

Pandas

PANEL DATA AND PYTHON DATA ANALYSIS

1. در کار ها و آزمایشات عملی و گاهی در تئوری با تعداد زیادی داده سر و کار داریم
 2. یا شاید تحلیل رابطه بین داده های مختلف برای ما اهمیت داشته باشد
 3. یا بررسی رفتار سیستم های مورد آزمایش ما حائز اهمیت باشد
 4. یا دستکاری و ایجاد تغییر در یک سری داده مطلوب نظر ما باشد
 5. و یا اضافه یا حذف کردن داده های یک آزمایش ضروری باشد
 6. و یا تعداد داده ها به قدری بالا باشد که عملا تحلیل و انجام تمام موارد بالا به صورت دستی غیر ممکن یا سخت باشد
- در این صورت پایتون با کتابخانه pandas که یک ابزار در زمینه data science است به کمک ما می آید
- کتابخانه pandas بر روی کتابخانه numpy ساخته شده و مبنی بر آن کار می کند به همین دلیل بسیاری از توابع آن را نیز به ارث برده



```
pip install pandas
```

```
import pandas as pd
```

با نحوه نصب کتابخانه ها

و استفاده از آنها آشنایی داریم

پس مستقیما به معرفی این کتابخانه میپردازیم

اولین مفاهیم مورد بحث در این کتابخانه عبارتند از

1. سری ها **series** یا دنباله ها یا سلسله ها
سری ها به زبان ساده آرایه از داده ها (هر نوع داده ای) هستند که در یک ستون به ترتیب زیر هم مرتب شده اند و ستون های یک جدول (data frame) را تشکیل میدهند
2. چهارچوب داده ها یا **data frame**
چهارچوب ها نیز به زبان ساده جداولی هستند که از یک یا چندین سری تشکیل شده اند مانند دیتا فریم زیر که از سه سری با نام (table) های duration , calories , max pale تشکیل شده

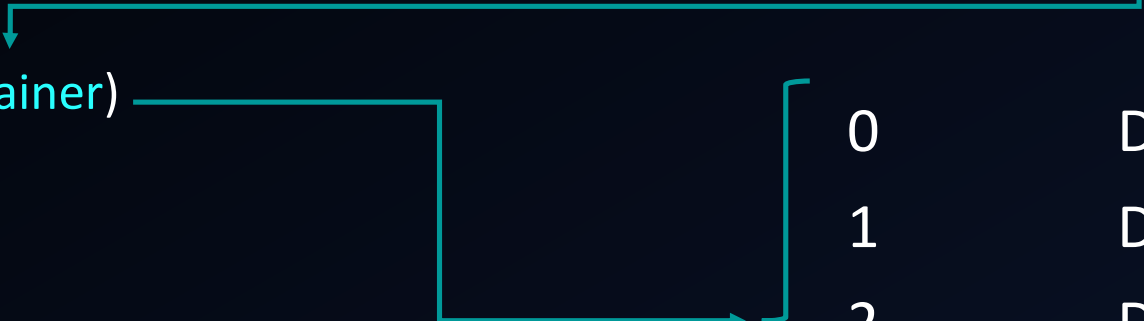
Series 1	Series 2	Series 3
Duration	Calories	Max pale
25	12	120
35	20	112
45	50	115
50	68	120

Data frame

ساخت سری ها

اگر یک لیست یا دیکشنری یا تاپل داشته باشیم به راحتی با دستور زیر می توان یک سری ساخت

`My_series = pd.Series(container)`



0	Data1
1	Data2
2	Data3
3	Data4
4	Data5
⋮	⋮

این بدین معنی است که با ایندکس دهی به `My_series` میتوان به داده های داخل این سری دسترسی داشت

مثال:

Import pandas as pd

My_list=[2,4,5,7,8,9]

My_series = pd.Series(My_list)

یا

My_tuple = (3,2,5,4,7)

My_series = pd.Series(My_tuple)

