תוכן

שבוע 2

המשך Pandas Data Frame

הגדרת פונקציה

רושמים def רווח שם פונקציה פתח סוגריים עגולים וסגור סוגריים עגולים.

בתוך הסוגריים אנחנו יכולים להגדיר מה הפרמטר/משתנה/ארגומנט שאנחנו מצפים לקבל בהפעלת הפונקציה. אחרי סגירת הסוגריים צריך לרשום נקודותיים.

יורדים שורה. ורושמים מה הפונקציה מחזירה אבל חייבים שהשורה השנייה תתחיל אחרי tab ואח"כ לרשום return ומה הערך המוחזר.

In [1]:
```
import pandas as pd
```

In [3]:
```
def my_func(x):
    return x

my_func(1)
```

Out[3]:
```
1
```

למה צריך לשים לב בכתיבת פונקציה:

1) מתחילה במילה def ואחריה שם פונקציה.
2) אחרי שם הפונקציה לרשום סוגריים עגולים () ואחריהם נקודותיים.
3) השורה השנייה חייבת להתחיל בtab ולא להתחיל באותו הקו עם שורה 1.

In [5]:
```
def my_func(my_argument):
    return my_argument+3

my_func(1)
```

Out[5]:
```
4
```

ניתן להגדיר בפונקציה ערך (ארגומנט) דיפולטי (שיהיה כברירת מחדל) למקרה שיפעילו את הפונקציה ללא ערך (ארגומנט), כלומר ישאירו את הסוגריים ריקים בהפעלת הפונקציה.

דוגמא לפונקציה עם ערך דיפולטי ופונקציה ללא ערך דיפולטי:

כשנפעיל פונקציה ללא ערך דיפולטי ולא ניתן לה ערך בסוגריים אז נקבל הודעת שגיאה.

```
def my_func_required(i_am_a_required_argument):
    return i_am_a_required_argument+3

def my_func_not_required(i_am_not_a_required_argument=1):
    return i_am_not_a_required_argument+3

print('Not Required')
print(my_func_not_required(3))
print(my_func_not_required())

print('Required')
print(my_func_required(3))
print(my_func_required())
```

```
Not Required
6
4
Required
6

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-df6f59d2ce88> in <module>
print('Required') 11
```

פונקציית apply

זו פונקציה שמפעילה פונקציה מסוימת על כל ערך ברשימה series.

```
df['Quantity'].head(3)
```

```
1      0
200    1
500    2
Name: Quantity, dtype: int64
```

```
df['Quantity'].apply(my_func)
```

```
4          0
203        1
503        2
203        3
```

פונקציית Lambda

פונקציית למדא זו פונקציה אנונימית קצרה שמוגדרת ללא שם.

התחביר שלה:

רושמים את המילה השמורה lambda

אחריה יכול להיות ארגומנט אחד או יותר (הפרמטר/הערך שהפונקציה מקבלת).

אחרי רשימת הארגומנטים רושמים נקודותיים.

ואז מגיע הביטוי עצמו – כלומר מה הערך המוחזר שרוצים שיהיה.

```
df['Quantity'].head(3)
```

```
1      0
200    1
500    2
Name: Quantity, dtype: int64
```

```
df['Quantity'].apply(my_func).head(3)
```

```
4      0
203    1
503    2
Name: Quantity, dtype: int64
```

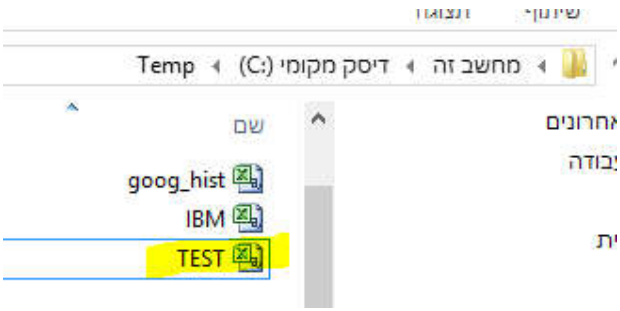
```
df['Quantity'].apply(lambda x: x+3).head(3)
```

```
4      0
203    1
503    2
Name: Quantity, dtype: int64
```

Group by

יוצר אגרגציה / קיבוץ של עמודה או עמודות לפי פונקציה שנרצה: למשל sum, min, max, count, mean ועוד.

לצורך ההמחשה של פונקציית groupby יצרתי קובץ csv בשם TEST שמכיל את המידע הבא:



C	B	A	
	letter	numbers	1
	b	2	2
	c	3	3
	a	1	4
	a	1	5
	b	2	6
	b	2	7
	a	1	8
	c	3	9
	b	2	10
			11
			12

In [19]:

```
df = pd.read_csv('c:\Temp\TEST.csv')
df
```

Out[19]:

	letter	numbers
0	b	2
1	c	3
2	a	1
3	a	1
4	b	2
5	b	2
6	a	1
7	c	3
8	b	2

In [20]:

```
df.groupby('letter')['numbers'].sum()
```

Out[20]:

```
letter
a    3
b    8
c    6
Name: numbers, dtype: int64
```

מה רשמנו לו כאן: שיבצע קיבוץ של העמודה letter ויבצע סכימה של העמודה numbers בהתאם. כך יש לנו שורה אחת לכל letter וסכום הnumbers שלה.

אפשר גם לבצע זאת עם פונקציות אחרות:

```
df.groupby('letter')['numbers'].sum()
```

סכום

```
letter
a    3
b    8
c    6
Name: numbers, dtype: int64
```

```
df.groupby('letter')['numbers'].mean()
```

ממוצע

```
letter
a    1
b    2
c    3
Name: numbers, dtype: int64
```

```
df.groupby('letter')['numbers'].count()
```

ספירה

```
letter
a    3
b    4
c    2
Name: numbers, dtype: int64
```

Groupby+apply

```
df.groupby('letter')['numbers'].apply(lambda x: x.sum())
```

```
letter
a    3
b    8
c    6
Name: numbers, dtype: int64
```

מה שרשמנו זהה לצורה שעשינו קודם groupby ללא apply.

```
print(df.groupby('letter')['numbers'].sum())
print(df.groupby('letter')['numbers'].apply(lambda x: x.sum()))
```

```
letter
a    3
b    8
c    6
Name: numbers, dtype: int64
letter
a    3
b    8
c    6
Name: numbers, dtype: int64
```

למה בכל זאת משתמשים בapply עם lambda- כי הצורה הזו יותר גמישה ומאפשרת להוסיף לgroupby גם תנאים ועוד.

איך מתבצעות הפקודות מאחורי הקלעים:

בפקודה הזו:

print(df.groupby('letter')['numbers'].sum())

מתבצע sum ישירות על הseries / העמודה numbers.

בפקודה הזו:

print(df.groupby('letter')['numbers'].apply(lambda x: x.sum()))

העמודה numbers מועברת כארגומנט לפונקציה האנונימית lambda, מתבצע sum ואז מחזירה את הערך בחזרה.

איזה עוד דברים ניתן לעשות בgroupby עם apply:

הוספת תנאים: לדוגמא שיבצע groupby ו sum אבל רק אם ערך בעמודה numbers לא שווה 1.

```
print(df.groupby('letter')['numbers'].apply(lambda x: x.sum()))
print(df.groupby('letter')['numbers'].apply(lambda x: x[x!=1].sum()))
```

```
letter
a    3
b    8
c    6
Name: numbers, dtype: int64
letter
a    0
b    8
c    6
Name: numbers, dtype: int64
```

דרך נוספת להפעיל את groupby עם apply.

הפעלה לא על עמודה מסוימת אלא על כל ה data frame.

לפונקציה lambda במקום להכניס עמודה מסוימת series, מכניסים את כל הdata frame.

letter	numbers	
b	2	**0**
c	3	**1**
a	1	**2**
a	1	**3**
b	2	**4**
b	2	**5**
a	1	**6**
c	3	**7**
b	2	**8**

לדוגמא: קיבוץ לפי עמודה letter והכפלת כל ערך בעמודה letter בערך בעמודהnumbers .

מכיוון שמדובר בהכפלת string במספר אז התוצאה שרשור הstring כמספר הפעמים שמופיע בערך של העמודה numbers בהתאמה.

```python
print(df.groupby('letter').apply(lambda x: print(x['letter']*x['numbers'])))
```

```
a    2
a    3
a    6
dtype: object
bb    0
bb    4
bb    5
bb    8
dtype: object
ccc    1
ccc    7
dtype: object
Empty DataFrame
[] :Columns
[] :Index
```

תרגילים עם פתרון

משתנה shape

נמצא בתוך ה dataframe וניתן להציג אותו.

הוא מתאר את מבנה הdataframe.

```
In [2]: df
```

Out[2]:

	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075
3	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377
4	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571
...	...	...	...	...	...	...	...
36191	20002020	ARCA	141	151.55	IBM	TRADE	18:27:09.391
36192	00002000	ARCA	100	152.00	IBM	TRADE	18:53:28.906
36193	20002020	ARCA	100	152.00	IBM	TRADE	19:28:11.889
36194	a0002020	ARCA	50	152.00	IBM	TRADE	19:31:30.532
36195	80002000	ARCA	50	152.00	IBM	TRADE	19:58:30.644

rows × 7 columns 36196

```
In [4]: df.shape
```

Out[4]:

```
(7 ,36196)
```

Columns- משתנה שמכיל את שמות העמודות. נמצא בתוך ה data frame

```
In [5]: df.columns
```

Out[5]:

```
,'Index(['Timestamp', 'EventType', 'Ticker', 'Price', 'Quantity', 'Exchange
,['Conditions'
('dtype='object
```

Index- משתנה שנמצא בתוך ה dataframe ומכיל את טווח האינדקסים שבטבלת ה dataframe.

```
In [6]: df.index
```

Out[6]:

```
RangeIndex(start=0, stop=36196, step=1)
```

הפונקציות Tail & Head

Head פונקציה שמציגה את מספר השורות הראשונות מתוך ה dataframe

Tail פונקציה שמציגה את מספר השורות האחרונות מתוך ה dataframe.

In[7]:
```
df.head(3)
```

Out[7]:

	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075

In[8]:
```
df.tail(3)
```

Out[8]:

	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
36193	20002020	ARCA	100	152.0	IBM	TRADE	19:28:11.889
36194	a0002020	ARCA	50	152.0	IBM	TRADE	19:31:30.532
36195	80002000	ARCA	50	152.0	IBM	TRADE	19:58:30.644

Slicing- בחירה של data מתוך מבנה טבלאי (שורות או עמודות)

Slicing של שורות

integer location =Iloc

בדוגמא כאן: הצגת השורות מstart index=0 עד end index= 6 תמיד השורה של הend index לא נכללת . כלומר משורה 0 עד 6 אך לא כולל את שורה 6.

