



جزوه شماره ۱



# PANDAS LIBRARY

ArcGIS Pro



ناهید نعمتی کوتنائی (تیسرا)  
دکتری جغرافیا و برنامه‌ریزی شهری  
مدرس دانشگاه

Dr.nemati.K  
 @Nemati\_k  
 09112230798



محمدطاهر طاهرپور  
دانشجوی ارشد مدیریت شهری  
دانشگاه تهران

mttaherpoor  
 @mtaherpoor  
 09336144947



پاییز ۱۴۰۲

## فهرست مطالب:

۳	Pandas Library
۳	Pandas چیست؟
۳	چطوری با pandas کار کنیم؟
۳	ساختار داده یا Data Structure
۳	ساختار داده Series
۸	ساختار داده DataFrame
۸	بخش اول:
۱۲	بخش دوم:
۱۵	بخش سوم:
۱۸	عملگرهای groupby
۲۱	ترکیب دیتافریم‌ها Combining DataFrames
۳۰	وارد کردن داده‌ها با فرمت csv و Excel
۳۲	خلاصه دستوره‌های pandas
۳۵	حل چندتا تمرین
۳۶	جواب تمرینها

## Pandas Library

تو این جزوه یکی دیگه از مهمترین کتابخانه‌های پایتون به اسم Pandas رو با هم یاد میگیریم. این کتابخانه هم مثل کتابخانه Numpy واسه علم داده ضروری هست. مطالب این جزوه از روی مطالب آموزشی Udemy - Data Visualization with Python Masterclass - Python A-Z بخش مربوط به Fundamentals of Data Science، سایت اصلی پانداس <https://pandas.pydata.org/> و پلتفرم گیت‌هاب <https://github.com/pandas-dev/pandas> گرفته شده.

پیش نیاز این جزوه، جزوه‌های شماره ۶ و ۷ هستن. قبل اینکه شروع کنی مطمئن شو که درک خوبی از اون دو تا جزوه بدست آوردی.

## Pandas چیه؟

یه کتابخانه open source پایتون هست که برای خوندن، دستکاری و تجزیه و تحلیل داده‌ها استفاده میشه. نکته باحال در مورد پانداس اینه که میتونی داده‌ها رو به صورت جدول مرتب کنی که واسه خواندنشون. ولی همیشه به صورت عددی هم تفسیرشون کرد. همیشه باهاشون محاسبات ریاضی هم انجام داد. در واقع داده‌ها رو load میکنی، prepare یا آماده‌شون میکنی، بعد manipulate یا دستکاریشون میکنی و بعد ازشون model میسازی و Analyse یا تجزیه و تحلیلشون میکنی.

## چطوری با pandas کار کنیم؟

مثل کتابخانه Numpy میتونی تو نوت بوک ArcGIS Pro با دستور import واردش کنی و باهاش تمرین انجام بدی.

پانداس رو با اسم مستعار pd وارد میکنیم:

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

## ساختار داده یا Data Structure

پانداس ابزارهای قوی واسه ساختار داده یا data structure داره. ساختار داده شیوه خاص تنظیم داده تو کامپیوتر هست که کمک میکنه که از داده‌ها موثرتر استفاده کنیم. پانداس سه تا ساختار داده داره:

**Series** 🚩: یه آرایه تک بعدی با نمایه‌سازی سفارشی یا custom indexing هست. میتونه شامل رشته‌ها

هم بشه. اگه کمبود داده داشته باشیم به صورت پیش‌فرض نادیده‌شون میگیره.

**DataFrame** 🚩: به جدول داده می‌گیم قاب داده یا DataFrame. در واقع همین کلیدواژه pandas رو از بقیه

کتابخانه‌ها متمایز میکنه.

**Panel** 🚩: Pandas مخفف panel data هست.

## ساختار داده Series

این ساختار داده به صورت **pandas.Series(data, index, dtype, name, copy)** تعریف میشه که چند تا پارامتر داره. این پارامترها فرمهای متنوعی میگیرن شامل:

۱- data: شامل constants یا یه عدد ثابت، lists یا لیستها، ndarray آرایه چندبعدی

۲- index: مقدار ایندکس هست

۳- dtype: همون data type هست. نوع داده سریهای خروجی رو تعیین میکنه.

۴- name: نام سری

۵- copy: داده ورودی رو کپی میکنه. مقدار پیش فرضش False هست.

```
import numpy as np
import pandas as pd
```

```
P_Series = pd.Series([7,11,19,113])
```

```
P_Series
```

```
0      7
1     11
2     19
3    113
dtype: int64
```

میخوایم تو نوت بوک Series درست کنیم. اول باید هم numpy و هم pandas رو وارد کنیم. بعدش یه سری به اسم P\_Series درست میکنیم و با کمک **pd.Series()** بهش یه لیست از داده‌های عددی رو اختصاص میدیم. (یعنی بین پارامترهایی که داره، فقط بخش data رو بهش میدیم)

نتیجه بهمون دو تا ستون میده که ستون اول ایندکسها هستن و ستون دوم اعضا. انتهایش هم نوع داده رو بهمون نشون میده که عدد صحیح هست.

```
P_Series_2 = pd.Series(29, range(5))
```

```
P_Series_2
```

```
0     29
1     29
2     29
3     29
4     29
dtype: int64
```

یه سری دیگه به اسم P\_Series\_2 تعریف میکنیم و اینبار برای بخش data یه عدد ثابت یا constant بهش میدیم و براش **range()** تعریف میکنیم که چندبار این عدد رو بهمون خروجی بده یا اینکه چند تا ایندکس داشته باشه.

```
P_Series[3]
```

```
113
```

اگه بخوایم به اعضای سری دسترسی داشته باشیم باید با **[]** چلوی اسمش نوشته بشه.

```
P_Series_3 = pd.Series(np.random.randint(1,150,20))
```

```
P_Series_3
```

```
0     49
1     16
2     72
3    119
4     32
5     65
6     32
7    104
8    121
9     19
10    116
11     45
12     74
13     57
14     44
15     71
16    109
17    105
18     70
19     37
dtype: int32
```

پاندا تابع **random.randint()** نداره واسه همین باید از ترکیب numpy و Pandas استفاده کنیم. یعنی عبارت **np.random.randint()** رو توی پرانتز **pd.Series()** مینویسیم که بهمون اعداد رندوم بده (دلیل اینکه اون اولش نامپی رو هم وارد کردیم همین بود که اینجا خطا بهمون نده). یه سری به اسم P\_Series\_3 درست میکنیم که ۲۰ تا عضو تو محدوده اعداد ۱ تا ۱۵۰ به صورت رندوم بهمون بده.

```
P_Series_3.max()
```

```
121
```

```
P_Series_3.min()
```

```
16
```

```
P_Series_3.mean()
```

```
67.85
```

```
P_Series_3.std()
```

```
34.31935927508892
```

حالا میشه ماکزیمم، مینیمم، میانگین و انحراف معیار رو هم برای اعضای سری محاسبه

کرد.

همه این مراحل بالا رو میشه با متد **describe()** انجام داد و نتیجه رو دید.

```
P_Series_3.describe()
```

```
count    20.000000
mean     67.850000
std      34.319359
min      16.000000
25%      42.250000
50%      67.500000
75%     104.250000
max     121.000000
dtype: float64
```

علاوه بر موارد بالا بهمون صدک بیست و پنجم، صدک پنجاهم و صدک هفتاد و پنجم و تعداد اعضا رو هم میده. صدک پنجاهم همون میانه هست.

نوع داده رو float یا عدد اعشاری میده.

```
P_Series_3.head()
```

```
0    49
1    16
2    72
3   119
4    32
dtype: int32
```

```
P_Series_3.head(2)
```

```
0    49
1    16
dtype: int32
```

```
P_Series_3.head(7)
```

```
0    49
1    16
2    72
3   119
4    32
5    65
6    32
dtype: int32
```

```
P_Series_3.tail()
```

```
15    71
16   109
17   105
18    70
19    37
dtype: int32
```

```
P_Series_3.tail(2)
```

```
18    70
19    37
dtype: int32
```

```
P_Series_3.tail(7)
```

```
13    57
14    44
15    71
16   109
17   105
18    70
19    37
dtype: int32
```

با متد **head()** میتونیم از ۵ سطر اول سری خروجی بگیریم. داخل پرانتزش هم میتونیم عدد بدیم که مشخص کنیم چند تا سطر از این سری رو نیاز داریم.

با متد **tail()** پنج سطر آخر رو میتونیم خروجی بگیریم. میتونیم عدد دیگه‌ای داخل پرانتزش بذاریم که از آخر میشماره.

میتونیم سری رو با ایندکس تعریف کنیم که به جای اعداد بهمون یه برچسب بده.

```
P_Series_4 = pd.Series([5,13,23,29], index=["Joe","William","Jack","Avare1"])
```

```
P_Series_4
```

```
Joe      5
William  13
Jack     23
Avare1   29
dtype: int64
```

```
P_Series_4["Avare11"]
```

```
29
```

```
P_Series_4.Joe
```

```
5
```

حالا به جای اینکه تو براکت [] عدد برای ایندکس وارد کنیم که به اعضا دسترسی داشته باشیم میتونیم از همین رشته‌های استفاده کنیم که عددش رو بهمون برگردونه.

به جای [] میتونیم رشته رو با نقطه به اسم سری وصل کنیم و همین نتیجه رو ازش بگیریم.

میتوانیم برای سریها دیکشنری یعنی key:value تعریف کنیم که داخل آکولاد میرن. در واقع وقتی دیکشنری براش تعریف کنیم تو خروجی مثل سریها دیده میشن. یعنی دو ستونه که ستون اول کلیدها هستن و ستون دوم مقادیر.

```
P_Series_D = pd.Series({"One":1,"Two":2,"Three":3,"Four":4,"Five":5})
```

```
P_Series_D
```

One	1
Two	2
Three	3
Four	4
Five	5

```
dtype: int64
```

اگه سری رو فقط با رشته‌ها بسازیم باز هم تو دو ستون بهمون خروجی میده که ستون اول ایندکسها هستن و ستون دوم مقادیر رشته‌ای. نوع داده رو هم بهمون object میده.

```
P_Series_S = pd.Series(["Tom","Jerry","Spike","Tyke","Nibbles","Quaker"])
```

```
P_Series_S
```

0	Tom
1	Jerry
2	Spike
3	Tyke
4	Nibbles
5	Quaker

```
dtype: object
```

```
P_Series_S.str.contains("e")
```

0	False
1	True
2	True
3	True
4	True
5	True

```
dtype: bool
```

حالا اگه بخوایم از بین اعضای کد قبلی اونایی که e دارن رو بررسی کن و به صورت True بهمون نشون بده و اونایی که e ندارن رو با False باید از متد **str.contains()** استفاده کنیم.

در واقع نوع داده رو به صورت بولین بهمون برمیگردونه.

اگه بخوایم که اعضای رشته‌ای مجموعه بالا رو به صورت حروف بزرگ بهمون نشون بده باید از متد **str.upper()** استفاده کنیم.

بسته به نوع داده ورودی میشه کلی متدهای متنوع براش استفاده کرد.

```
P_Series_S.str.upper()
```

0	TOM
1	JERRY
2	SPIKE
3	TYKE
4	NIBBLES
5	QUAKER

```
dtype: object
```

علاوه بر اون میتونیم از ایندکسهای شخصی شده هم استفاده کنیم. تو مثال پایین دو تا لیست یه اسم Data و Label درست کردیم که یکی اعداد صحیح داده و اون یکی رشته. بعد با تابع `pd.Series(data, index)` ایندکسهاش رو شخصی سازی کردیم.

```
Data = [1,2,3,4,5,6]

Label = ["apple", "cherry", "banana", "orange", "pear", "melon"]

pd.Series(data = Data, index = Label)

apple      1
cherry     2
banana     3
orange     4
pear       5
melon      6
dtype: int64
```

تو کد بالا تعداد اعضای Data و Label با هم برابر هستن اگه اعضا برابر نباشن بهمون خطا میده.

