תוכן

שבוע 1

התקנת סביבה

הורדת חבילה בשם אנקונדה שמכילה את שפת התכנות "פיתון"

הקדמה:

פיתון היא שפת שמאפשרת לכתוב מעט שורות קוד כדי להריץ הרבה דברים ותהליכים.

כל זאת הודות לחבילות הרבות שמתממשקות אליה.

החבילות השונות מכילות פונקציות רבות שנכתבו ע"י מומחים רבים וממשיכות להיכתב כל הזמן .

מה שנשאר לנו הוא להבין מה הן עושות ולהפעיל אותן בהתאם לצורך.

חבילת אנקונדה מאפשרת לנו את השימוש בשפת הפיתון וגם גישה פשוטה לספריות השונות.

דוגמאות לספריות:

Pandas- ספרייה שמאפשרת לבצע מניפולציה על נתונים לצורכי תחקור.

Matplotlib- ספרייה שמאפשרת לראות את הנתונים בצורה גרפית.

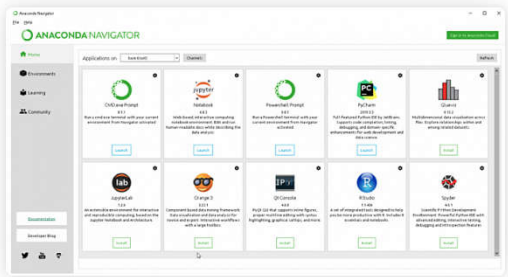
Scikit-Learn- ספרייה לבניית מודלי חיזוי- לחזות מה יהיה על סמך למידת מכונה של מה שהיה.

הורדת חבילת אנקונדה:

לינק להורדת חבילה אנקונדה:

https://www.anaconda.com/products/individua

נכנסים ללינק-> גוללים קצת למטה ומגיעים ל כפתור של "Install Anaconda"



User interface makes learning easier

Anaconda Navigator is a desktop GUI that comes with Anaconda Individual Edition. It makes it easy to launch applications and manage packages and environments without using command-line commands.

Expedite your data science journey with easy access to training materials, documentation, and community resources including Anaconda.org.

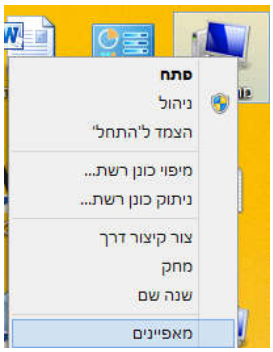
Install Anaconda

אח"כ הוא מביא אותנו לחלק הזה שבו אנחנו צריכים לבחור את ההתקנה Python 3.7 שמתאימה למחשב שיש ברשותנו.

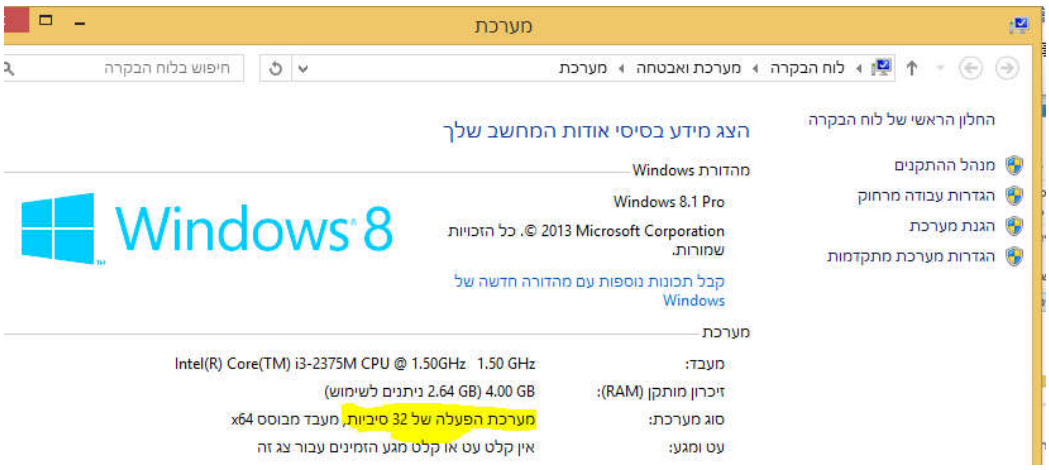
אם למשל יש לנו windows איך יודעים אם הוא 64bit או 32bit כדי לדעת באיזו התקנה לבחור?

בדרך כלל נראה בשולחן העבודה אייקון של "המחשב שלי"/"My Computer"/ "מחשב זה"

אם נלחץ עליו קליק ימני ונבחר באופציה "מאפיינים"



נראה לדוגמא: המחשב בדוגמא זו הוא 32bit.

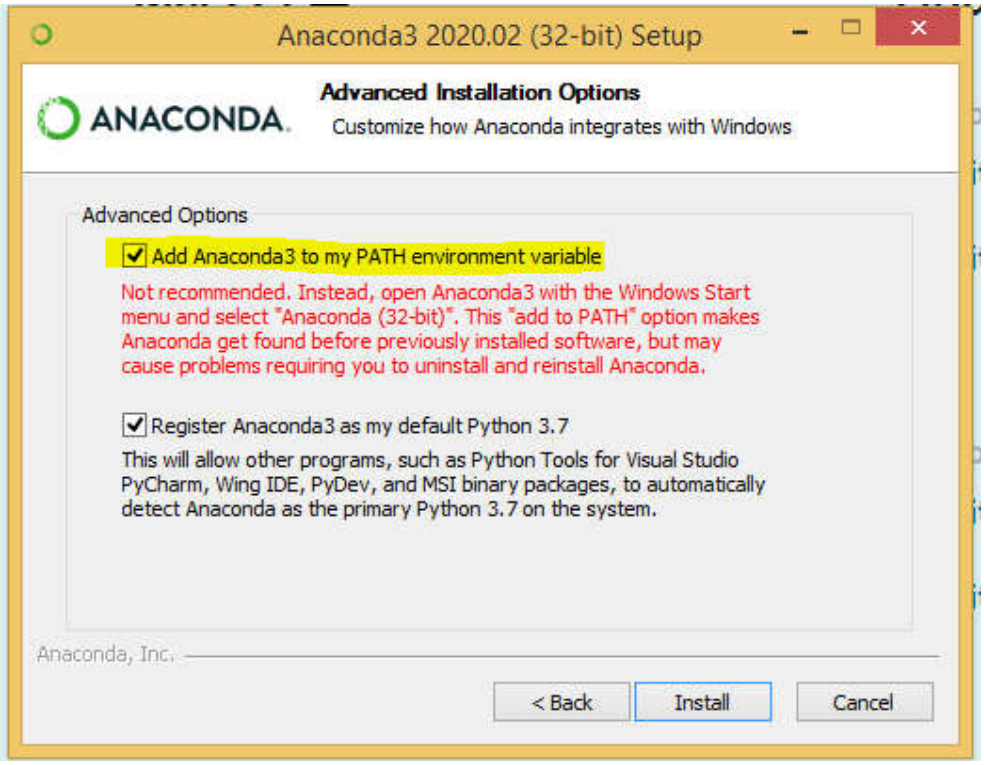


(אבל גם אם אנחנו לא מוצאים אז אל דאגה כי אם הורדנו קובץ לא מתאים אז בשלב ההתקנה כבר נקבל הודעה על כך שההתקנה לא מתאימה לגרסה של המחשב שלנו).

לוחצים על ההתקנה שמתאימה למחשב שלנו וההתקנה מתחילה לרדת.

אחרי שההתקנה מסיימת לרדת פותחים את הקובץ שירד ומתחילה התקנת התוכנה (חבילת אנקונדה).

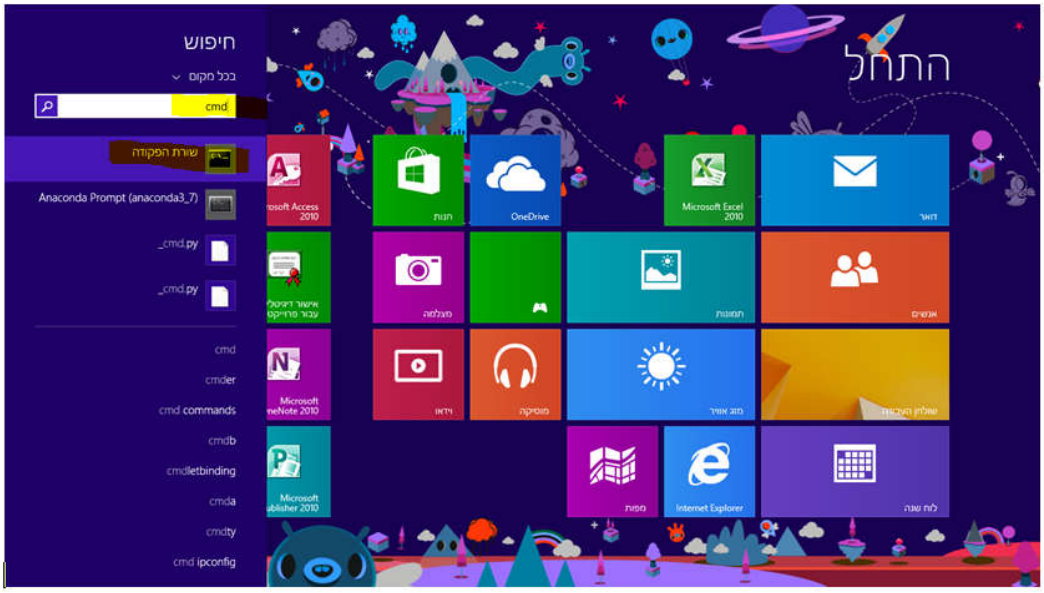
בהתקנה צריך ללחוץ next-> I agree -<... ובחלק של "Advanced Options" חשוב לסמן V באופציה העליונה.



אחרי שסיימנו להתקין ....

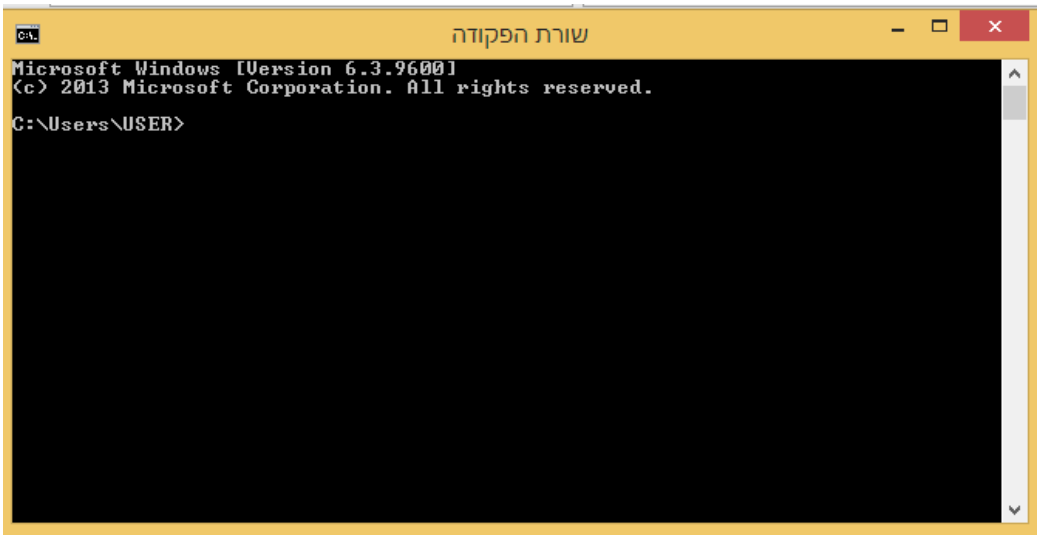
לוחצים על "התחל"/סמל חלונות בפינה הימנית או השמאלית של המסך.

ומקלידים את המילה cmd (זה קיצור למילה command prompt)

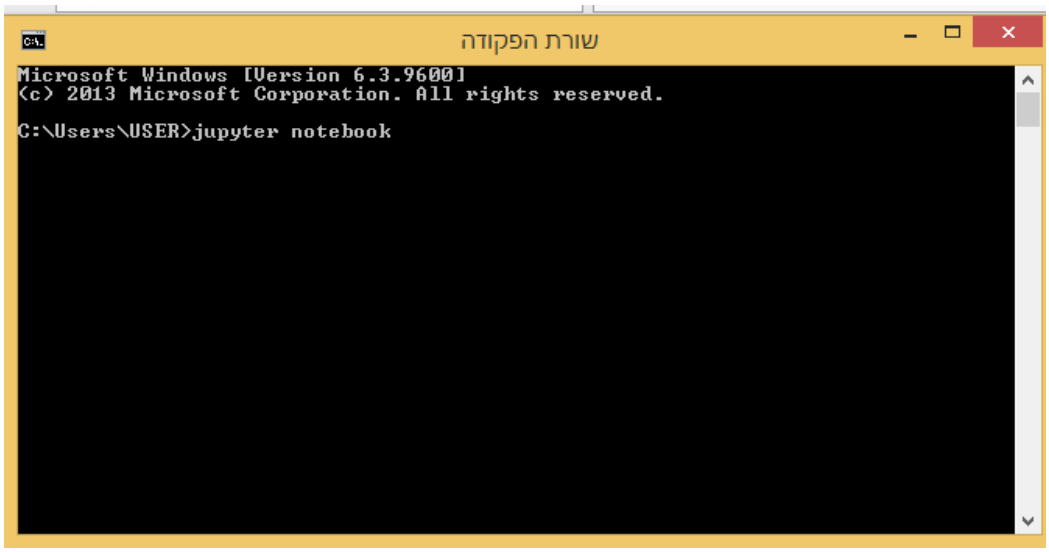


לוחצים על האופציה אם היא בעברית: "שורת פקודה" או אם באנגלית אז "command prompt"

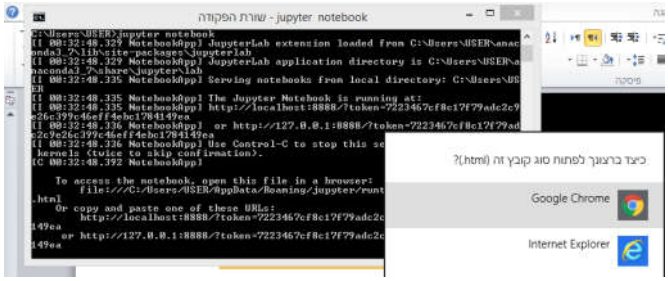
ויפתח לנו המסך הבא:



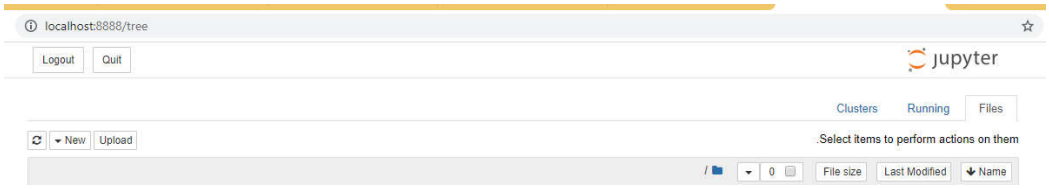
ואז נרשום: jupyter notebook ונלחץ על Enter.