In[9]:
```
df.iloc[0:6]
```

Out[9]:

	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075
3	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377
4	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571
5	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.732

אם לא מציינים את ה start index אז מציג נתונים החל מהשורה הראשונה הזמינה. במקרה שלנו שורה 0.

In[10]:
```
df.iloc[:6]
```

Out[10]:

	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075
3	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377
4	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571
5	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.732

אם לא מציינים את הend index אז מציג נתונים עד סוף הטבלה כולל השורה האחרונה.

אם היינו מציינים ב end index אז מספר השורה האחרונה שהוא 36195 אז היינו מקבלים נתונים עד שורה 36194 כי מה שמצוין בend index הוא תמיד לא כולל את אותו ערך.

In [16]:
```
df.iloc[36192:36195]
```

Out[16]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
00002000	ARCA	100	152.0	IBM	TRADE	18:53:28.906	**36192**
20002020	ARCA	100	152.0	IBM	TRADE	19:28:11.889	**36193**
a0002020	ARCA	50	152.0	IBM	TRADE	19:31:30.532	**36194**

In [15]:
```
df.iloc[36192:]
```

Out[15]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
00002000	ARCA	100	152.0	IBM	TRADE	18:53:28.906	**36192**
20002020	ARCA	100	152.0	IBM	TRADE	19:28:11.889	**36193**
a0002020	ARCA	50	152.0	IBM	TRADE	19:31:30.532	**36194**
80002000	ARCA	50	152.0	IBM	TRADE	19:58:30.644	**36195**

Index שלילי= אינדקסים שמתייחסים לאינדקסים האחרונים בטבלה.

כשאנחנו רושמים start index= -4 כלומר הערך בשורה הרביעית מהסוף.

End index= -1 הכוונה הערך בשורה האחרונה. אך תמיד מציג את הend index כעד אך לא כולל. לכן יציג לנו בדוגמא זו עד שורה 36194 ולא 36195.

In [17]:
```
df.iloc[-4:-1]
```

Out[17]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
00002000	ARCA	100	152.0	IBM	TRADE	18:53:28.906	**36192**
20002020	ARCA	100	152.0	IBM	TRADE	19:28:11.889	**36193**
a0002020	ARCA	50	152.0	IBM	TRADE	19:31:30.532	**36194**

כך כן נקבל את השורה האחרונה:

In [18]:
```
df.iloc[-4:]
```

Out[18]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
00002000	ARCA	100	152.0	IBM	TRADE	18:53:28.906	**36192**
20002020	ARCA	100	152.0	IBM	TRADE	19:28:11.889	**36193**
a0002020	ARCA	50	152.0	IBM	TRADE	19:31:30.532	**36194**
80002000	ARCA	50	152.0	IBM	TRADE	19:58:30.644	**36195**

אם רוצים לראות שורות מסוימות שלא נמצאות ברצף אחת אחרי השנייה:

אז רושמים במקום טווח של אינדקסים, מערך של אינדקסים:

In[19]:
```
df.iloc[[0,50,30,36195]]
```

Out[19]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	**0**
00002000	NASDAQ	100	153.96	IBM	TRADE	08:43:14.162	**50**
20002020	EDGX	100	154.12	IBM	TRADE	08:42:55.016	**30**
80002000	ARCA	50	152.00	IBM	TRADE	19:58:30.644	**36195**

אם רוצים לראות טווח שורות אבל לשלוט על הצעדים/קפיצות (steps) של הindex ים. כלומר שלא יציג לנו את אינדקסים 0,1,2,3,4 – קפיצות של 1

אלא יציג את האינדקסים 0,2,4,6...- קפיצות למשל של 2.

In[20]:
```
df.iloc[0:10:2]
```

Out[20]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	**0**
20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	**2**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	**4**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.817	**6**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:43.188	**8**

משורה -6 (6 מהסוף) עד הסוף בקפיצות של 2

In[21]:
```
df.iloc[-6::2]
```

Out[21]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
80002000	FINRA	5	151.55	IBM	TRADE	18:20:43.988	**36190**
00002000	ARCA	100	152.00	IBM	TRADE	18:53:28.906	**36192**
a0002020	ARCA	50	152.00	IBM	TRADE	19:31:30.532	**36194**

Slicing של עמודות

מכיל start index עד end index של שורות

מספר הקפיצות בשורות

פסיק מספר האינדקס של העמודה

index העמודה

```
df.iloc[-6::2,2]
```

```
IBM     36190
IBM     36192
IBM     36194
Name: Ticker, dtype: object
```

אם רוצים לבחור כמה עמודות אז נרשום את מספרי האינדקס של העמודות בתוך מערך:

In [23]:

```
df.iloc[-6::2,[5,2,3,6]]
```

Out[23]:

Conditions	Price	Ticker	Exchange	
80002000	151.55	IBM	FINRA	**36190**
00002000	152.00	IBM	ARCA	**36192**
a0002020	152.00	IBM	ARCA	**36194**

פקודה להצגת צבר של עמודות וצבר של שורות שלא נמצאים בסדר רציף:

נרשום מערך של מספרי שורות ומערך של מספרי עמודות

In [24]:

```
df.iloc[[2,5,23],[5,2,3,6]]
```

Out[24]:

Conditions	Price	Ticker	Exchange	
20002020	154.50	IBM	ARCA	**2**
00002000	154.50	IBM	ARCA	**5**
00002000	154.36	IBM	EDGX	**23**

בחירת טווח רציף של עמודות ושורות

גם כאן כאשר רושמים 2 עד 7 אז לא תיכלל עמודה 7

טווח עמודות

In [25]:

```
df.iloc[-7::2,2:7]
```

Out[25]:

Conditions	Exchange	Quantity	Price	Ticker	
80002000	FINRA	15	151.50	IBM	**36189**
20002020	ARCA	141	151.55	IBM	**36191**
20002020	ARCA	100	152.00	IBM	**36193**
80002000	ARCA	50	152.00	IBM	**36195**

מעמודה 2 עד 7 לא כולל 7 בקפיצות של 2

In[26]:
```
df.iloc[-7::2,2:7:2]
```

Out[26]:

Conditions	Quantity	Ticker	
80002000	15	IBM	**36189**
20002020	141	IBM	**36191**
20002020	100	IBM	**36193**
80002000	50	IBM	**36195**

מעמודה ה5 מהסוף עד 2 מהסוף לא כולל את השנייה מהסוף בקפיצות של 2

In[27]:
```
df.iloc[-7::2,-5:-2:2]
```

Out[27]:

Quantity	Ticker	
15	IBM	**36189**
141	IBM	**36191**
100	IBM	**36193**
50	IBM	**36195**

דרך נוספת שניתן לבחור מספר שורות מעמודה ספציפית:

In[28]:
```
df['Timestamp'].iloc[-7::2]
```

Out[28]:
```
18:20:34.674    36189
18:27:09.391    36191
19:28:11.889    36193
19:58:30.644    36195
Name: Timestamp, dtype: object
```

In[29]:
```
df.iloc[-7::2]['Timestamp']
```

Out[29]:
```
18:20:34.674    36189
18:27:09.391    36191
19:28:11.889    36193
19:58:30.644    36195
Name: Timestamp, dtype: object
```

Filters

חיתוך הdata באמצעות תוכן המידע הטבלאי.

מגדירים תנאי לוגי (==, <,>, <= וכ') ואת התוצאה מכניסים למשתנה שמשמש אותנו כfilter.

ה-filter הזה מקבל ערכים בוליאנים של true או false אם התנאי שהגדרנו מתקיים או לא מתקיים בכל שורה.

לדוגמא:

אם רוצים לבדוק באיזו שורה יש לנו את הכמות 200.

נרשום:

```
filtr=(df['Quantity']==200)
```

```
filtr
```

```
False        0
True         1
False        2
True         3
True         4
  ...
False    36191
False    36192
False    36193
False    36194
```

איך מסננים את הdata frame שלנו בהתאם לfilter שיצרנו

מציג לנו רק את הערכים שיצאו true בפילטר:

```
df[filtr]
```

:[32] In

Out[32]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	**1**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377	**3**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	**4**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.732	**5**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.817	**6**
...	...	...	...	...	...	...	**...**
20102000	FINRA	200	152.93	IBM	TRADE	16:11:58.392	**36145**
20106000	FINRA	200	152.93	IBM	TRADE	16:12:05.095	**36151**
20106000	FINRA	200	152.93	IBM	TRADE	16:12:29.602	**36156**

אם רוצים שיציג לנו רק את הערכים שיצאו false בפילטר:

נרשום את אותה הפקודה רק נוסיף את הסימן טילדה ~

```
df[~filtr]
```

:[33] In

Out[33]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	**0**
20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	**2**
00002000	ARCA	100	154.50	IBM	TRADE	07:49:46.676	**15**
a0002020	ARCA	1	154.53	IBM	TRADE	07:49:46.698	**16**
a0002020	ARCA	99	154.50	IBM	TRADE	07:49:46.698	**17**
...	...	...	...	...	...	...	**...**
20002020	ARCA	141	151.55	IBM	TRADE	18:27:09.391	**36191**
00002000	ARCA	100	152.00	IBM	TRADE	18:53:28.906	**36192**
20002020	ARCA	100	152.00	IBM	TRADE	19:28:11.889	**36193**

אם רוצים להציג רק עמודה אחת מהdataframe בהתאם לפילטר

```
df[filtr]['Quantity']
```

```
200        1
200        3
200        4
200        5
200        6
 ...
200    36145
200    36151
200    36156
200    36164
```

אם רוצים לראות כמה שורות עומדות בתנאי הפילטר

```
df[filtr]['Quantity'].count()
```

```
1516
```

אם רוצים לבחור להציג כמה עמודות אז נצטרך להכניס את שמות העמודות למערך:

```
df[filtr][['Quantity','Price']]
```

In [36]:

Out[36]:

	Price	Quantity
1	154.49	200
3	154.50	200
4	154.50	200
5	154.50	200

אם רוצים להרחיב את הפילטר עם עוד תנאי:

למשל אם רוצים ליצור פילטר שיציג את כל המקרים שיש לנו עסקאות של 200 מניות **או** (|) עסקאות של 300 מניות נרשום זאת כך:

```
filtr=((df['Quantity']==200) | (df['Quantity']==300))
filtr
```

```
False        0
True         1
False        2
True         3
True         4
  ...
False    36191
False    36192
False    36193
False    36194
False    36195
Name: Quantity, Length: 36196, dtype: bool
```

אם אנחנו רוצים לצמצם את הפילטר והפעם לחפש מקרים בהם היו לנו עסקאות של 200 מניות **וגם** (&) במחיר של 153 ומשהו.

