-1} = A^{-1}A = I$.

Esimerkki 3.13. *Matriisi* $B = \begin{pmatrix} -\frac{1}{5} & \frac{2}{5} & -\frac{2}{5} \\ -\frac{1}{5} & \frac{2}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{1}{5} & \frac{1}{5} \end{pmatrix}$ *on matriisin*

$A = \begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \\ -1 & 1 & 0 \end{pmatrix}$ *käänteismatriisi, sillä*

$$AB = \begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{5} & \frac{2}{5} & -\frac{2}{5} \\ -\frac{1}{5} & \frac{2}{5} & \frac{3}{5} \\ \frac{3}{5} & -\frac{1}{5} & \frac{1}{5} \end{pmatrix}$$

$$= \begin{pmatrix} 1 \cdot \left(-\frac{1}{5}\right) + 0 \cdot \left(-\frac{1}{5}\right) + 2 \cdot \frac{3}{5} & 1 \cdot \frac{2}{5} + 0 \cdot \frac{2}{5} + 2 \cdot \left(-\frac{1}{5}\right) & 1 \cdot \left(-\frac{2}{5}\right) + 0 \cdot \frac{2}{5} + 2 \cdot \frac{1}{5} \\ 2 \cdot \left(-\frac{1}{5}\right) + 1 \cdot \left(-\frac{1}{5}\right) + 1 \cdot \frac{3}{5} & 2 \cdot \frac{2}{5} + 1 \cdot \frac{2}{5} + 1 \cdot \left(-\frac{1}{5}\right) & 2 \cdot \left(-\frac{2}{5}\right) + 1 \cdot \frac{3}{5} + 1 \cdot \frac{1}{5} \\ -1 \cdot \left(-\frac{1}{5}\right) + 1 \cdot \left(-\frac{1}{5}\right) + 0 \cdot \frac{3}{5} & -1 \cdot \frac{2}{5} + 1 \cdot \frac{2}{5} + 0 \cdot \left(-\frac{1}{5}\right) & -1 \cdot \left(-\frac{2}{5}\right) + 1 \cdot \frac{3}{5} + 0 \cdot \frac{1}{5} \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{1}{5} + \frac{6}{5} & \frac{2}{5} - \frac{2}{5} & -\frac{2}{5} + \frac{2}{5} \\ -\frac{2}{5} - \frac{1}{5} + \frac{3}{5} & \frac{4}{5} + \frac{2}{5} - \frac{1}{5} & -\frac{4}{5} + \frac{3}{5} + \frac{1}{5} \\ \frac{1}{5} - \frac{1}{5} & -\frac{2}{5} + \frac{2}{5} & \frac{2}{5} + \frac{3}{5} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Kertolaskusta BA tulee myös vastaukseksi identiteettimatriisi, mutta kertolaskun suorittaminen jätetään harjoitustehtäväksi.

Kaikilla matriiseilla ei kuitenkaan ole käänteismatriisia. Seuraava lause kertoo, millaisilla matriiseilla on käänteismatriisi.

Lause 3.4. *Matriisilla A on käänteismatriisi, jos ja vain jos se on neliömatriisi ja det(A)$\neq 0$.*

Matriiseja, joilla on käänteismatriisi, kutsutaan säännölliksiksi. Lisäksi on myös hyvä muistaa, että jos matriisilla on käänteismatriisi, on se yksikäsitteinen.

Käänteismatriisin etsimiseksi on useita keinoja. Tässä materiaalissa esitellään kuitenkin vain periaatteeltaan yksinkertaisin. Tämä tapa perustuu suoraan käänteismatriisin määritelmään. Olkoon matriisi A matriisi, jolle käänteismatriisi pitää etsiä. Tiedetään että näiden kahden matriisin kertolaskun tuloksena on identiteettimatriisi. Merkitään siis käänteismatriisin alkioita tuntemattomilla

x_{ij} ja kerrotaan matriisi A käänteismatriisillaan. Koska tulos tiedetään, saadaan kertolaskun avulla yhtälöryhmä, josta voidaan ratkaista alkiot x_{ij}. Matemaattisin merkinnöin siis

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ . & . & . & . \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ . & . & . & . & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Kun kertolasku suoritetaan, saadaan tuloksena seuraava yhtälöryhmä, jossa on n^2 yhtälöä ja n^2 muuttujaa.

$$\begin{aligned} a_{11}x_{11} + a_{12}x_{21} \cdots + a_{1n}x_{n1} &= 1 \\ a_{11}x_{12} + a_{12}x_{22} + \cdots + a_{1n}x_{n2} &= 0 \\ &\ldots\ldots\ldots\ldots \\ a_{n1}x_{1n} + a_{n2}x_{2n} + \cdots + a_{nn}x_{nn} &= 1 \end{aligned}$$

Yhtälöryhmästä saadaan ratkaistua siis muuttujat x_{ij}.

Esimerkki 3.14. *Etsitään käänteismatriisi matriisille* $A = \begin{pmatrix} 1 & -1 \\ 3 & -2 \end{pmatrix}$. *Merkitään käänteismatriisin alkioita muuttujilla* x_{ij}. *Näin saadaan kertolasku.*

$$\begin{pmatrix} 1 & -1 \\ 3 & -2 \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Kun suoritetaan kertolasku, saadaan yhtälöryhmä

$$\begin{aligned} x_{11} - x_{21} &= 1 \\ x_{12} - x_{22} &= 0 \\ 3x_{11} - 2x_{21} &= 0 \\ 3x_{12} - 2x_{22} &= 1 \end{aligned}$$

Koska ensimmäinen ja kolmas yhtälö eivät riipu toisesta ja neljännestä, voidaan yhtälöryhmä hajottaa kahdeksi yhtälöryhmäksi. Tässä ei nyt käydä läpi sen tarkemmin yhtälöparien ratkaisemista. Koko yhtälöryhmän ratkaisuksi tulee

$$x_{11} = -2,\ x_{12} = 1,\ x_{21} = -3,\ x_{22} = 1$$

Näin ollen saadaan käänteismatriisiksi $A^{-1} = \begin{pmatrix} -2 & 1 \\ -3 & 1 \end{pmatrix}$

Harjoituksia

Tehtävä 14. *Mikä seuraavista on matriisin* $A = \begin{pmatrix} -3 & 1 \\ 2 & -4 \end{pmatrix}$ *käänteismatriisi?*

a) $\begin{pmatrix} -3 & 2 \\ 1 & -4 \end{pmatrix}$ *b)* $\begin{pmatrix} -\frac{2}{5} & -\frac{1}{10} \\ -\frac{1}{5} & -\frac{3}{10} \end{pmatrix}$ *c)* $\begin{pmatrix} \frac{2}{5} & -\frac{1}{10} \\ \frac{3}{10} & -\frac{1}{5} \end{pmatrix}$

Tehtävä 15. *Etsi matriisin* $A = \begin{pmatrix} 2 & -1 \\ 4 & 1 \\ -1 & -3 \end{pmatrix}$ *käänteismatriisi.*

Tehtävä 16. *Etsi matriisin* $A = \begin{pmatrix} 0 & -1 \\ 2 & 1 \end{pmatrix}$ *käänteismatriisi.*

3.6 Sovelluksia Python-ohjelmointiin

Tässä kappaleessa tutkitaan, kuinka Python-ohjelmointia voidaan käyttää matriisilaskennan työkaluna. Lähteenä on käytetty NumPy-kirjaston tutoriaalia [6].

Kuten edellä huomattiin, monet matriisien laskutoimituksista ovat työläitä suorittaa. Esimerkiksi kertolaskut isoilla matriiseilla, käänteismatriisin etsiminen ja lineaarisen yhtälöryhmän ratkaiseminen vievät paljon aikaa. Python-ohjelmointi tarjoaa helpotusta tähän ongelmaan. Pythonin NumPy-kirjastosta löytyy valmiit komennot kaikkien näiden operaatioiden suorittamiseen.

Python-kielessä on olemassa matriiseille oma tietorakenne. Tämä tietorakenne saadaan käyttöön NumPy-kirjaston mukana, joten aina matriiseja käsiteltäessä on otettava käyttöön NumPy-kirjasto. Tämä tapahtuu kirjoittamalla lähdekooditiedoston alkuun `from numpy import *`.

Matriisi luodaan `matrix`-tyyppisenä. Matriisissa kukin vaakarivi on ikäänkuin lista. Rivit siis erotetaan toisistaan hakasulkein ja alkiot pilkuin. Matriisissa voidaan viitata yksittäiseen alkioon hakasulkeiden avulla. Jälleen kerran on vain muistettava, että niin rivien kuin sarakkeidenkin indeksointi alkaa nollasta.

Esimerkki 3.15.

```
#otetaan Numpy-kirjasto käyttöön
from numpy import *
#luodaan matriisi A
A = matrix ([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
#tulostetaan matriisista alkio a_32
print A[2,1]
```

Ohjelma tulostaa siis luvun 8, koska se on kolmannella rivillä ja toisessa sarakkeessa.

Pythonilla voidaan matriiseille suoraan suoritttaa tavallisia laskuoperaatioita, kuten yhteen-, vähennys- ja kertolasku. Matriiseja voidaan kertoa niin reaa-

liluvuilla kuin keskenäänkin. Operaatiot tapahtuvat normaalisti operaattoreilla +,- ja *.

Esimerkki 3.16.

```
#otetaan NumpPy-kirjasto käyttöön
from numpy import *
#luodaan matriisi A
A = matrix ([[1, 2], [3, 4]])
#luodaan matriisi B
B = matrix ([[5, 6], [7, 8]])
#yhteenlasku
print A+B
#vähennyslasku
print A-B
#skalaarilla kertominen
print 5*A
#kertolasku
print A*B
```

Ohjelma siis tulostaa matriisit A ja B laskettuna yhteen, vähennettynä toisistaan, matriisin A kerrottuna viidellä sekä A:n ja B:n tulon. Tulosteena ovat siis seuraavat matriisit.

$$\begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}, \begin{pmatrix} -4 & -4 \\ -4 & -4 \end{pmatrix}, \begin{pmatrix} 5 & 10 \\ 15 & 20 \end{pmatrix}, \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

Myös tranpoosin ja käänteismatriisin etsimiselle on omat komentonsa, joiden avulla kyseiset matriisit saadaan suoraan. Transpoosi saadaan komennolla `matriisi.T`, missä `matriisi` on transponoitava matriisi. Käänteismatriisi puolestaan saadaan komennolla `matriisi.I`, missä `matriisi` on matriisi, jolle käänteismatriisi etsitään. Samoin determinantti saadaan laskettua suoraan. Se tapahtuu kommennolla `linalg.det(matriisi)`.

Esimerkki 3.17. *Lasketaan Pythonin avulla matriisille* $A = \begin{pmatrix} 1 & -4 & 0 \\ -2 & 1 & 4 \\ 2 & 0 & -1 \end{pmatrix}$ *transpoosi, käänteismatriisi ja determinantti.*

```
#otetaan NumPy-kirjasto käyttöön
from numpy import *
#luodaan matriisi A
A=matrix([[1,-4,0],[-2,1,4],[2,0,-1]])
#tulostetaan matriisin A tranpoosi
```

```
print A.T
# tulostetaan matriisin A käänteismatriisi
print A.I
#tulostetaan matriisin A determinantti
print linalg.det(A)
```

Ohjelma siis tulostaa matriisin $A^T = \begin{pmatrix} 1 & -2 & 2 \\ -4 & 1 & 0 \\ 0 & 4 & -1 \end{pmatrix}$, *matriisin* $A^{-1} = \begin{pmatrix} 0.04 & 0.16 & 0.64 \\ -0.24 & 0.04 & 0.16 \\ 0.08 & 0.32 & 0.28 \end{pmatrix}$ *sekä determinantin, joka on -25 .*

On huomioitava, että Python antaa esimerkiksi käänteismatriisien alkiot desimaalilukumuodossa. Mikäli murtolukuesitystä tarvitaan, se on osattava itse muodostaa.

Kuten edellä todettiin esimerkiksi matriisin kertolaskun suorittamiseen ja käänteismatriisin olemassaoloon on tiettyjä rajoitteita eli kaikille matriiseille kertolasku ei onnistu eikä kaikilla matriiseilla ole käänteismatriisia. Python ottaa huomioon myös useimmat näistä tilanteista, vaikka ehdot on helppo tarkistaa myös itse. Ainoa, mitä Python ei huomioi, on käänteismatriisin olemassaoloon liittyvä ehto. Jos Pythonilla yrittää etsiä käänteismatriisia muulle kuin neliömatriisille, ohjelma kyllä antaa matriisin, mutta se ei ole kyseisen matriisin käänteismatriisi. Tässä tapauksessa on siis ohjelmoijan itsensä oltava tarkkana.

Esimerkki 3.18. *Lasketaan Pythonilla matriisien* $A = \begin{pmatrix} 12 & 5 & -9 \\ 2 & 6 & 7 \end{pmatrix}$ *ja* $B = \begin{pmatrix} 11 & 10 \\ -8 & 6 \end{pmatrix}$ *tulo.*

```
#otetaan NumPy-kirjasto käyttöön
from numpy import *
#luodaan matriisit A ja B
A=matrix([[0,1],[1,1],[2,2]])
B=matrix([[11,10,-9],[-8,6,7],[9,7,-6]])
#tulostetaan matriisit A ja B kerrottuna keskenään
print A*B
```

Koska matriiseja A ja B ei voi kertoa keskenään, ohjelma antaa virheilmoituksen.

```
File "<stdin>", line 2, in <module>
File "/usr/lib/python2.5/site-packages/numpy/core/defmatrix.py",
line 157, in __mul__ return N.dot(self, asmatrix(other))
ValueError: objects are not aligned
```

Lasketaan seuraavaksi matriisin A käänteismatriisi.

```
#tulostetaan matriisin A käänteismatriisi
print A.I
```

Ohjelma tulostaa matriisin.

```
matrix([[ -1.00000000e+00,   2.00000000e-01,   4.00000000e-01],
        [  1.00000000e+00,  -7.01749856e-17,  -1.40349971e-16]])
```

Tämä ei kuitenkaan ole matriisin A käänteismatriisi, sillä kertolasku $A * A^{-1}$ *antaa tulokseksi jotain muuta kuin identiteettimatriisin.*

```
>>> A*(A.I)
matrix([[  1.00000000e+00,  -7.01749856e-17,  -1.40349971e-16],
        [  1.11022302e-16,   2.00000000e-01,   4.00000000e-01],
        [  2.22044605e-16,   4.00000000e-01,   8.00000000e-01]])
```

NumPy-moduulista löytyy myös valmis komento lineaarisen yhtälöryhmän ratkaisuun. Kyseinen komento on `linalg.solve(matriisi1, matriisi2)`, missä matriisi1 on yhtälöryhmän kerroinmatriisi ja toisena argumenttina on oikeasta puolesta saatava matriisi.