אם קופץ חלון לבחירה עם איזו תוכנת אינטרנט לפתוח אז לא משנה מה בוחרים.



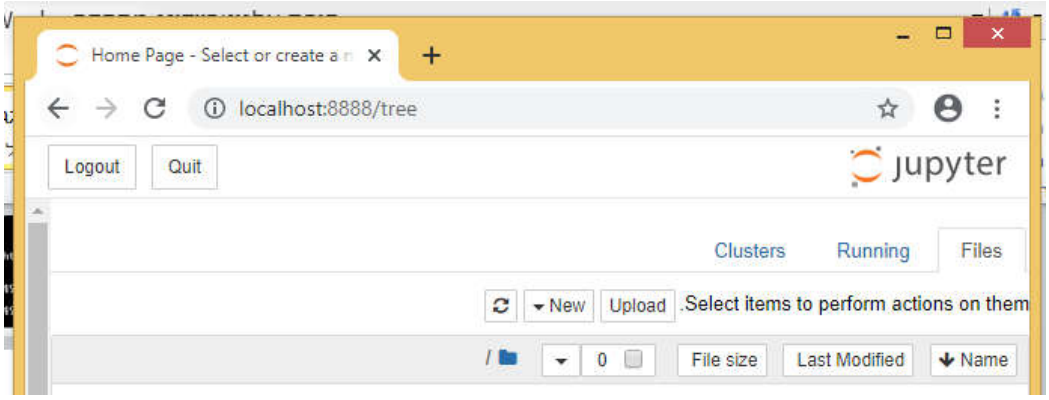
אם נפתח חלון דומה למה שרואים בצילום מסך זה אז אתם במקום הנכון.



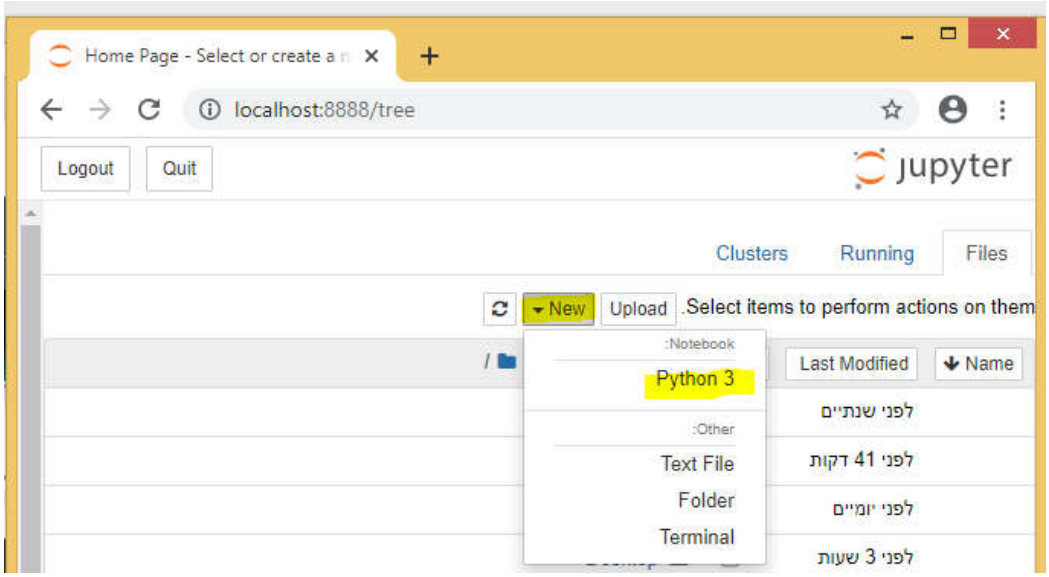
אם דרך זו לא עבדה לכם אז אפשר לפתוח את דפדפן האינטרנט (browser) ובשורת הכתובת לרשום לו:

http://localhost:8888/tree

ותגיעו לתוכנת ה jupyter



נלחץ על new-> Python3



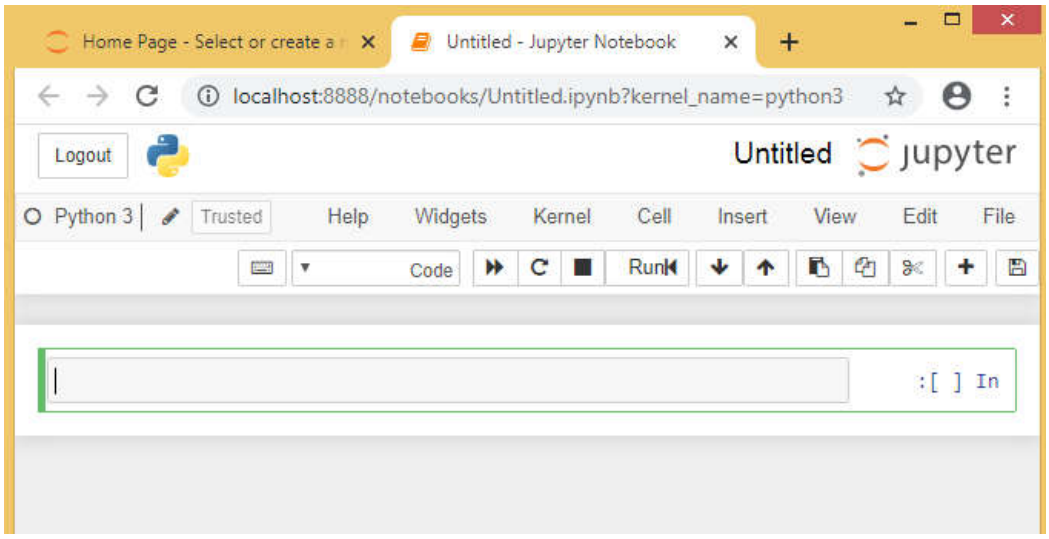
ונקבל מחברת חדשה של jupyter.

מה זה בכלל jupyter ולמה אנחנו צריכים אותו?

תוכנת הjupyter זו פלטפורמת פיתוח מאוד נוחה שמאפשרת לנו להוסיף תאים תאים של קוד, להפעיל כל תא בנפרד ולראות מה עושה קטע הקוד שכתבנו באותו תא.

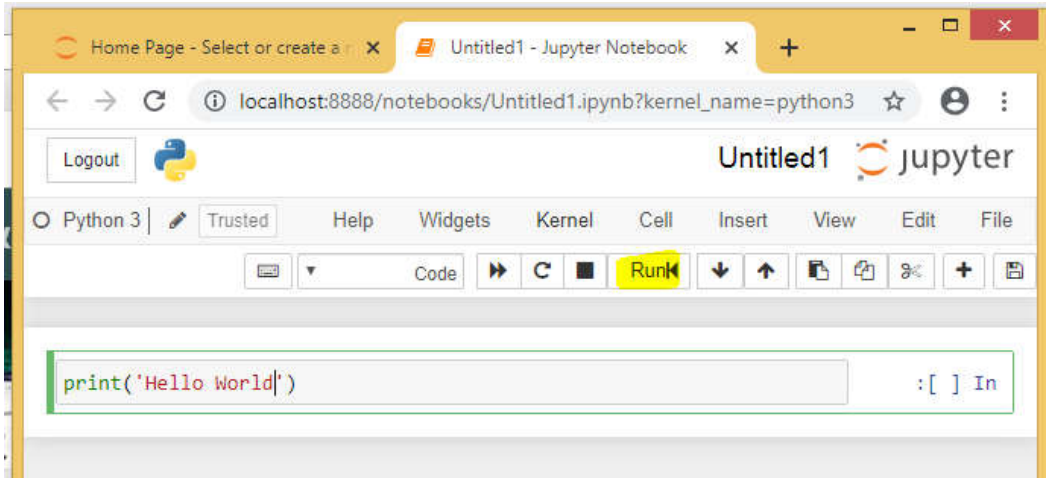
ניתן להוסיף גם תאי מלל ואז בסוף יש לנו תוצר ברור של קטעי קוד שמלווים בהסברים של קטעי טסט שכתבנו.
בנוסף, תמיד נוכל לראות באיזה תא יש לנו טעות ואילו תאים עובדים טוב. כך יהיה לנו יותר קל לזהות בעיות ולתקנן.

אחרי שקיבלנו מחברת חדשה של jupyter.



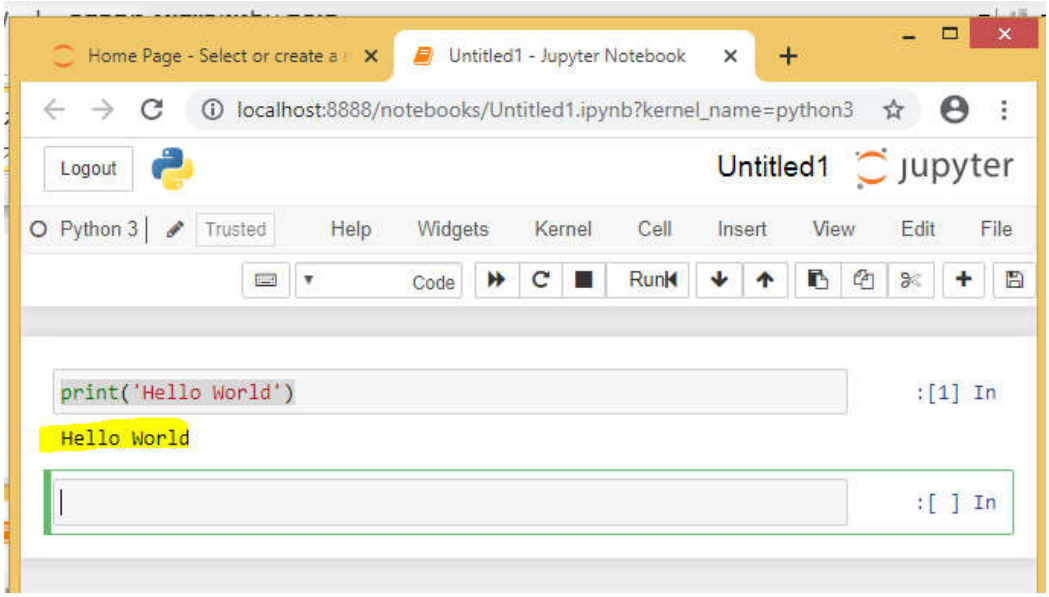
נרשום בתא את הקוד:
print('Hello World')

ונלחץ על האופציה "Run" שבתפריט

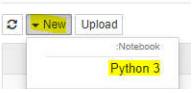


הקוד שרשמנו יתבצע. במקרה שלנו הקוד אומר להדפיס את המילה Hello World.

ומיד גם יפתח לנו חלון קוד חדש.


Home Page - Select or create a
Untitled1 - Jupyter Notebook
localhost:8888/notebooks/Untitled1.ipynb?kernel_name=python3
Logout
Untitled1
jupyter
Python 3
Trusted
Help
Widgets
Kernel
Cell
Insert
View
Edit
File
Code
Run
print('Hello World')
In [1]:
Hello World
In [ ]:

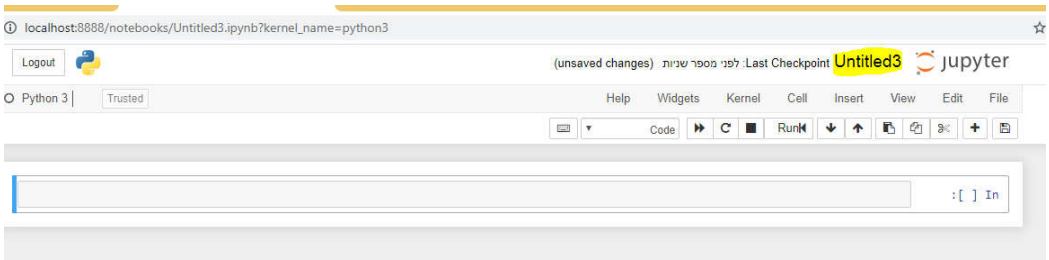
קריאת נתונים מקבצי CSV

נתחיל בפתיחת הjupyter וניצור מחברת חדשה (זוכרים... 

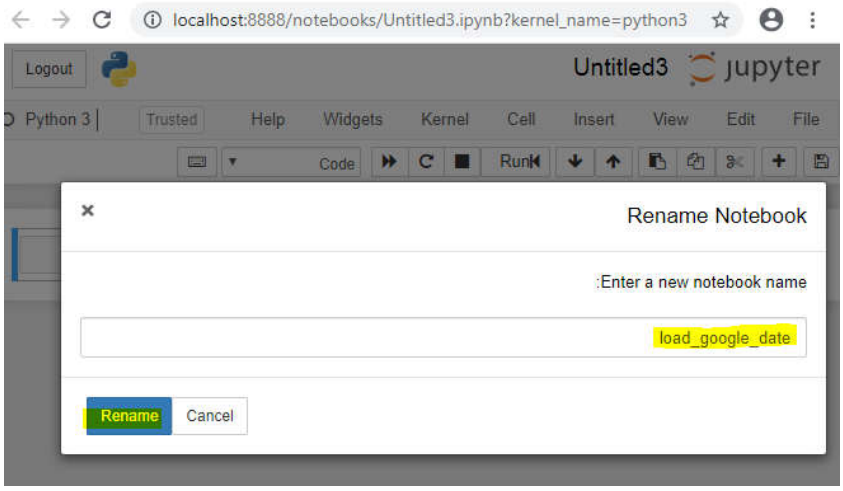
אם לא זוכרים רשום זאת בסיכום הקודם).