כדי שנוכל לסנן את המידע לפי מחיר 153 בלי להתייחס לנקודה העשרונית אז נמיר את המחיר מfloat ל int.

```
filtr=((df['Quantity']==200) & (df['Price'].astype(int)==153))
filtr
```

```
False        0
False        1
False        2
False        3
False        4
  ...
False    36191
False    36192
False    36193
False    36194
False    36195
Length: 36196, dtype: bool
```

```
df[filtr]
```

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
00000001	NYSE	200	153.92	IBM	TRADE NB	09:30:03.926	**248**
00000001	NYSE	200	153.91	IBM	TRADE NB	09:30:05.834	**302**

יצירת filter של שמסתמך על מערך:

יצרנו משתנה שמכיל מערך כמויות עסקאות שעל פיהן מסננים את המידע.

משתמשים בפונקציה isin ששואלת האם נמצא (האם ערך נמצא באיזו רשימה).

```
quantity_vals=[100,200,350,500]
filtr=df['Quantity'].isin(quantity_vals)
filtr
```

```
False        0
True         1
True         2
True         3
True         4
  ...
False    36191
True     36192
True     36193
False    36194
False    36195
Name: Quantity, Length: 36196, dtype: bool
```

```
df[filtr]
```

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	**1**
20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	**2**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377	**3**
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	**4**

אם למשל היינו רוצים לבצע פילטר על מלל string ולא על int, כמו שעשינו עד עכשיו כשרצינו תוצאות עם מחיר או כמות מסוימים.

אם אנחנו רוצים לבצע פילטר כך שיציג לנו רק שורות שהעמודה Exchange שלהם מכילה (contains) את המילה ARCA (באותיות גדולות כפי שרשמנו לכן מוסיפים כפרמטרים לפונקציה case=True הכוונה שיהיה רגיש לאותיות גדולות או קטנות כי אנחנו לא רוצים שהוא ייקח בפילטר גם arca שרשום באותיות קטנות ולא יחפש זאת כביטוי רגולרי בתבניות לכן רשמנו regex=False):

```
arca_filtr=(df['Exchange'].str.contains('ARCA',regex=False,case=True))
arca_filtr
```

:[47] In

Out[47]:

```
True        0
True        1
True        2
True        3
True        4
 ...
True    36191
True    36192
True    36193
True    36194
True    36195
Name: Exchange, Length: 36196, dtype: bool
```

```
df[arca_filtr]
```

:[48] In

Out[48]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	**0**
20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	**1**
20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	**2**

```
arca_filtr=(df['Exchange'].str.contains('ARCA',regex=False,case=Tr
arca_filtr
```

:[47] In

Out[47]:

```
True        0
True        1
True        2
True        3
True        4
 ...
True    36191
True    36192
True    36193
True    36194
True    36195
Name: Exchange, Length: 36196, dtype: bool
```

```
df[arca_filtr]
```

:[48] In

Out[48]:

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	**0**
20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	**1**
20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	**2**

אם רוצים פילטר משולב כך שיציג לנו נתונים מ2 בורסות: ARCA או FINRA.

וגם שורות עם כמות 200 אז נרשום:

```
arca_filtr=(df['Exchange'].str.contains('ARCA',regex=False,case=True))
finra_filtr=(df['Exchange'].str.contains('FINRA',regex=False,case=True))
filtr= ((df['Quantity']==200)&(arca_filtr|finra_filtr))
df[filtr]
```

Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	1
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377	3
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	4
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.732	5
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.817	6

Find Longest Idle Period

מציאת הזמן המת הגדול ביותר שעבר בין 2 תנועות מכירה וקניה.

נבצע המרה של הערכים בעמודה Timestamp מטקסט לסוג datetime ונשים אותם בעמודה חדשה.

אח"כ ניקח ערך של datetime פחות datetime מינימלי.

אח"כ נמיר את התוצאה לשניות (פונקציה total_seconds)

ונמיר את התוצאה ל int- (עם הפונקציה astype(int) כדי לקבל מספרים שלמים במקום עשרוניים.

עד כאן קיבלנו את הזמן בשניות מתחילת המסחר.

כדי לדעת מה ההבדל בזמנים בין תנועה לתנועה שקודמת לה.

נחסיר את הזמן בשניות של שורה מסוימת פחות הזמן בשניות של השורה שקודמת לה.

```
import pandas as pd
df = pd.read_csv('c:\Temp\IBM.csv')
```

```
df['datetime']=pd.to_datetime(df['Timestamp'])
df
```

datetime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
2020-06-01 07:12:38.577	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	0
2020-06-01 07:48:52.137	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	1
2020-06-01 07:49:41.075	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	2

```
df['diff_time']=df['datetime']-df['datetime'].min()
df
```

	diff_time	datetime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	00:00:00	07:12:38.577 2020-06-01	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	00:36:13.560000	07:48:52.137 2020-06-01	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	00:37:02.498000	07:49:41.075 2020-06-01	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075
3	00:37:03.800000	07:49:42.377 2020-06-01	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377
4	00:37:03.994000	07:49:42.571 2020-06-01	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571

```
df['secs']=df['diff_time'].dt.total_seconds().astype(int)
df
```

	secs	diff_time	datetime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	0	00:00:00	2020-06-01 07:12:38.577	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	2173	00:36:13.560000	2020-06-01 07:48:52.137	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	2222	00:37:02.498000	2020-06-01 07:49:41.075	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075

```
df['secs_diff']=df['secs']-df['secs'].shift(+1)
df
```

	secs_diff	secs	diff_time	datetime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	NaN	0	00:00:00	2020-06-01 07:12:38.577	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	2173.0	2173	00:36:13.560000	2020-06-01 07:48:52.137	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	49.0	2222	00:37:02.498000	2020-06-01 07:49:41.075	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075
3	1.0	2223	00:37:03.800000	2020-06-01 07:49:42.377	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377

```
df['secs'].shift(+1)
```

```
NaN            0
0.0            1
2173.0         2
2222.0         3
2223.0         4
   ...
40085.0    36191
40470.0    36192
42050.0    36193
```

נמיר את הערך הריק NaN (Nan=not a number) ל 0

integer location =Iloc

location =Loc (ניתן לעבוד גם מול שמות עמודות ולא רק מול אינדקסים, לא יכול לעבוד עם אינדקסים שליליים)

```
df.loc[0,'secs_diff']=0
```

```
df
```

secs_diff	secs	diff_time	datetime	Conditions	Exchange	Quantity	Price
0.0	0	00:00:00	07:12:38.577 2020-06-01	80002000	ARCA	1	155.61
2173.0	2173	00:36:13.560000	07:48:52.137 2020-06-01	20002020	ARCA	200	154.49
49.0	2222	00:37:02.498000	07:49:41.075 2020-06-01	20002020	ARCA	500	154.50
1.0	2223	00:37:03.800000	07:49:42.377 2020-06-01	00002000	ARCA	200	154.50

עכשיו נחשב את המקסימום מבין כל המרווחים שכבר חישבנו

```
max_idle=df['secs_diff'].max()
```

```
max_idle
```

```
2327.0
```

אנחנו צריכים את השורה/שורות שהערך שנמצא בה הוא המקסימלי.

לצורך זה נגדיר פילטר שיציג לנו רק את השורות שהsecs_diff שלה הוא עם הפרק זמן הכי ארוך שהוא max_idle, 2327.

```
filtr=(df['secs_diff']==max_idle)
filtr
```

```
False        0
False        1
False        2
False        3
False        4
  ...
False    36191
False    36192
```

```
df[filtr]
```

secs_diff	secs	diff_time	datetime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
2327.0	39510	10:58:30.857000	18:11:09.434 2020-06-01	00102000	FINRA	2282	151.55	IBM	TRADE	18:11:09.434	**36188**

Validate Ticker Data

בדיקה אם הערכים של עמודת הticker יהיו של idm ולא של משהו אחר

בדיקה שבעמודה ticker יש רק ערכים שכתובה בהן המילה 'IBM'.

איך בודקים זאת : יוצרים פילטר עם הפונקציה contains כדי לחפש רק ערכים של IBM באותיות גדולות.

ואז נחפש את ההופכי של הפילטר באמצעות טילדה ~ כדי לראות אם יש לנו שורות שהערך שלהן בעמודה ticker הוא לא IBM.

```
import pandas as pd
df = pd.read_csv('c:\Temp\IBM.csv')
```

```
df
```

Conditions	Exchange	Quantity	Price	Ticker	EventType
80002000	ARCA	1	155.61	IBM	TRADE
20002020	ARCA	200	154.49	IBM	TRADE
20002020	ARCA	500	154.50	IBM	TRADE
00002000	ARCA	200	154.50	IBM	TRADE
00002000	ARCA	200	154.50	IBM	TRADE
...	...	...	...	...	...
20002020	ARCA	141	151.55	IBM	TRADE

```
ibm_filtr=(df['Ticker'].str.contains('IBM',regex=False, case=True))
ibm_filtr
```

```
True        0
True        1
True        2
True        3
True        4
 ...
True    36191
```

```
df[ibm_filtr]
```

Conditions	Exchange	Quantity	Price	Ticker	EventType
80002000	ARCA	1	155.61	IBM	TRADE
20002020	ARCA	200	154.49	IBM	TRADE
20002020	ARCA	500	154.50	IBM	TRADE
00002000	ARCA	200	154.50	IBM	TRADE
00002000	ARCA	200	154.50	IBM	TRADE

נבצע ספירה של כמה ערכים בעמודה ticker יש לנו שלא מקיימים את הפילטר

```
cntr=df[~ibm_filtr]['Ticker'].count()
cntr
```

```
0
```

ראינו שאין לנו ערכים בעמודה ticker שהם לא IBM.

Find Busy Hours

חיפוש שעת המסחר הכי עמוסה. שעת המסחר שהיו בה הכי הרבה תנועות של קניה ומכירה.

איך עושים זאת:
ממירים את עמודת הtimestamp לשעות.

בודקים כמה פעולות מסחר יש בכל שעה ע"י הפונקציה groupby ו count.

השעה עם הcount המקסימלי היא השעה העמוסה ביותר.

איך כותבים זאת:

1) הוספת עמודה datetime וhour

```
import pandas as pd
df=pd.read_csv('c:\Temp\IBM.csv')
```

```
df.columns
```

```
,'Index(['Timestamp', 'EventType', 'Ticker', 'Price', 'Quantity', 'Exchange
,['Conditions'
('dtype='object
```

```
df['datetime']=pd.to_datetime(df['Timestamp'])
```

```
df['Hour']=df['datetime'].dt.hour
df
```

Hour	datetime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
7	07:12:38.577 2020-06-01	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	0
7	07:48:52.137 2020-06-01	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	1
7	07:49:41.075 2020-06-01	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	2
7	07:49:42.377 2020-06-01	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377	3
7	07:49:42.571 2020-06-01	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	4

2) יצירת משתנה בשם max_trades שבו נשמור את הכמות המקסימלית מתוך כלל שעות המסחר. מקבצים את המידע לפי שעה וכמות לכל שעה ומשם לוקחים את הערך המקסימלי.

```
max_trades=df.groupby('Hour')['Hour'].count().max()
max_trades
```

```
11558
```

איך המידע נראה לפני פונקציית המax

```
max_trades=df.groupby('Hour')['Hour'].count()
max_trades
```

```
Hour
20        7
47        8
3651      9
4647     10
4897     11
2944     12
3192     13
5157     14
11558    15
71       16
4        17
5        18
3        19
Name: Hour, dtype: int64
```

3) יצירת טבלה זמנית שתכיל את מידע זה
שלב ראשון יצירת טבלה ריקה עם עמודה בשם Trades

```
tmpData=pd.DataFrame(columns=['Trades'])
```

```
tmpData
```

Trades

שלב שני הכנסת הcount לפי שעה שעשינו קודם לטבלה זו

In [10]:
```
tmpData['Trades']=df.groupby('Hour')['Hour'].count()
```

In [11]:
```
tmpData
```

Out[11]:

Trades	Hour
20	7
47	8
3651	9
4647	10
4897	11
2944	12
3192	13
5157	14
11558	15
71	16
4	17
5	18

השעות הפכו להיות הindex של הdata frame הזה.