Esimerkki 3.19. *Ratkaistaan alla oleva yhtälöryhmä Pythonilla.*

$$2x - 3y = 10$$
$$5x + 6y = 20$$

Otetaan ensin yhtälöryhmän kerroinmatriisi

$$A = \begin{pmatrix} 2 & -3 \\ 5 & 6 \end{pmatrix}$$

Oikean puolen kertoimista saadaan matriisi $B = (10, 20)^T$. *Sitten kirjoitetaan Python-ohjelma, joka ratkaiseen yhtälöryhmän.*

```
#otetaan Numpy-kirjasto käyttöön
from numpy import *
#luodaan matriisit
A = matrix([[2, -3], [5, 6]])
```

```
B = matrix ([[10], [20]])
#ratkaistaan yhtälöryhmä
print linalg.solve(A, B)
```

Ratkaisuksi tulee $x = 4,44\ldots$, $y = -0,3700370\ldots$.
On kuitenkin huomattava, että Python antaa ratkaisut desimaalimuodossa. Mikäli jostain syystä tarvitaan murtolukuesitys, pitää se osata itse muuttaa. Tässäkin tapauksessa ratkaisut ovat jaksollisia desimaalilukuja, joten niille on olemassa murtolukuesitys. Muunto murtolukumuotoon on helppo tarkistaa testaamalla, toteutuuko yhtälöryhmä.

Harjoituksia

Olkoon matriisi $A = \begin{pmatrix} 1 & 1 & 2 \\ -3 & 1 & 0 \\ -1 & 2 & -2 \end{pmatrix}$ ja matriisi $B = \begin{pmatrix} 2 & -2 & 1 \\ 0 & -1 & 1 \\ -2 & 0 & 2 \end{pmatrix}$

Tehtävä 17. *Kirjoita Python-ohjelma, joka tulostaa matriisista A alkion a_{12} ja matriisista B alkion b_{32}*

Tehtävä 18. *Kirjoita Python-ohjelma, joka tulostaa matriisien A ja B summan, erotuksen sekä kummatkin matriisit kerrottuna luvulla 4.*

Tehtävä 19. *Kirjoita Python-ohjelma, joka tulostaa matriisit A ja B kerrottuna keskenään kummassakin järjestyksessä (AB,BA) sekä matriisin B determinantin.*

Tehtävä 20. *Kirjoita Python ohjelma, joka tulostaa matriisin A transpoosin sekä matriisin B käänteismatriisin.*

Tehtävä 21. *Ratkaise Pythonilla yhtälöryhmä*

$$5x_1 + 3x_2 = 9$$
$$4x_1 + 2x_2 = 8$$

Tehtävä 22. *Kirjoita Python-ohjelma, joka ratkaisee seuraavan ylioppilaskirjoituksista peräisin olevan tehtävän.*
Ratkaise lineaarinen yhtälöryhmä

$$3x - 2y = 1$$
$$4x + 5y = 2$$

(Pitkän matematiikan yo-koe, syksy 2001)

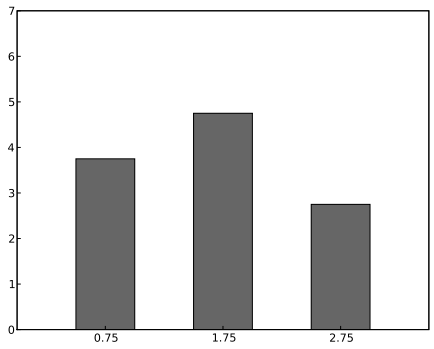
4 Tilastotiedettä

Tässä luvussa kerrataan hieman erilaisia tilastollisia kuvioita sekä menetelmistä pienimmän neliösumman menetelmä. Lopuksi katsotaan kuinka Pythonista on apua tilastotieteen sovelluksissa. Päälähteinä luvussa on käytetty Pii-sarjan kirjaa Tilastot ja todennäköisyys [7] sekä Laskutaito-sarjan kirjaa Laskutaito X [8].

4.1 Pylväsdiagrammit

Tässä kappaleessa palautetaan mieleen pylväsdiagrammi sekä sen piirtäminen.

Määritelmä 4.1. Pylväsdiagrammi on kuvio, jossa pystyakselilla on asteikko ja vaaka-akselilla on pystypylväitä. Yleensä pylväät piirretään yhtä leveiksi, jolloin pylväiden korkeus kuvaa kunkin arvon frekvenssiä.



Kuva 2: Pylväsdiagrammi

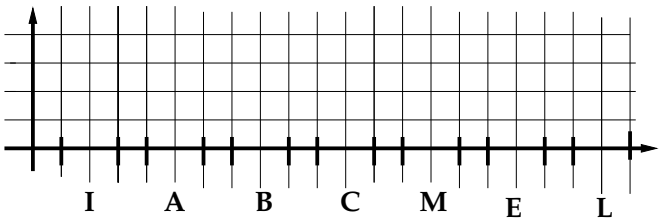
Pylväät voidaan piirtää erikseen tai yhteen. Pylväät on tapana piirtää yhteen, jos kuviolla kuvataan luokiteltua aineistoa, muuten pylväiden väliin jätetään pieni rako. Pylväsdiagrammia, jossa pylväät on piirretty yhteen, kutsutaan yleensä histogrammiksi.

Pylväsdiagrammin piirtäminen on yksinkertaista. Ensin vaaka-akselille merkitään havaintoarvot ja varataan kullekin havaintoarvolle yhtä leveä pylväs. Pylväiden väliin jätetään pieni rako. Jos diagrammi piirretään ruutupaperille, ruutuja on hyvä käyttää apuna. Pystyakselille valitaan sopiva, tasavälinen asteikko kuvaamaan frekvenssejä. Aina ei ole yksiselitteistä, mikä asteikkoväli on sopivin ja myös useampi asteikkoväli voi sopia kyseiseen kuvaajaan. Sitten kunkin havaintoarvon kohdallle piirretään pylväs, jonka korkeus määräytyy havainnon frekvenssin mukaan. Lopuksi vielä nimetään akselit ja lisätään kuvioon otsikko

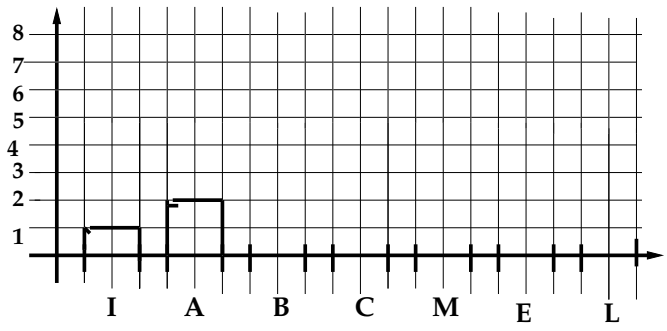
Esimerkki 4.1. *Piirretään pylväsdiagrammi, joka kuvaa erään lukion pitkän matematiikan lukijoiden arvosanoja ylioppilaskirjoituksissa. Havainnot sekä niiden frekvenssit on annettu alla olevassa taulukossa.*

Arvosana	*Oppilasmäärä*
L	*2*
E	*4*
M	*7*
C	*5*
B	*4*
A	*2*
I	*1*

Merkitään ensin vaaka-akselille arvosanat sekä varataan kullekin arvosanalle kahden ruudun levyinen pylväs. Jätetään yksi ruutu pylväiden väliin.

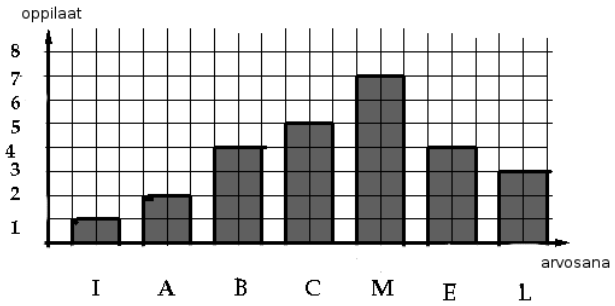


Sitten valitaan asteikko pystyakselille. Tässä tapauksessa on sopivinta käyttää asteikkoa, jossa yksi ruutu vastaa yhtä oppilasta. Sen jälkeen piirretään kutakin frekvenssejä vastaavat pylväät. Arvosanan improbatur sai yksi oppilas, joten pylväs tulee korkeudelle 1, arvosanan approbatur sai kaksi oppilasta, joten pylväs tulee korkeudelle 2 jne.



Lopuksi vielä nimetään akselit ja annetaan kuviolle otsikko. Näin saadaan siis seuraava pylväsdiagrammi.

Ylioppilaskirjoitusten arvosanat



Harjoituksia

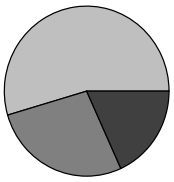
Tehtävä 23. *Alla annetussa taulukossa on tiedot syyskuun keskilämpötilasta Helsingissä vuosina 2004-2007 [9]. Piirrä annetusta aineistosta pylväsdiagrammi.*

Vuosi	*Lämpötila*
2004	*12,8*
2005	*13,1*
2006	*14,0*
2007	*11,9*

4.2 Piirakkakuviot

Tässä kappaleessa palautetaan mieleen piirakkakuvio sekä sen piirtäminen.

Määritelmä 4.2. Piirakkakuvio eli ympyräkuvio on ympyränmuotoinen kuvio. Ympyrä on jaettu sektoreihin ja kunkin sektorin keskuskulma vastaa havaintoarvon osuutta. Koko ympyrä vastaa kaikkia havaintoja.



Kuva 3: Piirakkakuvio

Piirakkakuviot ovat erittäin havainnollisia, jos pitää kuvata osuuksien suuruutta.

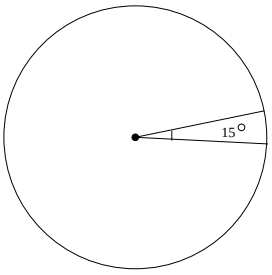
Myös piirakkakuvion piirtäminen on yksikertaista. Ensin lasketaan havaintojen suhteelliset frekvenssit, jos niitä ei ole jo valmiiksi annettu. Sitten lasketaan kuinka suurta keskuskulmaa kukin havainto vastaa ympyrässä. Sen jälkeen piirretään ympyrä ja siihen kutakin keskuskulmaa vastaava sektori. Lopuksi vielä kirjoitetaan kuvioon, mitä havaintoa kukin sektori vastaa ja annetaan kuviolle otsikko. Värejä käyttämälä kuviosta saa havainnollisemman.

Esimerkki 4.2. *Piirretään piirakkakuvio, joka kuvaa keväällä 2006 ylioppilaaksi kirjoittaineiden sijoittumista jatko-opintoihin. Aineisto on annettu alla olevassa taulukossa [9].*

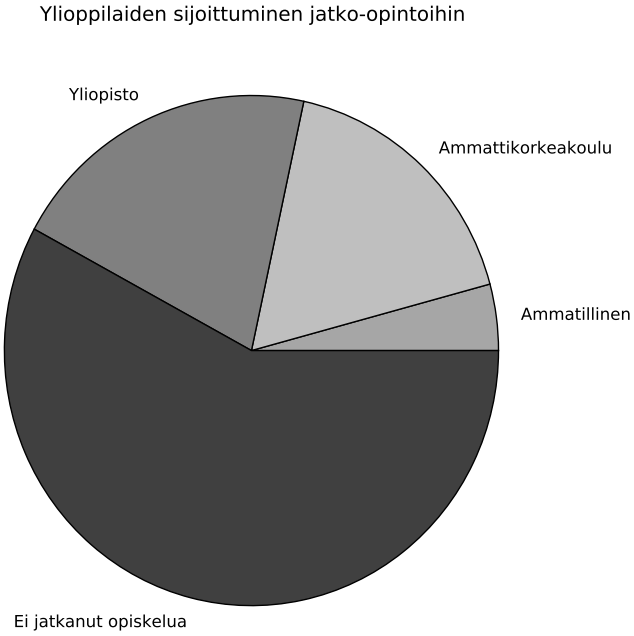
Sijoittumiskohde	Osuus ylioppilaista(%)
toisen asteen ammatillinen koulutus	4,2
ammattikorkeakoulu	17,4
yliopisto	20,5
ei jatkanut opiskelua syksyllä 2006	57,9

Tiedot on annettu jo valmiiksi prosenttiosuuksina, joten niitä ei tarvitse laskea. Lasketaan siis ensin kuinka suurta keskuskulmaa kukin osuus vastaa. Koko ympyrän asteluku on 360, joten kunkin keskuskulman asteluku saadaan laskemalla osuus luvusta 360. Esimerkiksi toisen asteen ammatillisen koulutukseen menijöille kulman suuruus lasketaan $0,042 \cdot 360^\circ \approx 15^\circ$. *Vastaavasti muille saadaan seuraavat keskuskulmat. Ammattikorkeakouluun menevien osuutta vastaa 63 asteen kulma, yliopistoon menevien osuutta 74 asteen kulma ja opiskeluista tauon pitävien osuutta 208 asteen kulma.*

Sitten piirretään ympyrä ja mitataan siitä keskuskulmia vastaavat sektorit.



Kun vielä kirjoitetaan kuvioon, mitä havaintoa kukin sektori vastaa sekä annetaan kuviolle otsikko, saadaan seuraava piirakkakuvio.



Harjoituksia

Tehtävä 24. *Alla olevassa aineistossa on vuoden 2006 joulukuusta vuoden 2007 marraskuuhun tapahtuneet liikennekuolemat jaoteltuna vuodenajoittain [9]. Piirrä piirakkakuvio, joka kuvaa liikennekuolemia vuodenajan mukaan.*

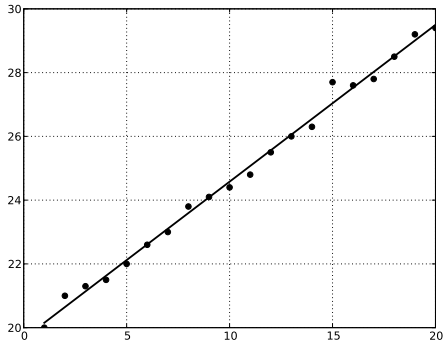
Vuodenaika	*Kuolleiden määrä*
talvi	*81*
kevät	*92*
kesä	*122*
syksy	*89*

4.3 Pienimmän neliösumman menetelmä

Joskus tulee tilanteita, jolloin havaintoihin pitää sovittaa kuvaaja. Kun havainnot merkitään sopivaan koordinaatistoon, usein ne asettuvat sellaiseen muotoon, että niihin on helppo sovittaa kuvaaja. Menetelmiä kuvaajan sovittamiseksi on monia. Yksi niistä on pienimmän neliösumman menetelmä, johon tässä nyt perehdytään. Yksikertaisuuden vuoksi tässä esityksessä tarkastellaan vain suoran sovitusta aineistoon, mutta pienimmän neliösumman menetelmällä voidaan löytää myös muita kuvaajia. Kun kyse on suoran sovittamisesta, menetelmää kut-

sutaan myös lineaariseksi pienimmän neliösumman menetelmäksi [5].