میتونیم تو تابع `pd.Series(data,index)` مستقیم اعضا و ایندکسها رو وارد کنیم.

```
Sd_LW = pd.Series([70,63,150,45,33,35],["Chocolate", "Bread", "Biscuits", "Potatoes(kg)", "Carrots(kg)", "Chips"])
```

```
Sd_TW = pd.Series([45,51,90,67,44,13],["Chocolate", "Bread", "Biscuits", "Potatoes(kg)", "Carrots(kg)", "Crackers"])
```

Sd\_LW

```
Chocolate      70
Bread          63
Biscuits       150
Potatoes(kg)   45
Carrots(kg)    33
Chips          35
dtype: int64
```

Sd\_TW

```
Chocolate      45
Bread          51
Biscuits       90
Potatoes(kg)   67
Carrots(kg)    44
Crackers       13
dtype: int64
```

Sd\_LW + Sd\_TW

```
Biscuits       240.0
Bread          114.0
Carrots(kg)    77.0
Chips          NaN
Chocolate      115.0
Crackers       NaN
Potatoes(kg)   112.0
dtype: float64
```

میتونیم دو تا سری بالا رو با هم جمع کنیم. اونایی که ایندکس مشابه دارن اعدادشون رو جمع میزنه و برای اونایی که مشابه نیستن خود ایندکس رو میده ولی جلوی مقدارش NaN (Not a Number) مینویسه. نوع داده‌ها رو هم به صورت اعشاری میده.

## ساختار داده DataFrame

### بخش اول:

یه آرایه دوبعدی هست. میتونه سطر و ستونهای شخصی سازی شده داشته باشه. میشه روش عملیات ریاضی انجام داد. برخلاف series، دیتافریم از انبوه داده‌ها پشتیبانی میکنه. میتونه انواع متفاوت داده رو در خودش داشته باشه.

واسه استفاده ازش هر دو تا کتابخونه numpy و pandas رو تو نوت بوک وارد میکنیم.

براش یه دیکشنری از وضعیت فروش یه سری اجناس مینویسیم که کلیدواژه‌هاش رشته هستن و مقادیرشون به حالت لیست نوشته میشه. بعد باید از تابع `pd.DataFrame()` استفاده کنیم که این دیکشنری رو تبدیلش کنیم به یه دیتافریم یا یه آرایه دو بعدی که کلیدواژه‌ها سرتیتر ستونهاش باشن و مقادیر لیست شده اطلاعات هر سطرش.

```
import numpy as np
import pandas as pd
```

```
P_Dict_Sales = {"Chocolate": [70, 45, 65], "Biscuits": [150, 90, 143],
                "Potatoes(kg)": [45, 67, 38], "Carrots(kg)": [33, 44, 45],
                "Chips": [35, 0, 28], "Crackers": [0, 13, 45]}
```

```
P_DFarms = pd.DataFrame(P_Dict_Sales)
```

P\_DFarms

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
0	70	150	45	33	35	0
1	45	90	67	44	0	13
2	65	143	38	45	28	45

یه ستون هم به جدول خروجی اضافه میشه که در واقع شماره ایندکس هر سطر هست و از ۰ شروع میشه. میتونیم به جای این ایندکسهای عددی یه سری مقادیر رشته‌ای وارد کنیم. اینکار تو پرانتز تابع `pd.DataFrame(name, index)` بعد از نام دیتافریم آورده میشه.

```
P_DFarms = pd.DataFrame(P_Dict_Sales, index = ["Week1",
                                              "Week2", "Week3"])
```

P\_DFarms

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	150	45	33	35	0
Week2	45	90	67	44	0	13
Week3	65	143	38	45	28	45



```
P_DFarms["Biscuits"]
```

```
Week1    150
Week2     90
Week3    143
Name: Biscuits, dtype: int64
```

```
P_DFarms.Crackers
```

```
Week1     0
Week2    13
Week3    45
Name: Crackers, dtype: int64
```

اگر تعداد ایندکسها از تعداد مقادیر سطرها بیشتر باشد خطا میدهد.

اگر بخوایم به یک ستون مشخص از دیتافریم دسترسی داشته باشیم باید به دو صورت عمل کنیم با اسم ستون رو (با کوتیشن) داخل [] جلوی اسم دیتافریم بیاوریم یا با نقطه (بدون کوتیشن) به اسم دیتافریم بچسبونیم.

تو خروجی همیشه ستون مربوط به ایندکسها رو هم بهمون میدهد. پایینش هم اسم ستون به همراه نوع داده آورده میشه.

```
P_DFarms.loc["Week1"]
```

```
Chocolate    70
Biscuits     150
Potatoes(kg)  45
Carrots(kg)   33
Chips         35
Crackers      0
Name: Week1, dtype: int64
```

```
P_DFarms.iloc[1]
```

```
Chocolate    45
Biscuits     90
Potatoes(kg)  67
Carrots(kg)   44
Chips         0
Crackers     13
Name: Week2, dtype: int64
```

از متد **loc()** میتونیم اطلاعات یک سطر رو به صورت کامل به همراه اسمی همه ستونها داشته باشیم. به جای پرانتز از [] به همراه اسم ایندکس اون سطر استفاده میکنیم.

همینکار رو میشه با **iloc()** هم انجام داد. که به جای پرانتز از [] به همراه شماره ایندکس سطر مورد نظر استفاده میکنیم. سطر اول ایندکس ۰ داره سطر دوم ۱ و به همین ترتیب.

میتونیم به دو شیوه بالا دیتافریم رو برش هم بزنیم که از [:] برای مشخص کردن سطر و ستون به همراه ویرگول که جداشون میکنه استفاده میکنیم.

```
In [24]: P_DFarms.loc["Week1":"Week2",["Chocolate","Biscuits","Carrots(kg)"]]
```

```
Out[24]:
```

	Chocolate	Biscuits	Carrots(kg)
Week1	70	150	33
Week2	45	90	44

یا میتونیم به عدد مشخص بهش بدیم که برای مثال مقادیر بزرگتر از این مقدار رو خروجی بده و مابقی اعداد رو NaN بنویسه.

```
P_DFarms[P_DFarms>= 50]
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70.0	150	NaN	NaN	NaN	NaN
Week2	NaN	90	67.0	NaN	NaN	NaN
Week3	65.0	143	NaN	NaN	NaN	NaN

از عملگرهایی & به معنی and و | یا هم می‌شود استفاده کرد که برای & باید هر دو تا شرط برقرار باشه که خروجی بده در غیر اینصورت NaN برمیگردونه ولی برای | به شرط هم برقرار باشه کافیه.

برای نوشتن علامت | باید Shift + \ رو نگه داری.

```
P_DFarms[(P_DFarms >= 0) & (P_DFarms < 50)]
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	NaN	NaN	45.0	33	35	0
Week2	45.0	NaN	NaN	44	0	13
Week3	NaN	NaN	38.0	45	28	45

```
P_DFarms[(P_DFarms >= 100) | (P_DFarms < 50)]
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	NaN	150.0	45.0	33	35	0
Week2	45.0	NaN	NaN	44	0	13
Week3	NaN	143.0	38.0	45	28	45

واسه انتخاب یه مقدار مشخص از دیتافریم از `.iat()` و `.at()` طبق قاعده بالا استفاده میکنیم. اولی از برچسب یا label استفاده میکنه و دومی از عدد صحیح. باید بین سطر و ستون تعیین اون عدد مشخص کاما بذاریم.

```
P_DFarms.at["Week1", "Chocolate"]
```

70

```
P_DFarms.iat[0,3]
```

33

میتونیم از طریق این دو تا متد، مقادیر دیتافریم رو هم تغییر بدیم.

```
P_DFarms.at["Week1", "Chocolate"] = 66
```

P\_DFarms

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	66	150	45	33	35	0
Week2	45	90	67	44	0	13
Week3	65	143	38	45	28	45

```
P_DFarms.iat[0,3] = 75
```

P\_DFarms

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	66	150	45	75	35	0
Week2	45	90	67	44	0	13
Week3	65	143	38	45	28	45

```
P_DFarms.describe()
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
count	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
mean	58.666667	127.666667	50.000000	54.666667	21.000000	19.333333
std	11.846237	32.807519	15.132746	17.616280	18.520259	23.158872
min	45.000000	90.000000	38.000000	44.000000	0.000000	0.000000
25%	55.000000	116.500000	41.500000	44.500000	14.000000	6.500000
50%	65.000000	143.000000	45.000000	45.000000	28.000000	13.000000
75%	65.500000	146.500000	56.000000	60.000000	31.500000	29.000000
max	66.000000	150.000000	67.000000	75.000000	35.000000	45.000000

مثل series اینجا هم متد `.describe()` بهمون یه خلاصه آماری از هر ستون جدول میده. اعداد با ۶ رقم اعشار دیده میشن.

واسه کم کردن عدد اعشاری که تو بخش قبلی داده میتونیم از تابع `pd.set_option()` استفاده کنیم و مقدار "precision" رو عدد ۲ بدیم که با کاما از هم جدا میشن.

مجدد با متد `.describe()` از دیتافریم خروجی میگیریم که میبینیم پیش فرض عوض میشه و عدد اعشار به دو رقم میرسه.

```
pd.set_option("precision",2)
```

```
P_DFarms.describe()
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
count	3.00	3.00	3.00	3.00	3.00	3.00
mean	58.67	127.67	50.00	54.67	21.00	19.33
std	11.85	32.81	15.13	17.62	18.52	23.16
min	45.00	90.00	38.00	44.00	0.00	0.00
25%	55.00	116.50	41.50	44.50	14.00	6.50
50%	65.00	143.00	45.00	45.00	28.00	13.00
75%	65.50	146.50	56.00	60.00	31.50	29.00
max	66.00	150.00	67.00	75.00	35.00	45.00

```
P_DFarms.mean()
```

```
Chocolate      58.67
Biscuits       127.67
Potatoes(kg)   50.00
Carrots(kg)    54.67
Chips          21.00
Crackers       19.33
dtype: float64
```

میتونیم مثل قبل از متدهای `.sum()/.min()/.max()/.mean()/.std()` هم استفاده کنیم که صرفا محاسبه آماری که نیاز داریم رو بهمون بده. مهمترین محاسبه آماری میانگین یا mean هست.

```
P_DFarms.T
```

	Week1	Week2	Week3
Chocolate	66	45	65
Biscuits	150	90	143
Potatoes(kg)	45	67	38
Carrots(kg)	75	44	45
Chips	35	0	28
Crackers	0	13	45

```
P_DFarms.T.describe()
```

	Week1	Week2	Week3
count	6.00	6.00	6.00
mean	61.83	43.17	60.67
std	50.58	33.28	42.12
min	0.00	0.00	28.00
25%	37.50	20.75	39.75
50%	55.50	44.50	45.00
75%	72.75	61.50	60.00
max	150.00	90.00	143.00

با متد `.T` (بدون پرانتز) میتونیم سطر و ستونها رو جابجا کنیم. میتونیم از ترکیبش با `.T.describe()` هم استفاده کنیم که برای هر سطر بهمون خلاصه آماری بده.

ب برای مرتب کردن سطرها بر اساس ایندکس‌شون از متد `sort_index()` استفاده میکنیم. تو پرانتزش اگر `ascending=False` بنویسیم. سطرها بصورت نزولی رو جابجا میکنه. اگه `ascending=True` باشه بصورت صعودی مرتب میشن که در این مثال سطرها جابجا نمیشن.

