Harjoitus 3 -- Ratkaisut

1

'≤'-merkki kirjoitetaan <=, '≥'-merkki >=, '≠'-merkki !=, '≒'-merkki ==. Yhtälöiden ratkaisusta puhutaan lisää myöhemmin.

2

a

```
f[x_, y_] := If[(*ehtolauseke*)x < y, y (*tämä palautetaan,
  jos ehto on tosi*), x(*tämä palautetaan, jos ehto on epätosi*)]

f[1, 2]
f[(√2)^π, e^√3]
```

```
2
```

$e^{\sqrt{3}}$

b

Summan laskemisen periaate: alustetaan summaa kuvaava muuttuja nollaksi, ja lisätään siihen summan termit yksi kerrallaan. Vastaavasti tulon laskemisen periaate: alustetaan tuloa kuvaava muuttuja *ykköseksi*, ja kerrotaan sitä tulon tekijöillä, yksi kerrallaan. Summat ja tulot voidaan Mathematicassa laskea valmiillakin funktioilla, mutta on hyödyllistä opetella laskemaan ne monessa ohjelmointikielessä esiintyvillä toistorakenteilla, For, While, ...

Muista, että jos funktio koostuu useammasta kuin yhdestä käskystä, käskyt on kirjoitettava sulkujen sisään ja viimeisen käskyn palauttama arvo on funktion arvo.

In[1]:=
```
g[x_, n_] :=
 (For[(*alustus*)i = 0; summa = 0, (*ehtolauseke*)i ≤ n, (*laskurin kasvatus*)i++,
   (*toistettavat lauseet*)summa += (-1)^i x^(2 i+1)/(2 i + 1)!]; summa // N)

g[2, 6]
```

```
0.909297
```

Laskuri ja summamuuttuja kannattaa määritellä paikallisiksi muuttujiksi.

```
i =.; summa =.;
h[x_, n_] := Module[(*paikalliset muuttujat*){i, summa},
  (*funktion runko*)For[i = 0; summa = 0, i ≤ n, i++, summa += (-1)^i x^(2 i+1)/(2 i + 1)!]; summa // N]

h[2, 6]
```

```
0.909297
```

Sama While-komennolla:

```
j[x_, n_] := Module[{i, summa},
  (*nyt alustukset on kirjoitettava toistorakenteen ulkopuolelle*)i = 0; summa = 0;
  While[(*ehto*)i ≤ n, (*toistettavat lauseet*)summa += (-1)^i x^(2 i+1)/(2 i + 1)!; i++]; N[summa]]
```

```
j[2, 6]
```

```
0.909297
```

Ja Do-komennolla. Tämä eroaa kahdesta edellisestä siinä, että varsinaista ehtolauseketta ei esiinny.

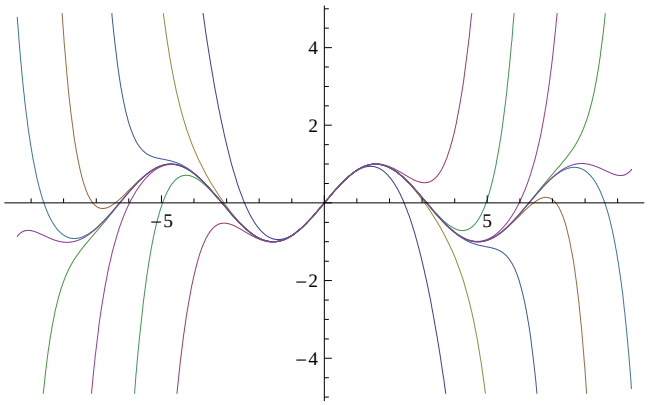
```
k[x_, n_] := Module[(*paikallisille muuttujille voi antaa alkuarvoja*){i, summa = 0},
  Do[(*toistettavat lauseet*)
   summa += (-1)^i x^(2 i+1)/(2 i + 1)!, (*i:n vaihteluväli*){i, 0, n}]; summa // N]
```

```
k[2, 6]
```

