Cvičenie 3 - jednorozmerné polia a práca so súbormi

Jednorozmerné pole

Úloha 1

Dané je jednorozmerné pole a, ktorého prvky sú $a = \{10, -2, 5, 9, 10, 7, -5, -3, -1, 5\}$.

Čo najjednoduchšie zistite:
- počet prvkov poľa a
- siedmy prvok od začiatku poľa a piaty prvok od konca poľa a
- do jednorozmerného poľa b vyberte a zapíšte druhý, tretí a štvrtý prvok poľa a (aspoň tromi spôsobmi)
- miesto (pozíciu), kde sa v poli nachádza prvok 10
- maximálny a minimálny prvok
- súčet všetkých prvkov a aritmetický priemer všetkých prvkov
- počet prvkov z poľa a, ktorých hodnota je menšia ako + 4
- súčet prvkov z poľa a, ktorých hodnota je menšia ako + 4

Najskôr zadefinujme pole A

In[1]:=
```
Clear[a]
a = {10, -2, 5, 9, 10, 7, -5, -3, -1, 5}
```

Out[2]=
```
{10, -2, 5, 9, 10, 7, -5, -3, -1, 5}
```

Môžeme použiť aj plný tvar príkazu (tento spôsob sa používa zriedkavejšie)

In[3]:=
```
Clear[a]
a = List[10, -2, 5, 9, 10, 7, -5, -3, -1, 5]
```

Out[4]=
```
{10, -2, 5, 9, 10, 7, -5, -3, -1, 5}
```

Počet prvkov jednorozmerného poľa určíme pomocou príkazu **Length**[ ]

In[5]:=
```
Length[a]
```
Out[5]=
```
10
```

Siedmy prvok od začiatku poľa A určíme

In[6]:=
```
a[[7]]
```
Out[6]=
```
-5
```

Piaty prvok od konca poľa A určíme

In[7]:=
```
a[[-5]]
```
Out[7]=
```
7
```

Do jednorozmerného poľa *b* vyberte a zapíšte druhý, tretí a štvrtý prvok poľa *a* (aspoň tromi spôsobmi)

1. spôsob - naprimitívnejší

In[8]:=
```
b = {a[[2]], a[[3]], a[[4]]}
```
Out[8]=
```
{-2, 5, 9}
```

2. spôsob

In[9]:=
```
b = a[[2 ;; 4]]
```
Out[9]=
```
{-2, 5, 9}
```

3. spôsob

In[10]:=
```
b = Take[a, {2, 4}]
```
Out[10]=
```
{-2, 5, 9}
```

4. spôsob

In[11]:=
```
b = a[[{2, 3, 4}]]
```
Out[11]=
```
{-2, 5, 9}
```

5. spôsob

In[12]:=
```
b = Part[a, {2, 3, 4}]
```
Out[12]=
```
{-2, 5, 9}
```

Miesto (pozíciu), kde sa v poli nachádza prvok 8

In[13]:=
```
Position[a, 10]
```
Out[13]= {{1}, {5}}

Maximálny a minimálny prvok

In[14]:=
```
Max[a]
```
Out[14]= 10

In[15]:=
```
Min[a]
```
Out[15]= -5

Súčet všetkých prvkov - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať

1. spôsob

In[16]:=
```
Sum[a[[i]], {i, 1, Length[a]}]
```
Out[16]= 35

2. spôsob

In[17]:=
$$\sum_{i=1}^{10} a[[i]]$$
Out[17]= 35

3. spôsob

In[18]:=
```
Apply[Plus, a]
```
Out[18]= 35

4. spôsob

In[19]:=
```
sucet = 0;
Do[sucet = sucet + a[[i]], {i, 1, Length[a]}]
sucet
```
Out[21]= 35

5. spôsob

In[22]:=
```
Total[a]
```
Out[22]= 35

Aritmetický priemer všetkých prvkov - ukážeme si niekoľko spôsobov ako danú

úlohu naprogramovať

1. spôsob

In[23]:=
```
1 / Length[a] * Sum[a[[i]], {i, 1, Length[a]}]
```

Out[23]= $\frac{7}{2}$

2. spôsob

In[24]:=
```
1 / 10 Sum[a[[i]], {i, 1, 10}]
```

Out[24]= $\frac{7}{2}$

3. spôsob

In[25]:=
```
Mean[a]
```

Out[25]= $\frac{7}{2}$

4. spôsob - funkcionálne programovanie

In[26]:=
```
1 / Length[a] * Apply[Plus, a]
```

Out[26]= $\frac{7}{2}$

Počet prvkov z poľa *a*, ktorých hodnota je menšia ako + 4 - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať
Stačí, ak budete rozumieť a ovládať prvý spôsob, tie ostatné sú pre tých, ktorých to zaujíma viac