אם רוצים שיהיה אינדקס רגיל שמתחיל ב0 וקופץ כל פעם בstep ים של 1 אז רושמים:

In [14]:
```
print(tmpData.index)
```

```
Int64Index([7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], dtype='int64', na
me='Hour')
```

In [15]:
```
tmpData=tmpData.reset_index()
tmpData
```

Out[15]:

Trades	Hour	
20	7	0
47	8	1
3651	9	2
4647	10	3
4897	11	4
2944	12	5
3192	13	6

```
print(tmpData.index)
```

```
RangeIndex(start=0, stop=13, step=1)
```

4) יצירת פילטר

מכניסים את השעה עם המספר המקסימלי של תנועות לתוך משתנה

```
max_trades=tmpData['Trades'].max()
max_trades
```

```
11558
```

יוצרים פילטר

```
fltr=(tmpData['Trades']==max_trades)
fltr
```

```
False     0
False     1
False     2
False     3
False     4
False     5
False     6
False     7
True      8
False     9
```

אנחנו לא יכולים לסנן לפיו כמו שאנחנו רגילים את כל המידע מטבלת המקור שלנו:
df[fltr]

כי הפילטר שלנו לא עם אותם אינדקסים כמו בטבלה df... הוא אחרי שביצענו group by ואחכ יצרנו אינדקסים חדשים.

אז נפעל בדרך אחרת:

אנחנו יכולים להפעיל אותה על הטבלה הזמנים שיצרנו:

```
tmpData[fltr]
```

In[20]:

Out[20]:

	Trades	Hour
11558	15	8

איך שולפים מתוך זה רק את השעה המקסימלית:

לוקחים רק את value 0 כך לא יופיע לנו האינדקס אלא יופיע לנו רק הערך של hour.

```
tmpData[fltr]
```

:[20] In

Out[20]:

	Trades	Hour
11558	15	8

```
max_hour=tmpData[fltr]
```

:[22] In

```
max_hour['Hour']
```

:[34] In

```
15    8
Name: Hour, dtype: int64
```

Out[34]:

```
max_hour['Hour'].values[0]
```

:[35] In

```
15
```

Out[35]:

במקום להשתמש במשתנה max_hour.

יכולנו לרשום הכל בפקודה אחת:

```
max_hour=tmpData[fltr]
```

```
max_hour['Hour']
```

```
15    8
Name: Hour, dtype: int64
```

```
max_hour['Hour'].values[0]
```

```
15
```

```
tmpData[fltr]['Hour'].values[0]
```

```
15
```

Daily Open Close

יצירת 5 המדדים

5 המדדים:

- Open- המחיר הראשון של המניה בשנייה מסוימת – מחיר בשורה [0]
- Close- המחיר האחרון של המניה בשנייה מסוימת- מחיר בשורה [-1]
- High- המחיר הכי גבוה שהמניה הגיעה אליו במהלך השנייה- max()
- Low- המחיר הכי נמוך שהמניה הגיעה אליו במהלך השנייה- min()
- Volume- כמות המניות שנסחרו באותה שניה- sum() של עמודה Quantity

```
new_df['Open']=df.groupby('TotalSeconds_Int')['Price'].apply(lambda x: x.iloc[0])
new_df['Close']=df.groupby('TotalSeconds_Int')['Price'].apply(lambda x: x.iloc[-1])
new_df['Low']=df.groupby('TotalSeconds_Int')['Price'].apply(lambda x: x.min())
new_df['High']=df.groupby('TotalSeconds_Int')['Price'].apply(lambda x: x.max())
new_df['Vol']=df.groupby('TotalSeconds_Int')['Quantity'].apply(lambda x: x.sum())
```

```
new_df
```

Vol	High	Low	Close	Open	TotalSeconds_Int
1	155.61	155.61	155.61	155.61	0
200	154.49	154.49	154.49	154.49	2173
500	154.50	154.50	154.50	154.50	2222

פקודת הדפסה print

%d למקרים בהם רוצים להדפיס נתון שהוא מסוג int.

%f אם רוצים להדפיס נתון מסוג float עם כך שיכלול מספרים אחרי הנקודה.

% נקודה מספר ספרות אחרי הנקודהf לדוגמא: %.2f הכוונה למספר עם 2 ספרות אחרי הנקודה העשרונית.

%s אם רוצים להדפיס נתון מסוג string.

%x/%X- אותיות גדולות או קטנות.

```
day_high=df['Price'].max()
day_low=df['Price'].min()
day_open=df['Price'].iloc[0]
day_close=df['Price'].iloc[-1]
```

```
print("Daily open:%d,close:%d, low:%d, high:%d" % (day_open,day_close,day_low,day_high))
```

```
Daily open:155,close:152, low:151, high:155
```

אם רוצים להדפיס ערכים כך שיופיעו עם מספר עשרוני:

```
print("Daily open:%f,close:%f, low:%f, high:%f" % (day_open,day_close,day_low,day_high))
```

```
Daily open:155.610000,close:152.000000, low:151.230000, high:155.610000
```

אם רוצים לקבוע כמה ספרות לאחר הנקודה העשרונית להציג:

בין האחוזים והf נרשום נקודה 2

```
print("Daily open:%.2f,close:%.2f, low:%.2f, high:%.2f" % (day_open,day_close,day_low,day_high))
```

```
Daily open:155.61,close:152.00, low:151.23, high:155.61
```

Shares Per Exchange

בדיקת מספר המניות שנסחרו פר exchange

סך כל כמות המניות שנסחרו:

```
df['Quantity'].sum()
```

```
4493928
```

איך מציגים זאת פר exchange:

מפעילים groupby

```
df.groupby('Exchange')['Quantity'].sum()
```

```
Exchange
ARCA           427056
BATS           323005
BATS Y         115807
CSE                 2
EDGA           119391
EDGX           191920
FINRA         1048401
NASDAQ        1190604
NASDAQ BX       69430
NASDAQ PSX      21251
NYSE           987061
Name: Quantity, dtype: int64
```

אנליזת נרות יפניים

Groupby

מתחילים ביצירת 5 המדדים: Open, Close, High, Low, Vol

כמו שעשינו בעבר, קודם ממירים את העמודה Timestamp לסוג datetime עם הפונקציה to_datetime.

אח"כ מחסירים את תאריך תחילת המסחר עם הערך ב Timestamp כדי לקבל את הזמן שעבר מתנועת המסחר הראשונה.

ממירים את עמודה זו לשניות עם הפונקציה total_seconds -> מקבלים תוצאה ב float.

```
In [3]: import pandas as pd
        df=pd.read_csv('c:\Temp\IBM.csv')

In [8]: df['date_time']=pd.to_datetime(df['Timestamp'])
        df['date_time']=df['date_time']-df['date_time'].min()
        df['date_time']=df['date_time'].dt.total_seconds()

In [9]: df['date_time']

Out[9]: 0            0.000
        1         2173.560
        2         2222.498
        3         2223.800
        4         2223.994
        5         2224.155
        6         2224.240
        7         2224.429
        8         2224.611
```

משתנים אותה למספר שלם ע"י שימוש בפונקציה astype(int)

```
In [10]: df['date_time_int']=df['date_time'].astype(int)
         df
```

Out[10]:

	Timestamp	EventType	Ticker	Price	Quantity	Exchange	Conditions	date_time	date_time_int
0	07:12:38.577	TRADE	IBM	155.61	1	ARCA	80002000	0.000	0
1	07:48:52.137	TRADE	IBM	154.49	200	ARCA	20002020	2173.560	2173
2	07:49:41.075	TRADE	IBM	154.50	500	ARCA	20002020	2222.498	2222
3	07:49:42.377	TRADE	IBM	154.50	200	ARCA	00002000	2223.800	2223

איך ממיינים את ה data frame לפי עמודה מסוימת

ממיינים את הdata frame לפי עמודה date_time ומבצעים השמה לdf, ה dataframe שלנו.
כדי שיכיל את המידע הממוין.

```
In [12]: df=df.sort_values('date_time')
```

יצירת data frame חדש שיכיל את 5 המדדים

```
In [13]: new_df=pd.DataFrame()
```

```
In [17]: new_df['Open']=df.groupby('date_time_int')['Price'].apply(lambda x: x.iloc[0])
         new_df['Close']=df.groupby('date_time_int')['Price'].apply(lambda x: x.iloc[-1])
         new_df['Low']=df.groupby('date_time_int')['Price'].apply(lambda x: x.min())
         new_df['High']=df.groupby('date_time_int')['Price'].apply(lambda x: x.max())
         new_df['Vol']=df.groupby('date_time_int')['Quantity'].apply(lambda x: x.sum())
```

```
In [18]: new_df
Out[18]:
```

	Open	Close	Low	High	Vol
date_time_int					
0	155.61	155.61	155.61	155.61	1
2173	154.49	154.49	154.49	154.49	200
2222	154.50	154.50	154.50	154.50	500
2223	154.50	154.50	154.50	154.50	400

בדיקה של התוצאות

שימוש בפונקציה info תיתן לנו לדוגמא מידע על ה data frame שלנו.

כמות הרשומות, טווח האינדקסים, מידע על העמודות.

אנחנו יצרנו data frame חדש (בשם new_df) שבו קיבצנו את המידע לפי שנייה.

ניתן לראות שב data frame החדש יש פחות שורות וזו בדיקה אחת שמראה שהמידע בו אכן מקובץ.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 36196 entries, 0 to 36195
Data columns (total 9 columns):
Timestamp        36196 non-null object
EventType        36196 non-null object
Ticker           36196 non-null object
Price            36196 non-null float64
Quantity         36196 non-null int64
Exchange         36196 non-null object
```

```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8365 entries, 0 to 45952
Data columns (total 5 columns):
Open     8365 non-null float64
Close    8365 non-null float64
Low      8365 non-null float64
High     8365 non-null float64
Vol      8365 non-null int64
dtypes: float64(4), int64(1)
```

איך בודקים באיזה מהשניות יש לנו הרבה שורות

מפעילים count לעמודת השניות ואז נראה כמה שורות יש לנו בכל שנייה .

אנחנו רואים בפלט אבל הוא לא מראה לנו שורות עם הרבה "בשר".