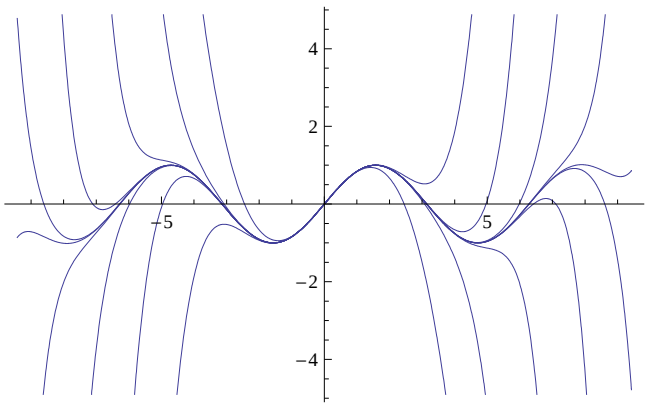
```
0.909297
```

Piirretään g[x,n]:n kuvaajat, kun n=1,...,10. Kun piirretään taulukollinen kuvaajia, kannattaa käyttää Evaluate-komentoa. Ilman sitä, piirtäminen kestää aivan tolkuttoman kauan, eikä *Mathematica* osaa piirtää eri käyriä eri väreillä.

```
Plot[Evaluate[Table[g[x, n], {n, 1, 10}]], {x, -3 π, 3 π}]
```




```
Plot[Table[g[x, n], {n, 1, 10}], {x, -3 π, 3 π}]
```

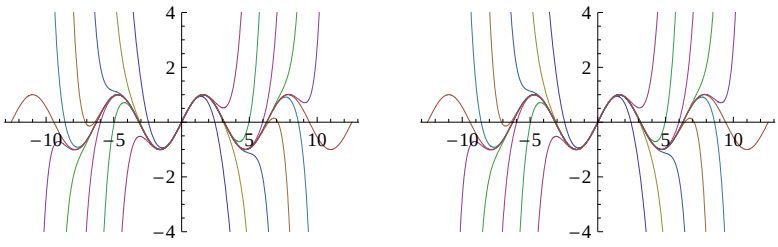


Jos halutaan piirtää Sin[x] yhtä aikaa yo. funktiotaulukon kanssa, kannattaa käyttää Join-funktiota, joka liittää kaksi taulukkoa yhteen, tai Append-funktiota, joka lisää taulukkoon alkion. Muista: Painamalla funktion nimen kohdalla F1 saat lisäti-

etoa. ImageSize-komento pakottaa piirrettävät kuvat normaalikokoisiksi (GraphicsArray tekisi niistä oletusarvoisesti pieniä). GraphicsArray-komennolla voi tulostaa kuvaajia taulukon muotooon järjestettynä.

In[2]:=
```
g1 = Plot[Evaluate[Join[Table[g[x, n], {n, 1, 10}], {Sin[x]}]],
   {x, -4 π, 4 π}, PlotRange → {-4, 4}, ImageSize → Small];
g2 = Plot[Evaluate[Append[Table[g[x, n], {n, 1, 10}], Sin[x]]],
   {x, -4 π, 4 π}, PlotRange → {-4, 4}, ImageSize → Small];
GraphicsArray[
 {g1,
  g2}]
```

Out[4]=



Tehtävänannossa esitetty summahan on tietysti sinifunktion Taylorin sarja.