```
P_DFarms.sort_index(ascending = False)
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week3	65	143	38	45	28	45
Week2	45	90	67	44	0	13
Week1	66	150	45	75	35	0

```
P_DFarms.sort_index(ascending = True)
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	66	150	45	75	35	0
Week2	45	90	67	44	0	13
Week3	65	143	38	45	28	45

برای مرتب کردن ستونها بر اساس اسامی باید داخل پرانتز تابع بالا `axis = 1` رو بهش بدیم. مقدار پیش فرض این محور ۰ هست. اگه مجدد بهش ۰ بدیم ستونها به حالت اولیه برمیگردن. (column l داره میشه ۱ ، row o داره میشه 0)

```
P_DFarms.sort_index(axis = 1)
```

	Biscuits	Carrots(kg)	Chips	Chocolate	Crackers	Potatoes(kg)
Week1	150	75	35	66	0	45
Week2	90	44	0	45	13	67
Week3	143	45	28	65	45	38

```
P_DFarms.sort_index(axis = 0)
```

	Chocolate	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	66	150	45	75	35	0
Week2	45	90	67	44	0	13
Week3	65	143	38	45	28	45

## بخش دوم:

مجدد یه دیتافریم تعریف میکنیم که مثل قبلی هست و فقط یه Bread اضافه داره. ایندکسهای برچسبی رو هم بهش میدیم و ارزش خروجی میگیریم.

```
P_Dict_Sales = {"Chocolate": [70, 45, 65], "Bread": [63, 51, 59],
                "Biscuits": [150, 90, 143], "Potatoes(kg)": [45, 67, 38],
                "Carrots(kg)": [33, 44, 45], "Chips": [35, 0, 28],
                "Crackers": [0, 13, 45]}
```

```
P_DFrames = pd.DataFrame(P_Dict_Sales)
```

```
P_DFrames = pd.DataFrame(P_Dict_Sales, index = ["Week1", "Week2",
                                                "Week3"])
```

```
P_DFrames
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45

میتونیم به روش ایندکسینگ بهش یه ستون جدید اضافه کنیم:

```
P_DFrames["Onions(kg)"] = [35, 49, 27]
```

```
P_DFrames
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers	Onions(kg)
Week1	70	63	150	45	33	35	0	35
Week2	45	51	90	67	44	0	13	49
Week3	65	59	143	38	45	28	45	27

```
New_Row = {"Chocolate": 77, "Bread": 69,
            "Biscuits": 105, "Potatoes(kg)": 33,
            "Carrots(kg)": 57, "Chips": 44,
            "Crackers": 30, "Onions(kg)": 30}
```

```
P_DFrames = P_DFrames.append(New_Row, ignore_index = True)
```

```
P_DFrames
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers	Onions(kg)
0	70	63	150	45	33	35	0	35
1	45	51	90	67	44	0	13	49
2	65	59	143	38	45	28	45	27
3	77	69	105	33	57	44	30	30
4	77	69	105	33	57	44	30	30

واسه اضافه کردن سطر جدید باید یه متغیر جدید اضافه کنیم و عناصر رو دونه دونه پیدا کنیم. یعنی به هر ستون مقدارش رو بدیم. بعد با متد **append()** این متغیر رو به دیتافریم اضافه کنیم.

مشکل اینه که لیبهامون رو از دست میدیم. باید **ignore\_value** که پیش فرضش **False** هست رو برگردونیم.

```
New_Row2 = {"Chocolate":77,"Bread":69,
            "Biscuits":105,"Potatoes(kg)":33,
            "Carrots(kg)":57,"Chips":44,
            "Crackers":30, "Onions(kg)":30}

P_DFrames = P_DFrames.append(New_Row2, ignore_index =False)

-----
TypeError                                 Traceback (most recent call last)
In [66]:
Line 1:     P_DFrames = P_DFrames.append(New_Row2, ignore_index =False)

File E:\ArcGIS Pro\bin\Python\envs\arcgispro-py3\Lib\site-packages\pandas\core\frame.py, in append:
Line 8931: raise TypeError("Can only append a dict if ignore_index=True")

TypeError: Can only append a dict if ignore_index=True
-----
```

با یه متغیر جدید امتحانش  
میکنیم ولی باز هم خطا دریافت  
میکنیم.

دلیلش اینه که باید موقع  
تعریف متغیر از تابع **pd.Series()**  
استفاده کنیم.

انتهاش هم ایندکس سطر  
چهارم رو بهش بدیم مثل کد زیر :

```
N_R = pd.Series(data = {"Chocolate":35,"Bread":43,
                        "Biscuits":77,"Potatoes(kg)":54,
                        "Carrots(kg)":55,"Chips":23,
                        "Crackers":33, "Onions(kg)":39}, name = "Week4")
```

```
P_DFrames = P_DFrames.append(N_R, ignore_index =False)
```

P\_DFrames

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers	Onions(kg)
Week1	70	63	150	45	33	35	0	NaN
Week2	45	51	90	67	44	0	13	NaN
Week3	65	59	143	38	45	28	45	NaN
Week4	35	43	77	54	55	23	33	39.0

واسه پاک کردن یه ستون از **del** استفاده میکنیم.

```
del P_DFrames["Chips"]
```

P\_DFrames

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Crackers	Onions(kg)
Week1	70	63	150	45	33	0	NaN
Week2	45	51	90	67	44	13	NaN
Week3	65	59	143	38	45	45	NaN
Week4	35	43	77	54	55	33	39.0

```
P_DFrames.pop("Potatoes(kg)")

Week1    45
Week2    67
Week3    38
Week4    54
Name: Potatoes(kg), dtype: int64
```

با متد **pop()** میشه اطلاعات ستونی که پاک میشه رو  
خروجی گرفت.

مجدد از P\_DFrames خروجی بگیر که ببینی کدام ستونها پاک شدن.

P_DFrames						
	Chocolate	Bread	Biscuits	Carrots(kg)	Crackers	Onions(kg)
Week1	70	63	150	33	0	NaN
Week2	45	51	90	44	13	NaN
Week3	65	59	143	45	45	NaN
Week4	35	43	77	55	33	39.0

با متد **drop()** هم میشه یه ستون رو پاک کرد ولی تو پرانتز حتما باید محور یا axis رو مشخص کنیم.

```
P_DFrames = P_DFrames.drop(["Bread", "Carrots(kg)"], axis = 1)
```

P_DFrames			
	Chocolate	Biscuits	Onions(kg)
Week1	70	150	NaN
Week2	45	90	NaN
Week3	65	143	NaN
Week4	35	77	39.0

با کمک متد **drop()** و دادن axis = 0 میتونیم سطرها رو هم پاک کنیم.

```
P_DFrames = P_DFrames.drop(["Crackers"], axis = 1)
```

P_DFrames					
	Chocolate	Bread	Biscuits	Carrots(kg)	Onions(kg)
Week1	70	63	150	33	NaN
Week2	45	51	90	44	NaN
Week3	65	59	143	45	NaN
Week4	35	43	77	55	39.0

فرق این متد با قبلی‌ها اینه که همیشه باهاش چندین ستون رو با هم پاک کرد.

```
P_DFrames = P_DFrames.drop(["Week4"], axis = 0)
```

P_DFrames			
	Chocolate	Biscuits	Onions(kg)
Week1	70	150	NaN
Week2	45	90	NaN
Week3	65	143	NaN

## بخش سوم:

یه دیتافریم جدید با ستونهای قبلی ولی با ۷ تا سطر ایجاد میکنیم.

ایندهای لیبل رو بهش اختصاص میدیم و ارزش خروجی میگیریم.

دیتافریم آماده است که روش تحلیل انجام بدیم.

```
P_Dict_Sales = {"Chocolate": [70, 45, 65, 35, 65, 60, 44],
                "Bread": [63, 51, 59, 46, 55, 71, 39],
                "Biscuits": [150, 90, 143, 87, 79, 56, 135],
                "Potatoes(kg)": [45, 67, 38, 33, 41, 29, 27],
                "Carrots(kg)": [33, 44, 45, 41, 46, 39, 51],
                "Chips": [35, 0, 28, 70, 36, 43, 0],
                "Crackers": [0, 13, 45, 39, 47, 46, 51]}
```

```
P_DFrames = pd.DataFrame(P_Dict_Sales)
```

```
P_DFrames = pd.DataFrame(P_Dict_Sales, index = ["Week1", "Week2",
                                                "Week3", "Week4",
                                                "Week5", "Week6",
                                                "Week7"])
```

P_DFrames							
	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
Week4	35	46	87	33	41	70	39
Week5	65	55	79	41	46	36	47
Week6	60	71	56	29	39	43	46
Week7	44	39	135	27	51	0	51

با متد **head()** میشه از ۵ سطر اول دیتافریم خروجی گرفت.

```
P_DFrames.head()
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
Week4	35	46	87	33	41	70	39
Week5	65	55	79	41	46	36	47

واسه خروجی گرفتن از تعداد سطر مشخص باید تو پرانتز عدد رو وارد کنیم.

```
P_DFrames.head(3)
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45

```
P_DFrames.head(6)
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
Week4	35	46	87	33	41	70	39
Week5	65	55	79	41	46	36	47
Week6	60	71	56	29	39	43	46

```
P_DFrames.tail()
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week3	65	59	143	38	45	28	45
Week4	35	46	87	33	41	70	39
Week5	65	55	79	41	46	36	47
Week6	60	71	56	29	39	43	46
Week7	44	39	135	27	51	0	51

```
P_DFrames.tail(2)
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week6	60	71	56	29	39	43	46
Week7	44	39	135	27	51	0	51

```
P_DFrames.tail(7)
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
Week4	35	46	87	33	41	70	39
Week5	65	55	79	41	46	36	47
Week6	60	71	56	29	39	43	46

واسه خروجی گرفتن از ۵ سطر آخر هم از متد **tail()** استفاده میکنیم. اگه تعداد سطر دیگه‌ای بخوایم میتونیم تو پرانتزش وارد کنیم.



واسه اضافه کردن یه ستون به دیتافریم از متد **insert()** استفاده میکنیم. اولین عدد داخل پرانتز ایندکس موقعیت ستون جدید رو تعیین میکنه.

```
P_DFrames.insert(3, "Onions(kg)", [25,33,31,36,44,19,27])
```

P\_DFrames

	Chocolate	Bread	Biscuits	Onions(kg)	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	25	45	33	35	0
Week2	45	51	90	33	67	44	0	13
Week3	65	59	143	31	38	45	28	45
Week4	35	46	87	36	33	41	70	39
Week5	65	55	79	44	41	46	36	47
Week6	60	71	56	19	29	39	43	46
Week7	44	39	135	27	27	51	0	51

```
P_DFrames["Potatoes(kg)"] = P_DFrames["Potatoes(kg)"] / 10
```

P\_DFrames

	Chocolate	Bread	Biscuits	Onions(kg)	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	25	4.5	33	35	0
Week2	45	51	90	33	6.7	44	0	13
Week3	65	59	143	31	3.8	45	28	45
Week4	35	46	87	36	3.3	41	70	39
Week5	65	55	79	44	4.1	46	36	47
Week6	60	71	56	19	2.9	39	43	46
Week7	44	39	135	27	2.7	51	0	51

میتونیم روی ستونها عملیات ریاضی انجام بدیم.

```
P_DFrames["Carrots(kg)"] = P_DFrames["Carrots(kg)"] + 5
```

P\_DFrames

	Chocolate	Bread	Biscuits	Onions(kg)	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	25	4.5	38	35	0
Week2	45	51	90	33	6.7	49	0	13
Week3	65	59	143	31	3.8	50	28	45
Week4	35	46	87	36	3.3	46	70	39
Week5	65	55	79	44	4.1	51	36	47
Week6	60	71	56	19	2.9	44	43	46
Week7	44	39	135	27	2.7	56	0	51

کد بالا رو همیشه به صورت += هم نوشت.

```
P_DFrames["Carrots(kg)"] += 5
```

P\_DFrames

	Chocolate	Bread	Biscuits	Onions(kg)	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	25	4.5	43	35	0
Week2	45	51	90	33	6.7	54	0	13
Week3	65	59	143	31	3.8	55	28	45
Week4	35	46	87	36	3.3	51	70	39
Week5	65	55	79	44	4.1	56	36	47
Week6	60	71	56	19	2.9	49	43	46
Week7	44	39	135	27	2.7	61	0	51

```
P_DFrames["Chocolate"].value_counts()
```

```
65    2
70    1
45    1
35    1
60    1
44    1
Name: Chocolate, dtype: int64
```

واسه اینکه ببینم از هر عدد تو یه ستون چند تا داریم از متد `value_counts()` استفاده می‌کنیم.