תפתח לנו מחברת חדשה:

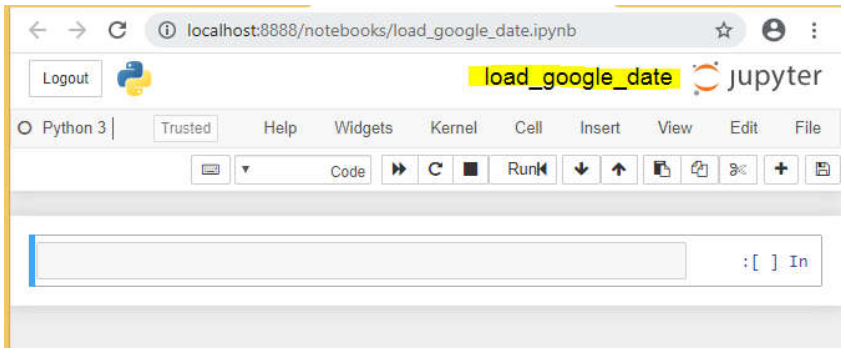
שם המחברת הזו הוא כרגע "Untitled3" .
נלחץ על שם זה ונוכל לשנות אותו לשם שיעיד איזה קוד הולך להירשם במחברת הזו.



נשנה את שם המחברת ל load_google_date ונלחץ על "Rename"



שם המחברת משתנה



פקודת import

בפייתון כמו שאמרנו יש הרבה ספריות שמכילות פונקציות רבות שנכתבו.
ומה שנשאר לנו הוא להפעיל אותן בהתאם לצורך.

כדי שנוכל ליהנות מכל הפונקציות שבספריות השונות ולהפעילן אנחנו צריכים לייבא אותן למחברת שלנו.

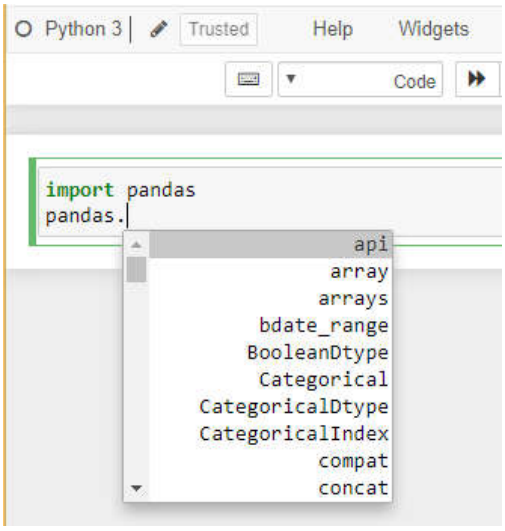
הייבוא הזה נעשה ע"י שימוש בפקודת הimport.
אנחנו קוראים לספרייה על מנת שהפונקציות שלה יהיו זמינות לנו לשימוש במחברת שאנחנו נמצאים בה.

בדוגמא שלנו, אנחנו נייבא את הספרייה pandas.

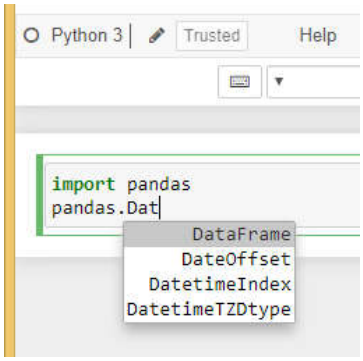
נרשום:
import pandas ואח"כ

בשורה שניה נרשום pandas נקודה ונלחץ על "Tab"

כך נראה את כל הפונקציות שזמינות לנו בספרייה זו.



נחפש פונקציה בשם DataFrame
איך מחפשים? מתחילים לרשום (רגיש לאותיות גדולות או קטנות, אם נרשום זאת עם d קטנה הוא לא ימצא את פונקציה זו) .
רושמים למשל את ההתחלה Dat, לוחצים על tab במקלדת ואז אנחנו יכולים לראות את כל הפונקציות בספריה זו שהשם שלהן מתחיל ב Dat..



נבחר DataFrame

ונקליד מיד אחרי – פתח סוגרים וסגור סוגרים – כלומר DataFrame() (למה? כי זו הדרך להריץ פונקציות - שם ספריה נקודה שם פונקציה, פתח סוגרים וסגור סוגריים).

אפשר ללחוץ על "run" ואז קטע קוד זה רץ.

לקטע קוד הזה אין תוצאה נראית לעין בחלק של "Out" כמו קטע הקוד שפעם הרצנו עם הפונקציה print.

print('Hello world')

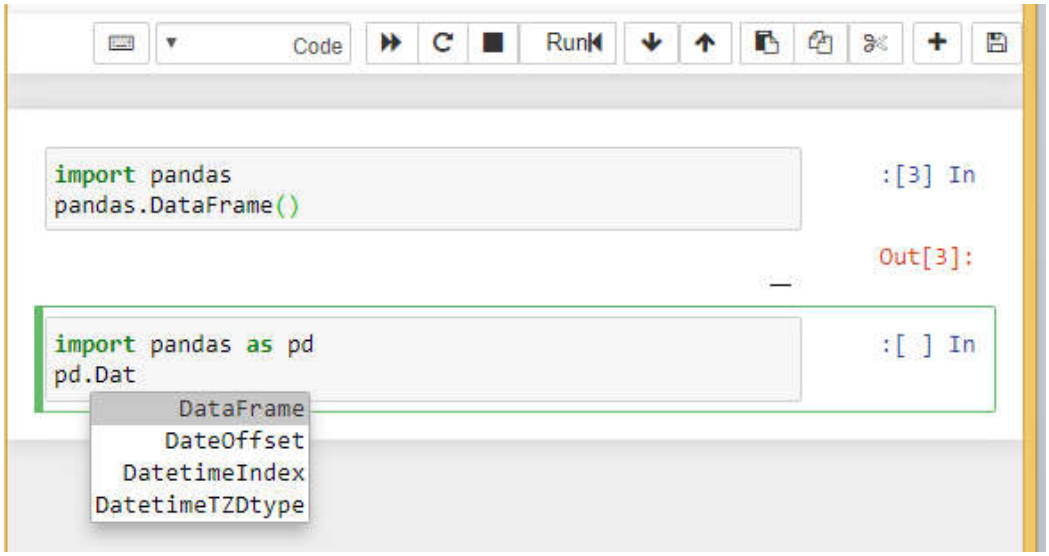
נמשיך עם pandas...

אם נרשום את פקודת ה import ככה:
import pandas as pd

זה כאילו כתבנו – תייבא את pandas ותקרא לה pd.

כאשר אנחנו רוצים לראות את הפונקציות שיש ל pandas או לחפש את הפונקציה DataFrame,

אז מספיק שנרשום pd נקודה ונלחץ על Tab
או
pd נקודה, נתחיל לרשום Dat ונלחץ על Tab כדי שיחפש לנו פונקציה שמתחילה באותיות Dat.



אם אנחנו לא רוצים לייבא את כל הספרייה למחברת שלנו אלא רק חלק מסוים מתוך הספרייה.

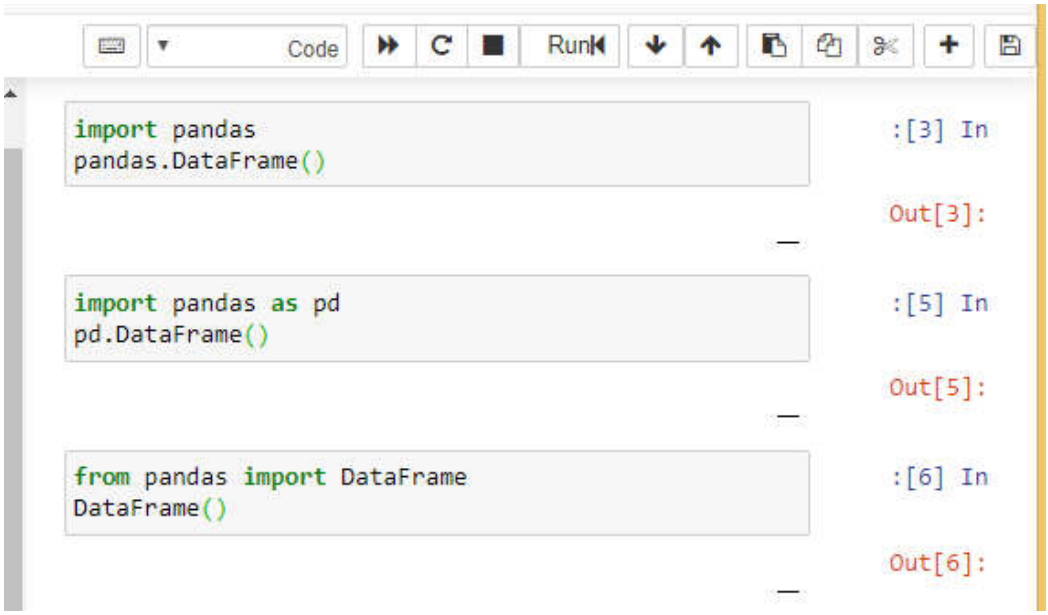
יש לנו דרך לייבא תת ספריה אם נרשום:

from pandas import DataFrame

בפקודה הזו רשמנו: מתוך הספרייה pandas תייבא רק את DataFrame.

גם כאן כאשר רושמים זאת אפשר לרשום Dat וללחוץ על tab והוא יראה לנו רשימה שמתוכה אפשר לבחור את DataFrame במקום להקליד את כל המילה.

עכשיו כשנרצה להפעיל את הפונקציה ()DataFrame כבר לא נצטרך לרשום שם ספריה נקודה שם פונקציה או שם מקוצר לספריה נקודה שם פונקציה כי כבר ייבאנו ישירות את הפונקציה עצמה.
עכשיו אפשר להפעיל אותה ולרשום רק ()DataFrame

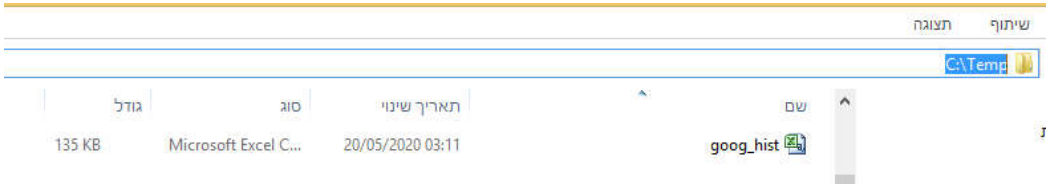


עד עכשיו ראינו דרכים שונות לייבא ספרייה ולהפעיל מתוכה פונקציה.

קריאה מקובץ

אם יש לנו קובץ נתונים שאנחנו רוצים לתחקר שהשם שלו goog_hist.csv.

נשמור אותו למשל ב: c:\Temp (יצרתי את הספרייה Temp תחת כונן c ושמרתי בה את הקובץ)



איזה קטע קוד נרשום על מנת שנוכל להתחיל להשתמש בקובץ זה במחברת שלנו?

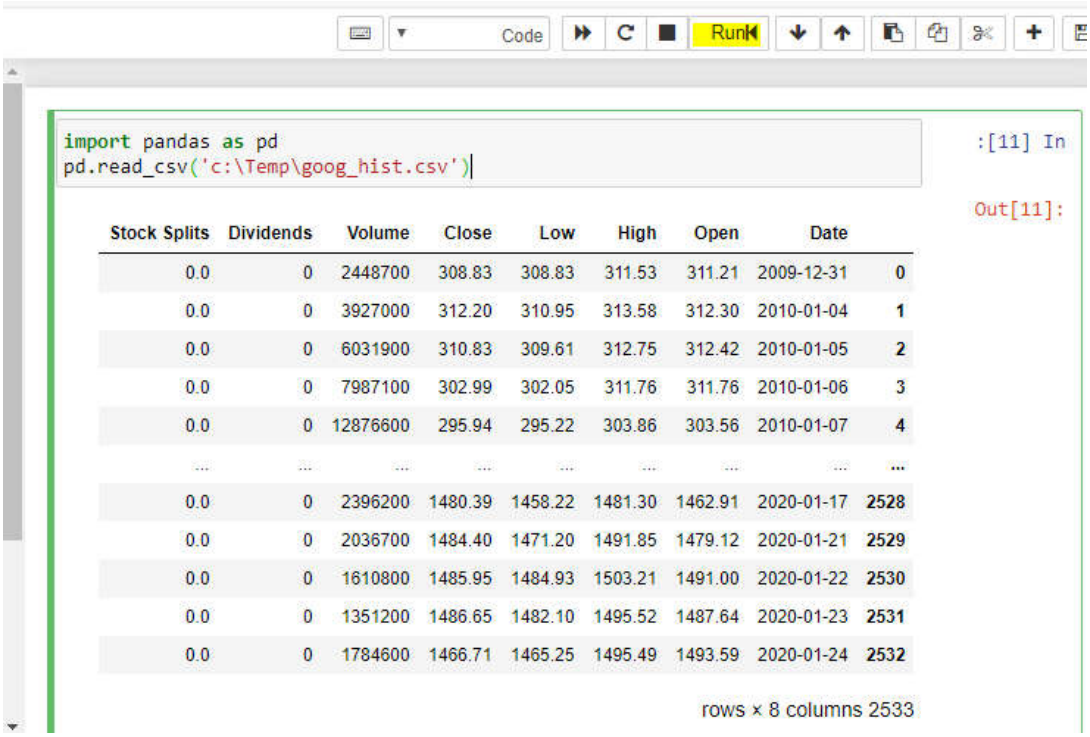
Pandas- ספרייה שמאפשרת לבצע מניפולציה על נתונים לצורכי תחקור.

אז נייבא את הספרייה pandas ונשתמש בפונקציה מתוכה שיודעת לקרא קבצי csv שקוראים לה read_csv.