0 2

1 4

2 5

3 7

4 8

5 9

0 3

1 2

2 5

3 4

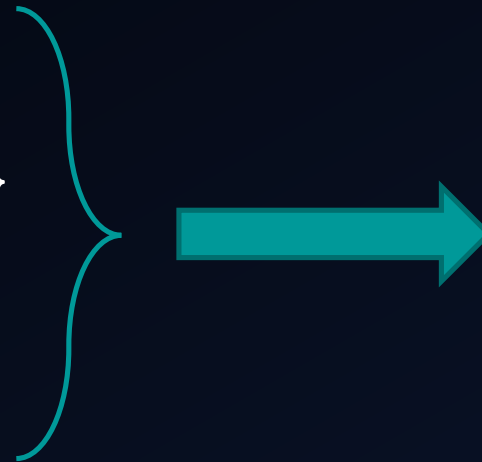
4 7

یا همچنین می‌توان از دیکشنری‌ها هم استفاده کرد
در این صورت key های دیکشنری به جای index ها در سری ظاهر می‌شوند

```
Import pandas as pd
```

```
my_dict = {'x':3,'y':2,'z':5,'k':4,'i':7}
```

```
My_series = pd.Series(my_dict)
```



x	3
y	2
z	5
k	4
i	7

حال چگونه سری ها را به یک data frame تبدیل کنیم؟

سری ها را می سازیم

```
s1 = pd.Series (list(range(20,31)))
```

```
s2 = pd.Series (list(range(40,51)))
```

یک data frame خالی می سازیم

```
df = pd.DataFrame()
```

سری ها را به data frame اضافه می کنیم و به هر سری، نام (lable) می دهیم
این نام می تواند عدد یا حرف باشد

```
df ["duration"] = s1
```

```
df ["calories"] = s2
```

حتی می توان ستونی درست کرد و محتویات آن را خالی گذاشت تا بعدا داده هایش تعیین شوند

```
df ["future"] =None
```

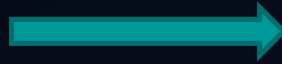
در این صورت ارزش تمام سطر های این ستون None خواهد بود

راه دیگر تشکیل یک data frame استفاده از دیکشنری ها است

```
my_dict = {"name":["BMW","KIA","Audi","volvo"],  
           "model":[2015,2019,2022,2020],  
           "country":["Germany","South Korea","Germany","Sweden"]}
```

```
df = pd.DataFrame(my_dict)
```

```
Print(df)
```



	name	model	country
0	BMW	2015	Germany
1	KIA	2019	South Korea
2	Audi	2022	Germany
3	Volvo	2020	Sweden

دسترسی به داده ها
در این مبحث 7 حالت برای دسترسی به داده ها وجود دارد

1. دسترسی به یک سطر خاص

```
df.loc[row]
```

2. دسترسی به دو یا چند سطر خاص

```
df.loc[ [ row , row , row , ...] ]
```

3. دسترسی به سطر ها به روش slicing

```
df.loc[ x : y : z ]
```

4. دسترسی به یک ستون خاص

```
df [ column name ]
```

5. دسترسی به دو یا چند ستون خاص

```
df [ [column , column , column , ...] ]
```

6. دسترسی به یک درایه خاص

```
df [ column ] [ row ]
```

7. دسترسی به چند سطر و چند ستون خاص

7. دسترسی به چند سطر و چند ستون خاص

فرض کنید دیتا فریم زیر را در اختیار داریم می‌خواهیم قسمت های رنگی را جدا کرده و با آن کار کنیم

با توجه به مطالب اسلاید قبل باید بتوان الگوریتمی طراحی کرد که این کار را برای ما انجام دهد

یعنی از دیتا فریم زیر فقط ستون های *a* , *c* , *e* و آن هم فقط سطر های 1 و 2 و 5 مطلوب نظر ما باشند

```
data = { "a" : [1,2,3,4,5,6], "b" : [2,4,5,7,8,9], "c" : [4,7,8,9,6,5], "d" : [5,4,7,8,9,6], "e" : [6,5,8,9,7,4]}
```

```
df = pd.DataFrame(data)
```

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
0	1	2	4	5	6
1	2	4	7	4	5
2	3	5	8	7	8
3	4	7	9	8	9
4	5	8	6	9	7
5	6	9	5	6	4