### عمگرهای groupby

این اصطلاحی هست که واسه کار کردن با گروهی از داده‌ها در دیتافریم استفاده میشه. از پانداس واسه گروه‌بندی، سازماندهی و کشف اطلاعات جدولی در حجم خیلی بالا استفاده میشه ولی بعضی اوقات نیاز هست که واسه انجام تحلیلهای بیشتر دیتافریم پانداس رو به زیرمجموعه‌های مختلفی تقسیم کنیم. در واقع تحلیلهای بعدی که روی این تقسیم‌بندی زیرمجموعه‌ها انجام میشه رو groupby انجام میده. یه دیتافریم میسازیم.

```
import numpy as np
import pandas as pd
```

```
Covid_data = {"Country": ["Japan", "Malesia", "Spain", "Singapore", "Italy", "Morocco", "Germany", "Egypt", "USA", "Canada", "Brazil"],
               "Continent": ["Asia", "Asia", "Europe", "Asia", "Europe", "Africa", "Europe", "Africa", "America", "America", "America"],
               "Case": [455, 93, 430, 138, 5883, 2, 795, 48, 213, 57, 19],
               "Death": ["Yes", "No", "Yes", "No", "Yes", "No", "No", "No", "Yes", "No", "No"],
               "Death_C": [6, 0, 5, 0, 234, 0, 0, 0, 11, 0, 0]}
```

```
DF_Cov = pd.DataFrame(Covid_data)
```

DF\_Cov

	Country	Continent	Case	Death	Death_C
0	Japan	Asia	455	Yes	6
1	Malesia	Asia	93	No	0
2	Spain	Europe	430	Yes	5
3	Singapore	Asia	138	No	0
4	Italy	Europe	5883	Yes	234
5	Morocco	Africa	2	No	0
6	Germany	Europe	795	No	0
7	Egypt	Africa	48	No	0
8	USA	America	213	Yes	11
9	Canada	America	57	No	0
10	Brazil	America	19	No	0

حالا میخوایم با متد **groupby()** کار کنیم. فقط قاره‌ها رو ازش میکشیم بیرون.

```
DF_Cov_G = DF_Cov.groupby("Continent")
```

```
DF_Cov_G
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000167977D80D0>
```

با متد **groups** بدون پرانتز همیشه از قاره‌ها که ۴ تا هستن و ایندکس‌هاشون خروجی گرفت.

```
DF_Cov_G.groups
```

```
{'Africa': [5, 7], 'America': [8, 9, 10], 'Asia': [0, 1, 3], 'Europe': [2, 4, 6]}
```

اگه بخوایم این خروجی رو سطر به سطر داشته باشیم باید از حلقه for استفاده کنیم.

```
for i,j in DF_Cov_G:
```

```
    print(i)
```

```
    print(j)
```

Africa

	Country	Continent	Case	Death	Death_C
5	Morocco	Africa	2	No	0
7	Egypt	Africa	48	No	0

America

	Country	Continent	Case	Death	Death_C
8	USA	America	213	Yes	11
9	Canada	America	57	No	0
10	Brazil	America	19	No	0

Asia

	Country	Continent	Case	Death	Death_C
0	Japan	Asia	455	Yes	6
1	Malaysia	Asia	93	No	0
3	Singapore	Asia	138	No	0

Europe

	Country	Continent	Case	Death	Death_C
2	Spain	Europe	430	Yes	5
4	Italy	Europe	5883	Yes	234
6	Germany	Europe	795	No	0

```
DF_Cov_G.get_group("Europe")
```

	Country	Continent	Case	Death	Death_C
2	Spain	Europe	430	Yes	5
4	Italy	Europe	5883	Yes	234
6	Germany	Europe	795	No	0

اگه بخوایم برای مثال فقط اطلاعات اروپا رو داشته باشیم

باید از متد **get\_group()** استفاده کنیم.

```
DF_Cov_G.size()
```

```
Continent
```

```
Africa      2
```

```
America     3
```

```
Asia        3
```

```
Europe      3
```

```
dtype: int64
```

با متد **size()** همیشه تعداد یه مقدار رو در سطرها شمرد.

همینکار رو برای بیش از یه ستون هم میشه انجام داد.

```
Df_Cov_M = DF_Cov.groupby(["Continent", "Death"])
```

```
Df_Cov_M.size()
```

```
Continent  Death
Africa     No      2
America    No      2
           Yes      1
Asia       No      2
           Yes      1
Europe     No      1
           Yes      2
dtype: int64
```

```
for i,j in Df_Cov_M:
    print(i)
    print(j)
```

```
('Africa', 'No')
   Country Continent  Case Death  Death_C
5  Morocco   Africa     2    No         0
7   Egypt   Africa    48    No         0
('America', 'No')
   Country Continent  Case Death  Death_C
9  Canada   America    57    No         0
10 Brazil   America    19    No         0
('America', 'Yes')
   Country Continent  Case Death  Death_C
8    USA   America   213    Yes        11
('Asia', 'No')
   Country Continent  Case Death  Death_C
1  Malesia   Asia     93    No         0
3  Singapore   Asia   138    No         0
('Asia', 'Yes')
   Country Continent  Case Death  Death_C
0   Japan   Asia   455    Yes          6
('Europe', 'No')
   Country Continent  Case Death  Death_C
6  Germany   Europe   795    No         0
('Europe', 'Yes')
   Country Continent  Case Death  Death_C
2   Spain   Europe   430    Yes          5
4   Italy   Europe  5883    Yes       234
```

کد زیر هم لیستی از قاره ها ایجاد میکنه که پشت سر هم نوشته میشن.

```
continent_names = DF_Cov["Continent"].unique()
continent_grouped = DF_Cov.groupby("Continent")
tables_continent = [continent_grouped.get_group(continent) for continent in continent_names]
```

```
tables_continent
```

```
[
   Country Continent  Case Death  Death_C
0   Japan   Asia   455    Yes          6
1  Malesia   Asia    93    No          0
3  Singapore   Asia   138    No          0,   Country Continent  Case Death  Death_C
2   Spain   Europe   430    Yes          5
4   Italy   Europe  5883    Yes       234
6  Germany   Europe   795    No          0,   Country Continent  Case Death  Death_C
5  Morocco   Africa     2    No          0
7   Egypt   Africa    48    No          0,   Country Continent  Case Death  Death_C
8    USA   America   213    Yes         11
9  Canada   America    57    No          0
10 Brazil   America    19    No          0]
```

```
Df_Cov_M.aggregate(np.sum)
```

```
Case  Death_C
Continent  Death
Africa     No      50         0
           Yes     213        11
America    No      76         0
           Yes     213        11
Asia       No     231         0
           Yes     455         6
Europe     No     795         0
           Yes    6313        239
```

از متد **aggregate()** واسه انجام عملیات تجمیع روی یه محور مشخص استفاده میشه که پیش فرض  $axis = 0$  هست. تو پرانتزش باید بنویسیم که چه کاری روی داده‌ها انجام بده.

با متد **agg()** هم همین نتیجه رو میگیریم. میتونیم تو پرانتزش به جای sum از mean هم استفاده کنیم.

```
Df_Cov_M.agg(np.sum)
```

		Case		Death_C
Continent	Death			
Africa	No	50		0
	Yes	213		11
America	No	76		0
	Yes	455		6
Asia	No	231		0
	Yes	455		6
Europe	No	795		0
	Yes	6313		239

```
Df_Cov_M.agg(np.mean)
```

		Case		Death_C
Continent	Death			
Africa	No	25.0		0.0
	Yes	213.0		11.0
America	No	38.0		0.0
	Yes	455.0		6.0
Asia	No	115.5		0.0
	Yes	455.0		6.0
Europe	No	795.0		0.0
	Yes	3156.5		119.5

```
Df_Cov_M.agg(np.mean).astype(int)
```

		Case		Death_C
Continent	Death			
Africa	No	25		0
	Yes	213		11
America	No	38		0
	Yes	455		6
Asia	No	115		0
	Yes	455		6
Europe	No	795		0
	Yes	3156		119

واسه اینکه اعدادمون حالت عدد صحیح پیدا کنن از **astype()** استفاده میکنیم.

میشه همینکار رو روی بیش از یه ستون هم انجام داد.

```
Df_Cov_M["Death_C", "Case"].agg([np.sum, np.mean, np.max]).astype(int)
```

		Death_C			Case		
		sum	mean	amax	sum	mean	amax
Continent Death							
Africa	No	0	0	0	50	25	48
	Yes	11	11	11	213	213	213
America	No	0	0	0	76	38	57
	Yes	6	6	6	455	455	455
Asia	No	0	0	0	231	115	138
	Yes	6	6	6	455	455	455
Europe	No	0	0	0	795	795	795
	Yes	239	119	234	6313	3156	5883

## ترکیب دیتافریمها Combining DataFrames

گاهی داده‌ها از منابع مختلفی دستمون میرسن که باید با هم یکی شن.

باز هم نامپی و پانداس رو وارد نوت بوک میکنیم.

باید چند تا دیتافریم درست کنیم که بعد بتونیم با هم ادغامشون کنیم.

```
import numpy as np
import pandas as pd
```

```
Sales_1 = {"Chocolate": [70, 45, 65, 35], "Bread": [63, 51, 59, 46],
           "Biscuits": [150, 90, 143, 87], "Potatoes(kg)": [45, 67, 38, 33],
           "Carrots(kg)": [33, 44, 45, 41], "Chips": [35, 0, 28, 70],
           "Crackers": [0, 13, 45, 39]}
```

```
DF_Months_1 = pd.DataFrame(Sales_1, index = ["Week1", "Week2",
                                             "Week3", "week4"])
```

DF\_Months\_1

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
week4	35	46	87	33	41	70	39

```
Sales_2 = {"Chocolate": [40, 35, 60, 44], "Bread": [23, 57, 39, 71],
           "Biscuits": [50, 70, 45, 56], "Potatoes(kg)": [25, 61, 33, 41],
           "Carrots(kg)": [53, 47, 44, 39], "Chips": [35, 13, 28, 36],
           "Crackers": [27, 13, 45, 47]}
```

```
DF_Months_2 = pd.DataFrame(Sales_2, index = ["Week1", "Week2",
                                             "Week3", "week4"])
```

DF\_Months\_2

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	40	23	50	25	53	35	27
Week2	35	57	70	61	47	13	13
Week3	60	39	45	33	44	28	45
week4	44	71	56	41	39	36	47

برای دیتافریم سوم هم از روی دیتافریم اول کپی کردیم.

```
DF_Months_3 = pd.DataFrame(Sales_1, index = ["Week1", "Week2",
                                             "Week3", "week4"])
```

DF\_Months\_3

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
week4	35	46	87	33	41	70	39

```
All_Sales = pd.concat([DF_Months_1,DF_Months_2,DF_Months_3])
```

All\_Sales

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
week4	35	46	87	33	41	70	39
Week1	40	23	50	25	53	35	27
Week2	35	57	70	61	47	13	13
Week3	60	39	45	33	44	28	45
week4	44	71	56	41	39	36	47
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
week4	35	46	87	33	41	70	39

با تابع **pd.concat()** با هم یکیشون میکنیم.