```
In [21]: df.groupby('date_time_int').count()
```

Out[21]:

date_time_int	Timestamp	EventType	Ticker	Price	Quantity	Exchange	Conditions	date_time
0	1	1	1	1	1	1	1	1
2173	1	1	1	1	1	1	1	1
2222	1	1	1	1	1	1	1	1
2223	2	2	2	2	2	2	2	2
2224	6	6	6	6	6	6	6	6
2225	3	3	3	3	3	3	3	3
2226	1	1	1	1	1	1	1	1
2228	4	4	4	4	4	4	4	4
2326	1	1	1	1	1	1	1	1

איך מחפשים שניות שיש להן כמות שורות גדולה

עושים את אותה פקודה שעשינו מקודם עם count , אנחנו נקבל בכל העמודות פר שורה את אותו מספר (כמות שורות לשנייה הספציפית).

מה שנשאר לנו לעשות כדי לראות את המקרים עם הרבה שורות לשנייה- הוא למיין את תוצאת ה count בסדר יורד (מהגדול לקטן) או פשוט למיין רגיל ולקחת את ה x שורות אחרונות כמו שלמדנו שקורה בהפעלת הפונקציה tail (שעושה פעולה הפוכה מהפונקציה head).

**לא משנה איזו עמודה נבחר למיין לפיה – כי זו אותה תוצאת count בכל העמודות פר שורה.

```
In [25]: df.groupby('date_time_int').count().sort_values('date_time').tail(10)
```

Out[25]:

date_time_int	Timestamp	EventType	Ticker	Price	Quantity	Exchange	Conditions	date_time
29027	48	48	48	48	48	48	48	48
24999	48	48	48	48	48	48	48	48
31354	50	50	50	50	50	50	50	50
30356	50	50	50	50	50	50	50	50
26092	53	53	53	53	53	53	53	53
30741	54	54	54	54	54	54	54	54
27086	56	56	56	56	56	56	56	56

ניקח לדוגמא את השנייה האחרונה מספר 27086 ונבדוק את התוצאות שלה.

נסנן את הdata frame המקורי (df) לפי אותה שנייה ונתחיל לבדוק את התוצאות של שנייה זו ב new_df המקובץ אל מול הנתונים המקוריים בdf.

```
In [26]: df[df['date_time_int']==27086]
```

Out[26]:

	Timestamp	EventType	Ticker	Price	Quantity	Exchange	Conditions	date_time	date_time_int
23547	14:44:04.720	TRADE NB	IBM	152.91	200	NYSE	20000020	27086.143	27086
23548	14:44:04.720	TRADE NB	IBM	152.91	100	EDGA	20000020	27086.143	27086
23549	14:44:04.721	TRADE NB	IBM	152.91	100	BATS Y	20000020	27086.144	27086
23550	14:44:04.721	TRADE NB	IBM	152.91	400	NASDAQ BX	20000020	27086.144	27086

מכיוון שה df כבר ממוין אז המחיר שנראה בשורה הראשונה של הפלט בשנייה זו הוא מחיר הפתיחה שלנו open=152.91

ובשורה האחרונה של שניה זו נראה את מחיר הסגירה close = 152.93.

נבדוק אם אכן כך גם ב data frame המקובץ new_df.

ב new_df השניות משמשות כאינדקס, לכן לא נוכל לסנן רגיל את השנייה הספציפית כמו שאנחנו מסננים לפי ערך בעמודה מסוימת.

```
In [27]: new_df
```

Out[27]:

date_time_int	Open	Close	Low	High	Vol
0	155.61	155.61	155.61	155.61	1
2173	154.49	154.49	154.49	154.49	200
2222	154.50	154.50	154.50	154.50	500
2223	154.50	154.50	154.50	154.50	400

סינון לפי index מסוים

```
In [28]: new_df[new_df.index ==27086]
```

Out[28]:

	Open	Close	Low	High	Vol
date_time_int					
27086	152.91	152.93	152.91	152.94	7969

רואים שהערכים בעמודה open ו close נכונים.

איך נבדוק את שאר הערכים- low, high, vol

Low זה הערך הנמוך ביותר של שנייה ספציפית.

נבדוק מה מחיר ה min של שנייה זו ב df.

נשים את כל השורות של שנייה 27086 במשתנה חדש ואז נבדוק מה המחיר המינימלי בעמודה price.

```
In [33]: df_bdika=df[df['date_time_int']==27086]
         df_bdika['Price'].min()
Out[33]: 152.91
```

```
In [28]: new_df[new_df.index ==27086]
```

Out[28]:

	Open	Close	Low	High	Vol
date_time_int					
27086	152.91	152.93	152.91	152.94	7969

רואים שנכון.

עושים אותו הדבר כדי לבדוק את המחיר המקסימלי.

נרשום max במקום min.

```
In [34]: df_bdika=df[df['date_time_int']==27086]
         df_bdika['Price'].max()

Out[34]: 152.94
```

```
In [28]: new_df[new_df.index ==27086]

Out[28]:
```

	Open	Close	Low	High	Vol
date_time_int					
27086	152.91	152.93	152.91	152.94	7969

רואים שהערך בעמודה high תקין.

עכשיו נבדוק את ערך הvol. צריך לבדוק את הsum של עמודת Quantity בשנייה זו.

```
In [35]: df_bdika=df[df['date_time_int']==27086]
         df_bdika['Quantity'].sum()

Out[35]: 7969L
```

```
In [28]: new_df[new_df.index ==27086]

Out[28]:
```

	Open	Close	Low	High	Vol
date_time_int					
27086	152.91	152.93	152.91	152.94	7969

גם תקין..

המרה לנרות של דקה

עכשיו יש לנו מידע פר שנייה. איך נמיר את המידע כך שיוצג פר דקה.

נחלק את עמודת השניות ל 60 וכך נמיר את השניות לדקות...

אח"כ נצטרך שוב לקבץ את המידע כך שתהיה לכל דקה שורה אחת.

עכשיו עמודת השניות ב new_df (ה data frame המקובץ עם 5 המדדים) היא index נצטרך לשנות זאת כך שתופיע כעמודה רגילה ולא כאינדקס.

נשתמש בפונקציה reset_index.

In [36]:
```
new_df=new_df.reset_index()
new_df
```

Out[36]:

	date_time_int	Open	Close	Low	High	Vol
0	0	155.61	155.61	155.61	155.61	1
1	2173	154.49	154.49	154.49	154.49	200
2	2222	154.50	154.50	154.50	154.50	500
3	2223	154.50	154.50	154.50	154.50	400
4	2224	154.50	154.50	154.50	154.50	1200

נוסיף עמודה חדשה בשם date_time_min שתייצג את הדקות

In [37]:
```
new_df['date_time_min']=new_df['date_time_int']/60
new_df
```

Out[37]:

	date_time_int	Open	Close	Low	High	Vol	date_time_min
0	0	155.61	155.61	155.61	155.61	1	0.000000
1	2173	154.49	154.49	154.49	154.49	200	36.216667
2	2222	154.50	154.50	154.50	154.50	500	37.033333
3	2223	154.50	154.50	154.50	154.50	400	37.050000

קיבלנו ערך מסוג float.

נמיר אותו ל int.

```
new_df['date_time_min']=new_df['date_time_min'].astype(int)
new_df
```

	date_time_int	Open	Close	Low	High	Vol	date_time_min
0	0	155.61	155.61	155.61	155.61	1	0
1	2173	154.49	154.49	154.49	154.49	200	36
2	2222	154.50	154.50	154.50	154.50	500	37
3	2223	154.50	154.50	154.50	154.50	400	37
4	2224	154.50	154.50	154.50	154.50	1200	37

עכשיו אנחנו צריכים לקבץ את המידע פר דקה.

ניצור data frame חדש שבו נשים את המידע המקובץ פר דקה.

ניתן לראות בצילום איך בנינו את טבלת 5 המדדים בפעם הקודמת לעומת עכשיו.

ההבדל- עכשיו יש לנו כבר את העמודות של 5 המדדים ורק נשאר לנו לקבץ אותן **פר דקה**:

ניקח את הערך של שורה 0 בעמודה Open עבור כל דקה

ניקח את הערך של השורה האחרונה בעמודה Close עבור כל דקה.

ניקח את הערך המינימלי של Low עבור כל דקה.

ניקח את הערך המקסימלי של High עבור כל דקה.

ניקח את ה sum של עמודה vol פר דקה.

In [46]:
```
min_candle=pd.DataFrame()
min_candle['Open']=new_df.groupby('date_time_min')['Open'].apply(lambda x: x.iloc[0])
min_candle['Close']=new_df.groupby('date_time_min')['Close'].apply(lambda x: x.iloc[-1])
min_candle['Low']=new_df.groupby('date_time_min')['Low'].apply(lambda x: x.min())
min_candle['High']=new_df.groupby('date_time_min')['High'].apply(lambda x: x.max())
min_candle['Vol']=new_df.groupby('date_time_min')['Vol'].apply(lambda x: x.sum())
min_candle

#new_df['Open']=df.groupby('date_time_int')['Price'].apply(lambda x: x.iloc[0])
#new_df['Close']=df.groupby('date_time_int')['Price'].apply(lambda x: x.iloc[-1])
#new_df['Low']=df.groupby('date_time_int')['Price'].apply(lambda x: x.min())
#new_df['High']=df.groupby('date_time_int')['Price'].apply(lambda x: x.max())
#new_df['Vol']=df.groupby('date_time_int')['Quantity'].apply(lambda x: x.sum())
```

Out[46]:

	Open	Close	Low	High	Vol
date_time_min					
0	155.61	155.61	155.61	155.61	1
36	154.49	154.49	154.49	154.49	200
37	154.50	154.50	154.50	154.53	3101
38	154.17	154.17	154.17	154.17	30
47	154.17	154.17	154.17	154.17	31333
54	154.37	154.37	154.37	154.37	100

נבדוק את התוצאות:

1) נשתמש בפונקציה info:

אנחנו מצפים לראות שב data frame החדש (min_candle) יש פחות שורות מאשר בnew_df

In [49]:
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8365 entries, 0 to 8364
Data columns (total 7 columns):
date_time_int    8365 non-null int64
Open             8365 non-null float64
Close            8365 non-null float64
Low              8365 non-null float64
High             8365 non-null float64
Vol              8365 non-null int64
```

```
In [48]: min_candle.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 451 entries, 0 to 765
Data columns (total 5 columns):
Open     451 non-null float64
Close    451 non-null float64
Low      451 non-null float64
High     451 non-null float64
Vol      451 non-null int64
```

הבדיקה הזו תקינה. יש פחות שורות ב data frame פר דקה מאשר ב data frame פר שנייה.