	<i>a</i>	<i>c</i>	<i>e</i>
1	2	7	5
2	3	8	8
5	6	5	4

و این نتیجه را برای ما چاپ کند

برای این کار ابتدا سطر های *a* , *c* , *e* را از دیتا فرم جدا کرده و آن را به عنوان یک دیتا فریم جدید ذخیره میکنیم
سپس سطر های دلخواه را در نظر گرفته و آنها را در یک دیتا فریم جدید ذخیره میکنیم

```
new_df = df[ ["a","c","e"] ]
```

	<i>a</i>	<i>c</i>	<i>e</i>
0	1	4	6
1	2	7	5
2	3	8	8
3	4	9	9
4	5	6	7
5	6	5	4

```
My_dataframe = new_df.loc[[1,2,5]]
```

	<i>a</i>	<i>c</i>	<i>e</i>
1	2	7	5
2	3	8	8
5	6	5	4

CSV (comma-separated values)

گاهی داده ها از قبل توسط شخص دیگر تهیه شده
یا داده ها به قدری زیاد هستند که به صورت دستی
قابل وارد کردن نیستند

معمولا این داده ها در یک فایل با پسوند CSV ذخیره شده اند

1	Duration,Pulse,Maxpulse,Calories
2	60,110,130,409.1
3	60,117,145,479.0
4	60,103,135,340.0
5	45,109,175,282.4
6	45,117,148,406.0
7	60,102,127,300.0
8	60,110,136,374.0
9	45,104,134,253.3
10	30,109,133,195.1
11	60,98,124,269.0
12	60,103,147,329.3
13	60,100,120,250.7

راه دیگر برای درست کردن دیتا فریم و تحلیل داده ها استفاده از داده های موجود در این نوع فایل ها هستند
در ادامه برای تمرین و آموزش از فایل CSV قرار داده شده در سایت W3schools.com استفاده کردیم
که داده های مربوط به مدت زمان، ضربان، ضربان بیشینه و میزان کالری مصرف شده در یک فعالیت ورزشی را نشان می دهد

<https://www.w3schools.com/python/pandas/data.csv>

در این فایل در سطر اول نام (لیبل) ستون ها آورده می شود و در سطر های بعدی مقادیر و داده ها آورده می شود
در اینجا در هر سطر 4 عدد وجود دارد که به ترتیب به ستون های متناظر خود مربوط هستند

قدم اول، خواندن داده ها

برای خواندن این داده ها از دستور `pd.csv_read('file name')` استفاده میکنیم و آن را به یک متغیر مانند `df` منصوب میکنیم

نکته مهم :

این متغیر به صورت خودکار یک دیتا فریم خواهد بود

برای چاپ این دیتا فریم میتوان از دستور `print()` استفاده کرد اما این تابع صرفاً 5 خط اول و آخر را چاپ میکند
برای دیدن تمام داده ها از دستور `print(df.to_string())` استفاده میکنیم

با اجرای دستور `print(pd.options.display.max_rows)` عددی به ما برگردانده میشود که اگر تعداد سطرهای داده بیشتر از آن مقدار باشد، دستور `print()` صرفاً 5 خط اول و آخر را نمایش دهد

اما میتوان این مقدار را تغییر داد

عدد دلخواه `pd.options.display.max_rows =`

برای داشتن یک دید کلی از داده ها بهتر است چند سطر اول داده ها را با دستور `df.head(x)` مشاهده کنیم که در آن `x` تعداد سطر هایی است که مایل به مشاهده آن هستیم و سپس نتیجه را پرینت می کنیم در این متد `x` می تواند خالی بماند در این صورت مانند تابع `print()` این متد نیز 5 سطر اول را **برمیگرداند** (چاپ نمی کند) به طور مشابه برای دیدن انتهای داده ها نیز متدی به نام `tail(x)` وجود دارد

همچنین برای کسب اطلاع از خصوصیات دیتا فریم خود میتوان از `df.info()` استفاده کرد که اطلاعات جامعی از داده ها را در اختیار ما قرار می دهد