ایندکسهاشون میتونه ما رو گمراه کنه. واسه همین باید ایندکسهاشون رو نادیده بگیریم. باید **ignore\_index** **= True** باشه.

```
All_Sales_reindex = pd.concat([DF_Months_1,DF_Months_2,
                                DF_Months_3], ignore_index = True)
```

All\_Sales\_reindex

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
0	70	63	150	45	33	35	0
1	45	51	90	67	44	0	13
2	65	59	143	38	45	28	45
3	35	46	87	33	41	70	39
4	40	23	50	25	53	35	27
5	35	57	70	61	47	13	13
6	60	39	45	33	44	28	45
7	44	71	56	41	39	36	47
8	70	63	150	45	33	35	0
9	45	51	90	67	44	0	13
10	65	59	143	38	45	28	45
11	35	46	87	33	41	70	39

برای اینکه مشخص کنیم هر داده از کدام دیتافریم آمده از **keys** استفاده میکنیم.

```
All_Sales_keys = pd.concat([DF_Months_1,DF_Months_2,DF_Months_3],
                             keys=["Month1","Month2","Manth3"])
```

All\_Sales\_keys

		Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Month1	Week1	70	63	150	45	33	35	0
	Week2	45	51	90	67	44	0	13
	Week3	65	59	143	38	45	28	45
	week4	35	46	87	33	41	70	39
Month2	Week1	40	23	50	25	53	35	27
	Week2	35	57	70	61	47	13	13
	Week3	60	39	45	33	44	28	45
	week4	44	71	56	41	39	36	47
Manth3	Week1	70	63	150	45	33	35	0
	Week2	45	51	90	67	44	0	13
	Week3	65	59	143	38	45	28	45
	week4	35	46	87	33	41	70	39

```
All_Sales_keys.loc["Month2"]
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	40	23	50	25	53	35	27
Week2	35	57	70	61	47	13	13
Week3	60	39	45	33	44	28	45
week4	44	71	56	41	39	36	47

با متد **loc()** میشه دیتافریمی که میخوایم رو مجدد دریافت کنیم.

```
All_Sales_keys = pd.concat([DF_Months_1,DF_Months_2,DF_Months_3],
                             keys=["Month1","Month2","Manth3"],
                             axis = 0)
```

All\_Sales\_keys

		Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Month1	Week1	70	63	150	45	33	35	0
	Week2	45	51	90	67	44	0	13
	Week3	65	59	143	38	45	28	45
	week4	35	46	87	33	41	70	39
Month2	Week1	40	23	50	25	53	35	27
	Week2	35	57	70	61	47	13	13
	Week3	60	39	45	33	44	28	45
	week4	44	71	56	41	39	36	47
Manth3	Week1	70	63	150	45	33	35	0
	Week2	45	51	90	67	44	0	13
	Week3	65	59	143	38	45	28	45
	week4	35	46	87	33	41	70	39

پیش فرض محور صفر هست یعنی چه  $axes=0$  رو بنویسیم چه ننویسیم نتیجه مشابه هست.



ولی وقتی  $axis = 1$  بدیم نتیجه متفاوت میشه. چون ایندکسها مشابه هست اطلاعات ۴ تا ایندکس رو روی سطر نگه میداره و اطلاعات دیتافریمها رو بر اساس ماهها روی ستونها پخش میکنه.

All\_Sales\_keys = pd.concat([DF\_Months\_1,DF\_Months\_2,DF\_Months\_3], keys=["Month1","Month2","Manth3"], axis = 1)

All\_Sales\_keys

	Month1							Month2							Manth3						
	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0	40	23	50	25	53	35	27	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13	35	57	70	61	47	13	13	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45	60	39	45	33	44	28	45	65	59	143	38	45	28	45
week4	35	46	87	33	41	70	39	44	71	56	41	39	36	47	35	46	87	33	41	70	39

اطلاعات دیافریم دو رو مجدد کپی میکنیم و اینبار ایندکسهای متفاوت بهش میدیم. مجدد با تابع `pd.concat()` جمعشون می‌کنیم. `key`ها رو بهش میدیم و یکبار با  $axis=0$  و یکبار هم با  $axis = 1$  امتحانش میکنیم.

```
All_Sales_keys = pd.concat([DF_Months_1,DF_Months_2,DF_Months_3], keys=["Month1","Month2","Manth3"], axis = 0)
```

		Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Month1	Week1	70	63	150	45	33	35	0
	Week2	45	51	90	67	44	0	13
	Week3	65	59	143	38	45	28	45
	week4	35	46	87	33	41	70	39
Month2	Week4	40	23	50	25	53	35	27
	Week5	35	57	70	61	47	13	13
	Week6	60	39	45	33	44	28	45
	week7	44	71	56	41	39	36	47
Manth3	Week1	70	63	150	45	33	35	0
	Week2	45	51	90	67	44	0	13
	Week3	65	59	143	38	45	28	45
	week4	35	46	87	33	41	70	39

```

All_Sales_keys = pd.concat([DF_Months_1,DF_Months_2,DF_Months_3],
                            keys=["Month1","Month2","Manth3"],
                            axis = 1)

```

All\_Sales\_keys

	Month1							Month2							Manth3						
	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70.0	63.0	150.0	45.0	33.0	35.0	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	70.0	63.0	150.0	45.0	33.0	35.0	0.0
Week2	45.0	51.0	90.0	67.0	44.0	0.0	13.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	45.0	51.0	90.0	67.0	44.0	0.0	13.0
Week3	65.0	59.0	143.0	38.0	45.0	28.0	45.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	65.0	59.0	143.0	38.0	45.0	28.0	45.0
week4	35.0	46.0	87.0	33.0	41.0	70.0	39.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	35.0	46.0	87.0	33.0	41.0	70.0	39.0
Week4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	40.0	23.0	50.0	25.0	53.0	35.0	27.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Week5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	35.0	57.0	70.0	61.0	47.0	13.0	13.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Week6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	60.0	39.0	45.0	33.0	44.0	28.0	45.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
week7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	44.0	71.0	56.0	41.0	39.0	36.0	47.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

با axis=1 تو ۷ تا ایندکس بهمون نشون میده و اگه تو ماه‌ها این ایندکس وجود نداشته باشه به صورت NaN سلولهای جدول رو پر میکنه.

مجدد اطلاعات دیتافریم دوم رو با ایندکسهای قبلی کپی میکنیم و بدون axis ازش خروجی میگیریم. و یکبار هم با axis=1 که اطلاعات به حالت اولیه برگرده.

```
M_A_DF = DF_Months_1.append([DF_Months_2,DF_Months_3])
```

M\_A\_DF

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
week4	35	46	87	33	41	70	39
Week1	40	23	50	25	53	35	27
Week2	35	57	70	61	47	13	13
Week3	60	39	45	33	44	28	45
week4	44	71	56	41	39	36	47
Week1	70	63	150	45	33	35	0
Week2	45	51	90	67	44	0	13
Week3	65	59	143	38	45	28	45
week4	35	46	87	33	41	70	39

میخوایم از متد **append()** واسه وصل کردن دیتافریمها استفاده کنیم.

شکل نوشتن کدش متفاوت هست باید یکی از دیتافریمها رو انتخاب کنیم مثلا اولی. بعد **append()** رو بهش بچسبونیم و تو پرانتزش اون دوتای دیگه رو بنویسیم که با هم یکی شن.

```
M_A_DF = DF_Months_1.append([DF_Months_2,DF_Months_3], ignore_index = True)
```

M\_A\_DF

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
0	70	63	150	45	33	35	0
1	45	51	90	67	44	0	13
2	65	59	143	38	45	28	45
3	35	46	87	33	41	70	39
4	40	23	50	25	53	35	27
5	35	57	70	61	47	13	13
6	60	39	45	33	44	28	45
7	44	71	56	41	39	36	47
8	70	63	150	45	33	35	0
9	45	51	90	67	44	0	13
10	65	59	143	38	45	28	45
11	35	46	87	33	41	70	39

یکبار دیگه با **ignore\_index = True** مینویسیمش که ببینیم چی برمیگردونه.

با توابع **merge()** و **join()** هم میشه سریها و دیتافریمها رو با هم ترکیب کرد.

میخوایم دیتافریم ۱ و ۲ رو با تابع **pd.merge()** با هم ترکیب کنیم. وقتی کد زیر رو بنویسیم، پایتون فقط سرستونها رو بهمون خروجی میده و داده‌ای رو نشون نمیده. واسه اینکه باید از آرگومان **on** استفاده کنیم.

```
M_Data_1 = pd.merge(DF_Months_1,DF_Months_2)
```

```
M_Data_1
```

Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
-----------	-------	----------	--------------	-------------	-------	----------

# merge()

right  
on  
left\_on  
right\_on

how  
sort  
suffixes  
copy

left\_index  
right\_index  
indicator  
validate

چون تابع merge() واسه ترکیب دیتافریمها نیاز به نام ستون داره. باید ستونی رو معرفی کنیم که بین دو دیتافریم داده مشترک توش داشته باشه. اینجا Chips دو تا مقدار مشترک تو هر دو تا دیتافریم داره. وقتی بهش بدیم فقط سطریایی رو برمیگردونه که داده مشترک این ستون شاملش هست.

```
In [54]: M_Data_1 = pd.merge(DF_Months_1,DF_Months_2, on ="Chips")
```

```
In [55]: M_Data_1
```

```
Out[55]:
```

	Chocolate_x	Bread_x	Biscuits_x	Potatoes(kg)_x	Carrots(kg)_x	Chips	Crackers_x	Chocolate_y	Bread_y	Biscuits_y	Potatoes(kg)_y	Carrots(kg)_y	Crackers_y
0	70	63	150	45	33	35	0	40	23	50	25	53	27
1	65	59	143	38	45	28	45	60	39	45	33	44	45

```
M_Data_2 = pd.merge(DF_Months_1,DF_Months_2, how ="outer")
```

```
M_Data_2
```

	Chocolate	Bread	Biscuits	Potatoes(kg)	Carrots(kg)	Chips	Crackers
0	70	63	150	45	33	35	0
1	45	51	90	67	44	0	13
2	65	59	143	38	45	28	45
3	35	46	87	33	41	70	39
4	40	23	50	25	53	35	27
5	35	57	70	61	47	13	13
6	60	39	45	33	44	28	45
7	44	71	56	41	39	36	47

همونطور که تو تصویر بالا می بینیم تابع merge() پارامتر دیگه به جز on هم داره که how هست. چند مدل وصل کردن داریم که شامل inner join, full other join, left other join, right outer join می شه. کد بالا رو با inner join نوشتیم. میخوایم با outer بنویسمش.

میتونیم پارامتر on رو هم بهش اضافه کنیم. میبینیم که دو تا از سطرها حذف میشن و تو بعضی از سطرها NaN دیده میشه. چون فقط اطلاعات فیلهایی رو نشون میده که با Chips بین دو تا دیتافریم مشترک هستن.

