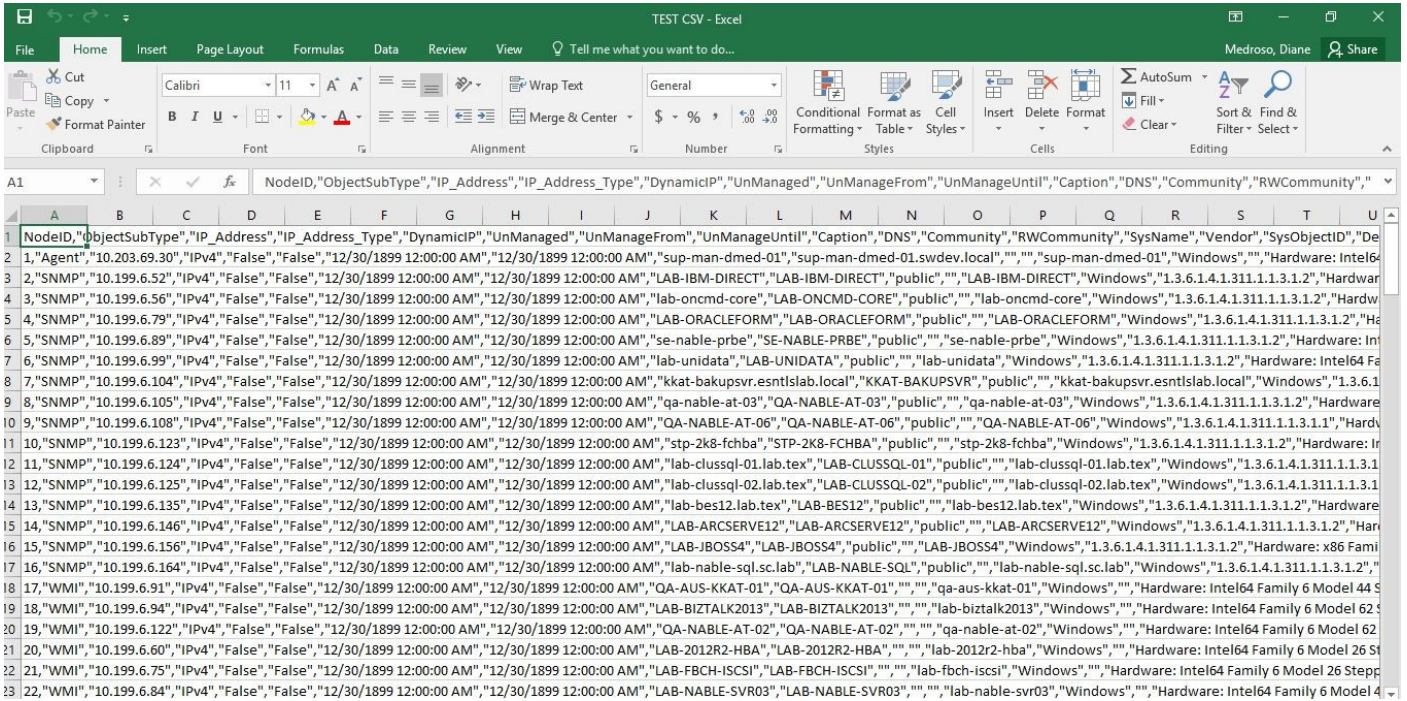
Pandas

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)



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)
```

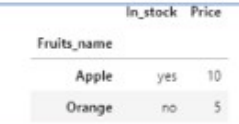
DataFrame adatszerkezetből

```
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

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

df[1:]		[1:] → Slicing row index position [Returns row with index 1 till the last row]
df[:1]		Returns row 0 only. End index is exclusive
df[::2]		[::2]→ Returns row 0 till last row by step 2 [alternate rows]
df["Banana":]		Slicing by row index values
df["Apple":"Banana"]		Returns row “Apple” till “Banana”. End index is inclusive.

Hagyományos indexing (oszlopok)

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

df

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

Hagyományos indexing (sorok és oszlopok)

df[1:]["Price"]	1 7 2 5 Name: Price, dtype: int64
df[:]["Price"]	0 10 1 7 2 5 Name: Price, dtype: int64
df[:][["Price"]]	Price 0 10 1 7 2 5
df[1:][["Price","Fruits_name"]]	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 = 'right'

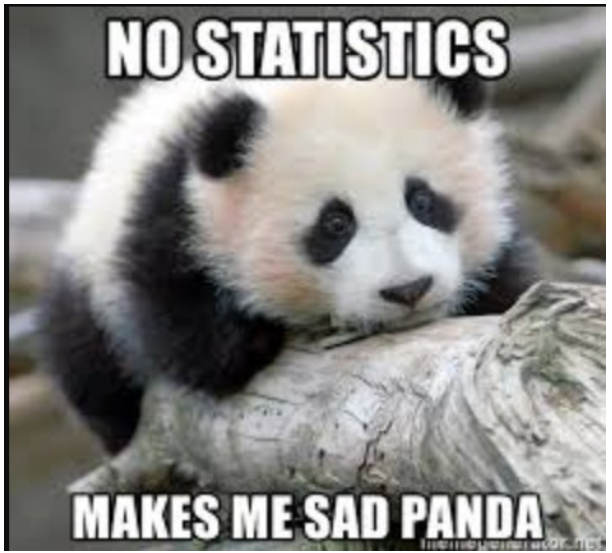
column_a	column_b_x	column_c_x	column_b_y	column_c_y
	NaN	NaN		
	NaN	NaN		

how = 'outer'

column_a	column_b_x	column_c_x	column_b_y	column_c_y
			NaN	NaN
			NaN	NaN
	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()
```

continent	country	population	area	population percentage of world
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 szirom 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

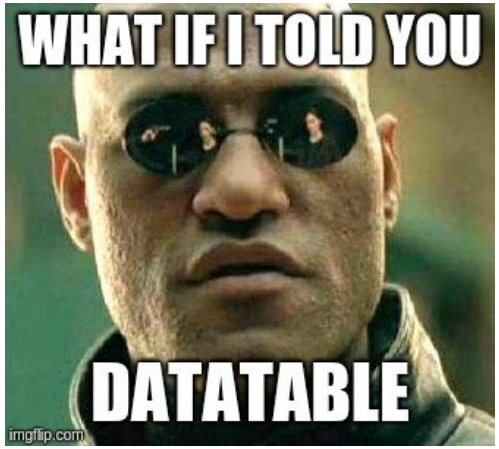
```
companies = ['Google', 'Microsoft', 'Facebook', 'Apple']

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 descenting 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