Esimerkki 4.3. *Alla on tehty mittaus, jossa huoneenlämpöistä rautakappaletta lämmitettään kaksikymmentä minuuttia ja sen lämpötila mitattaan minuutin välein. Alla olevassa kuvassa tulokset on sijoitettu aika-lämpötila-koordinaatistoon ja sovitettu suora ainestoon.*



Kuvaajan sovittamisesta aineistoon on useita hyötyjä. Eräs tärkeimmistä on se, että kuvaajasta on apua laadittaessa ennustetta. Edellisessä esimerkissä kuvaajan avulla voitaisiin ennustaa rautakappaleen lämpötilaa, kun sitä on lämmitetty kolmekymmentä minuuttia.

Pienimmän neliösumman menetelmä perustuu siihen, että minimoidaan havaintopisteiden suorasta pystysuoraa mitattujen etäisyyksien neliöiden summaa. Kyseinen summalauseke on muotoa

$$S(a,b) = \sum_{i=1}^{n} (y_i - (bx_i + a))^2.$$

Minimi kyseiselle summalle löydetään kun derivoidaan lauseke sekä muuttujan a että muuttujan b suhteen ja katsotaan, milloin kumpikin derivaatta on nolla. Tällä ehdolla saadaan yhtälöpari

$$\begin{aligned} na + b\sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} y_i \\ a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} x_i y_i \end{aligned}$$

missä n havaintojen lukumäärä. Yhtälöparista saadaan ratkaistua tuntemattomat a ja b, jotka ovat sopivimman suoran $y = bx + a$ kertoimet.

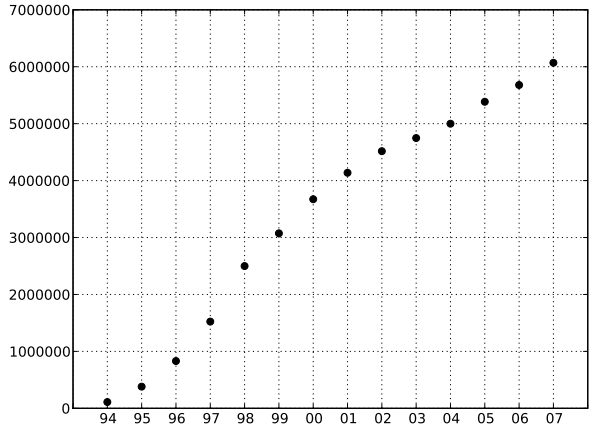
Esimerkki 4.4. *Oheisessa taulukossa on esitetty gsm-liittymien määrä Suomessa vuosina* $1995 - 2007$ *[9]. Sovitetaan suora tähän aineistoon pienimmän neliösumman menetelmällä.*

Vuosi	1994	1995	1996	1997	1998
Liittymien lkm	110 155	380 703	830 585	1 523 356	2 498 793

Vuosi	1999	2000	2001	2002	2003
Liittymien lkm	3 073 942	3 672 762	4 137 337	4 516 772	4 747 126

Vuosi	2004	2005	2006	2007
Liittymien lkm	4 999 060	5 384 572	5 679 010	6 069 463

Merkitään ensin pisteet vuosi-lukumäärä-koordinaatistoon



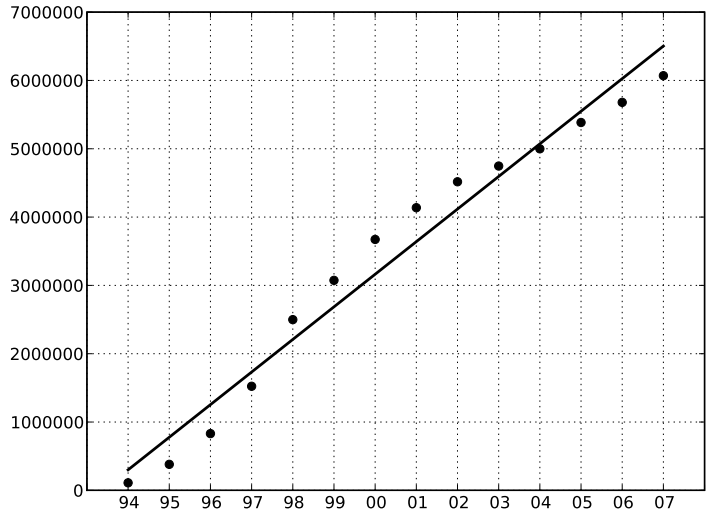
Sitten etsitään sopivin suora pienimmän neliösumman menetelmällä. Havaintoja on nyt 14*, joten* $n = 14$*. Laskentaa varten merkitään vuosia numeroilla siten, että vuosi* 1994 *vastaa lukua* 1*, vuosi* 1995 *lukua* 2 *jne. Summalauseke* $\sum_{i=1}^{n} x_i = 1 + 2 + 3 + \cdots + 14$ *saa siis arvon* 105*. Vastaavasti summalauseke* $\sum_{i=1}^{n} y_i = 110155 + 380703 + \cdots + 6069463$ *saa arvon* 47 623 636*. Yhtälöryhmästä ylempi yhtälö saa siis muodon* $14a + 105b = 47\,623\,636$*.*

Alemmassa yhtälössä tarvittavista summalausekkeista
$\sum_{i=1}^{n} x_i^2 = 1^2 + 2^2 + \cdots + 14^2$ *saa arvon* 1015 *ja lauseke*
$\sum_{i=1}^{n} x_i y_i = 1 \cdot 110155 + 2 \cdot 380703 + \cdots + 14 \cdot 6069463$ *saa arvon* 465 728 731*.*
Näin saadaan alempi yhtälö muotoon $105a + 1015b = 465\,728\,731$*.*

Yhtälöryhmä on siis nyt muotoa

$$14a + 105b = 47\,623\,636$$
$$105a + 1015b = 465\,728\,731$$

Kun ratkaistaan yhtälöryhmä, saadaan ratkaisuksi $a \approx -176931$ *ja* $b \approx 477149$. *Näin ollen optimaalisin suora on muotoa* $y = 477149x - 176931$. *Piirretään se siis samaan koordinaatistoon pisteiden kanssa.*



Tehdään vielä kuvaajan pohjalta ennuste, kuinka monta gsm-liittymää suomessa on vuonna 2009. *Nyt siis* $x = 16$, *joten* $y = 477149 \cdot 16 - 176931 = 7\ 457\ 453$. *Tämän mukaan Suomessa olisi siis vuonna* 2009 *gsm-liittymiä* 7 457 453 *kappaletta.*

Harjoituksia

Tehtävä 25. *Oheisessa aineistossa on keskimääräinen neliöhinta vuokra-asunnoissa vuodesta 1993 vuoteen 2006 [9]. Sovita suora aineistoon pienimmän neliösumman menetelmällä. Mikä näiden tietojen mukaan olisi keskimääräinen neliövuokra vuonna 2010?*

1990-luku

Vuosi	*93*	*94*	*95*	*96*	*97*	*98*	*99*
Neliöhinta (e/m^2)	*5,86*	*5,97*	*5,99*	*6,26*	*6,78*	*7,03*	*7,28*

2000-luku

Vuosi	*00*	*01*	*02*	*03*	*04*	*05*	*06*
Neliöhinta (e/m^2)	*7,67*	*7,99*	*8,21*	*8,50*	*8,60*	*8,81*	*8,93*

4.4 Sovelluksia Python-ohjelmointiin

Pythonilla voidaan piirtää erilaisia kuvioita Matplotlib-kirjaston avulla. Ensiksi tutustutaan siis hieman Matplotlib-kirjaston perusteisiin eli kuinka Pythonilla voidaan kuvaajia piirtää. Sitten katsotaan kuinka Pythonilla piirretään pylväsdiagrammeja ja piirakkakuvioita sekä sovitetaan suora pienimmän neliösumman menetelmää käyttäen. Päälähteenä on käytetty Matplotlib-kirjaston tutoriaalia [10].

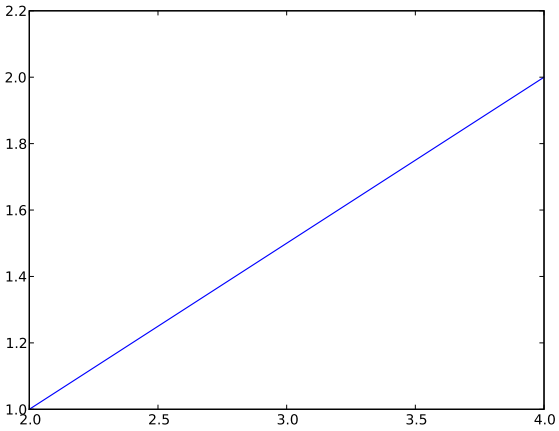
Kuvaajien piirtämiseen Pythonissa käytetään siis Matplotlib-kirjastoa. Se otetaan käyttöön kirjoittamalla lähdekooditiedoston alkuun `from pylab import *`. Tiedoston loppuun kirjoitetaan aina myös komento `show()`, jolloin ohjelma myös näyttää kuvaajan.

Suoran piirtäminen Pythonissa tapahtuu komennolla `plot`. Komento on itse asiassa funktio, joten se tarvitsee argumentteja. Komennolle annetaan listana tiedot x- ja y-koordinaateista, joiden kautta suora kulkee. Tässä vaiheessa on hyvä muistaa, että jo kaksi pistettä riittää määrittämään suoran yksikäsitteisesti. Pisteiden koordinaatit annetaan niin, että x-koordinaatit annetaan omana listana ja y-koordinaatit omana. Ensimmäisen pisteet x-koordinaatti on ensimmäisenä x-koordinaattien listassa ja ensimmäisen pisteen y-koordinaatti ensimmäisenä y-koordinaattien listassa.

Esimerkki 4.5. *Piirretään Pythonin avulla suora, joka kulkee pisteiden (2,1) ja (4,2) kautta.*

```
#luodaan lista x-koordinaateille
x=[2,4]
#luodaan lista y-koordinaateille
y=[1,2]
#piirretään suora
plot(x,y)
#näytetään kuvaaja
show()
```

Ohjelma tulostaa kuvan



Python piirtää oletuksena sinisen viivan. Komennolle `plot` voi antaa argumenttina myös värejä. Kullekin värille on olemassa sitä merkitsevä kirjain. Alla on taulukoitu yleisimpien värien kirjaintunnukset.

Taulukko 3: Värien kirjaintunnukset Pythonissa

Väri	Kirjaintunnus
sininen	b
vihreä	g
punainen	r
keltainen	y
musta	k
valkoinen	w

Tässä materiaalissa on käytetty painoteknisistä syistä harmaasävyjä. Harmaasävyjen kohdalla väriä merkitsevä kirjain korvataan luvulla väliltä [0,1]. Luku osoittaa sävyä siten, että 0 on musta ja 1 valkoinen.

Komennolla `plot` annettujen pisteiden väliin piirretään automaattisesti viiva. Värin perään voi myös laittaa määreen, jos haluaa piirtää annetut pisteet jollain muulla tavalla kuin suoran osana. Alla on taulukoitu muutamia tapoja.

Taulukko 4: Erilaisia piirtotapoja Pythonissa

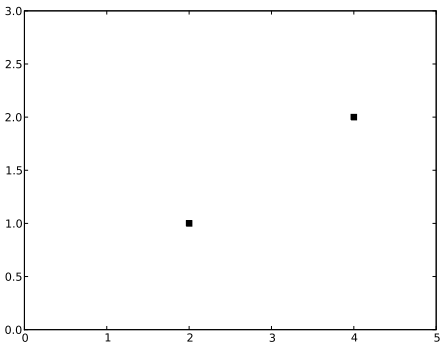
Piirtotapa	Kirjaintunnus
ympyrät	o
neliöt	s
kolmiot	^

Myös akseleita voi säädellä itselleen mieluisammiksi. Tämä tapahtuu kommennolla `axis`. Funktiolle annetaan argumenttina lista, johon on tallennettu x-akselin alku- ja päätepisteet sekä y-akselin alku- ja päätepisteet.

Esimerkki 4.6. *Piirretään edellisen esimerkin suoran sijasta nyt mustat neliöt sekä laitetaan x-akseli kulkemaan nollasta viiteen ja y-akseli nollasta kolmeen.*

```
#luodaan lista x-koordinaateille
x=[2,4]
#luodaan lista y-koordinaateille
y=[1,2]
#piirretään suora, määre 'ks' piirtää mustat neliöt
plot(x,y,'ks')
#luodaan lista akselien määräämistä varten
akselit=[0,5,0,3]
axis(akselit)
#näytetään kuvaaja
show()
```

Ohjelma tulostaa kuvan



Seuraavassa esimerkissä esitellään koodipätkä, jonka avulla voidaan piirtää pylväsdiagrammeja.

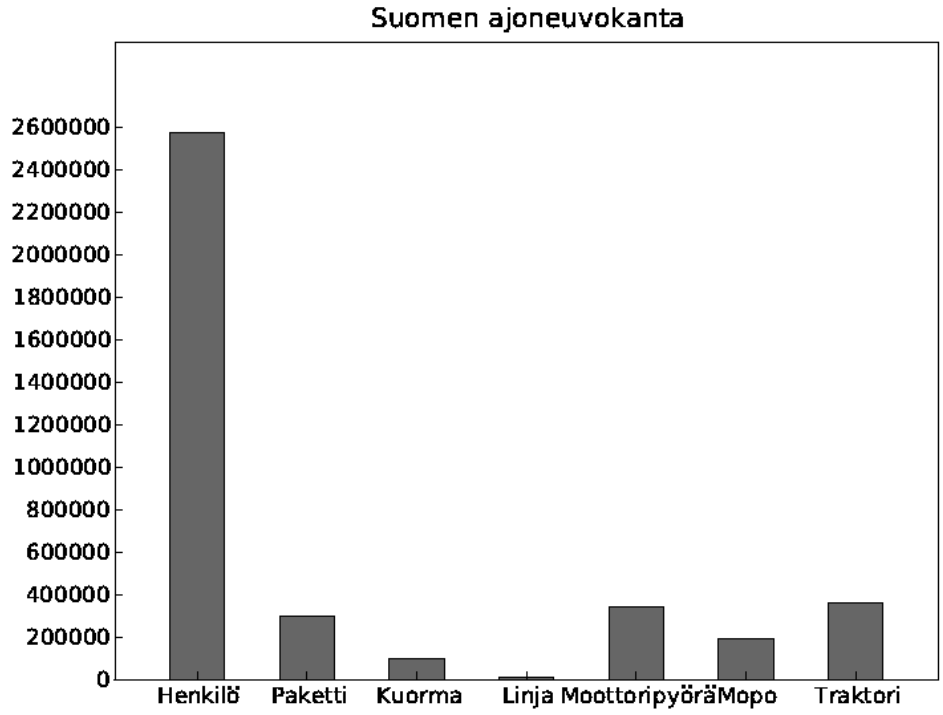
Esimerkki 4.7. *Piirretään Pythonilla pylväsdiagrammi, joka kuvaa eri moottoriajoneuvojen lajeja Suomessa vuonna 2007. Aineisto on annettu ohessa [9]. Alla on koodi, joka piirtää pylväsdiagrammin.*

Ajoneuvolaji	Määrä
Henkilöauto	2 570 356
Pakettiauto	297 531
Kuorma-auto	97 187
Linja-auto	11 543
Moottoripyörä	339 851
Mopo	188 388
Traktori	357 911

```
#koodissa käytetään apuna numarray-tietorakennetta, joten
#otetaan se käyttöön ja määritellään sille lyhennys-
#merkintä na
import numpy.numarray as na

#otetaan Matplotlib käyttöön
from pylab import *

#nimetään pylväät
labels = ["Henkilö", "Paketti", "Kuorma", "Linja","Moottoripyörä",
         "Mopo","Traktori"]

#pylväiden korkeudet eli frekvessit
data =   [2570356,297531,97187,11543,339851,188388,357911]

#luodaan numarray-tyyppinen lista, joka sisältää tiedot
#pylväiden sijainnista
xlocations = na.array(range(len(data)))+0.5
#määritetään pylvään leveys
width = 0.5
#bar-komento piirtää pylväät, color määrittää pylvään värin
bar(xlocations, data, color='0.4' , width=width)
#luodaan y-akseli asteikkoineen
yticks(range(0, 2600001, 100000))
#luodaan x-akseli
xticks(xlocations+width/2, labels)
xlim(0, xlocations[-1]+width*2)
```

```
#kuvaajan otsikko
title("Suomen ajoneuvokanta")
#näytetään asteikko ainoastaan kuvaajan alla
gca().get_xaxis().tick_bottom()
#näytetään asteikko ainoastaan kuvaajan vasemmalla
#puolella
gca().get_yaxis().tick_left()

#näytetään kuvaaja
show()
```

Ohjelma piirtää seuraavan kuvaajan.