```
M_Data_2 = pd.merge(DF_Months_1,DF_Months_2, how ="outer", on ="Chips")
```

	Chocolate_x	Bread_x	Biscuits_x	Potatoes(kg)_x	Carrots(kg)_x	Chips	Crackers_x	Chocolate_y	Bread_y	Biscuits_y	Potatoes(kg)_y	Carrots(kg)_y	Crackers_y
0	70.0	63.0	150.0	45.0	33.0	35	0.0	40.0	23.0	50.0	25.0	53.0	27.0
1	45.0	51.0	90.0	67.0	44.0	0	13.0	NaN	NaN	NaN	NaN	NaN	NaN
2	65.0	59.0	143.0	38.0	45.0	28	45.0	60.0	39.0	45.0	33.0	44.0	45.0
3	35.0	46.0	87.0	33.0	41.0	70	39.0	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	13	NaN	35.0	57.0	70.0	61.0	47.0	13.0
5	NaN	NaN	NaN	NaN	NaN	36	NaN	44.0	71.0	56.0	41.0	39.0	47.0

حالا بخش how رو با left امتحان می‌کنیم. اینبار سطرها کمتر میشن و اطلاعات دیتافریم اول کامل میمونه و از دومی فقط اونایی رو نشون میده که تو ستون Chips اطلاعات مشترک دارن.

```
In [60]: M_Data_3 = pd.merge(DF_Months_1,DF_Months_2, how ="left", on ="Chips")
```

```
In [61]: M_Data_3
```

```
Out[61]:
```

	Chocolate_x	Bread_x	Biscuits_x	Potatoes(kg)_x	Carrots(kg)_x	Chips	Crackers_x	Chocolate_y	Bread_y	Biscuits_y	Potatoes(kg)_y	Carrots(kg)_y	Crackers_y
0	70	63	150	45	33	35	0	40.0	23.0	50.0	25.0	53.0	27.0
1	45	51	90	67	44	0	13	NaN	NaN	NaN	NaN	NaN	NaN
2	65	59	143	38	45	28	45	60.0	39.0	45.0	33.0	44.0	45.0
3	35	46	87	33	41	70	39	NaN	NaN	NaN	NaN	NaN	NaN

اگه کد بالا رو با right امتحان کنیم اطلاعات دیتافریم دوم رو کامل نگه میداره و از اولی فقط اونایی رو نشون میده که با ستون Chips اطلاعات مشترک دارن.

```
M_Data_3 = pd.merge(DF_Months_1,DF_Months_2, how ="right", on ="Chips")
```

```
M_Data_3
```

	Chocolate_x	Bread_x	Biscuits_x	Potatoes(kg)_x	Carrots(kg)_x	Chips	Crackers_x	Chocolate_y	Bread_y	Biscuits_y	Potatoes(kg)_y	Carrots(kg)_y	Crackers_y
0	70.0	63.0	150.0	45.0	33.0	35	0.0	40	23	50	25	53	27
1	NaN	NaN	NaN	NaN	NaN	13	NaN	35	57	70	61	47	13
2	65.0	59.0	143.0	38.0	45.0	28	45.0	60	39	45	33	44	45
3	NaN	NaN	NaN	NaN	NaN	36	NaN	44	71	56	41	39	47

این تابع به کمی از تابع concat() پیچیده‌تر هست. با تابع pd.join() میشه دیتافریمهایی رو با هم ترکیب کرد و نیازی به اینکه اطلاعات ستون مشترک داشته باشن نداره.

```
Join_Right = pd.DataFrame({"Apple":["Apfel","Mela","Mansana","pomme"],
                           "Orange":["Orange","Arancia","naranja","Orange"],
                           "Banana":["Banana","Banana","Platano","Banana"]},
                           index = ["German","Italian","Spanish","French"])
```

```
Join_Right
```

	Apple	Orange	Banana
German	Apfel	Orange	Banana
Italian	Mela	Arancia	Banana
Spanish	Mansana	naranja	Platano
French	pomme	Orange	Banana

دو تا دیتافریم به اسم Join\_Left و Join\_Right تعریف می‌کنیم.

اولی ۴ تا ایندکس داره و دومی ۵ تا.

```
Join_Left = pd.DataFrame({"One":["Ein", "Uno", "Una", "Un", "Unus"],
                          "Two":["Zwei", "Duo", "Dos", "Duex", "Duo"],
                          "Three":["Drei", "Tre", "Tres", "Trois", "Treibus"]},
                          index = ["German", "Italian", "Spanish", "French",
                                   "Latin"])
```

Join\_Left

	One	Two	Three
German	Ein	Zwei	Drei
Italian	Uno	Duo	Tre
Spanish	Una	Dos	Tres
French	Un	Duex	Trois
Latin	Unus	Duo	Treibus

```
Result_1 = Join_Right.join(Join_Left)
```

Result\_1

	Apple	Orange	Banana	One	Two	Three
German	Apfel	Orange	Banana	Ein	Zwei	Drei
Italian	Mela	Arancia	Banana	Uno	Duo	Tre
Spanish	Mansana	naranga	Platano	Una	Dos	Tres
French	pomme	Orange	Banana	Un	Duex	Trois

وقتی این دو تا دیتافریم با هم جمع میشن، پایتون ایندکس غیرمشابه رو از دیتافریم دوم حذف میکنه.

```
Result_2 = Join_Left.join(Join_Right)
```

Result\_2

	One	Two	Three	Apple	Orange	Banana	key
German	Ein	Zwei	Drei	NaN	NaN	NaN	NaN
Italian	Uno	Duo	Tre	NaN	NaN	NaN	NaN
Spanish	Una	Dos	Tres	NaN	NaN	NaN	NaN
French	Un	Duex	Trois	NaN	NaN	NaN	NaN
Latin	Unus	Duo	Treibus	NaN	NaN	NaN	NaN

حالا اگه دیتافریم دوم رو که تعداد ایندکس بیشتری داره اول بیاریم، ایندکسی حذف نمیشه فقط جواب اطلاعات NaN پر میکنه.

یکبار دیگه دیتافریم اول رو به جای ایندکس با key مینویسم و با دیتافرسم دوم join میکنیم و پارامتر on رو معادل key قرار میدیم. می بینیم که اطلاعات کامل می آد.

```
Result_3 = Join_Right.join(Join_Left, on = "key")
```

Result\_3

	Apple	Orange	Banana	key	One	Two	Three
0	Apfel	Orange	Banana	German	Ein	Zwei	Drei
1	Mela	Arancia	Banana	Italian	Uno	Duo	Tre
2	Mansana	naranga	Platano	Spanish	Una	Dos	Tres
3	pomme	Orange	Banana	French	Un	Duex	Trois

## وارد کردن داده‌ها با فرمت csv و Excel

همیشه ما با داده‌هایی که خودمون تولید میکنیم کار انجام نمیدیم. گاهی داده‌هایی بدستمون میرسه که دیگران تولیدش کردن. این داده‌ها واسه ورود به کتابخونه‌های پایتون باید یا فرمت csv داشته باشن یا xlsx.

CSV(Comma Separated Values) داده‌هایی هستن که با ویرگول از هم جدا میشن.

واسه اضافه کردنشون باید از تابع `pd.read_csv()` استفاده کنیم و تو پرانتزش اسم فایل رو با پسوندش داخل کوتیشن بنویسیم. این فایل باید دقیقا تو مسیری باشه که نوت بوکمون توش ذخیره شده که بتونه بخوندش.

واسه اینکه بتونیم یه سری داده واسه تمرین پیدا کنیم میتونیم بریم تو سایتهای زیر و یه سری مجموعه داده رو دانلود کنیم:

<https://www.kaggle.com/>

<https://data.gov/>

<https://piktochart.com/>

من از سایت اول اطلاعات کرونا رو دانلود کردم (قبلش باید ثبتنام کنید. با وی پی ان وصل شدم). اطلاعات زیادی بهم داد. یکی از اطلاعات رو که مربوط به WHO بود رو کپی کردم و گذاشتن تو مسیری که نوت بوکمون توش ذخیره شده بود. یادت باشه موقعی که اسم فایل رو تو پرانتز داخل کوتیشن مینویسی حتما پسوند csv. رو وارد کنی.

این مجموعه داده ۳۷۹۵۵۰ تا سطر داده ولی واسه اینکه خلاصه نوشته شه ۵ سطر اول و ۵ سطر آخرش برامون

می‌آد.

```
import numpy as np
import pandas as pd
```

```
Covid = pd.read_csv("covid-19-county-level-data.csv")
```

Covid

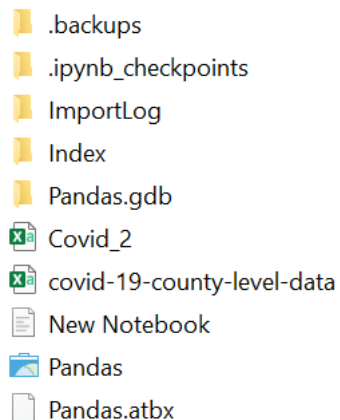
	Unnamed: 0	date	county	state	fips	cases	deaths
0	0	2020-01-21	Snohomish	Washington	53061.0	1	0
1	1	2020-01-22	Snohomish	Washington	53061.0	1	0
2	2	2020-01-23	Snohomish	Washington	53061.0	1	0
3	3	2020-01-24	Cook	Illinois	17031.0	1	0
4	4	2020-01-24	Snohomish	Washington	53061.0	1	0
...	...	...	...	...	...	...	...
379545	379545	2020-07-28	Sweetwater	Wyoming	56037.0	228	2
379546	379546	2020-07-28	Teton	Wyoming	56039.0	311	1
379547	379547	2020-07-28	Uinta	Wyoming	56041.0	243	1
379548	379548	2020-07-28	Washakie	Wyoming	56043.0	44	5
379549	379549	2020-07-28	Weston	Wyoming	56045.0	5	0

379550 rows × 7 columns

میتونیم یه مجموعه داده درست کنیم و به فرمت csv ذخیره‌اش کنیم. از متد `to_csv()` استفاده میکنیم.

یه فایل به اسمی که تو پرانتز وارد کردیم تو مسیری که نوت بوک ذخیره شده ساخته میشه.