1. spôsob - procedurálne programovanie

In[27]:=
```
pocet = 0;
Do[
  If[a[[i]] < 4, pocet = pocet + 1],
  {i, 1, Length[a]}];
pocet
```

Out[29]= 4

2. spôsob - pomocou pure function

In[30]:=
```
b = Select[a, # < 4 &]
Length[b]
```

Out[30]= {-2, -5, -3, -1}

Out[31]= 4

3. spôsob - pomocou predikátových funkcií - toto si ešte porobne budeme vysvetľovať na prednáške

In[32]:=
```
testQ[x_] = If[x < 4, True, False];
Map[testQ, a]
```

Out[33]= {False, True, False, False, False, False, True, True, True, False}

In[34]:=
```
Select[a, testQ]
```

Out[34]= {-2, -5, -3, -1}

In[35]:=
```
Select[a, testQ] // Length
```

Out[35]= 4

Súčet prvkov z poľa *a*, ktorých hodnota je menšia ako + 4 - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať
Stačí, ak budete rozumieť a ovládať prvý spôsob, tie ostatné sú pre tých, ktorých to zaujíma viac

1. spôsob - procedurálne programovanie

In[36]:=
```
sucet = 0;
Do[
  If[a[[i]] < 4, sucet = sucet + a[[i]] ],
  {i, 1, Length[a]}];
sucet
```

Out[38]= -11

2. spôsob - pomocou pure function

In[39]:=
```
b = Select[a, # < 4 &]
Total[b]
```

Out[39]= {-2, -5, -3, -1}

Out[40]= -11

3. spôsob - pomocou predikátových funkcií

In[41]:=
```
testQ[x_] = If[x < 4, True, False];
Select[a, testQ] // Total
```

Out[42]= -11

4. spôsob - pomocou predikátových funkcií a funkcion8lneho programovania

In[43]:=
```
testQ[x_] = If[x < 4, True, False];
b = Select[a, testQ]
Apply[Plus, b]
```

Out[44]= {-2, -5, -3, -1}

Out[45]= -11

Úloha 2

Dané je jednorozmerné pole b, ktorého prvky sú dané predpisom $\frac{i^2-1}{i^2+i-7}$ ak $i \in [-5, 12]$, i je typu Integer.

Vytvorte toto pole a nakreslite jeho prvky.

Čo najjednoduchšie zitite:

- maximálny a minimálny prvok
- súčet všetkých prvkov a aritmetický priemer všetkých prvkov
- počet prvkov z poľa *b*, ktorých hodnota je väčšia ako −0.2 a menšia ako +0.3
- súčet prvkov z poľa *b*, ktorých hodnota je väčšia ako −0.2 a menšia ako +0.3

In[46]:=
```
Clear[b]
b = Table[(i^2 - 1)/(i^2 + i - 7), {i, -5, 12}]
```

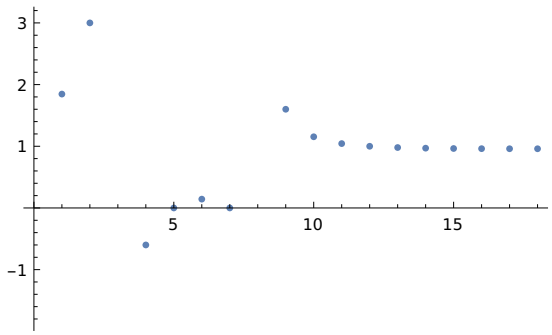
Out[47]= $\left\{\frac{24}{13}, 3, -8, -\frac{3}{5}, 0, \frac{1}{7}, 0, -3, \frac{8}{5}, \frac{15}{13}, \frac{24}{23}, 1, \frac{48}{49}, \frac{63}{65}, \frac{80}{83}, \frac{99}{103}, \frac{24}{25}, \frac{143}{149}\right\}$

In[48]:=
```
Clear[b]
b = Table[(i^2 - 1)/(i^2 + i - 7), {i, -5, 12}] // N
```

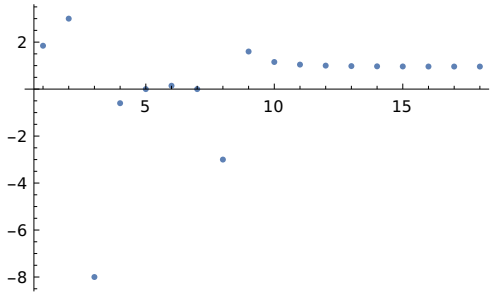
Out[49]= {1.84615, 3., -8., -0.6, 0., 0.142857, 0., -3., 1.6, 1.15385, 1.04348, 1., 0.979592, 0.969231, 0.963855, 0.961165, 0.96, 0.959732}

In[50]:=
```
ListPlot[b]
```

Out[50]=



In[51]:=
```
ListPlot[b, PlotRange → All]
```

Out[51]=


Maximálny a minimálny prvok

In[52]:=
```
Max[b]
```

Out[52]=
```
3.
```

In[53]:=
```
Min[b]
```

Out[53]=
```
-8.
```

Súčet všetkých prvkov - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať

1. spôsob

In[54]:=
```
Sum[b[[i]], {i, 1, Length[b]}]
```

Out[54]=
```
3.97991
```

2. spôsob

In[55]:=
$$\sum_{i=1}^{\text{Length}[b]} b[[i]]$$

Out[55]=
```
3.97991
```

3. spôsob

In[56]:=
```
Apply[Plus, b]
```

Out[56]=
```
3.97991
```

4. spôsob

In[57]:=
```
sucet = 0;
Do[sucet = sucet + b[[i]], {i, 1, Length[b]}]
sucet
```

Out[59]=
```
3.97991
```

5. spôsob

In[60]:=
```
Total[b]
```
Out[60]= 3.97991

Aritmetický priemer všetkých prvkov - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať

1. spôsob

In[61]:=
```
1 / Length[b] * Sum[b[[i]], {i, 1, Length[b]}]
```
Out[61]= 0.221106

2. spôsob

In[62]:=
```
Length[b]
```
Out[62]= 18

In[63]:=
```
1 / Length[b] Sum[b[[i]], {i, 1, 18}]
```
Out[63]= 0.221106

3. spôsob

In[64]:=
```
Mean[b]
```
Out[64]= 0.221106

4. spôsob

In[65]:=
```
1 / Length[b] * Apply[Plus, b]
```
Out[65]= 0.221106

Počet prvkov z poľa *b*, ktorých hodnota je väčšia ako −0.2 a menšia alebo rovná ako +0.3 - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať
Stačí, ak budete rozumieť a ovládať prvý spôsob, tie ostatné sú pre tých, ktorých to zaujíma viac

1. spôsob - procedurálne programovanie

In[66]:=
```
pocet = 0;
Do[
  If[-0.2 < b[[i]] ≤ 0.3, pocet = pocet + 1],
  {i, 1, Length[b]}];
pocet
```
Out[68]= 3

2. spôsob - pomocou pure function

In[69]:=
```
Select[b, -0.2 < # ≤ 0.3 &] // N
Length[%]
```

Out[69]=
```
{0., 0.142857, 0.}
```

Out[70]=
```
3
```

3. spôsob - pomocou predikátových funkcií

In[71]:=
```
testQ[x_] = If[-0.2 < x ≤ 0.3, True, False];
Map[testQ, b]
```

Out[72]=
```
{False, False, False, False, True, True, True, False, False,
 False, False, False, False, False, False, False, False, False}
```

In[73]:=
```
Select[b // N, testQ]
```

Out[73]=
```
{0., 0.142857, 0.}
```

In[74]:=
```
Select[b, testQ] // Length
```

Out[74]=
```
3
```

Súčet prvkov z poľa *b*, ktorých hodnota je väčšia ako −0.2 a menšia alebo rovná ako +0.3 - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať Stačí, ak budete rozumieť a ovládať prvý spôsob, tie ostatné sú pre tých, ktorých to zaujíma viac

1. spôsob - procedurálne programovanie

In[75]:=
```
sucet = 0;
Do[
  If[-0.2 < b[[i]] ≤ 0.3, sucet = sucet + b[[i]],
  {i, 1, Length[b]}];
sucet // N
```

Out[77]=
```
0.142857
```

2. spôsob - pomocou pure function

In[78]:=
```
Select[b, -0.2 < # ≤ 0.3 &] // N
Total[%]
```

Out[78]=
```
{0., 0.142857, 0.}
```

Out[79]=
```
0.142857
```

3. spôsob - pomocou predikátových funkcií

In[80]:=
```
testQ[x_] = If[-0.2 < x ≤ 0.3, True, False];
Select[b, testQ] // Total // N
```

Out[81]=
```
0.142857
```

4. spôsob - pomocou predikátových funkcií a funkcion8lneho programovania

In[82]:=
```
testQ[x_] = If[-0.2 < x ≤ 0.3, True, False];
b1 = Select[b, testQ] // N
Apply[Plus, b1] // N
```

Out[83]=
```
{0., 0.142857, 0.}
```

Out[84]=
```
0.142857
```

Práca s dátovým súborom

Úloha 3

V súbore data.txt, ktorý je uložený v AISe sú uložené dáta pochádzajúce z merania. Načítajte tieto dáta do poľa c a nakreslite jeho prvky.

Čo najjednoduchšie zitite:

- rozsah hodnôt poľa
- počet prvkov z poľa *c*, ktorých hodnota je väčšia ako –0.47 a menšia ako +0.75
- súčet absolútnych hodnôt prvkov z poľa *c*, ktorých hodnota je väčšia ako –0.47 a menšia ako +0.75

In[103]:=
```
$Path
Directory[]
FileNames[]
```

Out[103]=
```
{/wolframcloud/userfiles/WolframApplications ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/SystemFiles/Links ,
 /wolframcloud/userfiles/783/783d7413-175c-4501-9beb-69777cd3b6dd/Base/Kernel ,
 /wolframcloud/userfiles/783/783d7413-175c-4501-9beb-69777cd3b6dd/Base/Autoload ,
 /wolframcloud/userfiles/783/783d7413-175c-4501-9beb-69777cd3b6dd/Base/Applications ,
 /usr/share/Mathematica/Kernel , /usr/share/Mathematica/Autoload ,
 /usr/share/Mathematica/Applications , .,
 /wolframcloud/userfiles/783/783d7413-175c-4501-9beb-69777cd3b6dd ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/SystemFiles/Autoload ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/AddOns/Autoload ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/AddOns/Applications ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/AddOns/Packages ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/AddOns/ExtraPackages ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/SystemFiles/Kernel/Packages ,
 /usr/local/Wolfram/WolframEngine/12.3.1.7360342/Documentation/English/System}
```

Out[104]=
```
/wolframcloud/userfiles/783/783d7413-175c-4501-9beb-69777cd3b6dd/Cvicenia/3
```

Out[105]=
```
{03 cvicenie 1D .nb, .03 cvicenie 1D .nb.lock,
 03 cvicenie 1D - samostatna praca.nb, 03 cvicenie 2D.nb,
 03 cvicenie 2D - samostatna praca.nb, data1.txt, data1.xls, data2.txt,
 data3.txt, data3.xls, priemer.txt, priemer.xls, vzduch.txt, vzduch.xls}
```

In[102]:=
```
SetDirectory["Cvicenia/3"]
```

Out[102]=
```
/wolframcloud/userfiles/783/783d7413-175c-4501-9beb-69777cd3b6dd/Cvicenia/3
```

In[108]:=
```
Clear[c]
c = ReadList["data1.txt"]
```

Out[109]=
```
{11, -20, 5, -17, 10, 16, 2, 9, -6, -6, -7, -10, 13, 14, -15, -16, 14, -2, 9, -7, 19, -14,
 -3, 21, 7, -20, 1, 3, 21, -13, -14, 19, 18, 0, -18, -12, -5, 6, -9, 23, -9, 2, 16, 1,
 -12, 21, 12, 12, 3, -18, -14, 2, 6, 0, -7, 9, 9, -10, -8, 13, 12, 14, 4, -5, 14, 9,
 13, 25, -5, 21, 14, 21, -18, -1, 5, -14, 5, 7, 0, 9, -19, 18, 15, -16, 8, -2, 20, 24,
 11, -15, 8, 24, 1, 14, 16, -19, -14, -14, -8, 9, 24, 25, 11, 24, 17, 22, 6, -9, -11,
 22, 14, -4, 9, 13, 1, -11, -8, 0, -11, -4, -5, 18, -1, -19, -10, -13, 5, 25, 10, 14,
 7, 9, 25, -6, 18, 4, -15, -15, -11, 15, 20, -9, 19, -13, 13, 20, -1, 10, -10, -17}
```

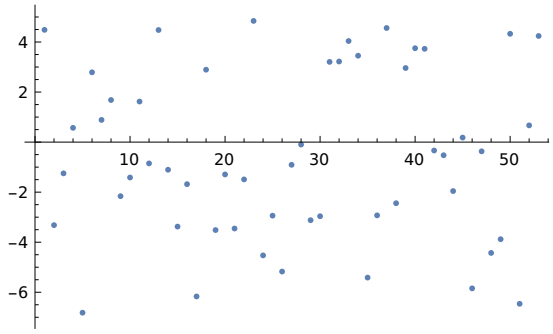
Ak máte problém s načítaním dát zo súboru, skús jednoduchý trik. Použite na na čítanie príkaz ReadList spolu so špecifikáciou očakávaných dát. Veľmi často je totiž problém skrytý práve v tom, ze *Mathematica* interpretuje číselné dáta ako text.

In[118]:=
```
Clear[c]
c = ReadList["data3.txt", Number]
```

Out[119]=
```
{4.48577 , -3.31824 , -1.24881 , 0.57016 , -6.81747 , 2.78922 , 0.886956 , 1.6822 ,
 -2.16072 , -1.41482 , 1.6207 , -0.851476 , 4.47931 , -1.10492 , -3.37613 , -1.68036 ,
 -6.16827 , 2.89324 , -3.51446 , -1.29168 , -3.45301 , -1.49089 , 4.84372 ,
 -4.52733 , -2.93878 , -5.17265 , -0.907472 , -0.0974924 , -3.12131 , -2.96187 ,
 3.20557 , 3.22031 , 4.03941 , 3.45295 , -5.41513 , -2.92822 , 4.5601 , -2.44213 ,
 2.961 , 3.75215 , 3.72837 , -0.335367 , -0.52454 , -1.95617 , 0.181377 , -5.84448 ,
 -0.368262 , -4.42884 , -3.87985 , 4.32817 , -6.46079 , 0.668655 , 4.24146}
```

In[120]:=
```
ListPlot[c]
```

Out[120]=



Podstatné pre vás je - dostať akýmkoľvek spôsobom dáta do Mathematice - a potom ich už budeme vedieť spracovať

Rozsah, v ktorom sa prvky nachádzajú

In[121]:=
```
Print["prvky poľa ležia v intervale (", Min[c], " , ", Max[c], ")"]
```

```
prvky poľa ležia v intervale (-6.81747 , 4.84372 )
```

Počet prvkov z poľa *c*, ktorých hodnota je väčšia ako −0.47 a menšia ako +0.75 - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať
Stačí, ak budete rozumieť a ovládať prvý spôsob, tie ostatné sú pre tých, ktorých to zaujíma viac

1. spôsob - procedurálne programovanie

In[122]:=
```
pocet = 0;
Do[
  If[-0.47 < c[[i]] < 0.75, pocet = pocet + 1],
  {i, 1, Length[c]}];
pocet
```

Out[124]=
```
6
```

2. spôsob - pomocou pure function

In[125]:=
```
Select[c, -0.47 < # < 0.75 &]
Length[%]
```

Out[125]= {0.57016, -0.0974924, -0.335367, 0.181377, -0.368262, 0.668655}

Out[126]= 6

3. spôsob - pomocou predikátových funkcií

In[127]:=
```
testQ[x_] = If[-0.47 < x < 0.75, True, False];
Map[testQ, c]
```

Out[128]= {False, False, False, True, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, True, False, False, False,
False, False, False, False, False, False, False, False, False, False, True,
False, False, True, False, True, False, False, False, False, True, False}

In[129]:=
```
Select[c, testQ]
```

Out[129]= {0.57016, -0.0974924, -0.335367, 0.181377, -0.368262, 0.668655}

In[130]:=
```
Select[c, testQ] // Length
```

Out[130]= 6

Súčet absolútnych hodnôt prvkov z poľa *c*, ktorých hodnota je väčšia ako −0.47 a menšia ako +0.75 - ukážeme si niekoľko spôsobov ako danú úlohu naprogramovať
Stačí, ak budete rozumieť a ovládať prvý spôsob, tie ostatné sú pre tých, ktorých to zaujíma viac

1. spôsob - procedurálne programovanie

In[131]:=
```
sucet = 0;
Do[
  If[-0.47 < c[[i]] < 0.75, sucet = sucet + Abs[c[[i]] ],
  {i, 1, Length[c]}];
sucet
```

Out[133]= 2.22131

2. spôsob - pomocou pure function

In[134]:=
```
c1 = Select[c, -0.47 < # < 0.75 &]
Total[Abs[c1]]
```

Out[134]= {0.57016 , -0.0974924 , -0.335367 , 0.181377 , -0.368262 , 0.668655}

Out[135]= 2.22131

3. spôsob - pomocou predikátových funkcií

In[136]:=
```
testQ[x_] = If[-0.47 < x < 0.75, True, False];
Select[c, testQ] // Abs // Total
```

Out[137]= 2.22131

4. spôsob - pomocou predikátových funkcií a funkcion8lneho programovania

In[138]:=
```
testQ[x_] = If[-0.47 < x < 0.75, True, False];
c1 = Select[c, testQ]
Apply[Plus, Abs[c1]]
```

Out[139]= {0.57016 , -0.0974924 , -0.335367 , 0.181377 , -0.368262 , 0.668655}

Out[140]= 2.22131

Úloha 4 - práca s dátovými súbormi v rôznych formátoch

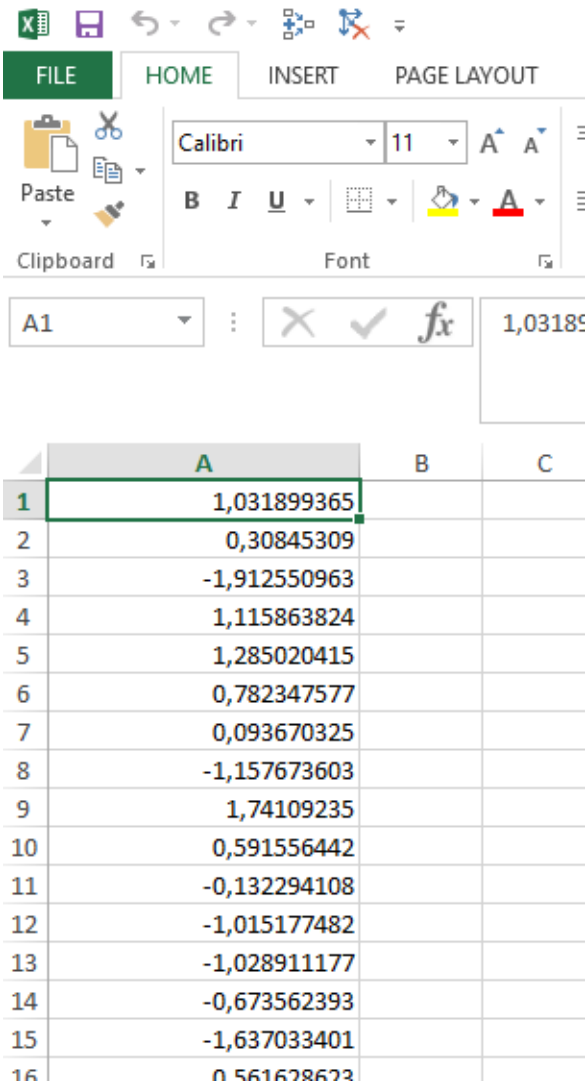
Úloha načítajte dáta zo súboru data.txt

In[141]:=
```
Clear[c]
c = ReadList["data3.txt"]
```

Out[142]= {4.48577 , -3.31824 , -1.24881 , 0.57016 , -6.81747 , 2.78922 , 0.886956 , 1.6822 ,
-2.16072 , -1.41482 , 1.6207 , -0.851476 , 4.47931 , -1.10492 , -3.37613 , -1.68036 ,
-6.16827 , 2.89324 , -3.51446 , -1.29168 , -3.45301 , -1.49089 , 4.84372 ,
-4.52733 , -2.93878 , -5.17265 , -0.907472 , -0.0974924 , -3.12131 , -2.96187 ,
3.20557 , 3.22031 , 4.03941 , 3.45295 , -5.41513 , -2.92822 , 4.5601 , -2.44213 ,
2.961 , 3.75215 , 3.72837 , -0.335367 , -0.52454 , -1.95617 , 0.181377 , -5.84448 ,
-0.368262 , -4.42884 , -3.87985 , 4.32817 , -6.46079 , 0.668655 , 4.24146}

Úloha načítajte dáta zo súboru data.xls

Tie isté dáta sa nachádzajú v súbore xls. Pozrime sa na ich štruktúru - takto vyzerá začiatok súboru v Exceli. Dáta načítame pomocou príkazu Import [ ]



	A	B	C
1	1,031899365		
2	0,30845309		
3	-1,912550963		
4	1,115863824		
5	1,285020415		
6	0,782347577		
7	0,093670325		
8	-1,157673603		
9	1,74109235		
10	0,591556442		
11	-0,132294108		
12	-1,015177482		
13	-1,028911177		
14	-0,673562393		
15	-1,637033401		
16	0,561628623		

In[143]:=
```
pomocna = Import["data3.xls"]
```

Out[143]=
```
{{{4.48577}, {-3.31824}, {-1.24881}, {0.57016}, {-6.81747}, {2.78922}, {0.886956},
  {1.6822}, {-2.16072}, {-1.41482}, {1.6207}, {-0.851476}, {4.47931}, {-1.10492},
  {-3.37613}, {-1.68036}, {-6.16827}, {2.89324}, {-3.51446}, {-1.29168}, {-3.45301},
  {-1.49089}, {4.84372}, {-4.52733}, {-2.93878}, {-5.17265}, {-0.907472},
  {-0.0974924}, {-3.12131}, {-2.96187}, {3.20557}, {3.22031}, {4.03941}, {3.45295},
  {-5.41513}, {-2.92822}, {4.5601}, {-2.44213}, {2.961}, {3.75215}, {3.72837},
  {-0.335367}, {-0.52454}, {-1.95617}, {0.181377}, {-5.84448}, {-0.368262},
  {-4.42884}, {-3.87985}, {4.32817}, {-6.46079}, {0.668655}, {4.24146}}}
```

Všimnite si, že tých zátvoriek je tam oproti predchádzajúcemu spôsobu nejako príliš veľa. Prečo? Každá bunka je považovaná za samostatný element a môžeme načítavať aj z viacerých Sheetov (Záložiek). Potrebujeme sa ich zbaviť. Najjednoduchší spôsob je pomocou príkazu Flatten odstrániť štruktúru viacrozmerného poľa

In[144]:=
```
data = Flatten[pomocna]
```

Out[144]=
```
{4.48577 , -3.31824 , -1.24881 , 0.57016 , -6.81747 , 2.78922 , 0.886956 , 1.6822 ,
  -2.16072 , -1.41482 , 1.6207 , -0.851476 , 4.47931 , -1.10492 , -3.37613 , -1.68036 ,
  -6.16827 , 2.89324 , -3.51446 , -1.29168 , -3.45301 , -1.49089 , 4.84372 ,
  -4.52733 , -2.93878 , -5.17265 , -0.907472 , -0.0974924 , -3.12131 , -2.96187 ,
  3.20557 , 3.22031 , 4.03941 , 3.45295 , -5.41513 , -2.92822 , 4.5601 , -2.44213 ,
  2.961 , 3.75215 , 3.72837 , -0.335367 , -0.52454 , -1.95617 , 0.181377 , -5.84448 ,
  -0.368262 , -4.42884 , -3.87985 , 4.32817 , -6.46079 , 0.668655 , 4.24146}
```

Úloha načítajte dáta zo súboru ocel.xls

Tieto dáta už oveľa viac pripomínajú reálne dáta. Uvedomte si, že reálne dáta budú mať rôznu štruktúru a nikto v praxi nebude tráviť čas tým, aby vám dáta upravil. Budete si to musieť vyriešiť sami. Takže - ako na to?

Skôr ako dáta načítate, premyslite si, čo je jednoduchšia cesta a ktoré dáta z dodanej tabuľky potrebujem.

Jedna možnosť je vymazať v xls súbore všetko čo nepotrebujem a dáta usporiadať tak, aby tvorili jeden stĺpec. Prípadne si vytvoriť viacero pomocných xls. súborov a nčítať dáta samostatne.

Druhá možnosť - lepšia je pozrieť sa na xls tabuľku ako na dvojrozmernú maticu. Bunka A1 bude prvok na pozícii (1,1) v našej imaginárnej matici. Bunka B4 bude prvok na pozícii (4,2) - 4. riadok, 2. stĺpec. S dátami potom môžeme manipulovať ako s maticou a povedať si, ktoré dáta chceme - napríklad stĺpec B bude vlastne len 2. stĺpec v matici.

	A	B	C
1	0:00	0,226905383	
2	1:00	0,065937173	
3	2:00	0,0781936	
4	3:00	0,141301743	
5	4:00	0,244768592	
6	5:00	0,180601819	
7	6:00	0,249825385	
8	7:00	0,054461107	
9	8:00	0,048258184	
10	9:00	0,053898433	
11	10:00	0,066076404	
12	11:00	0,14574465	
13	12:00	0,238735859	
14			

Nezabudnite, že *Mathematica* automaticky načítava dáta po jednotlivých sheetoch (záložkách), takže ak chceme pracovať v prvom pracovnom liste, musíme to Mathematice povedať

In[145]:=
```
HrubeNacitanie = Import["vzduch.xls"]
```

Out[145]=
```
{{{90.}, {120.}, {-18.}, {113.}, {54.}, {128.}, {84.}, {68.}, {104.}, {137.}, {110.}, {80.}, {116.},
   {72.}, {147.}, {67.}, {96.}, {60.}, {120.}, {133.}, {54.}, {111.}, {139.}, {134.}, {79.},
   {130.}, {60.}, {120.}, {139.}, {142.}, {88.}, {116.}, {125.}, {57.}, {137.}, {103.}, {128.},
   {70.}, {56.}, {124.}, {104.}, {58.}, {99.}, {131.}, {138.}, {121.}, {62.}, {58.}}, {{}}, {{}}}
```

In[146]:=
```
Import["vzduch.xls", "Elements"]
```

Out[146]=
```
{Data, Dataset, Dimensions, FormattedData, Formulas, Images, SheetCount, Sheets}
```

Načítaj len dáta z prvej záložky - dostaneme čisté dáta

In[147]:=
```
PrvyPracovnyList = Import["vzduch.xls", {"Data", 1}]
```

Out[147]=
```
{{90.}, {120.}, {-18.}, {113.}, {54.}, {128.}, {84.}, {68.}, {104.}, {137.}, {110.}, {80.},
  {116.}, {72.}, {147.}, {67.}, {96.}, {60.}, {120.}, {133.}, {54.}, {111.}, {139.}, {134.},
  {79.}, {130.}, {60.}, {120.}, {139.}, {142.}, {88.}, {116.}, {125.}, {57.}, {137.}, {103.},
  {128.}, {70.}, {56.}, {124.}, {104.}, {58.}, {99.}, {131.}, {138.}, {121.}, {62.}, {58.}}
```

Teraz už vidíme, že je to dvojrozmerná matica. Takto načítame prvý riadok

In[148]:=
```
PrvyPracovnyList[[1]]
```

Out[148]=
```
{90.}
```

Takto načítame prvý stĺpec

In[149]:=
```
PrvyPracovnyList[[All, 1]]
```

Out[149]=
```
{90., 120., -18., 113., 54., 128., 84., 68., 104., 137., 110.,
  80., 116., 72., 147., 67., 96., 60., 120., 133., 54., 111., 139.,
  134., 79., 130., 60., 120., 139., 142., 88., 116., 125., 57., 137.,
  103., 128., 70., 56., 124., 104., 58., 99., 131., 138., 121., 62., 58.}
```

Takto načítame druhý stĺpec.

In[150]:=
```
PrvyPracovnyList[[All, 2]]
```

```
Part: Part 2 of {{90.}, {120.}, {-18.}, {113.}, {54.}, {128.}, {84.}, {68.}, {104.}, {137.}, ≪38≫} does not exist.
```

Out[150]=
```
{{90.}, {120.}, {-18.}, {113.}, {54.}, {128.}, {84.}, {68.}, {104.}, {137.}, {110.}, {80.}, {116.},
   {72.}, {147.}, {67.}, {96.}, {60.}, {120.}, {133.}, {54.}, {111.}, {139.}, {134.}, {79.},
   {130.}, {60.}, {120.}, {139.}, {142.}, {88.}, {116.}, {125.}, {57.}, {137.}, {103.}, {128.},
   {70.}, {56.}, {124.}, {104.}, {58.}, {99.}, {131.}, {138.}, {121.}, {62.}, {58.}}[[All, 2]]
```

Potrebujeme napríklad len údaje od 8:00 do 12:00 (vrátane). V tabuľke vidíme, že je to posledných 5 riadkov

In[151]:=
```
PrvyPracovnyList[[-5 ;; -1]]
```

Out[151]=
```
{{131.}, {138.}, {121.}, {62.}, {58.}}
```

Takže vždy podľa potreby a na základe toho, čo sme sa naučili s maticami - použijeme potrebný formát.

Šikovný a lenivý programátor to môt´že urobiť aj v jednom kroku - v indexoch máme

1 - prvý sheet (záložka)

9 ;; 13 - riadky 9 až 13 z našej tabuľky

All - všetky stĺpce

In[152]:=
```
Import["vzduch.xls"][[1, 9 ;; 13, All]]
```

Out[152]=
```
{{104.}, {137.}, {110.}, {80.}, {116.}}
```

Pri importe súborov vždy najskôr rozmýšľajte a až potom konajte. Hľadajte tú najjednoduchšiu cestu.