בדיקה באיזו מהדקות יש לנו הרבה שורות כדי שאותן נחקור:

עושים count ו groupby לעמודה דקות ב data frame הקודם, שמכיל מידע של הדקות לפני הקיבוץ.

```
In [51]: new_df.groupby('date_time_min').count()
```

Out[51]:

date_time_min	date_time_int	Open	Close	Low	High	Vol
0	1	1	1	1	1	1
36	1	1	1	1	1	1
37	6	6	6	6	6	6
38	1	1	1	1	1	1
47	1	1	1	1	1	1
54	1	1	1	1	1	1
61	2	2	2	2	2	2
77	1	1	1	1	1	1
86	1	1	1	1	1	1
90	7	7	7	7	7	7
91	5	5	5	5	5	5
92	1	1	1	1	1	1
94	1	1	1	1	1	1

מחפשים דקות עם כמות גדולה של שורות באמצעות הפונקציה sort_value

ככה נראית הטבלה הקודמת שלנו

```
In [54]: new_df
```

Out[54]:

	date_time_int	Open	Close	Low	High	Vol	date_time_min
0	0	155.61	155.61	155.61	155.61	1	0
1	2173	154.49	154.49	154.49	154.49	200	36
2	2222	154.50	154.50	154.50	154.50	500	37
3	2223	154.50	154.50	154.50	154.50	400	37
4	2224	154.50	154.50	154.50	154.50	1200	37
5	2225	154.50	154.50	154.50	154.50	600	37
6	2226	154.50	154.50	154.50	154.50	200	37
7	2228	154.50	154.50	154.50	154.53	201	37

קיבוץ לפי דקה+ ספירת כמות ערכים לדקה+ מיון .

אנחנו רואים שיש 54 שורות בדקה 526.

```
new_df.groupby('date_time_min').count().sort_values('date_time_int').tail(10)
```

	date_time_int	Open	Close	Low	High	Vol
date_time_min						
520	43	43	43	43	43	43
516	43	43	43	43	43	43
512	48	48	48	48	48	48
517	48	48	48	48	48	48
522	48	48	48	48	48	48
523	49	49	49	49	49	49
511	50	50	50	50	50	50
524	52	52	52	52	52	52
525	52	52	52	52	52	52
526	54	54	54	54	54	54

נבדוק את הערכים בטבלה החדשה שלנו לפי דקה 526.

נשים את שורות אלו במשתנה חדש df_bdika

```
df_bdika=new_df[new_df['date_time_min']==526]
df_bdika.head(5)
```

	date_time_int	Open	Close	Low	High	Vol	date_time_min
8231	31560	151.74	151.75	151.74	151.75	165	526
8232	31561	151.74	151.75	151.74	151.75	414	526
8233	31562	151.75	151.74	151.74	151.75	580	526
8234	31563	151.74	151.75	151.74	151.75	1312	526
8235	31564	151.75	151.74	151.74	151.75	247	526

Open- מכיוון שהטבלה ממוינת לפי שנייה אז הערך בשורה הראשונה של open הוא המחיר פתיחה של המניה בדקה 526 והוא 151.74.

Close- יהיה השורה האחרונה, הclose בדקה האחרונה. במקרה שלנו: 151.64.

```
df_bdika.tail(5)
```

Out[62]:

	date_time_int	Open	Close	Low	High	Vol	date_time_min
8280	31614	151.70	151.67	151.64	151.70	1900	526
8281	31616	151.64	151.64	151.64	151.64	16	526
8282	31617	151.63	151.65	151.63	151.69	236	526
8283	31618	151.66	151.66	151.66	151.66	15	526
8284	31619	151.67	151.64	151.64	151.67	341	526

בדיקת הערך של דקה 526 בעמודה Low

```
df_bdika['Low'].min()
```

```
151.63
```

בדיקת הערך של דקה 526 בעמודה High

```
df_bdika['Low'].max()
```

```
151.78
```

בדיקת הערך של דקה 526 בעמודה vol

```
df_bdika['Vol'].sum()
```

```
74145L
```

עשינו בדיקה כדי לראות איזה ערכים אנחנו מצפים לראות בדקה 526 בכל 5 המדדים.

עכשיו נראה בכלל מה הערכים שקיבלנו ב data frame לפי דקה (min_candle) בדקה 526 ונראה אם הם זהים לערכים שבדקנו.

```
min_candle
```

date_time_min	Open	Close	Low	High	Vol
0	155.61	155.61	155.61	155.61	1
36	154.49	154.49	154.49	154.49	200
37	154.50	154.50	154.50	154.53	3101
38	154.17	154.17	154.17	154.17	30

נראה את הערכים בדקה 526:

```
min_candle[min_candle.index ==526]
```

date_time_min	Open	Close	Low	High	Vol
526	151.74	151.64	151.63	151.78	74145

נראה שתקין....הבדיקה זהה לערכים של דקה 526 בטבלת min_candle.

ניתוח מניות SNP500

קובץ חדש שמכיל את כל המניות שנמצאות בSNP ב5 שנים האחרונות.

מדד S&P500

28/06/2015

מהו מדד ה S&P500, ממה הוא מורכב ומה הוא מייצג ? המדד פותח על ידי חברת הדירוג Standard & Poor's ומכאן שמו והוא מורכב מ- 500 החברות בעלות שווי השוק הגדול ביותר הנסחרות בבורסות הראשיות בארה"ב בהן: הנאסד"ק - NASDAQ, בורסת ניו-יורק -NYSE.

המדד הושק ופורסם לראשונה בשנת 1957 ומהר מאוד הפך למדד מרכזי. המניות הכלולות במדד נבחרות מתוך המדד הרחב הכולל 1500 חברות (S&P 1500)

המניות נבחרות על ידי ועדה המורכבת מכלכלנים ואנליסטים בחברת Standard & Poor's, חלק מהקריטריונים לכניסה למדד הם: שווי השוק המינימלי, מידת הנזילות והשייכות הסקטוריאלית/הענפית.

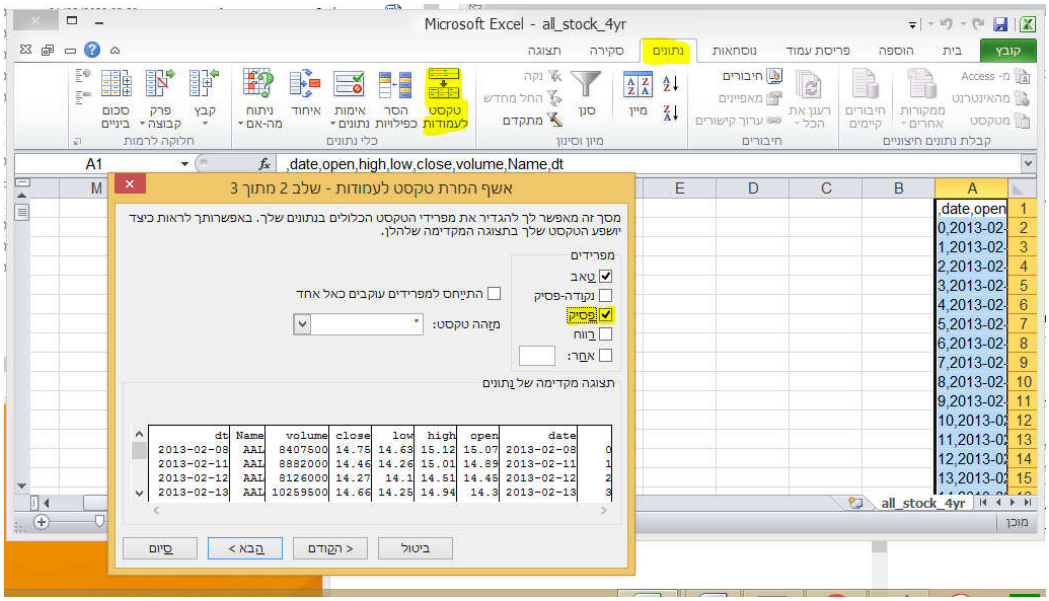
מדד S&P 500 משמש כמדד ייחוס (Benchmark) מרכזי לשוקי המניות בארה"ב ולמדדי מניות בעולם כולו, מכיוון שהמדד מהווה ברומטר לביצועי המניות והחברות הציבוריות בארה"ב. בעבר, מדד הדאו-ג'ונס 30 שימש כחלון הראווה לשוק המניות האמריקאי, אך מכיוון שמורכב ממספר קטן יחסית של 30 חברות , מדד ה-S&P 500 תפס את מקומו וביסס את מעמדו בתור האינדקס המרכזי של וול-סטריט המדד בנוי מ 500 חברות ממגוון ענפים וסקטורים של הכלכלה האמריקנית.

אופן חישוב והרכב המדד: המדד הוא מסוג שווי שוק, משקל כל מניה במדד נקבע בהתאם לשווי השוק היחסי שלה, כמו כן המדד הוא מסוג Price only, כך שהדיבידנדים המחולקים מרווחי החברות לא כלולים בו.

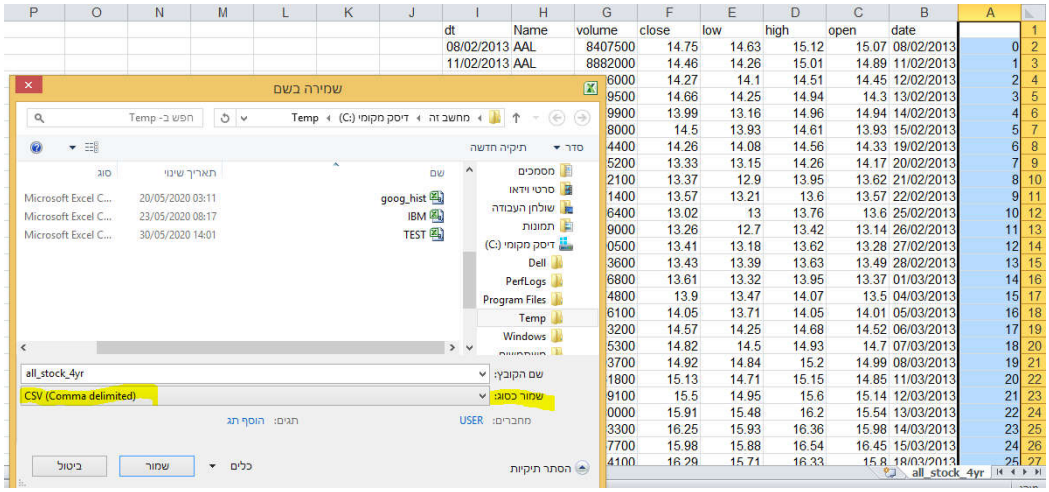
אם הקובץ מידע שמורידים נראה שכל העמודות בו נמצאות בעמודה אחת של אקסל במקום להיות מפוצלות:

איך מבצעים הפרדה לעמודות:

מסמנים את העמודה שרוצים לפצל-> בוחרים לשונית נתונים->טקסט לעמודות->ונפתח אשף ובו בוחרים מה המזהה של הפרדה בין עמודות. במקרה שלנו זה פסיק שמפריד.