```
Covid_2 = Covid  
Covid_2.to_csv("Covid_2.csv")
```



- .backups
- .ipynb\_checkpoints
- ImportLog
- Index
- Pandas.gdb
- Covid\_2
- covid-19-county-level-data
- New Notebook
- Pandas
- Pandas.atbx

همینکار رو با مجموعه داده‌های اکسل با فرمت xlsx هم میشه انجام داد. باید با `pd.read_excel()` بنویسیم:

```
Covid2 = pd.read_excel("canada-cumulative-case-counts.xlsx")
```

## خلاصه دستوره‌های pandas

توضیحات	توابع و متدها
<b>ساختار داده Series</b>	
<p>واسه ساخت <b>ساختار داده Series</b> استفاده میشه. که چند تا پارامتر داره.</p> <p>این پارامترها فرمهای متنوعی میگیرن شامل:</p> <p>۶- <b>data</b>: شامل constants یا یه عدد ثابت، lists یا لیستها، ndarray آرایه چندبعدی</p> <p>۷- <b>index</b>: مقدار ایندکس هست</p> <p>۸- <b>dtype</b>: همون data type هست. نوع داده سریهای خروجی رو تعیین میکنه.</p> <p>۹- <b>name</b>: نام سری</p> <p>۱۰- <b>copy</b>: داده ورودی رو کپی میکنه. مقدار پیشفرضش False هست.</p>	<p>pandas.Series(data, index, dtype, name, copy)</p>
واسه دریافت <b>خلاصه آماری</b> از Series	.describe()
واسه دریافت <b>خروجی از ۵ سطر اول</b> . ۵ پیش فرض هست، اگه عدد دیگه‌ای بخوایم باید تو پرانتزش وارد کنیم.	.head()
واسه دریافت <b>خروجی از ۵ سطر آخر</b> . ۵ پیش فرض هست، اگه عدد دیگه‌ای بخوایم باید تو پرانتزش وارد کنیم.	.tail()
واسه <b>وارد کردن برچسب به جای ایندکسهای عددی</b> در اول سطرهای خروجی. برای دسترسی به اعضا میشه این برچسبها رو با کوتیشن تو براکت جلوی اسم Series نوشت یا با نقطه بدون کوتیشن بهش چسبونند. به جای تعریف ایندکس میشه اعضا رو به صورت دیکشنری هم تعریف کرد: {key:value}	Index = []
از این متد واسه مشخص کردن اینکه یه سری <b>چه رشته‌هایی</b> داره یا اینکه اطلاعات رشته‌ای سری شامل <b>چه حروفی</b> هستن استفاده میشه. میتونیم تو پرانتز حرف رو وارد کنیم و ببینیم وجود داره یا نه. جواب رو به صورت داده بولینی بهمون برمیگردونه.	.str.contains()
اگه بخوایم که اعضای رشته یه سری رو به <b>صورت حروف بزرگ</b> بهمون نشون بده	.str.upper()
<b>ساختار داده DataFrame</b>	
<p>واسه <b>ساخت دیتافریم</b> یا جدول از داده‌ها استفاده میشه. اطلاعاتش به صورت دیکشنری وارد میشن که کلیدها سر ستونها هستن و مقادیر هم اطلاعات سلولها. یه ستون اضافه هم در خروجی داریم که ایندکسها رو نشون میده که میتونیم شخصی سازی کنیم. تعداد ایندکسها از تعداد مقادیر نباید بیشتر یا کمتر باشه.</p>	<p>pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)</p>



.loc()	میتونیم اطلاعات یک سطر یا ستون رو به صورت کامل به همراه اسامی همه ستونها داشته باشیم. به جای پرانتز از [] به همراه اسم ایندکس اون سطر استفاده میکنیم.
.iloc()	واسه دسترسی به اطلاعات به سطر یا ستون به جای پرانتز از [] به همراه شماره ایندکس سطر مورد نظر استفاده میشه. داخل [] میشه اطلاعات سطر و ستونهای مشخصی رو وارد کرد که برامون برش بزنه.
.at() / .iat()	واسه انتخاب به مقدار مشخص از دیتافریم از .at() و .iat() طبق قاعده بالا استفاده میکنیم. اولی از برچسب یا label استفاده میکنه و دومی از عدد صحیح. باید بین سطر و ستون تعیین اون عدد مشخص کما بذاریم.
.describe()	دریافت خلاصه آماری از هر ستون جدول. اعداد خروجی با ۶ رقم اعشار دیده میشن.
pd.set_option("precision" = 2)	برای تغییر رقم اعشار از ۶ به عدد دلخواه مثلا ۲ رقم اعشار استفاده میشه.
.sum()/.min()/.max()/.mean()/.std()	واسه دریافت مجموع، کمترین عدد، بیشترین عدد، میانگین اعداد و انحراف معیار استفاده میشه.
.T.describe()	جابجا کردن سطر و ستونها و دریافت خلاصه آماری از سطرها
.sort_index()	برای مرتب کردن سطرها بر اساس ایندکسشون استفاده میشه. اگه ascending = True باشه مقادیر از بالا به پایین مرتب میشن. اگه axis = 1 بنویسیم ستونها رو بر اساس اسامیشون مرتب میکنه.
.append()	برای اضافه کردن به سطر به دیتافریم استفاده میشه. واسه اینکه لیبلها رو از دست ندیم باید ignore_value=False باشه.
del	واسه پاک کردن به ستون استفاده میشه.
.pop()	واسه خروجی گرفتن از اطلاعات ستونی که پاک میشه استفاده میشه.
.drop()	چندین سطر و ستون رو با هم میشه با این متد پاک کرد. وقتی axis = 0 باشه سطرها پاک میشن و وقتی axis = 1 باشه ستونها
.insert()	واسه اضافه کردن به ستون به دیتافریم. اولین عدد داخل پرانتزش موقعیت ستون رو در دیتافریم مشخص میکنه.
.value_counts()	واسه اینکه ببینیم از هر عدد تو به ستون چند تا وجود داره.
.groupby()	میشه از به دیتافریم به سری زیر مجموعه درست کرد و تحلیلهای بیشتری روش انجام داد. بعد از تعریف باید با group. ارزش خروجی بگیریم. واسه اینکه خروجیها رو سطر به سطر بهمون بده باید براش حلقه for بنویسیم.
.get_group()	واسه داشتن اطلاعات به ستون از زیر مجموعه ای که با groupby ساختیم.
.size()	تعداد به مقدار رو در سطرها یا ستونها نشون میده.

<code>.aggregate()</code> / <code>.agg()</code>	واسه انجام <b>عملیات تجميع</b> روی یه محور مشخص استفاده میشه که پیش فرض <code>axis = 0</code> هست. تو پرانتزش باید بنویسیم که چه کاری روی داده‌ها انجام بده. مثلاً جمع کنه، میانگین بگیره و ...
<code>.astype()</code>	واسه <b>تبدیل داده‌ها</b> مثلاً از اعشار به عدد صحیح
<code>pandas.concat(objs, axis=0, join='outer', ignore_index=False, keys=None, levels=None, names=None, verify_integrity=False, sort=False, copy=None)</code>	واسه <b>ترکیب کردن دیتافریمها</b> استفاده میشه. واسه اینکه ایندکسهاشون گمراهمون نکنه میتونیم <code>ignore_index = True</code> بدیم چون پیش فرضش <code>False</code> هست. از پارامتر <code>key</code> که پیش فرضش <code>none</code> هست هم میشه واسه اینکه مشخص کنیم هر اطلاعات از کدوم دیتافریم اومده استفاده میشه. با <code>loc()</code> هم میشه مجدد دیتافریمی که میخوایم رو ازش بیرون بکشیم. <code>Axis</code> هم به صورت پیش فرض ۰ هست میتونیم بهش ۱ بدیم که اطلاعات رو روی ستونها پخش کنه.
<code>.append()</code>	واسه <b>وصل کردن دیتافریمها</b> استفاده میشه.
<code>pandas.merge(left, right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=False, suffixes=('_x', '_y'), copy=None, indicator=False, validate=None)</code>	واسه <b>ترکیب کردن دیتافریمها</b> استفاده میشه ولی باید حتماً یه ستون داشته باشیم که بینشون اطلاعات مشابه وجود داشته باشه و با پارامتر <code>on</code> معرفی بشه. شکل جوین شدن یا <code>how</code> رو هم پیش فرض <code>inner</code> داده که میتونیم <code>outer</code> هم بدیم که و تو بعضی از سطرها <code>NaN</code> دیده میشه. چون فقط اطلاعات فیلدهایی رو نشون میده که با <code>Chips</code> بین دو تا دیتافریم مشترک هستن. بخش <code>how</code> رو با <code>right</code> و <code>left</code> هم میشه مشخص کرد. اگه کد بالا رو با <code>right</code> امتحان کنیم اطلاعات دیتافریم دوم رو کامل نگه میداره و از اولی فقط اونایی رو نشون میده که با ستون انتخابی اطلاعات مشترک دارن.
<code>.join(other, on=None, how='left', lsuffix="", rsuffix="", sort=False, validate=None)</code>	میشه <b>دیتافریمهایی رو با هم ترکیب کرد</b> و نیازی به اینکه اطلاعات ستون مشترک داشته باشن نداره. بسته به اینکه اول کدون دیتافریم بیاد ایندکسها بر اساس اون گرفته میشن.
<code>pd.read_csv()</code> / <code>.to_csv()</code>	واسه <b>خوندن و ذخیره کردن</b> مجموعه داده با <b>فرمت csv</b>
<code>pd.read_excel()/to_excel()</code>	واسه <b>خوندن و ذخیره کردن</b> مجموعه داده با <b>فرمت xlsx</b>

## حل چندتا تمرین

**تمرین ۱:** یه فایل دلخواه با فرمت csv رو وارد نوت بوک کن. دو تا از ستونها رو بابه دلخواه متد drop() پاک کن.

**تمرین ۲:** مجدد از اسم مجموعه داده‌ات خروجی بگیر که اطلاعات اولیه برگردن اینبار با دستور del یکی از ستونها رو به دلخواه پاک کن.

**تمرین ۳:** بر اساس یکی از ستونها اطلاعات رو sort یا مرتب کن.

**تمرین ۴:** یکی از ستونها مهم رو از بزرگترین عدد به کوچکترین عدد مرتب کن و بعد از ۵ سطر اولش خروجی بگیر. مجدد از پنج سطر آخر همین ستون خروجی بگیر.

**تمرین ۵:** یکی از ستونها رو به دلخواه انتخاب کن و ببین از هر مقدار چند تا تو سلولهایش وجود داره.

**تمرین ۶:** یه زیرمجموعه از مجموعه داده‌ات با یکی از ستونهاش درست کن و ازش خروجی بگیر. بعد یه کد بنویس که از یکی از اطلاعات موجود در این ستون خروجی بگیره.

## جواب تمرینها

**جواب تمرین ۱:** اول باید نامپی و پانداس رو وارد کنیم. بعد یه متغیر درست میکنیم و با تابع `pd.read.csv()` میخوانیم که فایل `csv` ما رو بخونه. باید اسم فایل رو که جلوش پسوند `csv` داره بذاریم تو کوتیشن و تو پرانتز تابع. برای پاک کردن سطر و ستون از متد `drop()` استفاده می کنیم. اگه `axis=0` باشه سطر پاک میشه و اگه `axis=1` باشه ستون پاک میشه.

```
import numpy as np
import pandas as pd

Covid = pd.read_csv("covid-19-county-level-data.csv")

Covid
```

	Unnamed: 0	date	county	state	fips	cases	deaths
0	0	2020-01-21	Snohomish	Washington	53061.0	1	0
1	1	2020-01-22	Snohomish	Washington	53061.0	1	0
2	2	2020-01-23	Snohomish	Washington	53061.0	1	0
3	3	2020-01-24	Cook	Illinois	17031.0	1	0
4	4	2020-01-24	Snohomish	Washington	53061.0	1	0
...	...	...	...	...	...	...	...
379545	379545	2020-07-28	Sweetwater	Wyoming	56037.0	228	2
379546	379546	2020-07-28	Teton	Wyoming	56039.0	311	1
379547	379547	2020-07-28	Uinta	Wyoming	56041.0	243	1
379548	379548	2020-07-28	Washakie	Wyoming	56043.0	44	5
379549	379549	2020-07-28	Weston	Wyoming	56045.0	5	0

379550 rows × 7 columns

```
Covid.drop(["Unnamed: 0", "fips"], axis=1)
```

	date	county	state	cases	deaths
0	2020-01-21	Snohomish	Washington	1	0
1	2020-01-22	Snohomish	Washington	1	0
2	2020-01-23	Snohomish	Washington	1	0
3	2020-01-24	Cook	Illinois	1	0
4	2020-01-24	Snohomish	Washington	1	0
...	...	...	...	...	...
379545	2020-07-28	Sweetwater	Wyoming	228	2
379546	2020-07-28	Teton	Wyoming	311	1
379547	2020-07-28	Uinta	Wyoming	243	1
379548	2020-07-28	Washakie	Wyoming	44	5
379549	2020-07-28	Weston	Wyoming	5	0

379550 rows × 5 columns

**جواب تمرین ۲:** اسم متغیر رو مینویسم و `shift+enter` رو نگه میداریم که اجرا شه و مجدد اسمی ستونها برگردن. از دستور `del` واسه پاک کردن یکی از ستونها استفاده میکنیم.