Seuraavassa esimerkissä esitellään koodipätkä, jonka avulla voidaan piirtää piirakkakuvioita

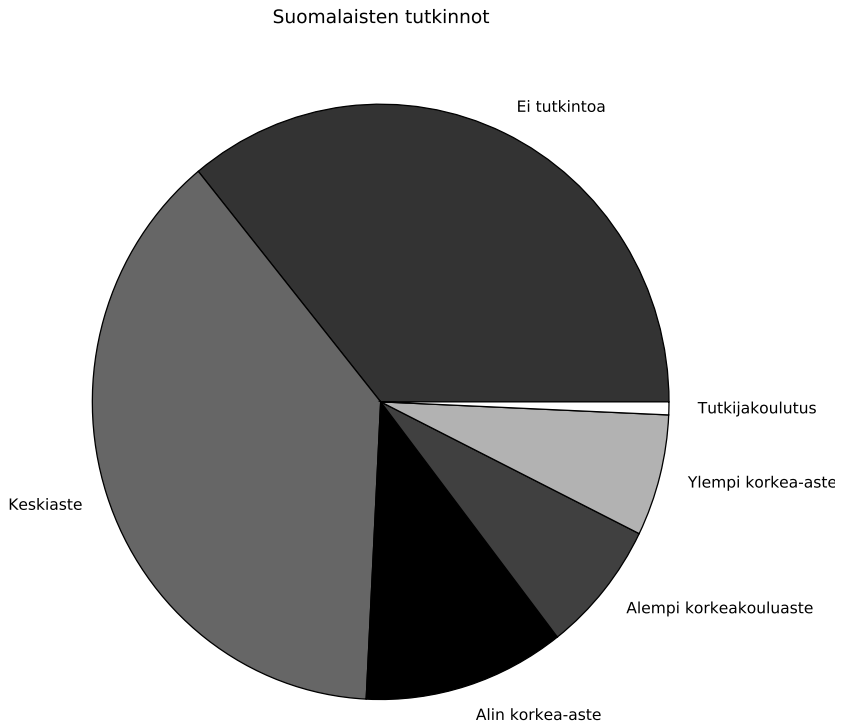
Esimerkki 4.8. *Piirretään Pythonin avulla piirakkakuvio, joka kuvaa 15 vuotta täyttäneiden suomalaisten suorittamia tutkintoja. Aineisto on annettu ohessa [9].*

Tutkinto	Osuus väestöstä (%)
Ei tutkintoa perusasteen jälkeen	35,9
Keskiaste	38,3
Alin korkea-aste	11,3
Alempi korkeakouluaste	7,2
Ylempi korkeakouluaste	6,6
Tutkijakoulutus	0,7

Alla on koodi, joka piirtää piirakkakuvion.

```
#otetaan matplotlib käyttöön
from pylab import *

#annetaan eri sektoreiden osuudet listana
fracs=[35.9,38.3,11.3,7.2,6.6,0.7]

#nimetään sektorit
labels=["Ei tutkintoa", "Keskiaste", "Alin korkea-aste",
"Alempi korkeakouluaste","Ylempi korkeakouluaste","Tutkijakoulutus"]
#annetaan kullekin sektorille väri
colors=('0.2','0.4','0','0.25', '0.7','1')

#määritellään taustäväri ja kuvan koko. Tässä tapauksessa
#se kannattaa olla neliönmuotoinen eli pituus ja
#leveys yhtäsuuret.
figure(figsize=(8,8),facecolor='w')
#pie-komento piirtää itse kuvion
pie(fracs, colors=colors, labels=labels)
#annetaan kuvaajalle otsikko
title("Suomalaisten tutkinnot")
#näytetään kuvaaja
show()
```

Ohjelma siis tulostaa kuvan



Seuraavassa esimerkissä on esitelty koodipätkä, jonka avulla voidaan piirtää suora pienimmän neliösumman menetelmällä.

Esimerkki 4.9. *Oheisessa aineistossa on annettu henkilöautojen lukumäärä Suomessa vuodesta 1984 vuoteen 2006 [9]. Sovitetaan Pythonilla suora kyseiseen aineistoon.*

1980-luku

Vuosi	*1984*	*1985*	*1986*	*1987*	*1988*	*1989*
Autojen määrä	*1473975*	*1546094*	*1619848*	*1698671*	*1795908*	*1908971*

1990-luku

Vuosi	*1990*	*1991*	*1992*	*1993*	*1994*	*1995*
Autojen määrä	*1938856*	*1922541*	*1936345*	*1872933*	*1872588*	*1900855*
Vuosi	*1996*	*1997*	*1998*	*1999*	*2000*	
Autojen määrä	*1942752*	*1948126*	*2021116*	*2082580*	*2134728*	

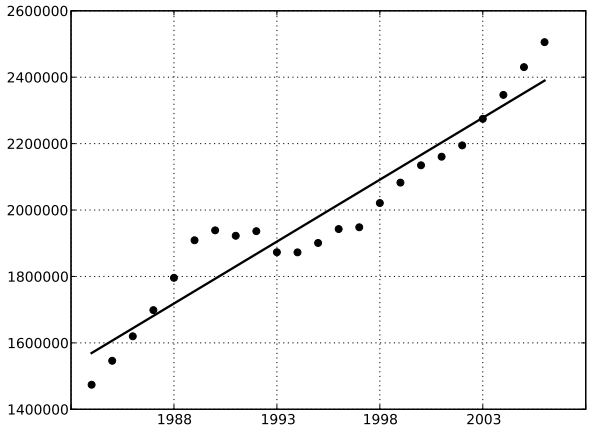
2000-luku

Vuosi	*2001*	*2002*	*2003*	*2004*	*2005*	*2006*
Autojen määrä	*2160603*	*2194683*	*2274577*	*2346726*	*2430345*	*2505543*

Alla olevalla koodilla saadaan suora piirrettyä.

```
#otetaan matplotlib käyttöön
from pylab import *
#annetaan pisteiden x-koordinaatit listana
x=[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0,
    14.0,15.0,16.0,17.0,18.0,19.0,20.0,21.0,22.0,23.0]
#annetaan pisteiden y-koordinaatit listana
y=[1473975,1546094,1619848,1698671,1795908,1908971,1938856,
   1922541,1936345,1872933,1872588,1900855,1942752,1948126,
    2021116,2082580,2134728,2160603,2194683,2274577,2346726,
   2430345,2505543]
#tehdään pienimmän neliösumman sovitus. Parametri 1 kertoo
#sovitettavan kuvaajan asteen. Tässä tapauksessa sovituksen
#ollessa lineaarinen aste on 1.
m,b=polyfit(x,y,1)
#luodaan x-akseli
xlocations=array(range(0,len(x)+4,5))
labels=["","1988 ","1993","1998 ","2003"]
xticks(xlocations,labels)
#piirretään suora
plot(x,y,'ko',x, [m*i+b for i in x], '-k',linewidth=2)
#otetaan ruudutus käytöön
grid(True)
#näytetään kuvaaja
show()
```

Ohjelma tulostaa seuraavan kuvaajan.



Harjoituksia

Tehtävä 26. *Alla on esitetty vuonna 2007 syntyneiden lasten määrä jaoteltuna äidin iän mukaan [9]. Piirrä aineistosta Pythonin avulla pylväsdiagrammi.*

Äidin ikä	*Syntyneiden määrä*
20-24	*9467*
25-29	*18673*
30-34	*18206*
35-39	*8721*
40-44	*2095*

Tehtävä 27. *Alla olevassa aineistossa on esitelty eri energialähteiden käyttöä Suomessa vuonna 2007 [11]. Piirrä aineistosta Pythonin avulla piirakkakuvio.*

Energialähde	*Osuus (%)*
Öljy	*24,4*
Maakaasu	*10,3*
Hiili	*12,9*
Ydinvoima	*16,6*
Sähkön nettotuonti	*3,1*
Turve	*7,2*
Vesi- ja tuulivoima	*3,4*
Puuperäiset polttoaineet	*20,2*
Muu	*1,9*

Tehtävä 28. *Oheisessa aineistossa on kuvattu työllisten määrää Suomessa vuodesta 1993 vuoteen 2006 [9]. Sovita aineistoon suora Pythonin avulla pienimmän neliösumman menetelmällä.*

Vuosi	1993	1994	1995	1996	1997
Työllisten määrä	1877721	1917051	1932752	1957144	2037997
Vuosi	1998	1999	2000	2001	2002
Työllisten määrä	2132704	2173885	2228557	2235317	2242303
Vuosi	2003	2004	2005	2006	
Työllisten määrä	2245780	2262359	2265211	2313788	

5 Numeerista matematiikkaa

Ohjelmoinnin hyöty matematiikassa tulee parhaiten esille numeerisen matematiikan osa-alueella. Tässä luvussa käydään läpi koulumatematiikasta tuttu Newtonin menetelmä ja sen sovelluksia Python-ohjelmointiin. Esimerkeissä ja harjoitustehtävissä käytetään jonkin verran ylioppilastehtäviä, koska Newtonin menetelmä on ollut viime vuosina suosittu kysymys ylioppilaskirjoituksissa. Päälähteenä luvussa on käytetty Burdenin ja Fairesin kirjaa Numerical Analysis [12].

5.1 Numeerisen laskennan luotettavuudesta

Tässä kappaleessa käsitellään lyhyesti numeerisen laskennan luotettavuutta. Vaikka tässä materiaalissa tulevat esimerkit ovat yksinkertaisia, jolloin tarkkuudella ei ole niin suurta merkitystä, on hyvä tietää, että tietokoneella laskettu vastaus harvoin on tarkka.

Numeerinen laskenta käyttää pääasiassa reaalilukuja, joita tietokoneessa käsitellään desimaaliesityksen avulla. Tässä yhteydessä desimaaliesityksestä käytetään nimitystä liukulukuesitys. Liukulukuesitykselle on käytettävissä kiinteänkokoinen tila koneen muistista, joten esityksen tarkkuus on rajoitettu. Pythonissa tämä tila on 32 bittiä. Pythonille on olemassa kirjastoja, joiden avulla tarkkuutta voidaan parantaa [13]. Kirjastoilla voidaan saavuttaa jopa mielivaltaisen tarkkoja arvoja. Kirjaston käyttö lisää toki laskenta-aikaa, joten on pohdittava kuinka tarpeellista tarkentaminen on. Kaikille ohjelmointikielille tällaisia kirjastoja ei ole saatavilla, joten siksi on hyvä huomioida tarkkuus numeerista laskentaa suoritettaessa.

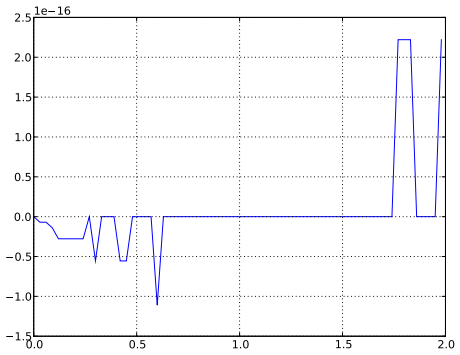
Seuraavalla yksinkertaisella esimerkillä voidaan havainnollistaa esitystarkkuuden rajallisuutta.

Esimerkki 5.1.

```
#otetaan Numpy-kirjasto käyttöön, sillä sieltä
#tarvitaan arange-funktiota, arange toimii kuten
#range, mutta se tulostaa arvot taulukoksi
from numpy import *
#otetaan Matplotlib-kirjasto käyttöön, sillä
#piirretään kuvaaja
from pylab import *

#määritellään muuttujaksi p jokin
#mielivaltainen liukuluku
p=1.2345
```

```
#määritellään fuktioksi jokin funktio,
#joka selvästi saa aina arvon nolla
def f(x):
        y=pow(pow(x,p),1./p) - x
        return y

#generoidaan arange-funktion avulla taulukko
#muuttujan t arvoista, jotka sijoitetaan funktioon
t=arange(0,2,0.03)
#piiretään kuvaaja
plot(t,f(t))
#otetaan ruudutus käyttöön
grid(True)
#näytetään kuvaaja
show()
```

Ohjelma tulostaa seuraavan kuvaajan.



Määritelty funktio on identtisesti nolla, joten kuvaajan pitäisi olla vain suora viiva. Kuten nähdään se ei ole suora viiva, vaan se kulkee vuoroin nollan alapuolella ja vuoroin yläpuolella. Tämä johtuu nimenomaan siitä, eksponentin p arvoa ei pystytä nyt tarkasti määrittämään.

Useat numeeriset menetelmät perustuvat iterointiin eli edellisestä iteraatioaskeelesta saatu tulos sijoitetaan seuraavaan askeleeseen. Näin myös laskennasta johtuva virhe kasvaa, mitä pidemmälle laskennassa edetään.

5.2 Newtonin menetelmä

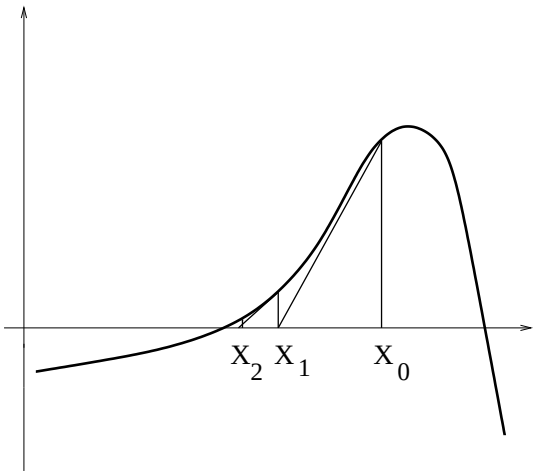
Newtonin menetelmällä ratkaistaan yleensä yhtälöitä, joilla ei ole analyyttista ratkaisua. Toisin sanoen etsitään funktiolle nollakohtaa.

Olkoon $f : [a, b] \to \mathbb{R}$ jatkuva ja derivoituva funktio, jonka arvot välin päätepisteissä ovat eri merkkiset. Koska funktion merkki vaihtuu määrittelyvälillä, tiedetään, että funktiolla on ainakin yksi nollakohta kyseisellä välillä. Nollakohdalle voidaan löytää likiarvo Newtonin menetelmällä.

Newtonin algoritmi toimii siten, että valitaan aloituspisteeksi $x_0 \in (a, b)$ ja määritellään rekursiivisesti

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ n = 0, 1, 2, \ldots$$

Määritelmä on mielekäs, jos $x_n \in (a, b)$ ja $f'(x) \neq 0$.



Kuva 4: Newtonin menetelmä geometrisesti esitettynä

Jonon (x_n) ei tarvitse supeta kohti mitään raja-arvoa. Suppenemistapauksessa Newtonin iteraatio on hyvin tehokas ja se suppenee useimmiten riittävään tarkkuuteen jo muutamalla askeleella. Jonon (x_n) luonne, suppeneeko se vai ei, riippuu aloituspisteen valinnasta. Näin ollen mikäli jono ei jo muutamien iteraatioiden jälkeen selvästi suppene kohti tiettyä arvoa, kannattaa tarkistaa aloituspisteen valinta. Ei ole myöskään selvää tapahtuuko suppeneminen kohti etsittyä juurta vai kohti jotain muuta juurta.

Aloituspisteen valintaan ei ole mitään yksikäsitteistä keinoa, vaan on vain tehtävä hyvä arvaus. Aloituspisteen valinnassa voidaan käyttää apuna esimerkiksi funktion kuvaajaa.

Newtonin menetelmällä saadaan siis vain arvio nollakohdalle, ei tarkkaa vastausta. Saadun tuloksen tarkuuden arviointiin on useita keinoja. Seuraavaksi esitellään niistä yksi.

Kuten aikaisemmin todettiin, funktiolla on nollakohta jollakin välillä, mikäli funktion arvot välin päätepisteissä ovat eri merkkiset. Näiden arvojen tulo on siis selvästi negatiivinen. Iteroitaessa Newtonin menetelmää eteenpäin voidaan laskea funktion arvojen tulo saadun nollakohdan ympäristössä. Mikäli tulo on negatiivinen, on saatu arvio riitävän lähellä nollakohtaa.

Ympäristön valinta riippuu halutusta tarkkuudesta. Mitä tarkempi vastaus halutaan, sitä pienempi ympäristö on valittava. Jos haluttu tarkkuus on esimerkiksi viisi desimaalia eli $1,0 \cdot 10^{-5}$, valitaan tarkasteltavaksi väliksi $[x_n - h, x_n + h]$, missä x_n on saatu arvio ja $h = 0,5 \cdot 10^{-5}$. Vakiolla h merkitään yleensä itseisarvoltaan pientä lukua. Tarkemmin tästä vakiosta kerrotaan seuraavassa kappaleessa.

Esimerkki 5.2. *Etsitään Newtonin menetelmällä nollakohta funktiolle* $f(x) = x - \sin\ x$ *eli ratkaisu yhtälölle* $x - \sin\ x = 0$ *kolmen desimaalin tarkkuudella.*

Funktiolla on nollakohta välillä [-1,1]. Tämä on helppo todeta laskemalla funktion arvot kyseisissä pisteissä. Kohdassa $x = -1$ *funktio saa arvon* $f(-1) = -1 - \sin(-1) \approx -0,159$ *ja vastaavasti kohdassa* $x = 1$ *arvon 0,159. Funktion arvon merkki vaihtuu, joten välillä on oltava nollakohta.*

Aloitetaan iteroiminen siis kohdasta $x = 1 = x_0$. *Iteraatiokaavassa tarvitaan funktion derivaattaa. Tällä kertaa se on* $f'(x) = 1 - \cos\ x$. *Nyt saadaan ensimmäinen arvio nollakohdalle. Se on siis* $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1 - \frac{f(1)}{f'(1)} \approx 0,655145$. *Arvioidaan sitten saadun arvion tarkkuus. Koska haluttu tarkkuus oli nyt kolme desimaalia, valitaan* $h = 0,5 \cdot 10^{-3}$. *Lasketaan siis* $f(0,655145 - 0,5 \cdot 10^{-3}) \cdot f(0,655145 + 0,5 \cdot 10^{-3})$. *Tulo saa nyt arvon 0,00210, joka on positiivinen. Jatketaan siis iterointia.*

Nyt $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \approx 0,433590$. *Arvioitaessa tarkkuutta saadaan* $f(0,433590 - 0,5 \cdot 10^{-3}) \cdot f(0,433590 + 0,5 \cdot 10^{-3}) = 0,000181 > 0$. *Jatketaan siis iterointia.*

Alla olevassa taulukossa on laskettu tietokoneen avulla iteraatiota eteenpäin.

n	x_n	$f(x_n - h) \cdot f(x_n + h)$
3	0,288148	$1.58 \cdot 10^{-5}$
4	0,191832	$1,38 \cdot 10^{-6}$
5	0,127810	$1,21 \cdot 10^{-7}$
6	0,085183	$1,06 \cdot 10^{-8}$
7	0,056782	$9,30 \cdot 10^{-10}$
8	0,037853	$8,17 \cdot 10^{-11}$
9	0,025234	$7,16 \cdot 10^{-12}$
10	0,016823	$6,28 \cdot 10^{-13}$
11	0,011215	$5,50 \cdot 10^{-14}$
12	0,0074768	$4,79 \cdot 10^{-15}$
13	0,0049845	$4,13 \cdot 10^{-16}$
14	0,0033230	$3,49 \cdot 10^{-17}$
15	0,0022153	$2,81 \cdot 10^{-18}$
16	0,0014769	$2,00 \cdot 10^{-19}$
17	0,0009846	$1,03 \cdot 10^{-20}$
18	0,0006564	$1,64 \cdot 10^{-22}$
19	0,0004376	$-5,56 \cdot 10^{-24}$

Kuten huomataan yhdesännellätoista iteraatiokerralla, saavutetaan haluttu tarkkuus, sillä silloin funktion arvojen tulo on ensimmäistä kertaa negatiivinen. Kolmen desimaalin tarkkuudella funktion nollakohta on siis $x = 0,000$. Mikäli haluttaisiin vielä tarkempi vastaus jatkettaisiin iterointia.

Harjoituksia

Tehtävä 29. *Etsi Newtonin menetelmällä funktion $f(x) = x + e^x$ nollakohta kolmen desimaalin tarkkuudella.*

5.3 Numeerinen derivointi

Laskettaessa tietokoneella Newtonin menetelmän avulla tarvitaan käyttöön numeerinen derivaatta. Tässä kappaleessa käydäänkin lyhyesti läpi numeerinen derivointi.

Palautetaan mieleen derivaatan määritelmä. Derivaattahan määriteltiin erotusosamäärän raja-arvona.

Määritelmä 5.1. Olkoon funktion f määritelty pisteen x ympäristössä. Silloin funktion f derivaatta pisteessä x on

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Mikäli derivaatalle ei saada muodostettua tarkkaa esitysmuotoa, voidaan sille muodostaa arvio sen määritelmän avulla. Muuttujan x paikalle sijoitetaan piste, jossa derivaattaa etsitään. Koska vakio h lähestyy nollaa, valitaan sen arvoksi riittävän läheltä nollaa jokin luku.

Esimerkki 5.3. *Lasketaan funktion* $f(x) = \cos\ 2x + e^x$ *kohdassa* $x = 1$ *silloin kun* $h = 10^{-5}$. *Sijoitetaan nyt siis derivaatan määritelmään muuttujan* x *paikalle luku 1 ja vakion* h *paikalle luku* 10^{-5}.

$$\begin{aligned} f'(1) &\approx \frac{f(1+10^{-5}) - f(1)}{10^{-5}} = \frac{(\cos\ 2 \cdot (1,0001) + e^{1,0001}) - (\cos\ 2 \cdot 1 - e^1)}{10^{-5}} \\ &\approx 0,89971 \end{aligned}$$

Jos derivoidaan funktio ja sijoitetaan derivaatta funktioon muuttujan x *arvo 1, niin tulokseksi saadaan* $0,89969$. *Derivoinnin yksityikohtainen suorittaminen jätetään harjoitustehtäväksi. Joka tapauksessa huomataan, että numeerisella derivoinnilla saatiin hyvä arvio derivaatan arvolle.*

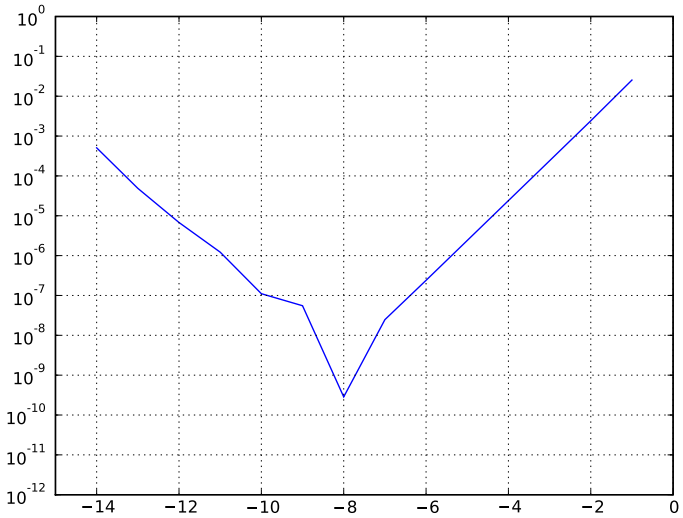
Numeerisen derivaatan virhe riippuu vakion h valinnasta. Seuraavaksi tarkastellaan, millä vakion h arvolla virhe on pienin.

Tarkastellaan funktion $\sin\ x$ derivaattaa pistessä $x = \frac{1}{2}$. Kirjoitetaan Python-ohjelma, joka laskee numeerisesti derivaatalle arvion eri vakion h arvoilla sekä virheen kullekin arviolle. Lisäksi ohjelma piirtää virheen vakion h arvon funktiona puolilogaritmiseen koordinaatistoon.

```
#koska piirretään kuvaajia, otetaan
#Matplotlib-kirjasto käyttöön
from pylab import *
#piste, jossa derivaattaa lasketaan
s=0.5
#luodaan tyhjä lista, johon tallennetaan h:n
#potenssit
x=[]
#luodaan tyhjä lista, johon tallennetaan
#virhe kullakin h:n arvolla
z=[]
#annetaan muuttujan j käydä läpi arvot
#yhdestä neljääntoista
for j in range(1,15):
```

```
        #tallennetaan listaan x muuttujan j
        #vastaluvut
        x=x+[-j]
        #laitetaan j:n vastaluku h:n
        #kymmenpotenssiksi
        h=10**(-j)
        #tallennetaan listaan z virhe kullakin
        #h:n arvolla
        z=z+[abs(cos(s)-((sin(s+h)-sin(s))/h))]
#piirretään kuvaaja puolilogaritmiseen koordinaatistoon
#semilogy-käsky toimii kuten plot-käsky
semilogy(x,z)
#otetaan ruudutus käyttöön
grid(True)
#muutetaan akselien asteikot sopiviksi
axis([-15,0,10**(-12),10**0])
#näytetään kuvaaja
show()
```

Ohjelma tulostaa seuraavan kuvaajan.



Kuva 5: Numeerisen derivoinnin virhe

Kuten kuvaajasta huomataan virhe on pienimillään, kun vakion h kymmenen potenssiksi valitaan luku -8. Käytetään jatkossa siis vakion h arvona $h = 10^{-8}$.

5.4 Sovelluksia Python-ohjelmointiin

Seuraavassa esimerkissä esitellään ohjelma, joka ratkaisee yhtälöitä käyttäen Newtonin menetelmää.

Esimerkki 5.4. *Etsitään arvio viiden desimaalin tarkkuudella funktion* $f(x) = \sin\ 2x + \frac{x}{2}$ *nollakohdalle Pythonia hyödyntäen. Alla on koodi, jossa on hyödynnetty Newtonin menetelmää.*

```
#otetaan math-kirjasto käyttöön
import math

#määritellään funktio f
def f(x):
        return math.sin(2*x)+x/2

#funktion f derivaatta pisteessä x
def derivaatta(f, x, h=1.0e-8):
        return (f(x+h)-f(x))/h

#Newtonin menetelmä
def newton(x0, tarkkuus, N=100):
        #alustetaan iteraatioiden määrä yhdeksi
        i=1
        while i<N:
                #lasketaan tarkempi x
                x=x0-(f(x0)/derivaatta(f,x0,h=1.0e-8))
                #testataan vastauksen tarkkuus
                if f(x-(tarkkuus/2))*f(x+(tarkkuus/2)) < 0:
                      return x
                #jos vastaus on riittävän tarkka,
                #lopetetaan ohjelman suoritus,
                #muuten jatketaan iterointia
                i=i+1
                x0=x
         #mikäli riittävän tarkkaa vastausta ei saada
         #sadassa iteraatiossa käyttäjää pyydetään
         #tarkistamaan alkuarvot
         print ("Tarkista alkuarvot")

#kysytään käyttäjältä alkuarvot
alkux=input("Anna aloituspiste:")
```

```
desimaalit=input("Anna tarkkuus:")

#suoritetaan Newtonin menetelmä annetuilla arvoilla
print newton(alkux,desimaalit)
```

Funktiolla on nyt nollakohta välillä $[-\frac{1}{2}, \frac{1}{2}]$*, sillä* $f(-\frac{1}{2}) \approx -1,091$ *ja* $f(\frac{1}{2}) = 1,091$*. Aloitetaan iteroiminen siis pisteestä* $x = \frac{1}{2}$*. Arviota pyydetiin viiden desimaalin tarkkuudella, joten tarkkuudeksi annetaan 0,00001. Kun nämä arvot syötetään ohjelmaan, ohjelma tulostaa luvun* $-4,89855422751 \cdot 10^{-7}$*. Viiden desimaalin tarkkuudella ratkaisuksi saadaan siis* 0.0000*.*

Tässä tapauksessa huomataan myös Newtonin menetelmän heikkous, derivaatan nollakohdat. Jos yrittää aloittaa iteraatiota esimerkiksi kohdasta $x = 1$*, ohjelma antaa nollakohdan aproksimaatioksi* $x \approx 1$*. Tämä johtuu siitä, että funktiolla on paikallinen huippukohta kohdassa* $x = 1$ *ja derivaatta saa siinä arvon nolla, joten Newtonin menetelmä ei toimi.*

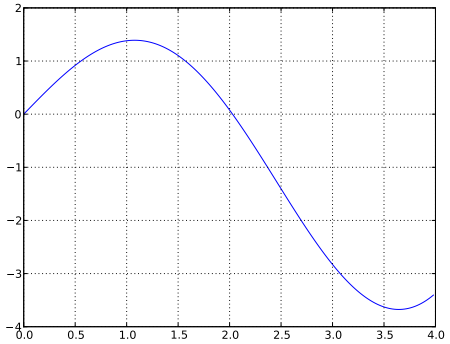
Huomattavaa on, että vaikka Python laskee iteraatiot, on käyttäjän itse osattava etsiä sopiva alkuarvo. Ohjelmassa toki on joitakin virhetilanteita huomioitu. Esimerkiksi tilanne, jossa funktio ei suppene, on huomioitu rajoittamalla iteraatioiden määrää.

Ohjelmassa on käytetty numeerista derivaattaa, koska ohjelma on haluttu tehdä mahdollisimman automaattiseksi käyttäjälle. Toki derivaatan voisi syöttää samoin kuin itse funktionkin. Käyttäjän olisi vain silloin itse derivoitava funktio.

Newtonin menetelmä on ollut suosittu kysymys myös ylioppilaskirjoituksissa. Tässä on esimerkkinä ratkaistu ylioppilastehtävä Pythonilla.

Esimerkki 5.5. *Määritä funktion* $f(x) = x \sin x$ *pienin positiivinen ääriarvokohta ja vastaava ääriarvo ratkaisemalla derivaatan nollakohta Newtonin menetelmällä. Anna vastauksesi viiden desimaalin tarkkuudella. (Pitkän matematiikan yo-koe, kevät 2005)*

Koska derivaatan nollakohta pitää ratkaista, muodostetaan funktion f *derivaattafunktio* g*. Kun funktio* f *derivoidaan, saadaan* $g(x) = \sin x + x \cos x$*. Piirretään ensiksi funktion* g *kuvaaja.*



Kuvaajasta nähdään, että funktion pienin positiivinen nollakohta on välillä $[1\frac{1}{2}, 2\frac{1}{2}]$*. Tämä voidaan myös todeta laskemalla funktion arvot kyseisissä pisteissä ja todeta, että merkki vaihtuu. Valitaan nyt iteraation aloituskohdaksi* $2\frac{1}{2}$*. Kirjoitetaan sitten koodi, joka laskee nollakohdalle approksimaation Newtonin menetelmällä.*

```
#otetaan math-kirjasto käyttöön
import math

#määritellään funktio f
def f(x):
        return math.sin(x)+x*math.cos(x)

#funktion f derivaatta pisteessä x
def derivaatta(f, x, h=1.0e-8):
        return (f(x+h)-f(x))/h

#Newtonin menetelmä
def newton(x0, tarkkuus, N=100):
        #alustetaan iteraatioiden määrä yhdeksi
        i=1
        while i<N:
                #lasketaan tarkempi x
                x=x0-(f(x0)/derivaatta(f,x0,h=1.0e-8))
                #testataan vastauksen tarkkuus
                if f(x-(tarkkuus/2))*f(x+(tarkkuus/2)) < 0:
                      return x
                #jos vastaus on riittävän tarkka,
                #lopetetaan ohjelman suoritus,
                #muuten jatketaan iterointia
```

```
            i=i+1
            x0=x
        #mikäli riittävän tarkkaa vastausta ei saada
        #sadassa iteraatiossa käyttäjää pyydetään
        #tarkistamaan alkuarvot
        print ("Tarkista alkuarvot")

#kysytään käyttäjältä alkuarvot
alkux=input("Anna aloituspiste:")
desimaalit=input("Anna tarkkuus:")

#suoritetaan Newtonin menetelmä annetuilla arvoilla
print newton(alkux,desimaalit)
```

Kun ohjelmalle annetaan aloituspisteeksi 2,5 ja tarkkuudeksi 0,00001, se tulostaa luvun 2.0287578417 7.Derivaatan nollakohta viiden desimaalin tarkkuudella on siis x=2,02876. Tämä on myös siis pienin positiivinen ääriarvokohta funktiolle f. Funktion f arvo tässä kohdassa on $f(2,02876) \approx 1,81971$.

Harjoituksia

Tehtävä 30. *Ratkaise Newtonin menetelmän avulla yhtälön $e^x + \sin\ x = 0$ suurin juuri viiden desimaalin tarkkuudella. (Pitkän matematiikan yo-koe, syksy 2002)*

Tehtävä 31. *Ratkaise yhtälö $x - \cos\ x = 0$ kolmen desimaalin tarkkudella (Pitkän matematiikan yo-koe, syksy 2007)*

6 Harjoitustehtävien ratkaisut

Tehtävä 1 Ohjelma kysyy käyttäjältä kaksi kokonaislukua. Luvut tallennetaan muuttujiin luku1 ja luku2. Sen jälkeen ohjelma laskee lukujen summan ja tallentaa sen muuttujaan tulos. Sen jälkeen ohjelma tulostaa muuttujan tulos sisällön eli lukujen summan.

Tehtävä 2 Ohjelma tulostaa käyttäjän antaman luvun kertotaulun. Ohjelma tekee sen siten, että se antaa ensin muuttujalle i arvon 1 ja sitten laskee annetun luvun ja i:n tulon sekä tulostaa sen. Sitten muuttujalle i annetaan arvoksi 2 ja toistetaan sama. Tätä jatketaan aina muuttujan i arvoon 10 saakka.

Tehtävä 3

```
kerrottava=input('Anna jokin kokonaisluku väliltä [1,10]:')
i=1
while i <= 10:
        tulos = kerrottava*i
        print kerrottava, '*', i, '=', tulos
        i=i+1
```

Tehtävä 4

```
luku = input('Anna jokin kokonaisluku:')
if luku%6==0:
        print 'Antamasi luku on jaollinen kuudella'
else:
        print 'Antamasi luku ei ole jaollinen kuudella'
```

Tehtävä 5

```
print range(1,41,2)
```

Tehtävä 6 Aloitetaan kertomalla ylempi yhtälö luvulla -2. Yhtälöryhmä siis muuntuu muotoon

$$\begin{aligned} -4x_1 - 2x_2 &= -10 \\ 4x_1 - 4x_2 &= 4 \end{aligned}$$

Lisätään sitten ylempi yhtälö alempaan, jolloin yhtälöryhmä saadaan muotoon

$$\begin{aligned} -4x_1 - 2x_2 &= -10 \\ 0 \cdot x_1 - 6x_2 &= -6 \end{aligned}$$

Alemmasta yhtälöstä saadaan nyt muuttujan x_2 ratkaisuksi $x_2 = 1$. Kun sijoitetaan tämä alkuperäiseen ylempään yhtälöön, saadaan muuttujalle x_1 yhtälö

$2x_1 + 1 = 5$. Tästä saadaan ratkaisuksi $x_1 = 2$. Koko yhtälöryhmän ratkaisu on siis

$$\begin{cases} x_1 = 2 \\ x_2 = 1 \end{cases}$$

Tehtävä 7 Yhtälöryhmän kerroinmatriisi on matriisi $A = \begin{pmatrix} 2 & 1 \\ 4 & -4 \end{pmatrix}$

Tehtävä 8 Matriisien summa

$$\begin{aligned} A + B &= \begin{pmatrix} 1 & 8 & -2 \\ 1 & 5 & 0 \\ 3 & 9 & 2 \end{pmatrix} + \begin{pmatrix} 1 & -4 & 0 \\ 6 & 7 & 12 \\ -9 & 8 & -3 \end{pmatrix} \\ &= \begin{pmatrix} 1+1 & 8-4 & -2+0 \\ 1+6 & 5+7 & 0+12 \\ 3-9 & 9+8 & 2-3 \end{pmatrix} = \begin{pmatrix} 2 & 4 & -2 \\ 7 & 12 & 12 \\ -6 & 17 & -1 \end{pmatrix} \end{aligned}$$

Matriisien erotus

$$\begin{aligned} A - B &= \begin{pmatrix} 1 & 8 & -2 \\ 1 & 5 & 0 \\ 3 & 9 & 2 \end{pmatrix} - \begin{pmatrix} 1 & -4 & 0 \\ 6 & 7 & 12 \\ -9 & 8 & -3 \end{pmatrix} \\ &= \begin{pmatrix} 1-1 & 8+4 & -2-0 \\ 1-6 & 5-7 & 0-12 \\ 3+9 & 9-8 & 2+3 \end{pmatrix} = \begin{pmatrix} 0 & 12 & -2 \\ -5 & -2 & -12 \\ 12 & 1 & 5 \end{pmatrix} \end{aligned}$$

Matriisit kerrottuna luvulla kolme

$$\begin{aligned} 3A &= 3 \begin{pmatrix} 1 & 8 & -2 \\ 1 & 5 & 0 \\ 3 & 9 & 2 \end{pmatrix} = \begin{pmatrix} 3 \cdot 1 & 3 \cdot 8 & 3 \cdot (-2) \\ 3 \cdot 1 & 3 \cdot 5 & 3 \cdot 0 \\ 3 \cdot 3 & 3 \cdot 9 & 3 \cdot 2 \end{pmatrix} \\ &= \begin{pmatrix} 3 & 24 & -6 \\ 3 & 15 & 0 \\ 9 & 27 & 6 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} 3B &= 3 \begin{pmatrix} 1 & -4 & 0 \\ 6 & 7 & 12 \\ -9 & 8 & -3 \end{pmatrix} = \begin{pmatrix} 3 \cdot 1 & 3 \cdot (-4) & 3 \cdot 0 \\ 3 \cdot 6 & 3 \cdot 7 & 3 \cdot 12 \\ 3 \cdot -9 & 3 \cdot 8 & 3 \cdot (-3) \end{pmatrix} \\ &= \begin{pmatrix} 3 & -12 & 0 \\ 18 & 21 & 36 \\ -27 & 24 & -9 \end{pmatrix} \end{aligned}$$

Tehtävä 9 $A^T = \begin{pmatrix} 1 & 1 & 3 \\ 8 & 5 & 9 \\ -2 & 0 & 2 \end{pmatrix}$

Tehtävä 10

$$\begin{vmatrix} 1 & -1 & 3 \\ 2 & 0 & 2 \\ -2 & 1 & -3 \end{vmatrix} = 1\begin{vmatrix} 0 & 2 \\ 1 & -3 \end{vmatrix} - (-1)\begin{vmatrix} 2 & 2 \\ -2 & -3 \end{vmatrix} + 3\begin{vmatrix} 2 & 0 \\ -2 & 1 \end{vmatrix}$$

$$= 1(0 \cdot (-3) - 2 \cdot 1) + 1(2 \cdot (-3) - 2 \cdot (-2)) + 3(2 \cdot 1 - 0 \cdot (-2))$$
$$= -2 - 2 + 6 = 2$$

Tehtävä 11 Yhtälöryhmän kerroinmatriisi on nyt matriisi $A = \begin{pmatrix} 5 & -1 \\ 1 & -2 \end{pmatrix}$ ja oikean puolen kertoimista muodostettu matriisi $B = (12, 6)^T$. Matriisin A determinantti on $\begin{vmatrix} 5 & -1 \\ 1 & -2 \end{vmatrix} = 5 \cdot (-2) - (-1) \cdot 1 = -9$.

Ratkaistaan ensin muuttuja x_1. Matriisi A_1 saadaan korvaamalla matriisista A ensimmäinen pystyrivi matriisilla B. Matriisi A_1 on siis matriisi $A_1 = \begin{pmatrix} 12 & -1 \\ 6 & -2 \end{pmatrix}$. Sen determinantti on $\begin{vmatrix} 12 & -1 \\ 6 & -2 \end{vmatrix} = 12 \cdot (-2) - 6 \cdot (-1) = -18$. Nyt saadaan ratkaistua muuttuja x_1, joka on siis $x_1 = \frac{-18}{-9} = 2$.

Samoin ratkaistaan muuttuja x_2. Matriisin A_2 determinantiksi saadaan 18, joten muuttujan x_2 ratkaisuksi saadaan $x_2 = \frac{18}{-9} = -2$. Yhtälöryhmän ratkaisu on siis $x_1 = 2$ ja $x_2 = -2$..

Tehtävä 12

a) Matriiseja ei voi kertoa keskenään, koska matriisi A on 3×2-matriisi ja matriisi B on 3×3 matriisi

b) $\begin{pmatrix} 3 & 2 \\ 4 & 6 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 & 2 \\ -2 & 3 & 1 & 2 \end{pmatrix}$

$$= \begin{pmatrix} 3 \cdot 1 + 2 \cdot (-2) & 3 \cdot 0 + 2 \cdot 3 & 3 \cdot (-1) + 2 \cdot 1 & 3 \cdot 2 + 2 \cdot 2 \\ 4 \cdot 1 + 6 \cdot (-2) & 4 \cdot 0 + 6 \cdot 3 & 4 \cdot (-1) + 6 \cdot 1 & 4 \cdot 2 + 6 \cdot 2 \end{pmatrix}$$

$$= \begin{pmatrix} 3 - 4 & 0 + 6 & -3 + 2 & 6 + 4 \\ 4 - 12 & 0 + 18 & -4 + 6 & 8 + 12 \end{pmatrix} = \begin{pmatrix} -1 & 6 & -1 & 10 \\ -8 & 18 & 2 & 20 \end{pmatrix}$$

c) Voidaan suoraan todeta, että kertolaskun tuloksena on matriisi A, koska matriisi B on identiteettimatriisi.

d)

$$\begin{pmatrix} 4 & -3 \\ 5 & 4 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 2 & -4 \end{pmatrix} = \begin{pmatrix} 4 \cdot 2 - 3 \cdot 2 & 4 \cdot 1 - 3 \cdot (-4) \\ 5 \cdot 2 + 4 \cdot 2 & 5 \cdot 1 + 4 \cdot (-4) \end{pmatrix} = \begin{pmatrix} 2 & 16 \\ 18 & -11 \end{pmatrix}$$

Tehtävä 13

a)

$$\begin{pmatrix} 1 & 2 & -4 \\ 1 & 2 & 0 \\ 3 & -4 & 3 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 4 \\ 8 & 3 \end{pmatrix} = \begin{pmatrix} 1 \cdot 5 + 2 \cdot 7 - 4 \cdot 8 & 1 \cdot 6 + 2 \cdot 4 - 4 \cdot 3 \\ 1 \cdot 5 + 2 \cdot 7 + 0 \cdot 8 & 1 \cdot 6 + 2 \cdot 4 + 0 \cdot 3 \\ 3 \cdot 5 - 4 \cdot 7 + 3 \cdot 8 & 3 \cdot 6 - 4 \cdot 4 + 3 \cdot 3 \end{pmatrix}$$

$$= \begin{pmatrix} 5+14-32 & 6+8-12 \\ 5+14+0 & 6+8+0 \\ 15-28+24 & 18-16+9 \end{pmatrix} = \begin{pmatrix} -13 & 2 \\ 19 & 14 \\ 11 & 11 \end{pmatrix}$$

b) Matriiseja ei voi kertoa keskenään, koska matriisi B on 2×4-matriisi ja matriisi A on 2×2 matriisi.
c) Voidaan suoraan todeta, että kertolaskun tuloksena on matriisi A, koska matriisi B on identiteettimatriisi.
d)

$$\begin{pmatrix} 2 & 1 \\ 2 & -4 \end{pmatrix} \begin{pmatrix} 4 & -3 \\ 5 & 4 \end{pmatrix} = \begin{pmatrix} 2 \cdot 4 + 1 \cdot 5 & 2 \cdot (-3) + 1 \cdot 4 \\ 2 \cdot 4 - 4 \cdot 5 & 2 \cdot (-3) - 4 \cdot 4 \end{pmatrix} = \begin{pmatrix} 13 & -2 \\ -12 & -22 \end{pmatrix}$$

Tehtävä 14 Matriisin A käänteismatriisi on vaihtoehdon b) matriisi, sillä

$$\begin{pmatrix} -3 & 1 \\ 2 & -4 \end{pmatrix} \begin{pmatrix} -\frac{2}{5} & -\frac{1}{10} \\ -\frac{1}{5} & -\frac{3}{10} \end{pmatrix} = \begin{pmatrix} -3 \cdot \left(-\frac{2}{5}\right) + 1 \cdot \left(-\frac{1}{5}\right) & -3 \cdot \left(-\frac{1}{10}\right) + 1 \cdot \left(-\frac{3}{10}\right) \\ 2 \cdot \left(-\frac{2}{5}\right) - 4 \cdot \left(-\frac{1}{5}\right) & 2 \cdot \left(-\frac{1}{10}\right) - 4 \cdot \left(-\frac{3}{10}\right) \end{pmatrix}$$
$$= \begin{pmatrix} \frac{6}{5} - \frac{1}{5} & \frac{3}{10} - \frac{3}{10} \\ -\frac{4}{5} + \frac{4}{5} & -\frac{2}{10} + \frac{12}{10} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Koska käänteismatriisi on yksikäsitteinen, ei välttämättä tarvitse kokeilla muita matriiseja.

Tehtävä 15 Matriisilla ei ole käänteismatriisia, koska se ei ole neliömatriisi.

Tehtävä 16 Käänteismatriisi saadaan ratkaisemalla yhtälö

$$\begin{pmatrix} 0 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Kun suoritetaan vasemman puolen kertolasku, päädytään yhtälöryhmään

$$\begin{aligned} -x_{21} &= 1 \\ -x_{22} &= 0 \\ 2x_{11} + x_{21} &= 0 \\ 2x_{12} + x_{22} &= 1 \end{aligned}$$

Kun yhtälöryhmä ratkaistaan, saadaan ratkaisuksi $x_{11} = \frac{1}{2}, x_{12} = \frac{1}{2}$, $x_{21} = -1, x_{22} = 0$. Käänteismatriisi on siis $A^{-1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -1 & 0 \end{pmatrix}$.

Tehtävä 17

```
#otetaan Numpy-kirjasto käyttöön
from numpy import *
#luodaan matriisi A
A = matrix ([[1, 1, 2], [-3, 1, 0], [-1, 2, -2]])
```

```
#luodaan matriisi B
B = matrix ([[2, -2, 1], [0, -1, 1], [-2, 0, 2]])
#tulostetaan matriisista A alkio a_12
print A[0,1]
#tulostetaan matriisista B alkio b_32
print B[2,1]
```

Ohjelma tulostaa luvut 1 ja 0.

Tehtävä 18

```
#otetaan Numpy-kirjasto käyttöön
from numpy import *
#luodaan matriisi A
A = matrix ([[1, 1, 2], [-3, 1, 0], [-1, 2, -2]])
#luodaan matriisi B
B = matrix ([[2, -2, 1], [0, -1, 1], [-2, 0, 2]])
#tulostetaan matriisien A ja B summa
print A+B
#tulostetaan matriisien A ja B erotus
print A-B
#tulostetaan matriisi A kerrottuna luvulla 4
print 4*A
#tulostetaan matriisi B kerrottuna luvulla 4
print 4*B
```

Ohjelma tulostaa matriisit $\begin{pmatrix} 3 & -1 & 3 \\ -3 & 0 & 1 \\ -3 & 2 & 0 \end{pmatrix}$, $\begin{pmatrix} -1 & 3 & 1 \\ -3 & 2 & -1 \\ 1 & 2 & -4 \end{pmatrix}$, $\begin{pmatrix} 4 & 4 & 8 \\ -12 & 4 & 0 \\ -4 & 8 & -8 \end{pmatrix}$ ja $\begin{pmatrix} 8 & -8 & 4 \\ 0 & -4 & 4 \\ -8 & 0 & 8 \end{pmatrix}$.

Tehtävä 19

```
#otetaan Numpy-kirjasto käyttöön
from numpy import *
#luodaan matriisi A
A = matrix ([[1, 1, 2], [-3, 1, 0], [-1, 2, -2]])
#luodaan matriisi B
B = matrix ([[2, -2, 1], [0, -1, 1], [-2, 0, 2]])
#tulostetaan matriisien tulo AB
print A*B
#tulostetaan matriisien tulo BA
print B*A
```

```
#tulostetaan matriisin B determinantti
print linalg.det(B)
```

Ohjelma tulostaa matriisit $\begin{pmatrix} -2 & -3 & 6 \\ -6 & 5 & -2 \\ 2 & 0 & -3 \end{pmatrix}$ ja $\begin{pmatrix} 7 & 2 & 2 \\ 2 & 1 & -2 \\ -4 & 2 & -8 \end{pmatrix}$ sekä luvun -2, joka on siis matriisin B determinantti.

Tehtävä 20

```
#otetaan Numpy-kirjasto käyttöön
from numpy import *
#luodaan matriisi A
A = matrix ([[1, 1, 2], [-3, 1, 0], [-1, 2, -2]])
#luodaan matriisi B
B = matrix ([[2, -2, 1], [0, -1, 1], [-2, 0, 2]])
#tulostetaan matriisin A transpoosi
print A.T
#tulostetaan matriisin B käänteismatriisi
print B.I
```

Ohjelma tulostaa matriisit $\begin{pmatrix} 1 & -3 & -1 \\ 1 & 1 & 2 \\ 2 & 0 & -2 \end{pmatrix}$ ja $\begin{pmatrix} 1 & -2 & \frac{1}{2} \\ 1 & -3 & 1 \\ 1 & -2 & 1 \end{pmatrix}$.

Tehtävä 21

```
#otetaan NumPy-kirjasto käyttöön
from numpy import *
#luodaan matriisit
A = matrix([[5, 3], [4, 2]])
B = matrix ([[9], [8]])
#ratkaistaan yhtälöryhmä
print linalg.solve(A, B)
```

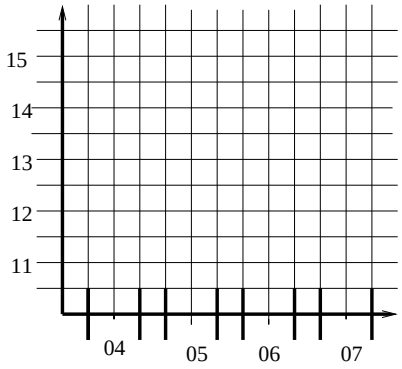
Ohjelma antaa ratkaisuksi $x_1 = 3, x_2 = -2$

Tehtävä 22

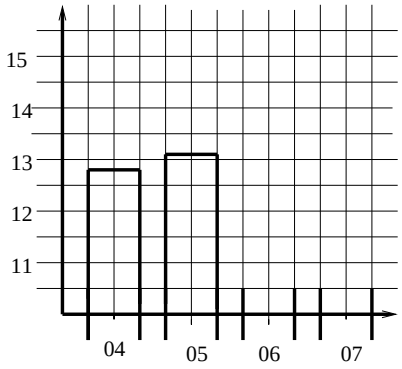
```
#otetaan NumPy-kirjasto käyttöön
from numpy import *
#luodaan matriisit
A = matrix([[3, -2], [4, 5]])
B = matrix ([[1], [2]])
#ratkaistaan yhtälöryhmä
print linalg.solve(A, B)
```

Ohjelma tulostaa ratkaisuksi $x \approx 0,391$ ja $y \approx 0,087$. Mikäli vastauksen haluaa murtolukumuodossa kuten yo-kokeessa vaaditaan, pitää se osata itse muuttaa.

Tehtävä 23 Merkitään ensin vaaka-akselille vuodet ja varataan kullekin vuodelle kahden ruudun levyinen pylväs. Jätettään pylväiden väliin yksi ruutu. Sitten valitaan asteikko pystyakselille. Valitaan asteikko nyt siten, että se alkaa 10 asteesta ja päättyy 15 asteeseen ja yksi ruutu vastaa puolta astetta.

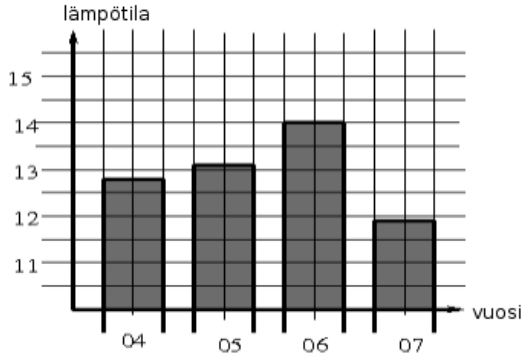


Piirretään sitten kunkin vuoden kohdalle pylväs, joka korkeus vastaa lämpötilaa.



Kun vielä annetaan kuviolle otsikko ja nimetään akselit saadaan seuraava pylväsdiagrammi.

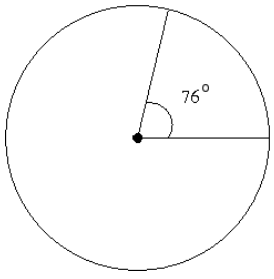
Syyskuun keskilämpötilat Helsingissä



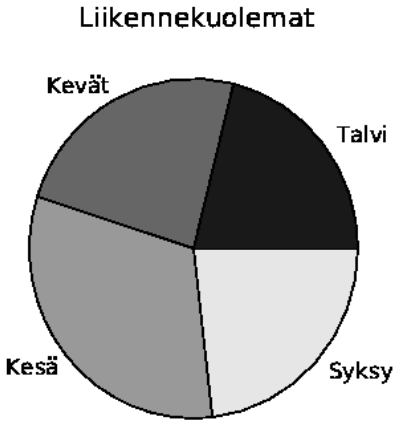
Tehtävä 24 Nyt tiedot ovat lukuina, joten ensiksi pitää laskea kuinka suurta prosenttiosuutta kuolleista kunakin vuodenaikana kuolleet vastaavat. Yhteensä kuolleita on siis 384. Näin ollen talvella kuolleiden osuus on $\frac{81}{384} \approx 0,211 = 21,1\%$. Vastaavasti keväällä kuolleiden osuudeksi saadaan 24,0 %, kesällä kuolleiden 31,8 % ja syksyllä kuolleiden osuudeksi 23,1 %.

Seuraavaksi lasketaan kuinka suurta keskuskulmaa kukin osuuksista vastaa. Koko ympyrän asteluku on 360, joten talvella kuolleita vastaa keskuskulma, jonka asteluku on $0,211 \cdot 360° = 76°$. Vastaavasti keväällä kuolleita vastaa 86 asteen keskuskulma, kesällä kuolleita 115 asteen kulma ja syksyllä kuolleita 83 asteen kulma.

Sitten piirretään ympyrä ja siihen kunkin asteluvun suuruinen sektori.



Kun vielä nimetään sektorit ja annetaan kuviolle otsikko, saadaan seuraava piirakkakuvio.



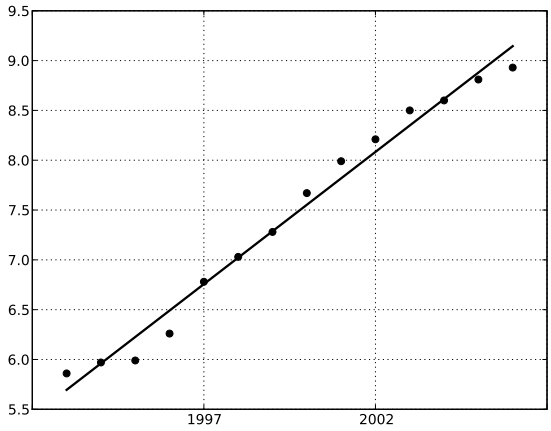
Tehtävä 25 Havaintoja on nyt 14, joten $n = 14$. Laskennallisista syistä merkitään vuosia niin, että vuosi 93 vastaa muuttujan x arvoa 1, vuosi 94 muuttujan arvoa 2 jne. Nyt siis summalauseke $\sum_{i=1}^{n} x_i = 1 + 2 + 3 + \cdots + 14$ saa arvon 105. Vastaavasti summalauseke $\sum_{i=1}^{n} y_i = 5,86 + 5,97 + \cdots + 8,93$ saa arvon 103,88. Näin ollen toinen ratkaistavan yhtälöryhmän yhtälöistä on muotoa $14a + 105b = 103,88$.

Summalauseke $\sum_{i=1}^{n} x_i^2 = 1^2 + 2^2 + \cdots + 14^2$ saa arvon 1015 ja summalauseke $\sum_{i=1}^{n} x_i y_i = 1 \cdot 5,86 + 2 \cdot 5,97 + \cdots + 14 \cdot 8,93$ arvon 839,47. Näin ollen toinen ratkaistavan yhtälöryhmän yhtälöistä on muotoa $105a + 1015b = 839,47$.

Ratkaistavana on siis nyt yhtälöpari

$$\begin{aligned} 14a + 105b &= 103,88 \\ 105a + 1015b &= 839,47 \end{aligned}$$

Kun tämä yhtälöpari ratkaistaan, saadaan ratkaisuksi $a = 5,43$ ja $b = 0,265$. Suoran yhtälö on siis $y = 0,265x + 5,43$. Piirrettynä suora näyttää tältä.