הקובץ מסתדר ואח"כ שומרים אותו בשם -> משנים את הסוג שלו ל(csv (Comma delimited



שניתי את שם הקובץ לstock_4yr כי לפני כן הפייתון לא הצליח למצוא אותו...

```
In [5]: import pandas as pd
        df=pd.read_csv('c:\Temp\stock_4yr.csv')
        df
```

Out[5]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt
0	0	08/02/2013	15.070	15.12	14.630	14.75	8407500	AAL	08/02/2013
1	1	11/02/2013	14.890	15.01	14.260	14.46	8882000	AAL	11/02/2013
2	2	12/02/2013	14.450	14.51	14.100	14.27	8126000	AAL	12/02/2013
3	3	13/02/2013	14.300	14.94	14.250	14.66	10259500	AAL	13/02/2013
4	4	14/02/2013	14.940	14.96	13.160	13.99	31879900	AAL	14/02/2013
5	5	15/02/2013	13.930	14.61	13.930	14.50	15628000	AAL	15/02/2013
6	6	19/02/2013	14.330	14.56	14.080	14.26	11354400	AAL	19/02/2013
7	7	20/02/2013	14.170	14.26	13.150	13.33	14725200	AAL	20/02/2013

אנחנו רואים שהמידע הוא לפי יום.

פונקציה בשם value_counts – מאפשרת לראות כמה ערכים יש לנו מכל ערך ייחודי.

```
In [6]: df['Name'].value_counts()

Out[6]: PNC     1233
        XL      1233
        VZ      1233
        TEL     1233
        NBL     1233
        DISH    1233
        HOG     1233
        HON     1233
        IDXX    1233
```

איך יודעים איזה פונקציות נמצאות בתוך pandas ומה הן עושות.

באתר pandas יש את כל המידע וכל הפקודות הזמינות + הסבר ודוגמאות.

באתר stack overflow יש שאלות ותשובות של אנשים.

pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value_counts.html

pandas Home What's New in 1.0.1 Getting started User Guide API reference Development Release Notes

Search the docs ...

Input/output
General functions
Series
pandas.Series
pandas.Series.index
pandas.Series.array
pandas.Series.values
pandas.Series.dtype
pandas.Series.shape
pandas.Series.nbytes
pandas.Series.ndim
pandas.Series.size
pandas.Series.T
pandas.Series.memory_usag

pandas.Series.value_counts

`Series.value_counts`(*self, normalize=False, sort=True, ascending=False, bins=None, dropna=True*)

Return a Series containing counts of unique values.

The resulting object will be in descending order so that the first element is the most frequently-occurring e
NA values by default.

Parameters: **normalize** : *bool, default False*
If True then the object returned will contain the relative frequencies of the unique valu

sort : *bool, default True*
Sort by frequencies.

ascending : *bool, default False*
Sort in ascending order.

bins : *int, optional*
Rather than count values, group them into half-open bins, a convenience for `pd.cut`, or
numeric data.

dropna : *bool, default True*

דרך אחרת לקבל תוצאה כמו ב value_counts:

```
In [9]: df.groupby('Name').count()
```

Out[9]:

	Unnamed: 0	date	open	high	low	close	volume	dt
Name								
A	1233	1233	1233	1233	1233	1233	1233	1233
AAL	1233	1233	1233	1233	1233	1233	1233	1233
AAP	1233	1233	1233	1233	1233	1233	1233	1233

אפשר לבחור עמודה בטבלה כדי שיוצג יותר יפה ולא אותם ערכים בכל העמודות.

לא משנה איזו עמודה בוחרים כי בכולן אותה תוצאה.

```
In [10]: df.groupby('Name')['date'].count()

Out[10]: Name
         A       1233
         AAL     1233
         AAP     1233
         AAPL    1233
         ABBV    1233
         ABC     1233
         ABT     1233
         ACN     1233
```

הכנה לעיבוד dt, חודש ושנה

המרת העמודה date לסוג תאריך במקום טקסט.

נשמור את ההמרה בעמודה חדשה בשם date_time.

```
df['date_time']=pd.to_datetime(df['date'])
```

איך לוקחים רק את השנה מתוך date_time:

```
df['date_time'].dt.year
```

באמצעות dt אפשר לגשת לתכונות של ערכים מסוג date_time.

```
In [3]: print(df['date_time'].head(3))
        print(df['date_time'].dt.year.head(3))
        print(df['date_time'].dt.month.head(3))
        print(df['date_time'].dt.day.head(3))

0   2013-08-02
1   2013-11-02
2   2013-12-02
Name: date_time, dtype: datetime64[ns]
0    2013
1    2013
2    2013
Name: date_time, dtype: int64
0     8
1    11
2    12
Name: date_time, dtype: int64
0    2
1    2
2    2
Name: date_time, dtype: int64
```

ניצור עמודות חדשות של year ו month

```
In [ ]: df['year']=df['date_time'].dt.year
        df['month']=df['date_time'].dt.month
```

סינון המידע

אם נרצה לראות רק את שנת 2013. נסנן כך:

```
In [7]: df[df['year']==2013]
```

Out[7]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
0	0	08/02/2013	15.07	15.120	14.630	14.75	8407500	AAL	08/02/2013	2013-08-02	2013	8
1	1	11/02/2013	14.89	15.010	14.260	14.46	8882000	AAL	11/02/2013	2013-11-02	2013	11
2	2	12/02/2013	14.45	14.510	14.100	14.27	8126000	AAL	12/02/2013	2013-12-02	2013	12
3	3	13/02/2013	14.30	14.940	14.250	14.66	10259500	AAL	13/02/2013	2013-02-13	2013	2
4	4	14/02/2013	14.94	14.960	13.160	13.99	31879900	AAL	14/02/2013	2013-02-14	2013	2

אם נרצה לראות רק חודש מסוים נסנן כך:

```
In [8]: df[df['month']==12]
```

Out[8]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
2	2	12/02/2013	14.4500	14.5100	14.1000	14.2700	8126000	AAL	12/02/2013	2013-12-02	2013	12
21	21	12/03/2013	15.1400	15.6000	14.9500	15.5000	8999100	AAL	12/03/2013	2013-12-03	2013	12
43	43	12/04/2013	16.1100	16.3900	15.9500	16.1400	3751800	AAL	12/04/2013	2013-12-04	2013	12
85	85	12/06/2013	16.9700	17.2000	16.6500	16.8800	3602200	AAL	12/06/2013	2013-12-06	2013	12

אם נרצה לראות חודש מסוים בשנה מסוימת:

```
In [9]: df[(df['month']==12) & (df['year']==2013)]
```

Out[9]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
2	2	12/02/2013	14.4500	14.5100	14.1000	14.2700	8126000	AAL	12/02/2013	2013-12-02	2013	12
21	21	12/03/2013	15.1400	15.6000	14.9500	15.5000	8999100	AAL	12/03/2013	2013-12-03	2013	12
43	43	12/04/2013	16.1100	16.3900	15.9500	16.1400	3751800	AAL	12/04/2013	2013-12-04	2013	12
85	85	12/06/2013	16.9700	17.2000	16.6500	16.8800	3602200	AAL	12/06/2013	2013-12-06	2013	12
106	106	12/07/2013	17.3600	17.8500	17.3500	17.5600	3947100	AAL	12/07/2013	2013-12-07	2013	12
127	127	12/08/2013	18.4900	18.8800	18.0700	18.8200	4674800	AAL	12/08/2013	2013-12-08	2013	12

אריתמטיקה בוליאנית

איך מחברים תנאים בוליאנים

1) אם רוצים שתנאי אחד יתקיים **וגם** השני יתקיים אז נשתמש בסימן **&**.
אם שניהם יתקיימו נקבל True
אם רק אחד יתקיים או אף אחד לא יתקיים נקבל False.

2) אם מספיק לנו שתנאי אחד יתקיים מתוך למשל 2 תנאים או יותר.
כלומר **או** שתנאי אחד יתקיים **או** ששני.
אז נשתמש בסימן |.
אם שני התנאים יתקיימו נקבל True
אם רק תנאי אחד יתקיים נקבל True
אם אף תנאי לא יתקיים נקבל False.

```
In [10]: grade=95
         (grade>90)|(grade<60)

Out[10]: True
```

```
In [11]: grade=80
         (grade>90)|(grade<60)

Out[11]: False
```

```
In [12]: grade=50
         (grade>90)|(grade<60)

Out[12]: True
```

```
In [17]: grade=50
         (grade>90)&(grade<60)

Out[17]: False
```

```
In [18]: grade=50
         (grade>40)&(grade<60)

Out[18]: True
```

אם רוצים ששני תנאים יתקיימו נשתמש ב &
אם מספיק שתנאי אחד יתקיים אז נשתמש ב |

חישוב ביצועים למניה ALL

נשמור במשתנה את הסינון של שנה אחת וחודש אחד שעשינו קודם.

```
In [19]: one_month=df[(df['month']==12) & (df['year']==2013)]
         one_month
```

Out[19]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
2	2	12/02/2013	14.4500	14.5100	14.1000	14.2700	8126000	AAL	12/02/2013	2013-12-02	2013	12
21	21	12/03/2013	15.1400	15.6000	14.9500	15.5000	8999100	AAL	12/03/2013	2013-12-03	2013	12
43	43	12/04/2013	16.1100	16.3900	15.9500	16.1400	3751800	AAL	12/04/2013	2013-12-04	2013	12
85	85	12/06/2013	16.9700	17.2000	16.6500	16.8800	3602200	AAL	12/06/2013	2013-12-06	2013	12
106	106	12/07/2013	17.3600	17.8500	17.3500	17.5600	3947100	AAL	12/07/2013	2013-12-07	2013	12
127	127	12/08/2013	18.4900	18.8800	18.0700	18.8200	4674800	AAL	12/08/2013	2013-12-08	2013	12

נסנן מניה אחת ממשתנה זה (מניית AAL):

```
In [20]: one_month[one_month['Name']=='AAL']
```

Out[20]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
2	2	12/02/2013	14.45	14.5100	14.1000	14.27	8126000	AAL	12/02/2013	2013-12-02	2013	12
21	21	12/03/2013	15.14	15.6000	14.9500	15.50	8999100	AAL	12/03/2013	2013-12-03	2013	12
43	43	12/04/2013	16.11	16.3900	15.9500	16.14	3751800	AAL	12/04/2013	2013-12-04	2013	12
85	85	12/06/2013	16.97	17.2000	16.6500	16.88	3602200	AAL	12/06/2013	2013-12-06	2013	12

****iloc[0]= תמיד שורה ראשונה גם אם האינדקס שלה הוא לא 0.

מהו התאריך הנמוך ביותר:

```
In [21]: one_month[one_month['Name']=='AAL']['date_time'].min()
Out[21]: Timestamp('2013-12-02 00:00:00')
```