נרשום:

```
import pandas as pd

pd.read_csv('c:\Temp\goog_hist.csv')
```



	Stock Splits	Dividends	Volume	Close	Low	High	Open	Date
0	0.0	0	2448700	308.83	308.83	311.53	311.21	2009-12-31
1	0.0	0	3927000	312.20	310.95	313.58	312.30	2010-01-04
2	0.0	0	6031900	310.83	309.61	312.75	312.42	2010-01-05
3	0.0	0	7987100	302.99	302.05	311.76	311.76	2010-01-06
4	0.0	0	12876600	295.94	295.22	303.86	303.56	2010-01-07
...	...	...	...	...	...	...	...	...
2528	0.0	0	2396200	1480.39	1458.22	1481.30	1462.91	2020-01-17
2529	0.0	0	2036700	1484.40	1471.20	1491.85	1479.12	2020-01-21
2530	0.0	0	1610800	1485.95	1484.93	1503.21	1491.00	2020-01-22
2531	0.0	0	1351200	1486.65	1482.10	1495.52	1487.64	2020-01-23
2532	0.0	0	1784600	1466.71	1465.25	1495.49	1493.59	2020-01-24

rows × 8 columns 2533

אז מה כתבנו כאן- תפעיל את הפונקציה read_csv מתוך pd, שזה השם המקוצר שנתנו לספרייה pandas ותקרא את הקובץ goog_hist.csv + רשמנו את המיקום שלו.

אז אוקיי הוא קורא מציג לנו כמה שורות ממנו אבל שוכח.

אנחנו רוצים שהוא יאכסן את מה שהוא קורא איפשהו כדי שנוכל ולהמשיך לעשות פעולות על אותו מידע שהוא קרא מהקובץ.

הדרך לאכסן נכתבת ככה:

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

אנחנו מאכסנים את הקובץ בdf יכולנו גם לרשום שם אחר...

אך מכיוון שקבצי csv הם מסגרת נתונים (Data Frame) החלטנו על שם קיצור df אך יכולים להחליט גם על שם אחר.

```
In [12]:
import pandas as pd
df=pd.read_csv('c:\Temp\goog_hist.csv')

In [ ]:
```

אם נרשום df לבד ונריץ אז נראה את השורות הראשונות והאחרונות מאותה מסגרת נתונים data frame שהכנסנו לתוך df

```
In [14]:
import pandas as pd
df = pd.read_csv('c:\Temp\goog_hist.csv')
df
```

Out[14]:

	Stock Splits	Dividends	Volume	Close	Low	High	Open	Date
0	0.0	0	2448700	308.83	308.83	311.53	311.21	2009-12-31
1	0.0	0	3927000	312.20	310.95	313.58	312.30	2010-01-04
2	0.0	0	6031900	310.83	309.61	312.75	312.42	2010-01-05
3	0.0	0	7987100	302.99	302.05	311.76	311.76	2010-01-06
4	0.0	0	12876600	295.94	295.22	303.86	303.56	2010-01-07

אם נרצה לדעת מה הסוג של אותו df שיצרנו אז נרשום את הפונקציה type ונשים בתוך הסוגרים את מה שמעניין אותנו הסוג שלו. במקרה שלנו df.

אחרי שמריצים רואים שהdf שיצרנו קיבל את התכונות של קובץ ה csv ועכשיו הוא גם מסוג Data frame.

```
In [15]:
import pandas as pd
df = pd.read_csv('c:\Temp\goog_hist.csv')
type(df)

Out[15]:
pandas.core.frame.DataFrame
```

הצגת עמודה אחת מתוך Data Frame

אם רוצים להציג רק עמודה אחת מאותו Data Frame נרשום את הפקודה הבאה:

שם הdata frame במקרה שלנו df

אח"כ בסוגרים מרובעים את שם העמודה שרוצים לראות.

```
df
```

:[18] In

Out[18]:

	Stock Splits	Dividends	Volume	Close	Low	High	Open	Date
0	0.0	0	2448700	308.83	308.83	311.53	311.21	2009-12-31

```
df['Date']
```

:[17] In

Out[17]:

```
2009-12-31       0
2010-01-04       1
2010-01-05       2
2010-01-06       3
2010-01-07       4
    ...
2020-01-17    2528
2020-01-21    2529
2020-01-22    2530
2020-01-23    2531
2020-01-24    2532
Name: Date, Length: 2533, dtype: object
```

הצגת שורה אחת מתוך Data Frame

אם רוצים לראות רק את שורה 0

```
df
```

:[20] In

Out[20]:

	Stock Splits	Dividends	Volume	Close	Low	High	Open	Date
0	0.0	0	2448700	308.83	308.83	311.53	311.21	2009-12-31
1	0.0	0	3927000	312.20	310.95	313.58	312.30	2010-01-04
2	0.0	0	6031900	310.83	309.61	312.75	312.42	2010-01-05
3	0.0	0	7987100	302.99	302.05	311.76	311.76	2010-01-06
4	0.0	0	12876600	295.94	295.22	303.86	303.56	2010-01-07

נרשום את שם הdata frame במקרה שלנו df נקודה iloc (פקודה לבחירת שורה) ובתוך סוגריים מרובעים נרשום את מספר השורה שאנחנו רוצים לראות. בדוגמא שלנו שורה 0.

ונראה את כל הערכים של שורה זו עם שמות העמודות.

```
df.iloc[0]
```

:[19] In

Out[19]:

```
Date            2009-12-31
Open                311.21
High                311.53
Low                 308.83
Close               308.83
Volume             2448700
Dividends                0
Stock Splits             0
Name: 0, dtype: object
```

אם אנחנו רוצים שמתוך שורה 0 יציג לנו רק ערך של עמודה אחת.

הצגת ערך של שורה ספציפית ועמודה ספציפית

אנחנו רוצים לראות רק את הערך של שורה 0 ועמודה "Open".

```
df.iloc[0]

Date            2009-12-31
Open                311.21
High                311.53
Low                 308.83
Close               308.83
Volume             2448700
Dividends                0
Stock Splits             0
Name: 0, dtype: object
```

אז נרשום

```
In [22]: df.iloc[0]['Open']
Out[22]: 311.21
```

מה רשמנו: מה data frame שבמקרה שלנו קוראים לו df תציג לי את שורה 0 ומתוכה רק את הערך שבעמודה "Open".

בdata frame הזה יש ערכי מניה לפי יום.

אם רוצים ביום ספציפי (בשורה מסוימת) לראות מה ההפרש בין הערך שכתוב בעמודה "Open"

לערך שכתוב בעמודה "Close" אז נרשום:

df.iloc[0]['Close']-df.iloc[0]['Open']

ונראה שהיתה ירידה של 2.37...בין הפתיחה של המניה לסגירה שלה.

```
In [22]: df.iloc[0]['Open']
Out[22]: 311.21

In [23]: df.iloc[0]['Close']
Out[23]: 308.83

In [24]: df.iloc[0]['Close']-df.iloc[0]['Open']
Out[24]: -2.3799999999999955
```

מה למדנו:

- **לשנות שם למחברת jupyter**
- **פקודת import לספרייה והפעלת פונקציה מתוכה.**
- **טעינת קובץ נתונים csv למחברת שלנו:**

א. אכסון שלו כדי שנוכל לתחקר אותו df=pd.read_csv('c:\Temp\goog_hist.csv')

ב. הצגת השורות הראשונות והאחרונות של הקובץ df

ג. הצגת עמודה אחת מהקובץ df['Date'] (עמודה Date)

ד. הצגת שורה אחת מהקובץ df.iloc[0] (שורה 0)
ה. הצגת ערך של שורה ספציפית ועמודה ספציפית df.iloc[0]['Open'] (הצגת הערך שבעמודה Open ובשורה 0).

הבורסה ומסחר אלגוריתמי

מה הן מניות

מניה- היא סוג של נייר ערך, המאפשר לחברה המנפיקה לגייס הון מציבור המשקיעים בבורסה. בכסף שמקבלת החברה מהציבור היא יכולה לרכוש ציוד, לגייס עובדים ולפתח את מוצריה, החדשים והקיימים, לטובת הרחבת פעילותה.

הנפקת מניות- גיוס כספים על ידי חברה באמצעות מכירת מניות לציבור.

הנפקה ראשונה לציבור initial public offering – **IPO** - היא הנפקה בה חברה פרטית מציעה בפעם הראשונה את מניותיה למכירה לציבור. בדרך כלל עושות זאת חברות אשר זקוקות להון כדי להתרחב, תוך שהן בוחרות לגייס אותו מהציבור.

NYSE- New York Stock Exchang הבורסה לניירות ערך בניו יורק היא הגדולה ביותר בארצות הברית ובעולם בשווי החברות הכולל והשנייה בגודלה במספר החברות הנסחרות בה. כיום, רווח השימוש בשם הרחוב וול סטריט ככינוי לבורסה עצמה.

טבלה שמייצגת את כל החברות שנסחרות בNYSE:
גודל הקובייה הוא ביחס לשווי החברה לעומת שאר החברות.
האותיות שמופיעות על כל קובייה מייצגות את סימול מנית החברה (הסבר למטה).

סימול מניה- Ticker Symbol- לכל מניה הנסחרת בבורסה בארה"ב משויך סימול הנקרא Ticker Symbol. ה-Ticker מורכב ממספר אותיות שמזכירות בדר"כ את שם החברה שהנפיקה את המניות.

ה-Ticker משמש את המשקיעים בכל מקרה בו יש צורך לציין את המניה הספציפית, למשל במתן פקודת קנייה או מכירה או בחיפוש מידע לגבי שער המניה.

מספר דוגמאות לסימולי ה-Ticker של מניות:

שם החברה	סימול ה-Ticker של המניה
KO	The Coca-Cola Company
GE	General Electric
MSFT	Microsoft Corporation
DIS	Walt Disney Company

סימולי ה-Ticker המבוקשים ביותר הם אלו המכילים אות אחת בלבד. מספר דוגמאות לסימולים כאלו הן:

שם החברה	סימול ה-Ticker של המניה
C	Citigroup
F	Ford Motor
G	Gillette
T&At	T

את סימול ה-Ticker של מניה ניתן למצוא ברשת האינטרנט וכן במדריכים היוצאים לאור אחת לשנה ומכילים את הסימולים של כל החברות הנסחרות בארה"ב.

דוגמא לזיהוי לא נכון של סימול מניה:


TheMarker | בעולם

משקיעים התבלבלו - והקפיצו את מניית זום הלא נכונה ב-900%

הפופולריות הגואה של שירותי שיחות הוועידה של זום וידיאו על רקע הסגר שנכפה על מיליארדים בעולם בשל הקורונה, גרם למשקיעים פזיזים שביקשו לגרוף רווח מהיר מהזינוק במנייתה - לקנות בטעות מניות של חברת טלקום בעלת שם דומה

איך נקבע מחיר המניה

מחיר המניות של חברה הנסחרת בבורסה נקבע על פי היצע וביקוש.

המשקיע שמחזיק במניה קובע את המחיר בו הוא מעוניין למכור אותה, הקונה קובע את המחיר שבו הוא מעוניין לקנות את המניה. המפגש של הקונים עם המוכרים בבורסה במחיר מוסכם, קובע את המחיר של המניה.

ההחלטה של המשקיעים שקובעים את מחירי המניות מושפעת בדרך כלל מדוחות החברות השונות, המלצות של אנליסטים ויועצים, אינדיקאטורים מאקרו כלכליים, נתוני החברות והענפים בו הן פועלות ועוד.
לכל סוחר יש רשימת מניות מומלצות שהוא עוקב אחריה ומתעדכן באופן קבוע מה קורה בכל אחת מהחברות האלה.

המחיר בבורסה לא תמיד מייצג את השווי הכלכלי של המניה ואת השווי האמיתי של החברה. אם המניה הייתה משקפת בצורה ישירה את שווי השוק של החברה, זה אומר שהיה צריך להתרחש איזשהו אירוע בחברה כדי שמחיר המניה ישתנה. בפועל, המניה היא מאוד תנודתית – עולה ויורדת

במהלך יום המסחר, בזמן ששום דבר לא באמת קורה בחברה עצמה. מניה יכולה לעלות ב-10% תוך דקות בודדות אך בפועל לא קרה דבר בחברה.

דוגמא לכך שמניות אינן משקפות את השווי שוק של החברה הן הבועות הפיננסיות, כאשר התלהבותם של המשקיעים גורמת למניות בבורסה לטפס למחירי בועה גבוהים מידי.
בדרך כלל אחרי "פיצוץ" בועות שכאלה מתבצע תיקון והמניות מתאזנות לערכן האמתי.

דבר נוסף שיכול לגרום לחוסר איזון במחירים, זה מידע שגוי שהמשקיעים מקבלים מהמקורות שלהם.
לכן כאשר מחפשים מניות מומלצות להשקיע בהן תמיד צריך לבדוק מהו הגורם האמתי לשינוי במחירה של המניה, אם הגורם הוא מהותי כמו שינויים אמתיים שמתרחשים או הולכים להתרחש בחברה או גורם שמועתי בלבד.
דוגמא לשינויים מהותיים: יציאה של מוצר חדש לשוק, גיוס מנכ"ל עם טרק-רקורד מעולה, פיטורים או גיוס של עובדים וכ'.

אין קשר ישיר בין פעילות החברה לבין מחיר המניה בבורסה לאותו הרגע.
בטווחי זמן יותר גדולים (כגון שנים), אז מחיר המניה משקף בצורה יחסית טובה את פעילות החברה ושווי השוק שלה לאורך זמן.

מושגים:

Market Model – מודל תמחור מניה

Ask- האנשים שמציעים למכור את המניה.

Bids- אנשים שרוצים לקנות את המניה. המחיר שהם מציעים הוא תמיד מתחת לAsk אחרת מתבצעת רכישה והם לא היו ברשימת הרוצים לקנות את המניה.

Spread- המרווח בין ההצעה הנמוכה ביותר של המוכרים לבין ההצעה הגבוהה ביותר של הקונים.

מחיר המניה- המפגש של הקונים עם המוכרים בבורסה במחיר מוסכם. המחיר שהתבצעה העסקה בין ה Bids לAsk (בין המוכרים לקונים).

ארביטראז'- הוא ניצול של פער במחיריו של נכס מסוים בשני שווקים או יותר, למטרת רווח.
פעולת ארביטראז' פשוטה עשויה לכלול רכישת נכס בשוק שבו מחירו זול ומכירתו בשוק אחר במחיר גבוה יותר.
פעולות ארביטראז' מורכבות יותר עשויות לכלול פעולות קנייה ומכירה של נכסים אחדים במספר רב של שווקים, המתקזזות האחת עם השנייה ומותירות רווח ביד מבצען.