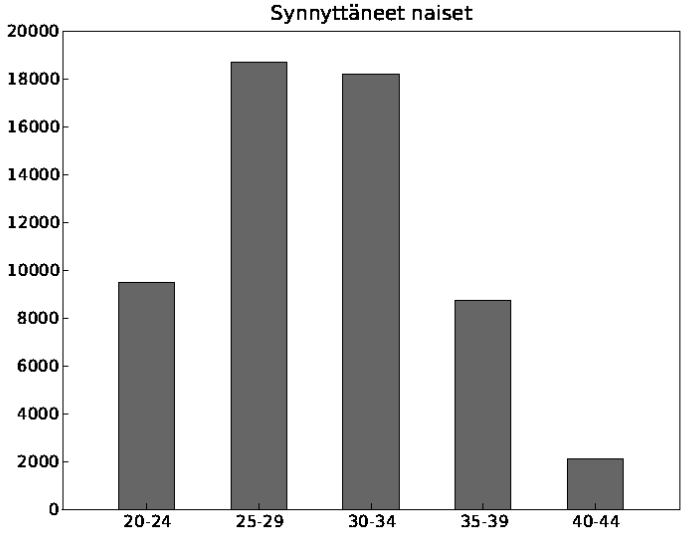
Vuosi 2010 vastaa muuttujan x arvoa 18. Näin ollen neliöhinta vuonna 2010 on $y = 0,265 * 18 + 5,43 = 10,2$.

Tehtävä 26

```
#koodissa käytetään apuna numarray-tietorakennetta, joten
#otetaan se käyttöön
import numpy.numarray as na

#otetaan Matplotlib käyttöön
from pylab import *

#nimetään pylväät
labels = ["20-24","25-29","30-34","35-39","40-44"]

#pylväiden korkeudet eli frekvessit
data =   [9467,18673,18206,8721,2095]
#luodaan numarray-tyyppinen lista, joka sisältää tiedot
#pylväiden sijainnista
xlocations = na.array(range(len(data)))+0.5
#määritetään pylvään leveys
width = 0.5
#bar-komento piirtää pylväät, color määrittää pylvään värin
bar(xlocations, data, color='0.4' , width=width)
#luodaan y-akseli asteikkoineen
yticks(range(0, 20001, 2000))
#luodaan x-akseli
```

```
xticks(xlocations+ width/2, labels)
xlim(0, xlocations[-1]+width*2)
#kuvaajan otsikko
title("Synnyttäneet naiset")
#näytetään asteikko ainoastaan kuvaajan vasemmalla
#puolella
gca().get_xaxis().tick_bottom()
gca().get_yaxis().tick_left()
#näytetään kuvaaja
show()
```

Ohjelma tulostaa kuvan

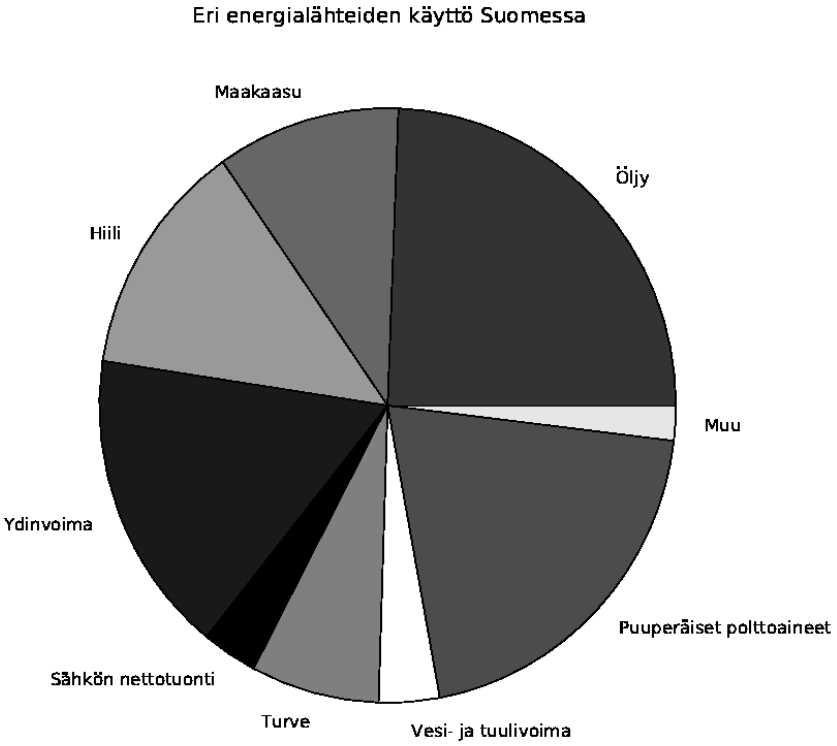


Tehtävä 27

```
 #otetaan matplotlib käyttöön
from pylab import *

#annetaan eri sektoreiden osuudet listana
fracs=[24.4,10.3, 12.9, 16.6, 3.1, 7.2, 3.4,20.2, 1.9]
#nimetään sektorit
labels=["Öljy", "Maakaasu","Hiili", "Ydinvoima", "Sähkön nettotuonti",
        "Turve", "Vesi- ja tuulivoima", "Puuperäiset polttoaineet", "Muu"]
#annetaan kullekin sektorille väri
colors=('0.2','0.4','0.6','0.1', '0', '0.5' ,'1','0.3','0.9' )
```

```
#määritellään kuvan koko, tässä tapauksessa se kannattaa olla
#neliönmuotoinen eli pituus ja leveys yhtäsuuret sekä taustaväri
figure(figsize=(9,9),facecolor='w')
#pie-komento piirtää itse kuvion
pie(fracs, colors=colors, labels=labels)
#annetaan kuvaajalle otsikko
title("Eri energialähteiden käyttö Suomessa")
#näytetään kuvaaja
show()
```

Koodista tulostuu seuraava kuva.

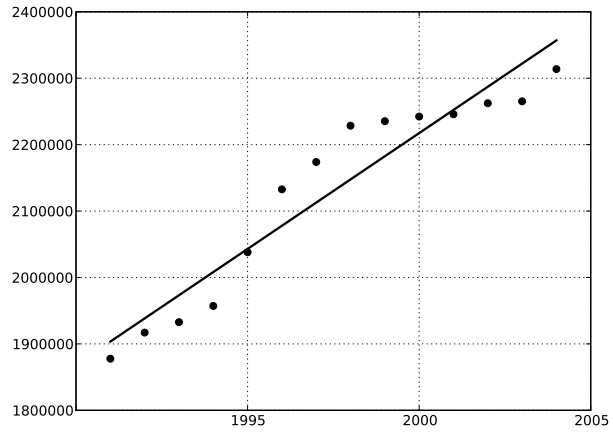


Tehtävä 28

```
#otetaan matplotlib käyttöön
from pylab import *
#annetaan pisteiden x-koordinaatit listana
```

```
x=[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0,14.0]
#annetaan pisteiden y-koordinaatit listana
y=[1877721,1917051,1932752,1957144,2037997,2132704,2173885,2228557,
    2235317,2242303,2245780,2262359,2265211,2313788]
#tehdään pienimmän neliösumman sovitus
m,b=polyfit(x,y, 1)
#luodaan x-akseli
xlocations=array(range(0,len(x)+5,5))
labels=["","1995 ","2000","2005"]
xticks(xlocations,labels)
#piirretään suora
plot(x,y,'ko',x, [m*i+b for i in x], '-k',linewidth=2)
#otetaan ruudutus käytöön
grid(True)
#näytetään kuvaaja
show()
```

Ohjelma tulostaa kuvan



Tehtävä 29 Funktiolla on nollakohta välillä [-1,0] sillä $f(-1) = -1 + e^{-1}$ $\approx -0,6321$ ja $f(0) = 0 + e^0 = 1$. Valitaan siis aloituspisteeksi $x_0 = 0$.

Koska ratkaisu pyydettiin kolmen desimaalin tarkkuudella, valitaan vakio h siten että $h = 0,5 \cdot 10^{-3}$.

Koska iteraatiokaavassa tarvitaan funktion derivaattaa, niin derivoidaan funktion. Funktion derivaatta on nyt $f'(x) = 1 + e^x$.

Lasketaan ensimmäinen arvio nollakohdalle ja saadaan $x_1 = 0 - \frac{f(0)}{f'(0)}$

$= -0,5$. Virhetarkastelusta saadaan nyt

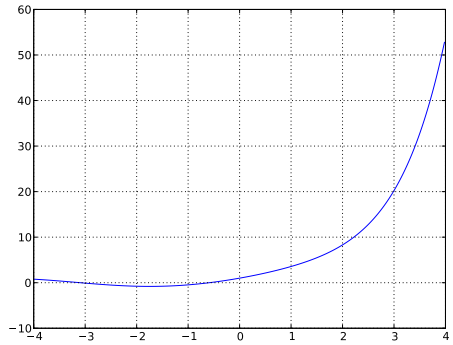
$$f(-0,5-0,5\cdot 10^{-3})\cdot f(-0.5+0,5\cdot 10^{-3}) = f(-0,5005)\cdot f(-0,4995)$$
$$= (-0,5005+e^{-0,5005})\cdot(-0,4995+e^{-0,4995}) \approx 0,011348 > 0$$

Jatketaan siis iterointia vielä.

Seuraava, parempi arvio on $x_2 = -0,5 - \frac{f(-0,5)}{f'(-0,5)} \approx -0,566311$. Virhetarkastelusta saadaan nyt $1,09\cdot 10^{-6} > 0$. Jatketaan siis iterointia.

Kolmannella iteraatiolla tulokseksi tulee $-0,567142$ ja virhetarkastelusta saadaan nyt $-6,14\cdot 10^{-7} < 0$. Jo kolmannella iteraatiolla saadaan siis riittävän tarkka vastaus. Ratkaisu kolmen desimaalin tarkkuudella on siis -0,567.

Tehtävä 30 Piirretään ensiksi funktion kuvaaja.



Kuvaajasta nähdään, että suurin ääriarvokohta on välillä [-1,0]. Iteroinnin aloituspisteeksi voidaan siten valita esimerkiksi $x_0 = 0$.

Koska vastaus pyydettiin viiden desimaalin tarkkuudella, annetaan tarkkuudeksi 0,00001.

Kun nämä syötetään alla olevaan ohjelmaan, se antaa tulokseksi $-0,588529412584$. Näin ollen vastaus on $x \approx -0,58853$.

```
#otetaan math-kirjasto käyttöön
import math

#määritellään funktio f
def f(x):
        return (math.e)**x+math.sin(x)

#funktion f derivaatta pisteessä x
def derivaatta(f, x, h=1.0e-8):
        return (f(x+h)-f(x))/h
```

```
#Newtonin menetelmä
def newton(x0, tarkkuus, N=100):
        #alustetaan iteraatioiden määrä yhdeksi
        i=1
        while i<N:
                #lasketaan tarkempi x
                x=x0-(f(x0)/derivaatta(f,x0,h=1.0e-8))
                #testataan vastauksen tarkkuus
                if f(x-(tarkkuus/2))*f(x+(tarkkuus/2)) < 0:
                      return x
                #jos vastaus on riittävän tarkka,
                #lopetetaan ohjelman suoritus,
                #muuten jatketaan iterointia
                i=i+1
                x0=x
         #mikäli riittävän tarkkaa vastausta ei saada
         #sadassa iteraatiossa käyttäjää pyydetään
         #tarkistamaan alkuarvot
         print ("Tarkista alkuarvot")

#kysytään käyttäjältä alkuarvot
alkux=input("Anna aloituspiste:")
desimaalit=input("Anna tarkkuus:")

#suoritetaan Newtonin menetelmä annetuilla arvoilla
print newton(alkux,desimaalit)
```

Tehtävä 31 Kirjoitetaan ensin ohjelma, joka ratkaisee yhtälön Newtonin menetelmällä.

```
#otetaan math-kirjasto käyttöön
import math

#määritellään funktio f
def f(x):
        return x-math.cos(x)

#funktion f derivaatta pisteessä x
def derivaatta(f, x, h=1.0e-8):
        return (f(x+h)-f(x))/h
```

```
#Newtonin menetelmä
def newton(x0, tarkkuus, N=100):
        #alustetaan iteraatioiden määrä yhdeksi
        i=1
        while i<N:
                #lasketaan tarkempi x
                x=x0-(f(x0)/derivaatta(f,x0,h=1.0e-8))
                #testataan vastauksen tarkkuus
                if f(x-(tarkkuus/2))*f(x+(tarkkuus/2)) < 0:
                      return x
                #jos vastaus on riittävän tarkka,
                #lopetetaan ohjelman suoritus,
                #muuten jatketaan iterointia
                i=i+1
                x0=x
         #mikäli riittävän tarkkaa vastausta ei saada
         #sadassa iteraatiossa käyttäjää pyydetään
         #tarkistamaan alkuarvot
         print ("Tarkista alkuarvot")

#kysytään käyttäjältä alkuarvot
alkux=input("Anna aloituspiste:")
desimaalit=input("Anna tarkkuus:")

#suoritetaan Newtonin menetelmä annetuilla arvoilla
print newton(alkux,desimaalit)
```

Yhtälöllä on nollakohta välillä [-1,1], sillä $f(-1) \approx -1,540$ ja $f(1) \approx 0,459$. Valitaan iteraation aloituspisteeksi siis $x = 1$.

Vastausta pyydettiin kolmen desimaalin tarkkuudella, joten tarkkuudeksi syötetään $0,001$.

Näillä arvoilla ohjelma tulostaa luvun 0,739112891031, joten vastaus on $x = 0,739$.

Lähteet

[1] J. Kasurinen, *Python-ohjelmointiopas, versio 1.1* (Lappeenrannan teknillinen yliopisto , 2007).

[2] Dive Into Python, www.diveintopython.org, 25.3.2008.

[3] J. B. Fraleigh and R. A. Beauregard, *Linear algebra* (Addison-Wesley , 1986).

[4] T. Metsänkylä, *Lineaarialgebra, opintomoniste* (Turun Yliopisto , 2003).

[5] Wolfram MathWorld, http://mathworld.wolfram.com, 15.7.2008.

[6] NumPy-tutoriaali, http://www.scipy.org/Tentative_NumPy_Tutorial, 3.7.2008.

[7] M. Heinonen, M. Luoma, L. Mannila, and T. Tikka, *Pii - Tilastot ja todennäköisyys* (Otava , 2007).

[8] T. Laurinolli *et al.*, *Laskutaito X* (WSOY , 2005).

[9] Tilastokeskus, www.tilastokeskus.fi, 26.10.2008.

[10] Matplotlib, http://matplotlib.sourceforge.net, 22.7.2008.

[11] Öljy- ja kaasualan keskusliitto, www.oilgas.fi, 22.7.2008.

[12] R. L. Burden and J. D. Faires, *Numerical Analysis* (PWS-Kent Publishing Company , 1989).

[13] Mpmath-kirjaston kotisivu, http://code.google.com/p/mpmath/, 11.10.2008.