



جزوه شماره ۱۳



# FUNDAMENTALS OF DATA SCIENCE

Data Visualization with Python



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

 Dr.nemati.K  
 @Nemati\_k  
 ۰۹۱۱۲۲۳۰۷۹۸



محمدطاهر طاهرپور  
دانشجوی ارشد مدیریت شهری  
دانشگاه تهران

 mttaherpoor  
 @mtaherpoor  
 ۰۹۳۳۶۱۴۴۹۴۷



پاییز ۱۴۰۲

## فهرست مطالب:

۳.....	Fundamentals of Data Science
۳.....	Data Science چیست؟
۴.....	زبانهای برنامه‌نویسی در کشورهای توسعه یافته و در حال توسعه
۵.....	کتابخانه‌های پایتون
۶.....	خلاصه دستورهای numpy
۱۰.....	خلاصه دستورهای pandas
۱۳.....	خلاصه دستورهای Matplotlib.pyplot
۱۵.....	ترسیم نمودارها در Matplotlib
۲۲.....	خلاصه دستورهای seaborn
۲۶.....	نمودارهای ترسیم شده در seaborn
۳۱.....	خلاصه دستورهای squarify
۳۱.....	نمودارهای ترسیم شده در squarify
۳۲.....	خلاصه دستورهای missingno
۳۲.....	نمودارهای ترسیم شده در missingno

## Fundamentals of Data Science

تو این جزوه اصول پایه Data Science یا علم داده رو با هم یاد میگیریم و یکبار دیگه دستورها و نمودارهایی که با کتابخانه‌های مورد نیاز علم داده یاد گرفتیم (numpy, pandas, matplotlib, seaborn, squarify, missingno) رو اینجا با هم مرور میکنیم. بخش مربوط به توضیحات تئوری data science این جزوه از روی مطالب آموزشی Udemey - Data Visualization with Python Masterclass - