

Pandas



DataFrame

```
import pandas as pd
```

	CustomerId	CreditScore	NumOfProducts
0	15634602	619	1
1	15647311	608	1
2	15619304	502	3
3	15701354	699	2
4	15737888	850	1



Comma Separated Values (CSV)

TEST CSV - Excel																									
File Home Insert Page Layout Formulas Data Review View Tell me what you want to do... Medroso, Diane Share																									
Clipboard Font Alignment Number Styles Cells Editing																									
A1 NodeID,"ObjectSubType","IP_Address","IP_Address_Type","DynamicIP","UnManaged","UnManagedFrom","UnManagedUntil","Caption","DNS","Community","RWCommunity","Vendor","SysObjectID","De																									
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U					
1	NodeID,"ObjectSubType","IP_Address","IP_Address_Type","DynamicIP","UnManaged","UnManagedFrom","UnManagedUntil","Caption","DNS","Community","RWCommunity","SysName","Vendor","SysObjectID","De																								
2	1,"Agent","10.203.69.30","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","sup-man-dmed-01","sup-man-dmed-01.swdev.local","","","sup-man-dmed-01","Windows","","Hardware: Intel6																								
3	2,"SNMP","10.199.6.52","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-IBM-DIRECT","LAB-IBM-DIRECT","public","","LAB-IBM-DIRECT","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardwar																								
4	3,"SNMP","10.199.6.56","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","lab-oncmd-core","LAB-ONCMD-CORE","public","","lab-oncmd-core","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardw																								
5	4,"SNMP","10.199.6.79","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-ORACLEFORM","LAB-ORACLEFORM","public","","LAB-ORACLEFORM","Windows","1.3.6.1.4.1.311.1.3.1.2","Ha																								
6	5,"SNMP","10.199.6.89","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","se-nable-prbe","SE-NABLE-PRBE","public","","se-nable-prbe","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardware: In																								
7	6,"SNMP","10.199.6.99","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","lab-unidata","LAB-UNIDATA","public","","lab-unidata","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardware: Intel64 Fa																								
8	7,"SNMP","10.199.6.104","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","kkat-bakupsvr.esntslab.local","KKAT-BAKUPSVR","public","","kkat-bakupsvr.esntslab.local","Windows","1.3.6.1																								
9	8,"SNMP","10.199.6.105","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","qa-nable-at-03","QA-NABLE-AT-03","public","","qa-nable-at-03","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardware																								
10	9,"SNMP","10.199.6.108","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","QA-NABLE-AT-06","QA-NABLE-AT-06","public","","QA-NABLE-AT-06","Windows","1.3.6.1.4.1.311.1.3.1.1","Hardv																								
11	10,"SNMP","10.199.6.123","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","stp-2k8-fchba","STP-2K8-FCHBA","public","","stp-2k8-fchba","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardware: In																								
12	11,"SNMP","10.199.6.124","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","lab-clussql-01.lab.tex","LAB-CLUSSQL-01","public","","lab-clussql-01.lab.tex","Windows","1.3.6.1.4.1.311.1.3.1.1																								
13	12,"SNMP","10.199.6.125","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","lab-clussql-02.lab.tex","LAB-CLUSSQL-02","public","","lab-clussql-02.lab.tex","Windows","1.3.6.1.4.1.311.1.3.1.1																								
14	13,"SNMP","10.199.6.135","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","lab-bes12.lab.tex","LAB-BES12","public","","lab-bes12.lab.tex","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardware																								
15	14,"SNMP","10.199.6.146","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-ARCSERVE12","LAB-ARCSERVE12","public","","LAB-ARCSERVE12","Windows","1.3.6.1.4.1.311.1.3.1.2","Har																								
16	15,"SNMP","10.199.6.156","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-JBOSS4","LAB-JBOSS4","public","","LAB-JBOSS4","Windows","1.3.6.1.4.1.311.1.3.1.2","Hardware: x86 Fami																								
17	16,"SNMP","10.199.6.164","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","lab-nable-sql.sc.lab","LAB-NABLE-SQL","public","","lab-nable-sql.sc.lab","Windows","1.3.6.1.4.1.311.1.3.1.2","																								
18	17,"WMI","10.199.6.91","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","QA-AUS-KKAT-01","QA-AUS-KKAT-01","","","qa-aus-kkat-01","Windows","","Hardware: Intel64 Family 6 Model 44 S																								
19	18,"WMI","10.199.6.94","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-BIZTALK2013","LAB-BIZTALK2013","","","lab-biztalk2013","Windows","","Hardware: Intel64 Family 6 Model 62 S																								
20	19,"WMI","10.199.6.122","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","QA-NABLE-AT-02","QA-NABLE-AT-02","","","qa-nable-at-02","Windows","","Hardware: Intel64 Family 6 Model 62																								
21	20,"WMI","10.199.6.60","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-2012R2-HBA","LAB-2012R2-HBA","","","lab-2012r2-hba","Windows","","Hardware: Intel64 Family 6 Model 26 S																								
22	21,"WMI","10.199.6.75","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-FBCH-ISCSI","LAB-FBCH-ISCSI","","","lab-fbch-iscsi","Windows","","Hardware: Intel64 Family 6 Model 26 Step																								
23	22,"WMI","10.199.6.84","IPv4","False","False","12/30/1899 12:00:00 AM","12/30/1899 12:00:00 AM","LAB-NABLE-SVR03","LAB-NABLE-SVR03","","","lab-nable-svr03","Windows","","Hardware: Intel64 Family 6 Model 4																								



Adatok betöltése

```
df = pd.read_csv("Churn_Modelling.csv")
```

```
df = pd.read_csv("Churn_Modelling.csv", usecols=cols, nrows=500)
```

```
cols = ['CustomerId', 'CreditScore', 'NumOfProducts']  
df = pd.read_csv("Churn_Modelling.csv", usecols=cols)
```



DataFrame adatszerkezetből

```
df = pd.DataFrame({'a':np.random.rand(10),  
                   'b':np.random.randint(10, size=10),  
                   'c':[True,True,True,False,False,np.nan,np.nan,  
                        False,True,True],  
                   'b':['London','Paris','New York','Istanbul',  
                        'Liverpool','Berlin',np.nan,'Madrid',  
                        'Rome',np.nan],  
                   'd':[3,4,5,1,5,2,2,np.nan,np.nan,0],  
                   'e':[1,4,5,3,3,3,3,8,8,4]})
```



Hagyományos indexing (sorok)

df

	In_stock	Price
Fruits_name		
Apple	yes	10
Banana	yes	7
Orange	no	5

df[1:]	<table><thead><tr><th></th><th>In_stock</th><th>Price</th></tr></thead><tbody><tr><th>Fruits_name</th><td></td><td></td></tr><tr><td>Banana</td><td>yes</td><td>7</td></tr><tr><td>Orange</td><td>no</td><td>5</td></tr></tbody></table>		In_stock	Price	Fruits_name			Banana	yes	7	Orange	no	5	[1:] → Slicing row index position [Returns row with index 1 till the last row]
	In_stock	Price												
Fruits_name														
Banana	yes	7												
Orange	no	5												
df[:1]	<table><thead><tr><th></th><th>In_stock</th><th>Price</th></tr></thead><tbody><tr><th>Fruits_name</th><td></td><td></td></tr><tr><td>Apple</td><td>yes</td><td>10</td></tr></tbody></table>		In_stock	Price	Fruits_name			Apple	yes	10	Returns row 0 only. End index is exclusive			
	In_stock	Price												
Fruits_name														
Apple	yes	10												
df[::2]	<table><thead><tr><th></th><th>In_stock</th><th>Price</th></tr></thead><tbody><tr><th>Fruits_name</th><td></td><td></td></tr><tr><td>Apple</td><td>yes</td><td>10</td></tr><tr><td>Orange</td><td>no</td><td>5</td></tr></tbody></table>		In_stock	Price	Fruits_name			Apple	yes	10	Orange	no	5	[::2] → Returns row 0 till last row by step 2 [alternate rows]
	In_stock	Price												
Fruits_name														
Apple	yes	10												
Orange	no	5												
df["Banana":]	<table><thead><tr><th></th><th>In_stock</th><th>Price</th></tr></thead><tbody><tr><th>Fruits_name</th><td></td><td></td></tr><tr><td>Banana</td><td>yes</td><td>7</td></tr><tr><td>Orange</td><td>no</td><td>5</td></tr></tbody></table>		In_stock	Price	Fruits_name			Banana	yes	7	Orange	no	5	Slicing by row index values
	In_stock	Price												
Fruits_name														
Banana	yes	7												
Orange	no	5												
df["Apple":"Banana"]	<table><thead><tr><th></th><th>In_stock</th><th>Price</th></tr></thead><tbody><tr><th>Fruits_name</th><td></td><td></td></tr><tr><td>Apple</td><td>yes</td><td>10</td></tr><tr><td>Banana</td><td>yes</td><td>7</td></tr></tbody></table>		In_stock	Price	Fruits_name			Apple	yes	10	Banana	yes	7	Returns row "Apple" till "Banana". End index is inclusive.
	In_stock	Price												
Fruits_name														
Apple	yes	10												
Banana	yes	7												



Hagyományos indexing (oszlopok)

<code>df["Price"]</code>	<pre>Fruits_name Apple 10 Banana 7 Orange 5 Name: Price, dtype: int64</pre>	Return Type → Series
<code>df.Price</code>	<pre>Fruits_name Apple 10 Banana 7 Orange 5 Name: Price, dtype: int64</pre>	Return Type → Series
<code>df[["Price"]]</code>	<pre> Price Fruits_name Apple 10 Banana 7 Orange 5</pre>	Return Type → Dataframe
<code>df[["Price", "In_stock"]]</code>	<pre> Price In_stock Fruits_name Apple 10 yes Banana 7 yes Orange 5 no</pre>	Return Type → Dataframe

`df`

	In_stock	Price
Fruits_name		
Apple	yes	10
Banana	yes	7
Orange	no	5



Hagyományos indexing (sorok és oszlopok)

<code>df[1:]["Price"]</code>	<pre>1 7 2 5 Name: Price, dtype: int64</pre>									
<code>df[:]["Price"]</code>	<pre>0 10 1 7 2 5 Name: Price, dtype: int64</pre>									
<code>df[:][["Price"]]</code>	<table><tr><th></th><th>Price</th></tr><tr><td>0</td><td>10</td></tr><tr><td>1</td><td>7</td></tr><tr><td>2</td><td>5</td></tr></table>		Price	0	10	1	7	2	5	
	Price									
0	10									
1	7									
2	5									
<code>df[1:][["Price", "Fruits_name"]]</code>	<table><tr><th></th><th>Price</th><th>Fruits_name</th></tr><tr><td>1</td><td>7</td><td>Banana</td></tr><tr><td>2</td><td>5</td><td>Orange</td></tr></table>		Price	Fruits_name	1	7	Banana	2	5	Orange
	Price	Fruits_name								
1	7	Banana								
2	5	Orange								

`df`

	Fruits_name	In_stock	Price
0	Apple	yes	10
1	Banana	yes	7
2	Orange	no	5



Indexing függvényekkel

	a	b	c	d	e
0	0.041211	London	True	3.0	1
1	0.571258	Paris	True	4.0	4
2	0.676766	New York	True	5.0	5
3	0.674262	Istanbul	False	1.0	3
4	0.477875	Liverpool	False	5.0	3
5	0.532795	Berlin	NaN	2.0	3
6	0.823467	NaN	NaN	2.0	3
7	0.929395	Madrid	False	NaN	8
8	0.652076	Rome	True	NaN	8
9	0.722104	NaN	True	0.0	4

`df.iloc[]`

```
df.iloc[1]
a    0.571258
b      Paris
c        True
d          4
e          4
Name: 1, dtype: object
```

```
df.iloc[0,1]
'London'
```

`df.loc[]`

```
df.loc[:2, 'b']
0      London
1      Paris
2    New York
Name: b, dtype: object
```



Boolean indexing

	City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_gender	Medal
0	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	100m freestyle	M	Gold
1	Athens	1896	Aquatics	Swimming	HERSCHMANN, Otto	AUT	Men	100m freestyle	M	Silver
2	Athens	1896	Aquatics	Swimming	DRIVAS, Dimitrios	GRE	Men	100m freestyle for sailors	M	Bronze
3	Athens	1896	Aquatics	Swimming	MALOKINIS, Ioannis	GRE	Men	100m freestyle for sailors	M	Gold
4	Athens	1896	Aquatics	Swimming	CHASAPIS, Spiridon	GRE	Men	100m freestyle for sailors	M	Silver
5	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	1200m freestyle	M	Bronze
6	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	1200m freestyle	M	Gold
7	Athens	1896	Aquatics	Swimming	ANDREOU, Joannis	GRE	Men	1200m freestyle	M	Silver
8	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	400m freestyle	M	Bronze
9	Athens	1896	Aquatics	Swimming	NEUMANN, Paul	AUT	Men	400m freestyle	M	Gold
10	Athens	1896	Aquatics	Swimming	PEPANOS, Antonios	GRE	Men	400m freestyle	M	Silver
11	Athens	1896	Athletics	Athletics	LANE, Francis	USA	Men	100m	M	Bronze
12	Athens	1896	Athletics	Athletics	SZOKOLYI, Alajos	HUN	Men	100m	M	Bronze
13	Athens	1896	Athletics	Athletics	BURKE, Thomas	USA	Men	100m	M	Gold
14	Athens	1896	Athletics	Athletics	HOFMANN, Fritz	GER	Men	100m	M	Silver
15	Athens	1896	Athletics	Athletics	CURTIS, Thomas	USA	Men	110m hurdles	M	Gold

```
data.Medal == 'Silver'
```

```
data[data.Medal == 'Silver']
```

```
data[(data.Medal == 'Silver') & (data.Gender == 'Men')]
```



Hiányzó értékek

	column_a	column_b	column_c	column_d	column_e
0	1.0	1.2	a	True	1
1	2.0	1.4	NaN	True	2
2	4.0	NaN	c	NaN	<NA>
3	4.0	6.2	d	None	4
4	NaN	NaN	NaN	False	<NA>
5	NaN	1.1	NaN	True	5
6	6.0	4.3	d	False	<NA>

```
df.replace(['?', '--'], np.nan, inplace=True)
```

```
df.dropna(axis=0, how='all', inplace=True)
```

```
df.dropna(axis=0, thresh=3)
```

```
df.fillna(25)
```

```
df.fillna(axis=0, method='ffill')
```



Feladat

- 1. Olvassátok be az Iris.csv-t egy DataFrame-be, jelenítsétek meg a DF első 10 sorát, majd számoljátok meg, hogy mennyi hiányzó érték van benne oszlopoként és összesítve egy számként
- 2. Számoljátok ki az oszlopok értékeinek az átlagát külön-külön, és az adott oszlop hiányzó értékeit cseréljétek ki a megfelelő átlagra



DataFrame kombinálás

df1

	column_a	column_b	column_c
0	1	a	True
1	2	b	True
2	3	c	False
3	4	d	True

df2

	column_a	column_b	column_c
0	1	a	False
1	2	k	False
2	9	l	False
3	10	m	True



pd.concat()

```
df = pd.concat([df1,df2])  
df
```

	column_a	column_b	column_c
0	1	a	True
1	2	b	True
2	3	c	False
3	4	d	True
0	1	a	False
1	2	k	False
2	9	l	False
3	10	m	True

```
df = pd.concat([df1,df2], axis=1)  
df
```

	column_a	column_b	column_c	column_a	column_b	column_c
0	1	a	True	1	a	False
1	2	b	True	2	k	False
2	3	c	False	9	l	False
3	4	d	True	10	m	True

```
df = pd.concat([df1,df2], ignore_index=True)  
df
```

	column_a	column_b	column_c
0	1	a	True
1	2	b	True
2	3	c	False
3	4	d	True
4	1	a	False
5	2	k	False
6	9	l	False
7	10	m	True



pd.merge()

df1

	column_a	column_b	column_c
0	1	a	True
1	2	b	True
2	3	c	False
3	4	d	True

df2

	column_a	column_b	column_c
0	1	a	False
1	2	k	False
2	9	l	False
3	10	m	True

```
df_merge = pd.merge(df1, df2, on='column_a')  
df_merge
```

	column_a	column_b_x	column_c_x	column_b_y	column_c_y
0	1	a	True	a	False
1	2	b	True	k	False

```
df_merge = pd.merge(df1, df2, on=['column_a','column_b'])  
df_merge
```

	column_a	column_b	column_c_x	column_c_y
0	1	a	True	False



```
df_merge = pd.merge(df1, df2, how='left', on='column_a')
```

First DataFrame		
column_a	column_b	column_c

Second DataFrame		
column_a	column_b	column_c

how = 'left'				
column_a	column_b_x	column_c_x	column_b_y	column_c_y
			NaN	NaN
			NaN	NaN

how = 'inner'				
column_a	column_b_x	column_c_x	column_b_y	column_c_y

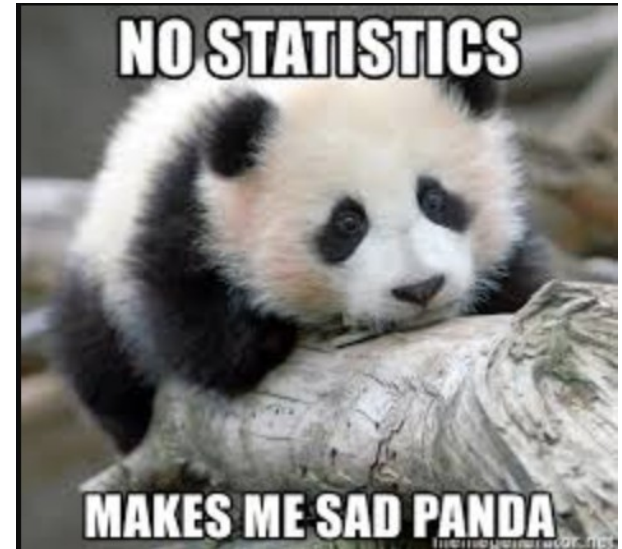
how = 'outer'				
column_a	column_b_x	column_c_x	column_b_y	column_c_y
			NaN	NaN
			NaN	NaN
	NaN	NaN		
	NaN	NaN		