שורט- מכירה בחסר.
עמדה, עסקה בה מחזיק של נייר ערך מסוים צופה ירידה במחיר ותנועה של השוק כלפי מטה.
היא שיטה לעשיית רווחים כשערכו של נכס הסחיר בשוק ההון נמצא בירידה. בשיטה זו נמכרים ניירות ערך או כספים שאינם בבעלות המוכר במטרה לגריפת רווחים במועד מאוחר יותר, אם ערכם אכן ירד.
המשמעות היא שהסוחר מקבל בהשאלה נייר ערך מבעליו ומוכר אותו בשוק. לאחר זמן מסוים עליו לרכוש אותו מחדש על מנת להחזירו לבעליו. אם בינתיים חלה ירידה בשערו של נייר הערך, הרי שמחיר המכירה גבוה ממחיר הקנייה, והעסקה הייתה רווחית.

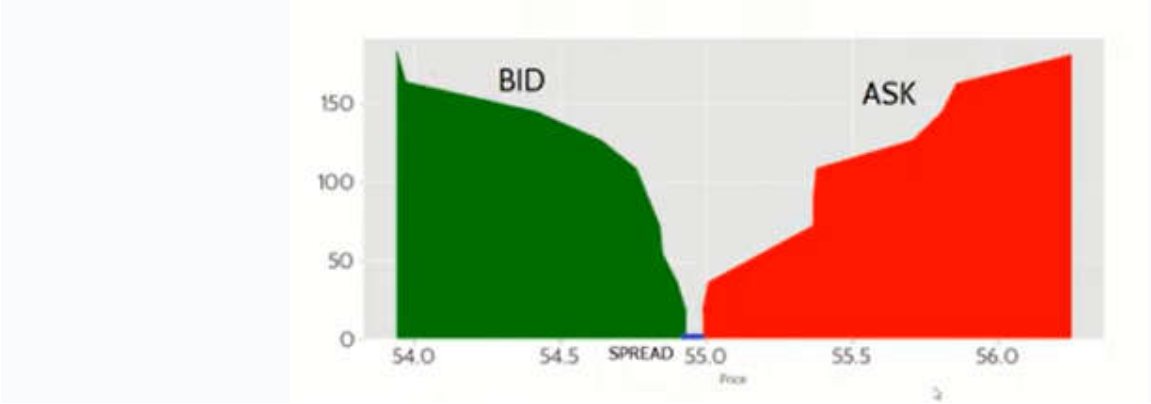
לונג- עמדה, עסקה בה מחזיק של נייר ערך מסוים צופה עליה במחיר ותנועה של השוק כלפי מעלה.

יש לנו את כל האנשים שרוצים לקנות וכל אלו שרוצים למכור.

ניתן לראות את המדרגות המחיר והכמות שמאפיינות את הקונים והמוכרים.
ה-Spread- יכול ללמד אותנו כיצד מעריכים השחקנים בשוק, הBids וה-Ask, את המניה.

אם היינו רוצים למכור במחיר 56 כמות עצומה של מניות (ראו צילום מסך למטה). היה נוצר רושם שיש הרבה אנשים שרוצים למכור ויכול להטעות את האלגוריתמים.

מה שחשוב באיזה מחיר התבצעה העסקה.



מה זה מסחר אלגוריתמי

מסחר אלגוריתמי- ידוע גם כאַלְגו-טריידינג או (Black Box trading) הוא מסחר בבורסה מבוסס תוכנת מחשב ,אשר שולחת פקודות מסחר שנוצרות באמצעות אלגוריתם ממוחשב. אלגוריתם זה יוזם הוראות קנייה ומכירה בהתבסס על פרמטרים של תזמון, מחיר, סחירות ועוד. הוראות אלה מועברות לביצוע ללא הפעלת שיקול דעת נוסף או התערבות אדם.

מקובל לחשוב שמסחר אלגוריתמי תרם רבות להגדלת הנזילות (הקטנת המרווח בין הקונה למוכר Spread) בשווקים בשנים האחרונות, מה שמקטין עלויות מסחר עבור עסקאות קטנות.

באלגו-טריידינג הסיכון לא נובע מפעילות על פי דחפים ורגשות, אלא הוא סיכון שקשור לתוכנה/ למוח, האם הנוסחאות שמשמשות את המסחר הן נכונות, כדי להפיק תשואה בשוק ההון. הפתרון כאן הוא ביכולת לתרגם את הנוסחה ולהפעיל אותה על עשרות שווקים ומצבים שונים בעבר ולראות אם אכן

היא ידעה לייצר כסף. לא כל מה שהיה הוא שיהיה. יכול להיות שנוסחה שעבדה לא תעבוד או תעבוד פחות טוב בעתיד, אבל האלגו מאפשר במקרים רבים למידה תוך כדי תנועה - הוא לומד אם השיטה מתאימה ואם לא, הוא יודע לעצור ולעבור לשיטה אחרת.

כל בית השקעות/ גוף פיננסי/ משקיע יבקש את המערכת מסחר המתאימה לו.
כפי שהמכונות ייצור החליפו את פועלי הייצור, מכונות האלגו יחליפו את מנהלי ההשקעות - כמובן שלא את כולם. יותר ויותר אנשים יבינו שהניהול מנוטרל הרגשות שעובד פשוט לפי נוסחאות, עובד ומייצר תשואות.

אלא שלא הכל וורוד באלגו-טריידינג – מאחר שבני האדם הם אלו ששולטים במכונות יש מקום למניפולציות ותנועות מכוונות למטרות רווח. למשל מתן הוראות קניה ומכירה כוזבות שברגע האחרון מתבטלות. החברות ששולטות במסחר האלגו בעיקר בארה"ב עלולות לעשות מניפולציות במניות בעזרת האלגו. בחלק מהמדינות בעולם חל איסור בשימוש באלגו, אבל, כך או כך קשה לאכוף שימוש באלגו וגם אם קיימת החמרה או רגולציה בנושא כמעט ולא ניתן לעקוב אחר המסחר באלגו.

סוגים של מסחר אלגוריתמי

מסחר אלגוריתמי מתחלק ל 3 סוגים עיקריים:

1) Sell side
2) Buy side
3) HFT

Sell side- בעיקר אלגוריתמים של קרנות הון סיכון ושחקנים גדולים בשוק. שמטרתם לממש קניה ומכירה של מניות בכמות גדולה מבלי שמחיר המניה יזוז.

יש כל מיני אלגוריתמים שמאפשרים למכור מניות מבלי לגלות לשוק שאנחנו מוכרים כמות גדולה של מניות, כדי שמחיר המניה לא יזוז (וכך לא יפסידו בכך הרבה כסף).
יש כל מיני אסטרטגיות שניתן לנסות לבצע על מנת להשפיע על מחיר המניה.

Buy side- יכולים לעשות גם קניה של מניה, גם מכירה של מניה, גם שורט וגם לונג.
לרוב הם מתחילים בלי מניות או בלי הרבה מניות והמטרה שלהם הוא להרוויח מהשינויים של מניה. ניבוי עתידי של מחיר מניה וקביעת אסטרטגית קניה ומכירה. הרעיון לקנות מניות ולמכור אותן בטווחי זמן קצרים.

HFT (high frequency trading) - "מסחר בתדירות גבוהה" שהוא לעיתים רווחי ביותר כשהתנודתיות בשוק גבוהה. בד"כ מדובר במכירה וקניה בקצבים מאוד מהירים של ננו שניות.
מאוד קשה להיכנס אליו כי חייבים מערכות מסחר מאוד מהירות.

בשוק קיימים סוגים רבים של אלגוריתמים שמתוכנתים אחרת. כלומר, כל חברה והאלגוריתם שלה. מזהה דברים אחרים, פועלת בהתניות אחרות. מדובר במעין שרשרת ארוכה של פקודות שונות שמבקשות לאתר ניירות שונים מפלטפורמות מסחר שונות.

מכונות האלגו תופסות נתח הולך וגדל במסחר העולמי ובעיקר במסחר בוולסטריט.
חלק מהטריק של מכונות האלגו זה הסוואת המסחר בהן. בארה"ב יש הערכה כי יותר מ-75% מ-8 מיליארד הפעולות שמתרחשות מדי יום בשוק האופציות, מתבצעות ע"י מחשבי אלגו. הדרך לעשות זאת בהסוואה היא לפרק את פעולות המסחר לאלפי פעולות קטנות, ולגרום לשוק לפעול כאילו סוחרים קטנים מצויים בו וסוחרים בו. כך לשדר שלא מדובר בגוף אחד אלא באלפים רבים שלדוגמה מוכרים עכשיו מניות נפט. ההסוואה היא לטשטש את העקבות וליצור בשוק מומנטום אמיתי. אך יש בזה כמובן סוג של הטעייה שהיא לפעמים על גבול ההונאה, לכך נדרשות עכשיו רשויות ניירות הערך

בעולם.

האלגו הוא חלק מרכזי מהמסחר, אבל הרגולציה עליו ברוב המקומות עדיין לא גובשה באופן מוחלט. מסחר אנושי לא יכול להתמודד מול הטכנולוגיה של מסחר אוטומטי - קבלת ההחלטות האנושית לא מהירה כמו באלגו, עיבוד הנתונים לצורך קבלת ההחלטות איטי כמובן מהאלגו, הביצוע של האלגו מהיר יותר, האלגו משתלט על כמה "זירות" מסחר, בעוד שבן אדם, לא יכול לעשות כמה פעולות ביצוע במקביל. **בקיצור, אם אי אפשר להתחרות באלגו-טריידינג , צריך להצטרף אליו.**

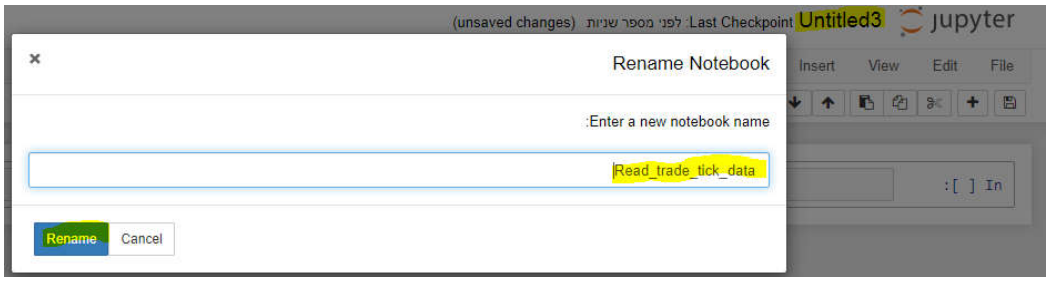
האם ניתן לעבוד במערכות אלגו-טריידינג עם הבנקים בישראל?
לא (לפחות בינתיים). הסיבה שלא עובדים מול הבנקים הישראלים נובעת בעיקר מרגולציה. הבנקים בישראל פועלים במערכות מסחר סגורות שאינן יכולות/מסוגלות/רוצות להיפתח מול גורמי מסחר חיצוניים.

קריאת נתונים מקובץ טיקים

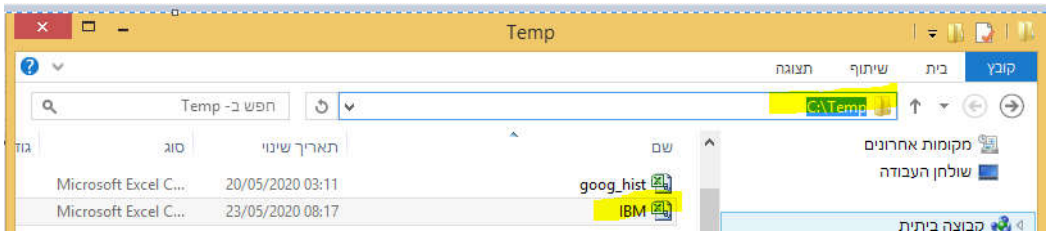
מפעילים jupyter notebook ופותחים מחברת חדשה.

לא זוכרים איך... אז מציצים ב"סיכום חלק 2....".

נשנה את שם המחברת ל Read_trade_tick_data



נשמור את הקובץ IBM במחשב שלנו



Tick = פעולה של מסחר שקרתה בין מוכר לקונה.

נייבא את הספרייה של Pandas ונקרא את קובץ IBM.csv

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

:[3] In

Out[3]:

	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
...	...	...	...	...	...	...	...

איך שרשמנו כרגע את הפקודה הוא מראה לנו את תוכן הקובץ אבל הוא רץ ושוכח מהמידע שראה. כדי שנוכל להשתמש במידע שיש בתוך קובץ הcsv אנחנו צריכים להכניס אותו לתוך "משתנה" ואז נוכל לשחק עם המידע ע"י מניפולציות על אותו משתנה (לדוגמא: שיראה לנו רק עמודה, שיראה רק שורה, שיראה רק ערך שבשורה ועמודה מסוימת וכ').

משתנה (Variable) הוא חלק זיכרון בתוכנית המכיל נתון שיכול להשתנות בזמן הריצה, על פי הפקודות הניתנות לו. בנוסף לפעולת ההשמה ניתן לבצע גם פעולה של שליפת המידע שבמשתנה.

השמה פקודת השמה היא פקודה המציבה ערך חדש במשתנה. פקודה זו קיימת, כפקודה בסיסית, במרבית שפות התכנות. הסימן הנפוץ לפקודת השמה הוא =.

נבצע השמה לתוך המשתנה df שבו נשים את תוכן ה csv

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

הקובץ מכיל תיעוד של עסקות מכירה בזמן מדויק (דקות, שניות, מילי שניות).

אם אנחנו רוצים לראות **שורה ספציפית מתוך הטבלה** אז נשתמש בפקודה iloc

בפקודה זו רשמנו שיציג לנו את שורה 0 בטבלה.

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

```
df.iloc[0]
```

```
Timestamp     07:12:38.577
EventType            TRADE
Ticker                 IBM
Price               155.61
Quantity                 1
Exchange              ARCA
Conditions        80002000
Name: 0, dtype: object
```

אם רוצים לראות **טווח של שורות** (משורה עד שורה) אז רושמים:

df.iloc[0:5]

בפקודה הזו אנחנו מבקשים לראות את שורה 0 עד 5 לא כולל שורה 5.

```
df.iloc[0:5]
```

In [6]:

Out[6]:

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
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377	3
00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	4

המטרה שלנו היא שהערכים בעמודה Timestamp יהיו בשניות כך שלא נקבל פירוט כל כך רב ברמת מילי שניות. אנחנו רוצים לקבץ/לסכם את המידע ברמת שניה.