■ b-ekstra

Otetaan esimerkki tulon laskemisesta: n:n kertoma on $n! = n(n-1)(n-2)\cdots 3\cdot 2\cdot 1$.

```
kertoma[n_] := Module[{i, tulo}, For[i = n; tulo = 1, i ≥ 1, i--, tulo *= i]; tulo]
kertoma[5]
```

```
120
```

Kertolaskun järjestyksellä ei ole väliä:

```
kertoma[n_] := Module[{i, tulo}, For[i = 1; tulo = 1, i ≤ n, i++, tulo *= i]; tulo]
kertoma[5]
```

```
120
```

■ c

Halutaan, että $A_{ij} = -2$, kun $i = j$, ja $A_{ij} = 1$, kun $i = \pm j$, ja $A_{ij} = 0$ kaikissa muissa tapauksissa. Toisin sanoen, $A_{ij} = -2$, kun $i - j = 0$, ja $A_{ij} = 1$, kun $i - j = \pm 1$, ja $A_{ij} = 0$ kaikille muille $i - j$:n arvoille. Lauseke $i - j$ kelpaa Switch-komennon valinta-lausekkeeksi. Switch haluaa siis lausekkeen (joka on joidenkin muuttujien, esim. i ja j, funktio), joka saa tilanteesta riippuen eri *vakioarvoja*. Valintalausekkeen arvoa verrataan järjestyksessä jokaiseen vakioarvoon, ja kun yhtäsuuruus sattuu kohdalle, kyseistä arvoa vastaavat komennot suoritetaan.

```
alkio[i_, j_] := Switch[(*valintalauseke*)i - j,
  0(*lausekkeen ensimmäinen mahdollinen arvo (tähän verrataan ensin)*),
  -2(*komento, joka suoritetaan, kun valintalausekkeen arvo on 0*),
  1(*toinen mahd. arvo
   (jos valintalauseke ei täsmännyt 1. mahd. arvoon, sitä verrataan tähän)*),
  1(*mitä tällöin suoritetaan*),
  -1, (*jne*)
  1,
  _(*valintalausekkeen "oletusarvo", vastaa mitä tahansa lauseketta*),
  0(*tämä suoritetana,
  jos valintalausekkeen arvo ei ole mikään kolmesta ensimmäisestä vaihtoehdosta*)]
```

```
alkio[1, 1]
alkio[3, 4]
alkio[52, 21]
```

```
-2
```

```
1
```

```
0
```

```
a = Table[alkio[i, j], {i, 10}, {j, 10}];
a // MatrixForm
```

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

Kompaktimmin esitettynä:

```
a = Table[Switch[i - j, 0, -2, 1, 1, -1, 1, _, 0], {i, 10}, {j, 10}]; a // MatrixForm
```

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

Keinotekoinen esimerkki Switch-lauseen käytöstä (toimii paremmin If-lauseella):

```
f[x_, y_] := Switch[x < y, True, y, False, x];
f[2, 1]
```

```
2
```

■ c -- toinen tapa

Matriisin luominen onnistuu myös Which-lauseella:

In[141]:=
```
a = Table[Which[
      i == j(*ehto 1*),
      -2(*jos ehto 1 on tosi*),
      i == j + 1(*ehto 2*),
      1,
      i == j - 1,
      1,
      1 == 1(*loppuun pitää kirjoittaa jokin ehto, joka on aina tosi*),
      0], {i, 10}, {j, 10}];
a // MatrixForm
```

Out[142]//MatrixForm=

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

3

Windowsiin löytyy Oulun yliopiston tietohallinnon ohjelmistojakelusta (www.oulu.fi/tietohallinto) SSH Secure Shell Client -ohjelma, jolla voi ottaa yhteyden yliopiston tuomi-, haapa-, paju-, jne. koneisiin. Tuomella ja haavalla on asennettu *Mathematica*. Se käynnistyy tekstipohjaisena komennolla math. Linux-käyttöjärjestelmissä SSH on valmiiksi asennettuna, ja yhteydenotto tapahtuu komentoriviltä komennolla ssh omatunnus@tuomi.oulu.fi. Voit kokeilla myös graafista käyttöliittymää komennolla ssh -X omatunnus@tuomi.oulu.fi (tässä on joitain fonttiongelmia).

Huom. jos askelpalautin (backspace) ei toimi, komentokehotteen asetuksista pääsee säätämään, mitä backspace- ja delete-näppäimet tekevät.

4

Ei kommenttia.