```
Covid
```

	Unnamed: 0	date	county	state	fips	cases	deaths
0	0	2020-01-21	Snohomish	Washington	53061.0	1	0
1	1	2020-01-22	Snohomish	Washington	53061.0	1	0
2	2	2020-01-23	Snohomish	Washington	53061.0	1	0
3	3	2020-01-24	Cook	Illinois	17031.0	1	0
4	4	2020-01-24	Snohomish	Washington	53061.0	1	0
...	...	...	...	...	...	...	...
379545	379545	2020-07-28	Sweetwater	Wyoming	56037.0	228	2
379546	379546	2020-07-28	Teton	Wyoming	56039.0	311	1
379547	379547	2020-07-28	Uinta	Wyoming	56041.0	243	1
379548	379548	2020-07-28	Washakie	Wyoming	56043.0	44	5
379549	379549	2020-07-28	Weston	Wyoming	56045.0	5	0

379550 rows × 7 columns

```
del Covid["fips"]
```

```
Covid
```

	Unnamed: 0	date	county	state	cases	deaths
0	0	2020-01-21	Snohomish	Washington	1	0
1	1	2020-01-22	Snohomish	Washington	1	0
2	2	2020-01-23	Snohomish	Washington	1	0
3	3	2020-01-24	Cook	Illinois	1	0
4	4	2020-01-24	Snohomish	Washington	1	0
...	...	...	...	...	...	...
379545	379545	2020-07-28	Sweetwater	Wyoming	228	2
379546	379546	2020-07-28	Teton	Wyoming	311	1
379547	379547	2020-07-28	Uinta	Wyoming	243	1
379548	379548	2020-07-28	Washakie	Wyoming	44	5
379549	379549	2020-07-28	Weston	Wyoming	5	0

379550 rows × 6 columns

**جواب تمرین ۳:** باید از متد `sort_values()` استفاده کنیم و تو پرانتزش اسم ستون مورد نظر رو بنویسیم.

```
Covid.sort_values("county")
```

	Unnamed: 0	date	county	state	cases	deaths
58466	58466	2020-04-14	Abbeville	South Carolina	9	0
141437	141437	2020-05-13	Abbeville	South Carolina	34	0
233124	233124	2020-06-12	Abbeville	South Carolina	62	0
308409	308409	2020-07-06	Abbeville	South Carolina	135	0
289442	289442	2020-06-30	Abbeville	South Carolina	113	0
...	...	...	...	...	...	...
346754	346754	2020-07-18	Ziebach	South Dakota	3	0
375616	375616	2020-07-27	Ziebach	South Dakota	7	0
208461	208461	2020-06-04	Ziebach	South Dakota	2	0
168677	168677	2020-05-22	Ziebach	South Dakota	1	0
314860	314860	2020-07-08	Ziebach	South Dakota	1	0

379550 rows × 6 columns

**جواب تمرین ۴:** اینجا ستون cases رو انتخاب کردم که میزان مبتلایان به کرونا رو نشون میده. False رو ascending در نظر گرفتم. بعد با متد head() خواستم که از ۵ سطر اولش خروجی بگیره. واسه خروجی گرفتن از ۵ سطر آخر هم از tail() استفاده میکنیم.

```
Covid.sort_values("cases", ascending = False).head()
```

	Unnamed: 0	date	county	state	cases	deaths
378184	378184	2020-07-28	New York City	New York	228939	22977
374968	374968	2020-07-27	New York City	New York	228740	22970
371756	371756	2020-07-26	New York City	New York	228445	22956
368544	368544	2020-07-25	New York City	New York	228220	22947
365335	365335	2020-07-24	New York City	New York	227882	22936

```
Covid.sort_values("cases", ascending = False).tail()
```

	Unnamed: 0	date	county	state	cases	deaths
282002	282002	2020-06-28	Unknown	Maryland	0	26
219599	219599	2020-06-08	Unknown	Maryland	0	44
244438	244438	2020-06-16	Unknown	Maryland	0	45
26030	26030	2020-04-01	Unknown	West Virginia	0	1
185759	185759	2020-05-28	Unknown	Maryland	0	69

**جواب تمرین ۵:** واسه انتخاب یه ستون باید اسمش رو تو کوتیشن بذاریم و داخل براکت جلوی اسم مجموعه داده بنویسیم. بعد برای اینکه مقادیر رو بشمریم باید از متد `value_count()` استفاده کنیم.

```
Covid["cases"].value_counts()

1      24827
2      16063
3      13265
4      10925
5       9559
...
29626      1
16425      1
15527      1
6565       1
18528      1
Name: cases, Length: 11264, dtype: int64
```

**جواب تمرین ۶:** واسه ساخت یه زیرمجموعه از `groupby()` استفاده میکنیم و واسه خروجی گرفتن از `groups`.  
 واسه گرفتن یه اطلاعات مشخص از یه ستون از `get_group()` استفاده میکنیم و اطلاعات مشخص شده رو تو  
 پرانتزش مینویسیم.

```
Covid_Co = Covid.groupby("county")
```

Covid\_Co.groups

{ 'Abbeville': [5000, 5873, 6877, 7997, 9208, 10533, 12004, 13626, 15373, 17227, 19167, 21218, 23355, 25552, 27825, 30160, 32551, 34987, 37462, 39985, 42545, 45131, 47748, 50405, 53078, 55763, 58466, 61181, 63907, 66653, 69411, 72178, 74953, 77735, 80525, 83418, 86285, 89174, 92072, 95000, 97953, 100922, 103814, 106729, 109668, 112631, 115618, 118629, 121664, 124724, 127808, 130917, 134051, 137210, 140394, 143603, 146837, 150096, 153380, 156689, 159993, 163323, 166673, 169999, 173351, 176726, 180124, 183546, 186992, 190462, 193956, 197475, 200999, 204539, 208094, 211665, 215252, 218856, 222477, 226114, 229768, 233439, 237126, 240829, 244548, 248283, 252034, 255801, 259584, 263383, 267198, 271029, 274876, 278739, 282618, 286513, 290424, 294351, 298294, 302253, 306228, 310219, 314226, 318249, 322288, 326343, 330414, 334501, 338604, 342723, 346858, 350999, 355156, 359329, 363518, 367723, 371944, 376181, 380434, 384703, 388988, 393289, 397606, 401939, 406288, 410653, 415034, 419431, 423844, 428273, 432718, 437179, 441656, 446149, 450658, 455183, 459724, 464281, 468854, 473443, 478048, 482669, 487306, 491959, 496628, 501313, 506014, 510731, 515464, 520213, 524978, 529759, 534556, 539369, 544198, 549043, 553904, 558781, 563674, 568583, 573508, 578449, 583406, 588379, 593368, 598373, 603394, 608431, 613484, 618553, 623638, 628739, 633856, 638989, 644138, 649303, 654484, 659681, 664894, 670123, 675368, 680629, 685906, 691199, 696508, 701833, 707174, 712531, 717904, 723293, 728708, 734139, 739586, 745049, 750528, 756023, 761534, 767061, 772604, 778163, 783738, 789329, 794936, 800559, 806198, 811853, 817524, 823211, 828914, 834633, 840368, 846119, 851886, 857669, 863468, 869283, 875114, 880961, 886824, 892703, 898608, 904529, 910466, 916419, 922388, 928373, 934374, 940391, 946424, 952473, 958538, 964619, 970716, 976829, 982958, 989103, 995264, 1001441, 1007634, 1013843, 1020068, 1026309, 1032566, 1038839, 1045128, 1051433, 1057754, 1064091, 1070444, 1076813, 1083198, 1089599, 1096016, 1102449, 1108898, 1115363, 1121844, 1128341, 1134854, 1141383, 1147928, 1154489, 1161066, 1167659, 1174268, 1180893, 1187534, 1194191, 1200864, 1207553, 1214258, 1220979, 1227716, 1234469, 1241238, 1248023, 1254824, 1261641, 1268474, 1275323, 1282188, 1289069, 1295966, 1302879, 1309808, 1316753, 1323714, 1330691, 1337684, 1344693, 1351718, 1358759, 1365816, 1372889, 1379978, 1387083, 1394204, 1401341, 1408494, 1415663, 1422848, 1430049, 1437266, 1444499, 1451748, 1458993, 1466254, 1473531, 1480824, 1488133, 1495458, 1502799, 1510156, 1517529, 1524918, 1532323, 1539744, 1547181, 1554634, 1562103, 1569588, 1577089, 1584606, 1592139, 1599688, 1607253, 1614834, 1622431, 1630044, 1637673, 1645318, 1652979, 1660656, 1668349, 1676058, 1683783, 1691524, 1699281, 1707054, 1714843, 1722648, 1730469, 1738306, 1746159, 1754028, 1761913, 1769814, 1777731, 1785664, 1793613, 1801578, 1809559, 1817556, 1825569, 1833598, 1841643, 1849704, 1857781, 1865874, 1873983, 1882108, 1890249, 1898406, 1906579, 1914768, 1922973, 1931194, 1939431, 1947684, 1955953, 1964238, 1972539, 1980856, 1989189, 1997538, 2005903, 2014284, 2022681, 2031094, 2039523, 2047968, 2056429, 2064906, 2073399, 2081908, 2090433, 2098974, 2107531, 2116104, 2124693, 2133308, 2141939, 2150586, 2159249, 2167928, 2176623, 2185334, 2194061, 2202804, 2211563, 2220338, 2229129, 2237936, 2246759, 2255598, 2264453, 2273324, 2282211, 2291114, 2300033, 2308968, 2317919, 2326886, 2335869, 2344868, 2353883, 2362914, 2371961, 2381024, 2390103, 2399198, 2408309, 2417436, 2426579, 2435738, 2444913, 2454104, 2463311, 2472534, 2481773, 2491028, 2500299, 2509586, 2518889, 2528208, 2537543, 2546894, 2556261, 2565644, 2575043, 2584458, 2593889, 2603336, 2612799, 2622278, 2631773, 2641284, 2650811, 2660354, 2669913, 2679488, 2689079, 2698686, 2708309, 2717948, 2727603, 2737274, 2746961, 2756664, 2766383, 2776118, 2785869, 2795636, 2805419, 2815218, 2825033, 2834864, 2844711, 2854574, 2864453, 2874348, 2884259, 2894186, 2904129, 2914088, 2924063, 2934054, 2944061, 2954084, 2964123, 2974178, 2984249, 2994336, 3004439, 301455
---

```
Covid Co.get group("New York City")
```

	Unnamed: 0	date	county	state	cases	deaths
416	416	2020-03-01	New York City	New York	1	0
448	448	2020-03-02	New York City	New York	1	0
482	482	2020-03-03	New York City	New York	2	0
518	518	2020-03-04	New York City	New York	2	0
565	565	2020-03-05	New York City	New York	4	0
...	...	...	...	...	...	...
365335	365335	2020-07-24	New York City	New York	227882	22936
368544	368544	2020-07-25	New York City	New York	228220	22947
371756	371756	2020-07-26	New York City	New York	228445	22956
374968	374968	2020-07-27	New York City	New York	228740	22970
378184	378184	2020-07-28	New York City	New York	228939	22977

150 rows  $\times$  6 columns

تا اینجا دو تا از کتابخانه‌های بزرگ و مهم پایتون رو که برای کار با داده حیاتی هستن، یاد گرفتیم. تو جزوه بعد سومین کتابخانه مهم به اسم matplotlib که اون هم برای کار با داده‌ها ضروری هست و داده‌ها رو برامون روی نمودار نمایش میده رو با هم یاد میگیریم.