הבעיה: הערכים בעמודה זו הם בכלל לא מסוג מספר אלא מסוג טקסט (String).

בpandas יש פונקציה שמאפשרת להמיר שדה שהוא טקסט string לסוג של תאריך ושעה date time.

כשאנחנו בודקים מה סוג העמודה Timestamp אנחנו רואים שהיא מסוג Series.

```
type(df['Timestamp'])
```

In [9]:

```
pandas.core.series.Series
```

Out[9]:

Data Frame

מידע שמסודר בעמודות ושורות. בדומה לגיליון אקסל- דו מיימדי.

Series - עמודה אחת ב-Data Frame- חד מיימדי.

אין אנחנו יודעים שהערכים בתוך העמודה (Series) Timestamp הם מסוג טקסט (string) ולא מסוג מספר או תאריך ושעה?

זוכרים את הפקודה שמציגה לנו ערך שנמצא בעמודה ושורה ספציפיים?

בפקודה הזו רשמנו שיציג לנו את הערך שבשורה 0 בעמודה Timestamp

```
In [10]: df.iloc[0]['Timestamp']
Out[10]: '07:12:38.577'
```

אם על כל זה נפעיל את פקודה type אז נראה מה הסוג של הערך.

קיבלנו: Str, כלומר string, כלומר ערך מסוג טקסט.

```
In [11]: type(df.iloc[0]['Timestamp'])
Out[11]: str
```

איך משנים את הערכים בעמודה כך שיהיו מסוג תאריך ושעה ולא String טקסט?

בספרייה pandas יש פונקציה שממירה ערכי עמודה לסוג date time.

הפונקציה היא **to_datetime()**

איך כתבנו את הפקודה:

רשמנו לשים במשתנה a את עמודה Timestamp שנמצאת במשתנה df שיצרנו קודם.

אבל אחרי שמפעילים עליה את הפונקציה to_datetime() שנמצאת בתוך הספרייה pandas שקראנו לה בקיצור pd.

מה קיבלנו: בתוך המשתנה a עמודה בשם Timestamp שהערכים שלה הפכו להיות מסוג datetime.

אמורה

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

```
a

07:12:38.577 2020-05-23       0
07:48:52.137 2020-05-23       1
07:49:41.075 2020-05-23       2
07:49:42.377 2020-05-23       3
07:49:42.571 2020-05-23       4
                ...
18:27:09.391 2020-05-23   36191
18:53:28.906 2020-05-23   36192
19:28:11.889 2020-05-23   36193
19:31:30.532 2020-05-23   36194
19:58:30.644 2020-05-23   36195
Name: Timestamp, Length: 36196, dtype: datetime64[ns]
```

איך מוסיפים עמודה לתוך טבלה

איך מוסיפים את העמודה מהסוג החדש לתוך הטבלה שלנו ששמורה במשתנה df?

תשובה: במקום לעשות השמה לa נעשה השמה לעמודה חדשה ב df שלנו שמכיל את כל הטבלה.

הפקודות זהות והשוני הוא: לאן מוסיפים את העמודה מהסוג המתוקן. בשורה ראשונה הוספנו אותה למשתנה a ובשורה השנייה הוספנו אותה לעמודה בשם 'DateTime'. העמודה הזו לא היתה קיימת במשתנה שלנו df אז אם עושים השמה לעמודה שלא שקיימת היא מתווספת לבד.

אם היינו עושים השמה לשם עמודה קיים אז הפקודה היתה דורסת את הנתונים שבעמודה הקיימת ושמה במקומם את הנתונים החדשים.

```
In [12]: a=pd.to_datetime(df['Timestamp'])
In [14]: df['DateTime']=pd.to_datetime(df['Timestamp'])
```

נבדוק אם זה עבד:

נרשום df ונריץ

```
In [12]: a=pd.to_datetime(df['Timestamp'])
In [59]: df['DateTime']=pd.to_datetime(df['Timestamp'])
In [15]: df
Out[15]:
```

	DateTime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp
0	07:12:38.577 2020-05-23	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577
1	07:48:52.137 2020-05-23	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137
2	07:49:41.075 2020-05-23	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075
3	07:49:42.377 2020-05-23	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377
4	07:49:42.571 2020-05-23	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571
...	...	...	...	...	...	...	...	...

רואים שנוספה לנו עמודה חדשה בשם DateTime וגם נוסף לנו תאריך של נוכחי (של הזמן שבו הרצנו את הפקודה) שזה כבר מידע לא נכון וגם לא שימושי כי הקובץ מכיל מסחר של יום אחד בלבד שלא בוצע היום.

אם אנחנו רוצים לחשב כמה זמן עבר משעת פתיחת המסחר בכל שורה

אנחנו צריכים לדעת את הזמן המינימלי – הזמן הראשון שבו בוצעה מכירה/קניה.

ואח"כ מכל ערך בעמודה של DateTime להחסיר את זמן זה.

בכל שורה נראה אחרי כמה זמן מרגע תנועת המסחר הראשונה בוצעה תנועת המסחר שבשורה שאנחנו מסתכלים עליה.

איך כותבים זאת:

עמודת "DateTime" פחות הזמן המינימאלי שבו בוצעה מכירה/קניה.

הזמן המינימלי שבו בוצע מכירה/קנייה

```
df['DateTime'].min()
```
In [16]:

```
Timestamp('2020-05-23 07:12:38.577000')
```
Out[16]:

כמה זמן עבר מזמן תנועת המסחר הראשונה (מאותו זמן מינימלי):

```
df['DateTime'].min()
```
In [16]:

```
Timestamp('2020-05-23 07:12:38.577000')
```
Out[16]:

```
df['DateTime']-df['DateTime'].min()
```
In [17]:

Out[17]:

```
00:00:00              0
00:36:13.560000       1
00:37:02.498000       2
00:37:03.800000       3
00:37:03.994000       4
        ...
11:14:30.814000   36191
11:40:50.329000   36192
12:15:33.312000   36193
```

נוסיף עמודה עם מידע זה:

נקרא לעמודה "TimeDelta"

```
df['TimeDelta']=df['DateTime']-df['DateTime'].min()
```
In [18]:

```
df
```
In [19]:

Out[19]:

TimeDelta	DateTime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
00:00:00	07:12:38.577 2020-05-23	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	0
00:36:13.560000	07:48:52.137 2020-05-23	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	1
00:37:02.498000	07:49:41.075 2020-05-23	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	2
00:37:03.800000	07:49:42.377 2020-05-23	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377	3
00:37:03.994000	07:49:42.571 2020-05-23	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	4

המרת העמודה TimeDelta כך שתציג את הזמן שעבר רק בשניות ולא רק במילי שניות

לצורך פעולה זו יש פונקציה שנקראת **total_seconds()**

איך מפעילים אותה: רושמים שם עמודה נקודה dt נקודה total_seconds()

```
df['TimeDelta'].dt.total_seconds()
```

```
0.000            0
2173.560         1
2222.498         2
2223.800         3
2223.994         4
    ...
40470.814    36191
42050.329    36192
44133.312    36193
44331.955    36194
45952.067    36195
Name: TimeDelta, Length: 36196, dtype: float64
```

נוסיף גם את עמודה זו לטבלה שלנו (df):

```
df['TotalSeconds']=df['TimeDelta'].dt.total_seconds()
```

:[66] In

```
df
```

:[67] In

Out[67]:

TotalSeconds	TimeDelta	DateTime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
0.000	00:00:00	07:12:38.577 2020-05-24	80002000	ARCA	1	155.61	IBM	TRADE	07:12:38.577	0
2173.560	00:36:13.560000	07:48:52.137 2020-05-24	20002020	ARCA	200	154.49	IBM	TRADE	07:48:52.137	1
2222.498	00:37:02.498000	07:49:41.075 2020-05-24	20002020	ARCA	500	154.50	IBM	TRADE	07:49:41.075	2
2223.800	00:37:03.800000	07:49:42.377 2020-05-24	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.377	3
2223.994	00:37:03.994000	07:49:42.571 2020-05-24	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.571	4

איך מאחדים את השניות כך שלא יופיעו כמספר עשרוני אלא כמספר שלם

תשובה: ממירים את הערכים כך שיהיו מסוג של מספר שלם Integer (int) ולא סוג עשרוני (float)

כרגע הערכים בעמודה זו הם מסוג float

```
df['TotalSeconds']
```

```
0.000            0
2173.560         1
2222.498         2
2223.800         3
2223.994         4
    ...
40470.814    36191
42050.329    36192
44133.312    36193
44331.955    36194
45952.067    36195
Name: TotalSeconds, Length: 36196, dtype: float64
```

דוגמאות איך נראים ערכים מסוגים שונים:

String	integer	float
"two"	0	3.0
'd"d'	223	0.0
'123'	-10	1.234

מפעילים את הפונקציה (int)satype ורושמים לה להמיר לסוג int (ניתן להמיר גם לסוגים אחרים).

```
df['TotalSeconds']
```

```
0.000            0
2173.560         1
2222.498         2
2223.800         3
2223.994         4
     ...
40470.814    36191
42050.329    36192
44133.312    36193
44331.955    36194
45952.067    36195
Name: TotalSeconds, Length: 36196, dtype: float64
```

```
df['TotalSeconds'].astype(int)
```

```
0            0
2173         1
2222         2
2223         3
2223         4
  ...
40470    36191
42050    36192
44133    36193
44331    36194
45952    36195
Name: TotalSeconds, Length: 36196, dtype: int32
```

ניצור עמודה חדשה בטבלה שלנו df עם עמודת int ים זו שנקרא לה "TotalSeconds_Int"

```
df['TotalSeconds'].astype(int)
```

```
0            0
2173         1
2222         2
2223         3
2223         4
  ...
40470    36191
42050    36192
44133    36193
44331    36194
45952    36195
Name: TotalSeconds, Length: 36196, dtype: int32
```

```
df['TotalSeconds_Int']=df['TotalSeconds'].astype(int)
```

איך מסננים מידע מטבלה df כך שנראה רק שורות שיש להן ערך מסוים בעמודה מסוימת

df[df['TotalSeconds_Int']==2223]

```
df[df['TotalSeconds_Int']==2223]
```

:[72] In

Out[72]:

TotalSeconds_Int	TotalSeconds	TimeDelta	DateTime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
2223	2223.800	00:37:03.800000	07:49:42.377 2020-05-24	00002000	ARCA	200	154.5	IBM	TRADE	07:49:42.377	3
2223	2223.994	00:37:03.994000	07:49:42.571 2020-05-24	00002000	ARCA	200	154.5	IBM	TRADE	07:49:42.571	4

מה שאלנו בחלק הפנימי של פקודה זו:

df['TotalSeconds_Int']==2223

האם הערכים בעמודה TotalSeconds_Int שווים ל 2223 (הסימן == זה כמו לשאול האם שווה)

אם נריץ זאת נקבל את התשובות true או false בכל שורה. נקבל true בשורות שהערכים בהם שווים ל 2223.

```
df['TotalSeconds_Int']==2223
```

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

אך ברגע שסוגרים את כל הפקודה הזו בסוגריים מרובעים ולפניהם רושמים את שם הטבלה שלנו שבמקרה שלנו היא df. אז מתבצע סינון ונקבל את כל השורות בטבלה שהערכים בעמודה מסוימת שווים למה שביקשנו.

המטרה שלנו בסוף היא לקבל שורה אחת עם נתונים פר שניה.

מה נעשה אם בשנייה מסוימת יש כמה עסקאות ולכל עסקה מחיר שונה...

איך נאחד אותן כך שיוצגו בשורה אחת?

תשובה: נמיר את הנתונים כך שיציגו 5 מדדים שיגרמו לאותה שניה של מסחר להכיל את כל הנתונים שהיו מפוצלים לנו בכמה שורות.

המדדים:

1) Open- המחיר הראשון של המניה בשנייה מסוימת.
2) Close- המחיר האחרון של המניה בשנייה מסוימת.
3) High- המחיר הכי גבוה שהמניה הגיעה אליו במהלך השנייה.
4) Low- המחיר הכי נמוך שהמניה הגיעה אליו במהלך השנייה.
5) Volume- כמות המניות שנסחרו באותה שניה.

אם נסנן את המידע לפי שנייה 30139 נקבל המון עסקות שהתבצעו בשנייה זו במחירים שונים ובכמויות שונות.

```
df[df['TotalSeconds_Int']==30139]
```

TotalSeconds_Int	TotalSeconds	TimeDelta	DateTime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
30139	30139.430	08:22:19.430000	15:34:58.007 2020-05-24	00000001	FINRA	100	152.00	IBM	TRADE NB	15:34:58.007	**28314**
30139	30139.431	08:22:19.431000	15:34:58.008 2020-05-24	00000001	FINRA	100	152.00	IBM	TRADE NB	15:34:58.008	**28315**
30139	30139.445	08:22:19.445000	15:34:58.022 2020-05-24	80000000	FINRA	30	152.00	IBM	TRADE	15:34:58.022	**28316**
30139	30139.446	08:22:19.446000	15:34:58.023 2020-05-24	a0000020	BATS	30	152.00	IBM	TRADE	15:34:58.023	**28317**
30139	30139.446	08:22:19.446000	15:34:58.023 2020-05-24	a0000020	BATS	6	152.00	IBM	TRADE	15:34:58.023	**28318**
...	...	...	...	...	...	...	...	...	...	...	**...**
30139	30139.926	08:22:19.926000	15:34:58.503 2020-05-24	a0000020	NASDAQ	53	152.04	IBM	TRADE	15:34:58.503	**28435**
30139	30139.926	08:22:19.926000	15:34:58.503 2020-05-24	20000020	NASDAQ	100	152.04	IBM	TRADE NB	15:34:58.503	**28436**
30139	30139.926	08:22:19.926000	15:34:58.503 2020-05-24	a0000020	NASDAQ	47	152.04	IBM	TRADE	15:34:58.503	**28437**

איך מחלצים מנתונים משנייה זו (30139) את כל 5 המדדים:

Open- המחיר הראשון של המניה בשנייה מסוימת- ערך [0]

למה המחיר הראשון של המניה בשנייה מסוימת הוא המחר בשורה הראשונה שלו שהיא שורה 0 כי הנתונים ממוינים לפי זמן.

```
Open=df[df['TotalSeconds_Int']==30139]['Price'].iloc[0]
```

```
Open
```

```
152.0
```

Close- המחיר האחרון של המניה בשנייה מסוימת- ערך [-1]

ערך מינוס 1 זה הערך האחרון בעמודה, בSeries

אם למשל רוצים את הערך האחד לפני האחרון אז נרשום מינוס 2.

```
Close=df[df['TotalSeconds_Int']==30139]['Price'].iloc[-1]
```

```
Close
```

```
152.05
```

High- המחיר הכי גבוה שהמניה הגיעה אליו במהלך השנייה – max

```
High=df[df['TotalSeconds_Int']==30139]['Price'].max()
```

```
High
```

```
152.05
```

Low- המחיר הכי נמוך שהמניה הגיעה אליו במהלך השנייה – min

```
Low=df[df['TotalSeconds_Int']==30139]['Price'].min()
```

```
Low
```

```
151.98
```

Volume- כמות המניות שנסחרו באותה שניה – sum

```
Vol=df[df['TotalSeconds_Int']==30139]['Quantity'].sum()
```

```
Vol
```

```
25760
```

עד עכשיו ביצענו את 5 המדדים רק לשנייה 30139.

איך מבצעים אותם לכל הטבלה df?

נרשום את הפקודות כרגע ללא הסברים. ההסברים יהיו בקבצים הבאים.

ניצור משתנה שמכיל טבלה חדשה שהיא תהיה הטבלה שמסכמת את כל הנתונים שבטבלה df פר שנייה עם 5 המדדים.

קראנו לטבלה החדשה new_df והיא כרגע ריקה.

In [99]:

```
new_df=pd.DataFrame()
new_df
```

Out[99]:

נתחיל להוסיף לה עמודות לפי 5 המדדים:

In [101]:
```
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())
```

In [102]:
```
new_df
```

Out[102]:

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
400	154.50	154.50	154.50	154.50	2223
1200	154.50	154.50	154.50	154.50	2224
...	...	...	...	...	...
141	151.55	151.55	151.55	151.55	40470

מה למדנו?

1) **איך רואים טווח של שורות**

df.iloc[0:5]

פקודה זו תציג לנו את שורות 0 עד 5 אבל לא כולל 5.

2) **איך משנים את הערכים בעמודה כך שיהיו מסוג תאריך ושעה ולא String טקסט**
משתמשים בפונקציה to_datetime()

צאוזה

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

```
a
```

```
07:12:38.577 2020-05-23    0
07:48:52.137 2020-05-23    1
07:49:41.075 2020-05-23    2
07:49:42.377 2020-05-23    3
```

3) **איך מוסיפים את עמודה חדשה לתוך הטבלה שלנו ששמורה במשתנה df?**
רושמים את שם העמודה החדש ושמים בו את התוכן שרוצים והעמודה החדשה מתווספת לטבלה. אם היינו רושמים שם עמודה קיים אז אותה עמודה היתה נדרסת עם התוכן החדש שרשמנו.

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

שם צאוזה תוכן

4) **אם אנחנו רוצים לחשב כמה זמן עבר משעת פתיחת המסחר בכל שורה**
שעת מסחר פחות שעת מסחר מינימלית.

```
df['DateTime'].min()
```

:[16] In

```
Timestamp('2020-05-23 07:12:38.577000')
```

Out[16]:

```
df['DateTime']-df['DateTime'].min()
```

:[17] In

Out[17]:

```
00:00:00              0
00:36:13.560000       1
00:37:02.498000       2
00:37:03.800000       3
00:37:03.994000       4
       ...
11:14:30.814000   36191
11:40:50.329000   36192
12:15:33.312000   36193
```

5) **המרת זמן למשל במילי שניות לשניות**

רושמים שם עמודה נקודה dt נקודה total_seconds()

```
df['TimeDelta'].dt.total_seconds()
```

```
0.000            0
2173.560         1
2222.498         2
2223.800         3
2223.994         4
    ...
40470.814    36191
42050.329    36192
44133.312    36193
44331.955    36194
45952.067    36195
Name: TimeDelta, Length: 36196, dtype: float64
```

6) **איך מאחדים את השניות כך שלא יופיעו כמספר עשרוני אלא כמספר שלם-** astype(int)

ממירים את הערכים כך שיהיו מסוג של מספר שלם Integer (int) ולא סוג עשרוני (float).

```
df['TotalSeconds'].astype(int)
```

```
0            0
2173         1
2222         2
2223         3
2223         4
```

7) **איך מסננים מידע מטבלה df כל שיראה לנו רק שורות שיש להן ערך מסוים בעמודה מסוימת.**

```
df[df['TotalSeconds_Int']==2223]
```

:[72] In

Out[72]:

	TotalSeconds_Int	TotalSeconds	TimeDelta	DateTime	Conditions	Exchange	Quantity	Price	Ticker	EventType	Timestamp	
	2223	2223.800	00:37:03.800000	07:49:42.377 2020-05-24	00002000	ARCA	200	154.5	IBM	TRADE	07:49:42.377	3
	2223	2223.994	00:37:03.994000	07:49:42.571 2020-05-24	00002000	ARCA	200	154.5	IBM	TRADE	07:49:42.571	4

8) **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

פיתון בסיסי

פקודה המציגה שמות עמודות של טבלה Data Frame

In [106]:
```
df.columns
```

Out[106]:
```
,'Index(['Timestamp', 'EventType', 'Ticker', 'Price', 'Quantity', 'Exchange
,['Conditions'
('dtype='object
```

פקודה המציגה X שורות ראשונות מתוך הטבלה Data Frame

אם לא רושמים כלום בתוך הסוגריים יציג לנו כברירת מחדל את 5 השורות הראשונות

In [107]:
```
df.head()
```

Out[107]:

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
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 נרשום מספר (פרמטר) בתוך הסוגריים, אז יציג לנו את כמות השורות הראשונות שרשמנו.

In[109]:
```
df.head(10)
```

Out[109]:

	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
6	00002000	ARCA	200	154.50	IBM	TRADE	07:49:42.817
7	00002000	ARCA	200	154.50	IBM	TRADE	07:49:43.006
8	00002000	ARCA	200	154.50	IBM	TRADE	07:49:43.188
9	00002000	ARCA	200	154.50	IBM	TRADE	07:49:43.191

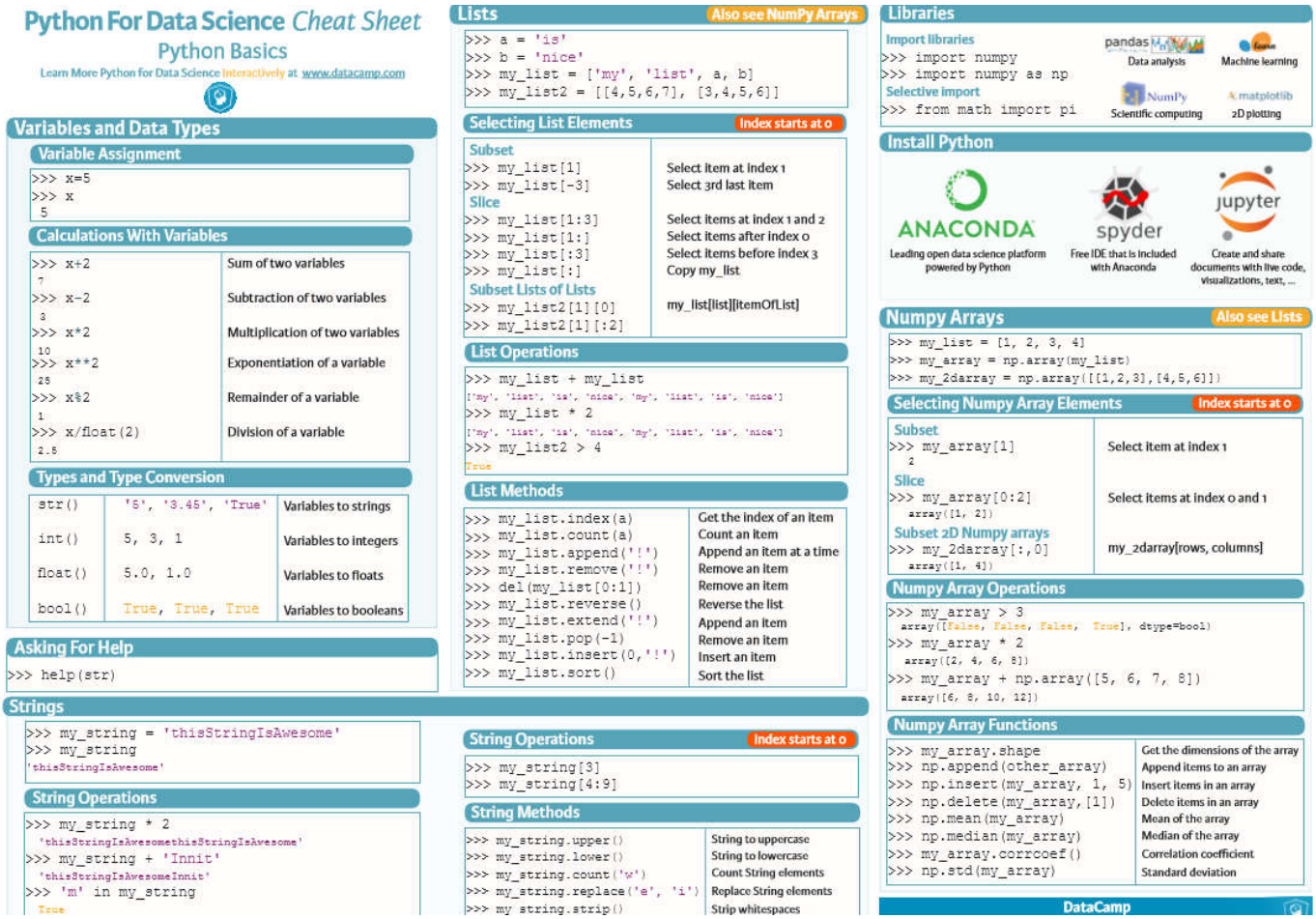
הצגת כמות הערכים הראשונים מעמודה מסוימת.

גם כאן, אם לא נזין שום מספר (פרמטר) בסוגריים אז יציג לנו כברירת מחדל את 5 הערכים הראשונים בעמודה.

In[108]:
```
df['Timestamp'].head()
```

Out[108]:
```
07:12:38.577    0
07:48:52.137    1
07:49:41.075    2
07:49:42.377    3
07:49:42.571    4
Name: Timestamp, dtype: object
```

Cheat Sheet -פקודות בסיסיות בפיתון



משתנים

משתנה שקראנו לו X שעושים אליו השמה של המספר 5. הערך שלו נהיה =5.

חישובים לדוגמא שניתן לבצע עם משתנה:

1) **אופרטור החיבור**: להוסיף למשתנה X 2
2) **אופרטור החיסור**: להחסיר מהמשתנה X 2
3) **אופרטור הכפל**: להכפיל את המשתנה X ב 2
4) **אופרטור חזקה**: לבצע X בחזקת 2
5) **אופרטור מודולו**: לחשב מה השארית של x חלקי 2.

דוגמא למה אופרטור % (מודולו) שימושי: לדעת אם מדובר במספר זוגי או לא.

במספר זוגי כאשר נעשה 2% אז התוצאה תהיה תמיד 0 (כי לא יהיה לו שארית אחרי חלוקה ל 2).
במספר אי זוגי כשנעשה לו 2% אז התוצאה תהיה תמיד 1 (כי תהיה לו שארית בחלוקה ל 2).

6) **אופרטור החילוק**: לבצע פעולת חלקי רגילה.
בגרסאות פיתון הנוכחיות אם מחלקים שני מספרים שלמים אז התוצאה תהיה עשרונית,
הערך שיתקבל יהיה מסוג float.

בגרסאות ישנות של פיתון אם היינו רוצים תוצאה שכוללת ספרות עשרוניות ולא רוצים שיעגל את התוצאה למספר שלם,
אז היה צריך שאחד מהמספרים יהיה מסוג float. אחרת היינו מקבלים תוצאה עם מספר.

Calculations With Variables

`>>> x+2` `7`	Sum of two variables
`>>> x-2` `3`	Subtraction of two variables
`>>> x*2` `10`	Multiplication of two variables
`>>> x**2` `25`	Exponentiation of a variable
`>>> x%2` `1`	Remainder of a variable
`>>> x/float(2)` `2.5`	Division of a variable

המרת משתנים

Str= string= טקסט
Int=integer= מספר שלם
Float= מספר עשרוני
Booleans= מחזיר ערך True או False

Types and Type Conversion

`str()`	`'5', '3.45', 'True'`	Variables to strings
`int()`	`5, 3, 1`	Variables to integers
`float()`	`5.0, 1.0`	Variables to floats
`bool()`	`True, True, True`	Variables to booleans

```
In [2]:
x=5
x

Out[2]:
5

In [3]:
str(x)

Out[3]:
'5'

In [4]:
int(x)

Out[4]:
5

In [5]:
float(x)

Out[5]:
5.0

In [6]:
bool(x)

Out[6]:
True

In [7]:
1=='1'

Out[7]:
False

In [8]:
x=1=='1'
x

Out[8]:
False
```

פונקציית help

רושמים help ובתוך הסוגריים מה שרוצים לקבל מידע לגביו.