נשמור זאת במשתנה בשם start

```
In [23]: start=one_month[one_month['Name']=='AAL']['date_time'].min()
         start
Out[23]: Timestamp('2013-12-02 00:00:00')
```

מהו התאריך הגבוה ביותר:

```
In [24]: end=one_month[one_month['Name']=='AAL']['date_time'].max()
         end
Out[24]: Timestamp('2013-12-31 00:00:00')
```

נבדוק מה ערך פתיחה של מניה ומה ערך סגירה בחודש ושנה שסיננו:

ערך סגירה:

```
In [30]: one_month[(one_month['date_time']==end) & (one_month['Name']=='AAL')]
```

Out[30]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
225	225	31/12/2013	24.74	25.25	24.63	25.25	7168395	AAL	31/12/2013	2013-12-31	2013	12

```
In [29]: one_month[(one_month['date_time']==end) & (one_month['Name']=='AAL')]['close']
Out[29]: 225    25.25
         Name: close, dtype: float64
```

ערך פתיחה:

```
In [28]: one_month[(one_month['date_time']==start) & (one_month['Name']=='AAL')]
```

Out[28]:

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
2	2	12/02/2013	14.45	14.51	14.1	14.27	8126000	AAL	12/02/2013	2013-12-02	2013	12

```
In [31]: one_month[(one_month['date_time']==start) & (one_month['Name']=='AAL')]['open']
Out[31]: 2    14.45
         Name: open, dtype: float64
```

נשמור הכל בתוך משתנים:

```
In [32]: start=one_month[one_month['Name']=='AAL']['date_time'].min()
         end=one_month[one_month['Name']=='AAL']['date_time'].max()
         open_month=one_month[(one_month['date_time']==start) & (one_month['Name']=='AAL')]['open']
         close_month=one_month[(one_month['date_time']==end) & (one_month['Name']=='AAL')]['close']
```

חישוב ביצועים באחוזים למניה

מחיר מקורי של מניה open_month

איך מחשבים את השינוי במניה:

Close_month פחות מחיר פתיחה open_month = כמה שעלתה או ירדה.

כדי לקבל זאת באחוזים נחלק את התוצאה ב start_month כפול 100.

הערך של open_month הוא מסוג series אז הדרך הקלה לחלץ ממנו את הערך היא להשתמש ב iloc[0]

```
In [34]: type(open_month)
Out[34]: pandas.core.series.Series

In [35]: open_month.iloc[0]
Out[35]: 14.449999999999999
```

כך שעדיף לשמור במשתנים אברים בודדים ולא עמודה כדי שנוכל לבצע את פעולת החיסור והחילוק בקלות ולקבל את תוצאת האחוזים:

```
In [41]: start=open_month.iloc[0]
         end=close_month.iloc[0]
         print(start,end)
         (14.449999999999999, 25.25)
```

חישוב אחוזים:

```
In [42]: ((end-start)/start)*100
Out[42]: 74.740484429065759
```

אנחנו רואים שהמניה עלתה ב 74%

חיבור סטרינג חודש ושנה

יש לנו תאריך שמופרד לחודש ושנה.

```
In [43]: df
Out[43]:
```

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month
0	0	08/02/2013	15.070	15.12	14.630	14.75	8407500	AAL	08/02/2013	2013-08-02	2013	8
1	1	11/02/2013	14.890	15.01	14.260	14.46	8882000	AAL	11/02/2013	2013-11-02	2013	11
2	2	12/02/2013	14.450	14.51	14.100	14.27	8126000	AAL	12/02/2013	2013-12-02	2013	12
3	3	13/02/2013	14.300	14.94	14.250	14.66	10259500	AAL	13/02/2013	2013-02-13	2013	2

אם אנחנו רוצים שתהיה לנו עמודה של חודש ושנה ביחד. יש כמה דרכים לבצע זאת:

1) עמודה date היא מסוג string אז אנחנו יכולים פשוט לקחת ממנה חלק מהתווים כך שיהיה לנו רק את החודש והשנה ללא היום.

```
In [45]: '2013-02-08'[:-3]
Out[45]: '2013-02'

In [49]: '08/02/2013'[3:]
Out[49]: '02/2013'
```

איך כותבים זאת כך שיבוצע על עמודה שלמה:

```
In [48]: df['date'].apply(lambda x: x[3:])

Out[48]: 0         02/2013
         1         02/2013
         2         02/2013
         3         02/2013
         4         02/2013
         5         02/2013
         6         02/2013
         7         02/2013
         8         02/2013
         9         02/2013
         10        02/2013
         11        02/2013
         12        02/2013
         13        02/2013
         14        03/2013
         15        03/2013
```

נשמור זאת במשתנה:

```
In [50]: df['string_hacking']=df['date'].apply(lambda x: x[3:])
```

דרך נוספת ליצירת עמודה של חודש ושנה:

```
In [51]: df['date_time'].dt.year.astype(str)+'-'+df['date_time'].dt.month.astype(str)

Out[51]: 0          2013-8
         1         2013-11
         2         2013-12
         3          2013-2
         4          2013-2
         5          2013-2
         6          2013-2
         7          2013-2
         8          2013-2
```

נשמור זאת כעמודה חדשה ב df:

```
In [52]: df['dt_string']=df['date_time'].dt.year.astype(str)+'-'+df['date_time'].dt.month.astype(str)

In [53]: df

Out[53]:
```

	Unnamed: 0	date	open	high	low	close	volume	Name	dt	date_time	year	month	string_hacking	dt_string
0	0	08/02/2013	15.070	15.12	14.630	14.75	8407500	AAL	08/02/2013	2013-08-02	2013	8	02/2013	2013-8
1	1	11/02/2013	14.890	15.01	14.260	14.46	8882000	AAL	11/02/2013	2013-11-02	2013	11	02/2013	2013-11
2	2	12/02/2013	14.450	14.51	14.100	14.27	8126000	AAL	12/02/2013	2013-12-02	2013	12	02/2013	2013-12
3	3	13/02/2013	14.300	14.94	14.250	14.66	10259500	AAL	13/02/2013	2013-02-13	2013	2	02/2013	2013-2
4	4	14/02/2013	14.940	14.96	13.160	13.99	31879900	AAL	14/02/2013	2013-02-14	2013	2	02/2013	2013-2

חישוב תשואה למניה

נבצע groupby כדי לחשב נתונים פר חודש ושנה.

למשל נראה את הנתונים של עמודת הclose לפי חודש ושנה:

מה שרשמנו נכון אם המידע ממוין ובאמת השורה האחרונה מייצגת את הclose האחרון בחודש.

```
In [54]: df.groupby('dt_string')['close'].apply(lambda x: x.iloc[-1])

Out[54]: dt_string
         2013-1      31.84
         2013-10     31.36
         2013-11     31.09
         2013-12     32.69
         2013-2      31.15
         2013-3      31.09
         2013-4      31.21
```

אחרי sort_values נוכל להיות בטוחים שהשימוש במיקום 1- ייתן לנו את הערך הנכון, של סוף החודש.

```
In [55]: df=df.sort_values('date_time')
         df.groupby('dt_string')['close'].apply(lambda x: x.iloc[-1])

Out[55]: dt_string
         2013-1      37.3700
         2013-10     34.7000
         2013-11     70.5400
```

קיבלנו לכל המניות ביחד תוצאת close לחודש ושנה אך לא התוצאה לא מופרדת פר מנייה...

איך גורמים לכך שיבצע groupby גם לפי חודש ושנה וגם לפי שם מניה:

פתרון: במקום לתת בפרמטר של groupby של שם עמודה. ניתן לו list של כמה שמות עמודות.

**list צריך להיות בסוגריים מרובעים וחשוב סדר השדות ששמים בlist. אם אנחנו רוצים שקודם יקבץ לפי חודש ושנה ויראה לנו את כל המניות פר חודש ושנה או שרוצים שיקבץ פר שם מניה ופר שם מניה יראה לנו תוצאות של חודש ושנה.

```
In [56]: df=df.sort_values('date_time')
         df.groupby(['Name','dt_string'])['close'].apply(lambda x: x.iloc[-1])
```

```
Out[56]: Name  dt_string
         A     2013-1       51.10
               2013-10      50.76
               2013-11      53.57
               2013-12      57.19
               2013-2       41.48
               2013-3       41.97
               2013-4       41.44
               2013-5       45.45
               2013-6       42.76
               2013-7       44.73
               2013-8       46.64
               2013-9       51.25
               2014-1       58.15
               2014-10      55.28
               2014-11      42.74
               2014-12      40.94
               2014-2       56.93
               2014-3       55.92
               2014-4       54.04
               2014-5       56.94
               2014-6       57.44
               2014-7       56.09
               2014-8       57.16
               2014-9       56.98
               2015-1       37.77
               2015-10      37.76
               2015-11      41.82
               2015-12      41.81
               2015-2       42.21
               2015-3       41.55
                            ...
         ZTS   2015-4       44.42
               2015-5       49.77
```

חישוב תשואה למניה באחוזים

המחיר ביום המסחר הראשון בחודש פחות המחיר ביום המסחר האחרון בחודש

```
In [58]: df.groupby(['Name','dt_string']).apply(lambda x: x['open'].iloc[0]-x['open'].iloc[-1])
```

```
Out[58]: Name  dt_string
         A     2013-1       -9.74
               2013-10      -9.42
               2013-11      -8.68
               2013-12     -12.64
               2013-2       -0.66
               2013-3       -1.45
               2013-4        0.55
               2013-5       -3.56
               2013-6        0.00
               2013-7       -1.74
               2013-8       -1.82
               2013-9       -9.29
               2014-1       -2.41
               2014-10       3.30
               2014-11      16.26
               2014-12      18.61
               2014-2        0.65
               2014-3        1.42
               2014-4        2.89
```

עכשיו אנחנו רואים את הירידה והעליה של המניה בכסף.

עכשיו ננסה להפוך זאת לאחוזים:

מחלקים זאת במחיר הopen הראשון, המחיר ההתחלתי ואח"כ כפול 100.

ונשמור זאת במשתנה results

```
In [59]: results = df.groupby(['Name','dt_string']).apply(lambda x: 100*(x['open'].iloc[0] - x['close'].iloc[-1]) / x['open'].iloc[0])

In [60]: results

Out[60]: Name  dt_string
         A     2013-1       -24.089364
               2013-10      -21.639109
               2013-11      -18.596414
               2013-12      -27.627762
               2013-2        -0.875486
               2013-3        -3.298056
               2013-4         0.742515
               2013-5        -7.319953
               2013-6         0.558140
               2013-7        -3.302540
               2013-8        -3.483470
               2013-9       -22.813324
               2014-1        -3.469751
               2014-10        5.520424
               2014-11       27.559322
               2014-12       31.766667
```