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype  
---  -
0   Duration    169 non-null   int64   
1   Pulse       169 non-null   int64   
2   Maxpulse    169 non-null   int64   
3   Calories    164 non-null   float64  
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
```

سطر های خالی (null ها)، داده های غیر منطقی، دوگانگی و تکرار در بررسی و تحلیل داده ها برای ما مزاحم هستند

پس بهتر است این داده های بد حذف شوند

برای تمیز کردن داده های خالی دو راه وجود دارد

1. حذف تمامی سطر:

از آنجا که معمولا دیتا فریم ها دارای تعداد زیادی داده هستند پس حذف کردن چندی از این داده ها در نتیجه نهایی بررسی (برای مثال یک آزمایش فیزیکی) تاثیر چندانی ندارد

2. جایگزینی داده خالی با میانگین داده ها:

اما اگر تعداد داده ها نسبتا کم بود و تک تک داده ها برای ما ارزش داشتند در این صورت حذف کامل سطر خطای قابل ملاحظه ای در نتیجه آزمایش ایجاد می کند. پس مناسب تر آن است تا داده های خالی با مقدار میانگین داده ها جایگزین شوند

برای انجام راه حل اول از دستور `df.dropna()` استفاده می کنیم. با اجرای این دستور یک کپی از `df` ایجاد شده و سطر های خالی آن حذف می شوند و داده های اصلی `df` آسیبی نخواهند دید

اما اگر خواهان تغییر روی داده های اصلی هستید میتوان از آرگومان `inplace = True` استفاده کنید در این صورت چهارچوب اصلی دچار تغییر میشود
مثال و توضیح بیشتر در صفحه بعد

همچنین اگر فقط یک یا چند ستون خاص مد نظر بود میتوان از آرگومان `subset=[column name , ...]` کمک گرفت

راه دوم، جایگزینی داده ها:

اولین روش، جایگزینی داده های خالی با داده دلخواه است. بدین معنی که می توان با دستور `df.fillna()` به جای تمامی داده های خالی (در تمامی ستون ها)، یک مقدار دلخواه قرار داد.

برای مثال دستور زیر تمامی داده های خالی را با عدد 60 جایگزین می کند

```
df.fillna(60)
```

لازم به ذکر است که برای این متد نیز بحث `inplace = True` صادق است به این معنی که :

```
print(df.fillna(60))
```

بر داده های اصلی تغییری اعمال نمیکند و صرفاً یک کپی از داده های اصلی را ایجاد کرده و بر روی آن تغییرات را اعمال میکند

پس اگر `df` را چاپ کنیم خواهیم دید که هنوز داده های خالی دارد

```
df.fillna(60 , inplace = True)
```

داده های اصلی را تغییر می دهد

اما اگر بخواهیم فقط داده های یک ستون خاص را با این روش جایگزین کنیم نام ستون دلخواه را نیز باید ذکر کنیم

```
df["column name"].fillna( value , inplace = True)
```

جایگزین کردن داده ها با یک عدد دلخواه و رندوم باعث ایجاد خطا در نتایج خواهد شد
راه مطمئن تر آن است که داده های خالی هر ستون را با میانگین داده های همان ستون جایگزین کنیم

ابتدا از متد `mean()` استفاده میکنیم تا میانگین اعداد ستون مورد نظر (برای مثال `Calories`) را به دست آوریم

```
df["Calories"].mean()
```

سپس داده های خالی در این ستون را با مقدار میانگین جایگزین می‌کنیم

```
df["Calories"].fillna( df["Calories"].mean() )
```

گاهی پیش می‌آید که یک یا چند داده به اشتباه وارد شده باشند در این صورت میتوان به صورت دستی آنها را اصلاح کرد

```
df.loc[ Row , Column name] = new value
```

راه دیگر نیز در صورت بزرگ بودن تعداد داده ها و صدمه ندیدن نتیجه حذف سطر مورد نظر است

```
df.drop( Row , inplace = True or False)
```

همچنین میتوان الگوریتمی طراحی کرد که داده ها را برای ما اصلاح کند
برای مثال تمام داده های بزرگ تر از یک مرز مشخص را اصلاح کرده و به عدد خاصی تبدیل کند
برای این کار نیاز به حلقه زدن بر روی تمامی ایندکس های df هستیم
بنا بر این df.index به کمک ما خواهد آمد

```
for i in df.index :  
    if (condition):  
        algorithm  
    ...
```