Asking For Help

```
>>> help(str)
```

```
help(bool)
```

```
:Help on class bool in module builtins

class bool(int)
bool(x) -> bool  |
  |
.Returns True when the argument x is true, False otherwise  |
.The builtins True and False are the only two instances of the class bool  |
.The class bool is a subclass of the class int, and cannot be subclassed  |
  |
```

String

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
 'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
 'thisStringIsAwesomeInnit'
>>> 'm' in my_string
 True
```

String Operations — Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

`>>> my_string.upper()`	String to uppercase
`>>> my_string.lower()`	String to lowercase
`>>> my_string.count('w')`	Count String elements
`>>> my_string.replace('e', 'i')`	Replace String elements
`>>> my_string.strip()`	Strip whitespaces

```
mystring='Hello World'
mystring*2
```

```
'Hello WorldHello World'
```

```
mystring+'2222'
```

```
'Hello World2222'
```

```
'ing' in mystring
```

```
False
```

```
'llo' in mystring
```

```
True
```

String מתנהג כמו מערך לכן ניתן לבצע עליו פקודות כמו שאנחנו מבצעים על מערך.
אם רוצים שיחזיר לנו את האיבר הראשון במילה נרשום בתוך סוגריים מרובעים [0]. אם נרצה את האיבר השלישי במילה נרשום [2] (כי מתחיל מ0).
אם רוצים טווח איברים לדוגמא מהאיבר הראשון עד האיבר החמישי אז נרשום [0:5] האיבר החמישי הוא איבר באינדקס 4 אבל כרושמים [0:4] הוא מתייחס לכך כאיבר 0 עד איבר 4 אבל לא כולל 4 ואנחנו רוצים שכן יכלול את איבר 4 לכן נרשום [0:5].

```
mystring='Hello World'
```

```
mystring[0]
```

```
'H'
```

```
mystring[1]
```

```
'e'
```

```
mystring[0:3]
```

```
'Hel'
```

```
mystring[4:9]
```

```
'o Wor'
```

```
mystring.upper()
```

```
'HELLO WORLD'
```

```
mystring.count('l')
```

```
3
```

```
mystring.count('W')
```

```
1
```

```
mystring.replace('ll','RR')
```

```
'HeRRo World'
```

```
mystring='Hello World  '
mystring
```

```
'  Hello World'
```

```
mystring.strip()
```

```
'Hello World'
```

***יכול גם להסיר תווים מסוימים מתחילת ה-string או מסופו.

```
In [129]: mystring='hello world'
          mystring

Out[129]: 'hello world'

In [133]: mystring2=mystring.strip('h')
          mystring2

Out[133]: 'ello world'

In [135]: mystring2=mystring2.strip('d')
          mystring2

Out[135]: 'ello worl'
```

List

List בפיתון הוא מערך. אך בניגוד למערכים משפות תכנות אחרות. ב list הערכים בכל שדה במערך יכולים להיות מסוגים שונים לדוגמא בשדה אינדקס 0 ערך מסוג string ואילו בשדה באינדקס 1 ערך מסוג int.

ואפילו ניתן לדוגמא שבאינדקס 3 יהיה ערך מסוג list כך שייווצר list שאחד הערכים שלו הוא עוד list.

ה list מאופיין בסוגריים מרובעים שהערכים בו מופרדים בפסיקים.

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset	
`>>> my_list[1]`	Select item at index 1
`>>> my_list[-3]`	Select 3rd last item
Slice	
`>>> my_list[1:3]`	Select items at index 1 and 2
`>>> my_list[1:]`	Select items after index 0
`>>> my_list[:3]`	Select items before index 3
`>>> my_list[:]`	Copy my_list
Subset Lists of Lists	
`>>> my_list2[1][0]`	my_list[list][itemOfList]
`>>> my_list2[1][:2]`	

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

`>>> my_list.index(a)`	Get the index of an item
`>>> my_list.count(a)`	Count an item
`>>> my_list.append('!')`	Append an item at a time
`>>> my_list.remove('!')`	Remove an item
`>>> del(my_list[0:1])`	Remove an item
`>>> my_list.reverse()`	Reverse the list
`>>> my_list.extend('!')`	Append an item
`>>> my_list.pop(-1)`	Remove an item
`>>> my_list.insert(0,'!')`	Insert an item
`>>> my_list.sort()`	Sort the list

דוגמאות לשימוש ב List:

In [40]:
```
a='is'
b='nice'
my_list=['my','list',a,b]
my_list
```

Out[40]:
```
['my', 'list', 'is', 'nice']
```

In [41]:
```
my_list2=[1,'aaa',a,[1,2,3,'bbbb']]
my_list2
```

Out[41]:
```
[aaa', 'is', [1, 2, 3, 'bbbb']' ,1]
```

שליפת נתונים מlist

```
my_list2[0]                    :[42] In
1                              Out[42]:

my_list2[1]                    :[43] In
'list'                         Out[43]:

my_list2[2]                    :[44] In
'is'                           Out[44]:

my_list2[3]                    :[45] In
['bbbb' ,3 ,2 ,1]              Out[45]:
```

```
my_list2[-1]                   :[47] In
['bbbb' ,3 ,2 ,1]              Out[47]:

my_list2[-2]                   :[48] In
'is'                           Out[48]:
```

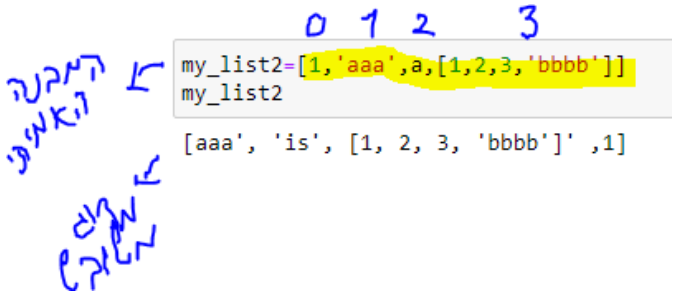
```
my_list2[1:3]                  :[49] In
['aaa', 'is']                  Out[49]:

my_list2[1:]                   :[50] In
[aaa', 'is', [1, 2, 3, 'bbbb']']  Out[50]:

my_list2[:3]                   :[51] In
['aaa', 'is' ,1]               Out[51]:

my_list2[:]                    :[52] In
[aaa', 'is', [1, 2, 3, 'bbbb']' ,1]  Out[52]:
```

***כאשר רצינו שיראה לנו את כל המערך הוא מציג זאת לעיתים עם מיקומים משובשים בחלק מהמקרים. אבל אם אנחנו שולפים לפי index
אנחנו רואים שהמיקום האמתי הוא כמו שהגדרנו בהתחלה את המערך ורק הסדר בתצוגה, כאשר מציג כמה ערכים ביחד, לפעמים משתבש.



אם יש list שאחד הערכים שלו הוא גם list אז איך אנחנו שולפים נתונים מה list הפנימי?

בדוגמא שלנו, בmy_list2 הערך באינדקס 3 הוא list.
איך נשלוף לדוגמא את הערך באינדקס 0 מהlist שנמצאת באינדקס 3 של my_list2....
(נשמע מבלבל).
הכוונה איך נשלוף את הערך המסומן:

```
my_list2=[1,'aaa',a,[1,2,3,'bbbb']]
```

```
In [53]: my_list2[3][0]
Out[53]: 1
```

ואם אנחנו רוצים לשלוף את הערך באינדקס 3 של ה list הפנימי:

```
In [54]: my_list2[3][3]
Out[54]: 'bbbb'
```

אופרטורים שאפשר להשתמש בהם ב list

```
In [55]: my_list
Out[55]: ['my', 'list', 'is', 'nice']

In [56]: my_list+my_list
Out[56]: ['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']

In [57]: my_list*2
Out[57]: ['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
```

פונקציות לList

Index

מספקים לה ערך מlist והיא מחזירה את מיקומי- את הindex שהוא ממוקם בו.

```
In [66]:
a='is'
b='nice'
my_list=['my','list',a,b]
my_list

Out[66]:
['my', 'list', 'is', 'nice']

In [67]:
my_list.index(a)

Out[67]:
2

In [68]:
my_list.index('list')

Out[68]:
1

In [69]:
my_list.index('nice')

Out[69]:
3
```

פונקציה count

מחזירה את כמות המופעים של ערך בlist (מכיוון שגם string מתנהל כמו מערך/LIST אז גם שימושי להשתמש בפונקציה לו עליו)

```
In [70]:
my_list.count(a)

Out[70]:
1

In [71]:
my_list.count('is')

Out[71]:
1

In [111]:
example_for_string='hello world'
example_for_string.count('l')

Out[111]:
3
```

פונקציה append להוספת ערך חדש ל list

פונקציה שמאפשרת להכניס פריט יחיד בסופה של הרשימה.

****גם כאן לפעמים התצוגה משבשת את סדר הערכים אבל ערך שמתווסף, מתווסף לסוף הרשימה, לאינדקס הגדול ביותר.**

```
my_list.append('!')                                   :[72] In

my_list                                               :[73] In
['!' ,'my', 'list', 'is', 'nice']                     Out[73]:

my_list.append('abcd')                                :[74] In

my_list                                               :[75] In
['my', 'list', 'is', 'nice', '!', 'abcd']             Out[75]:

my_list.index('!')                                    :[76] In
4                                                     Out[76]:
```

פונקציה נוספת להוספת ערכים לרשימה- extend

מאפשרת להוסיף יותר מערך אחד בהפעלת הפונקציה (להבדיל מappend שמאפשר להוסיף ערך אחד בלבד בכל הפעלה של הפונקציה)

```
my_list                                               :[91] In
['nice', 'is', 'list', 'my']                          Out[91]:

my_list.extend('!')                                   :[92] In

my_list                                               :[93] In
['!' ,'nice', 'is', 'list', 'my']                     Out[93]:

my_list.index('!')                                    :[94] In
4                                                     Out[94]:
```

```
my_list.extend(['aa','bb','cc','dd','ee'])
my_list

['nice', 'is', 'list', 'my', '!', 'aa', 'bb', 'cc', 'dd', 'ee']
```

עוד פונקציה להוספת ערכים ל list- פונקציה insert

פונקציה שמאפשרת להכניס פריט לתוך list- מספקים לה את מיקום הערך (index) פסיק ואח"כ את הערך עצמו שרוצים להוסיף לlist.

```
my_list                                                    :[112] In
['list', 'my', '!', 'aa', 'bb', 'dd']                      Out[112]:

my_list.insert(0,'yyyy')                                   :[113] In
my_list
['yyyy', 'list', 'my', '!', 'aa', 'bb', 'dd']              Out[113]:

my_list.insert(-1,'zzzz')                                  :[114] In
my_list
['yyyy', 'list', 'my', '!', 'aa', 'bb', 'zzzz', 'dd']      Out[114]:

len(my_list)                                               :[116] In
8                                                          Out[116]:

my_list.insert(len(my_list),'wwwww')                       :[117] In
my_list
['yyyy', 'list', 'my', '!', 'aa', 'bb', 'zzzz', 'dd', 'wwwww']  Out[117]:
```

***הפונקציה len מחזירה את כמות הערכים בlist. ברשימה שלנו היו 8 ערכים. אך מכיוון שהאינדקס מתחיל מ0 אז הערך האחרון הוא באינדקס 7.
לכן טריק זה יכול לעזור לנו כשנרצה להוסיף ערכים לסוף הרשימה.

פונקציה remove להסרת ערך מlist

```
my_list                                                    :[75] In
['my', 'list', 'is', 'nice', '!', 'abcd']                  Out[75]:

my_list.index('!')                                         :[76] In
4                                                          Out[76]:

my_list.remove('!')                                        :[77] In

my_list                                                    :[78] In
['my', 'list', 'is', 'nice', 'abcd']                       Out[78]:

my_list.index('abcd')                                      :[79] In
4                                                          Out[79]:
```

דרך נוספת להסרת ערכים מ list פונקציה del

```
my_list                                                    :[80] In
['my', 'list', 'is', 'nice', 'abcd']                       Out[80]:

del(my_list[0:1])                                          :[81] In
my_list
['list', 'is', 'nice', 'abcd']                             Out[81]:

del(my_list[:])                                            :[82] In
my_list
[]                                                         Out[82]:

del(my_list)                                               :[83] In

my_list                                                    :[84] In

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-84-9af163499740> in <module>
```

פונקציה נוספת להסרת ערכים מlist – פונקציה pop

פונקציה המוחקת איבר ברשימה ומדפיסה אותו למשתמש. אם לא מוגדר מיקומו של האיבר בסוגרים, ברירת המחדל הוא מחיקת האיבר האחרון.

```
my_list                                                    :[102] In
['nice', 'is', 'list', 'my', '!', 'aa', 'bb', 'cc', 'dd', 'ee']   Out[102]:

my_list.pop()                                              :[103] In
my_list
['nice', 'is', 'list', 'my', '!', 'aa', 'bb', 'cc', 'dd']  Out[103]:

my_list.pop(-2)                                            :[108] In
'cc'                                                       Out[108]:

my_list.pop(0)                                             :[109] In
'nice'                                                     Out[109]:

my_list.pop(0)                                             :[110] In
my_list
['list', 'my', '!', 'aa', 'bb', 'dd']                      Out[110]:
```

פונקציה reverse

משנה את סדר האינדקסים ב list

```
a='is'
b='nice'
my_list=['my','list',a,b]
my_list
```
:[86] In

```
['my', 'list', 'is', 'nice']
```
Out[86]:

```
my_list.reverse()
```
:[87] In

```
my_list
```
:[88] In

```
['nice', 'is', 'list', 'my']
```
Out[88]:

```
my_list.index('nice')
```
:[90] In

```
0
```
Out[90]:

פונקציה sort

פונקציה שמאפשרת למיין את הערכים ברשימה

```
my_new_list=[1,0,5,3,2,7,9,20,10,12]
my_new_list
```
:[118] In

```
[12 ,10 ,20 ,9 ,7 ,2 ,3 ,5 ,0 ,1]
```
Out[118]:

```
my_new_list.sort()
my_new_list
```
:[119] In

```
[20 ,12 ,10 ,9 ,7 ,5 ,3 ,2 ,1 ,0]
```
Out[119]:

```
my_new_list.index(20)
```
:[127] In

```
9
```
Out[127]:

```
my_new_list.index(12)
```
:[128] In

```
8
```
Out[128]:

אתר מומלץ שמציג פונקציות בפיתון כולל הסבר ודוגמאות:

https://he.wikibooks.org/wiki/%D7%A4%D7%99%D7%99%D7%AA%D7%95%D7%9F/%D7%A4%D7%99%D7%99%D7%AA%D7%95%D7%9F_%D7%92%D7%A8%D7%A1%D7%94_3/%D7%A8%D7%A9%D7%99%D7%9E%D7%AA_%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%95%D7%AA_%D7%9E%D7%A2%D7%A8%D7%9B%D7%AA_built-in