how = 'right'				
column_a	column_b_x	column_c_x	column_b_y	column_c_y
	NaN	NaN		
	NaN	NaN		



Groupby

	country	continent	population	area	population percentage of world
0	United States	North America	332722557	3796742	4.180
1	Canada	North America	38711108	3855100	0.487
2	United Kingdom	Europe	67081234	93628	0.843
3	France	Europe	67853000	247368	0.853
4	Germany	Europe	83222442	137882	1.050
5	China	Asia	1412600000	3705407	17.800
6	Japan	Asia	125502000	145937	1.580
7	South Korea	Asia	51745000	38690	0.650



```
df.groupby("continent").count()
```

	country	population	area	population percentage of world
continent				
Asia	3	3	3	3
Europe	3	3	3	3
North America	2	2	2	2

```
df_agg = df.groupby("continent").agg({"country": "count",  
"population": ["sum", "min", "max"]})
```

	country	population		
	count	sum	min	max
continent				
Asia	3	1589847000	51745000	1412600000
Europe	3	218156676	67081234	83222442
North America	2	371433665	38711108	332722557



Feladat

- 1. Számoljátok ki a fajonkénti darabszámot, a legnagyobb szirm hosszt, a legkisebb levél szélességet, a szirmok szélességének az átlagát
- 2. Olvassátok be újból az Iris.csv-t egyszer úgy, hogy csak az első 30 sort és első 3 oszlopot, egyszer úgy, hogy csak az első 30 sort és a maradék oszlopokat, egyszer úgy, hogy a maradék sorokat és az első 3 oszlopot és egyszer a maradék sorokat és a maradék oszlopokat.

Ezután fűzzétek össze úgy ezeket a df-eket, hogy megegyezzen az eredetivel



Series

```
companies = ['Google', 'Microsoft', 'Facebook', 'Apple']
```

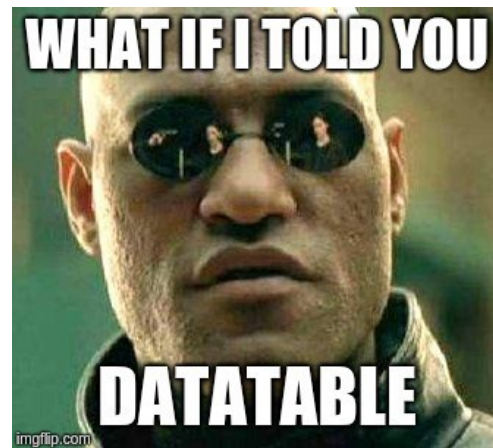
```
pd.Series(companies)
```

```
0      Google
1  Microsoft
2    Facebook
3       Apple
dtype: object
```

```
# Pass string
```

```
pd.Series(companies, index=['GOOGL', 'MSFT', 'FB', 'AAPL'])
```

```
GOOGL      Google
MSFT       Microsoft
FB         Facebook
AAPL        Apple
dtype: object
```



```
s = pd.Series(['a','b','b','a','a'])
```

```
s.unique()  
array(['a', 'b'], dtype=object)
```

```
s.nunique()  
2
```

```
s.agg(['mean', 'sum', 'product'])
```

```
mean      6.583333  
sum      39.500000  
product  33000.000000  
dtype: float64
```

```
s.value_counts()
```

```
a      3  
b      2  
dtype: int64
```

```
s.nlargest()
```

```
s.nlargest(2)
```

gt(): greater than

ge(): greater than or equal

eq(): equal

le(): less than or equal

lt(): less than

ne(): not equal

```
# ascending by default  
s.sort_values()
```

```
# To sort it in descending order  
s.sort_values(ascending=False)
```

```
# To modify the original series  
s.sort_values(inplace=True)
```



Feladat

- 1. Keverjétek össze a df-et úgy, hogy külön-külön rendezitek az oszlopokat, majd külön-külön rendezitek a sorokat

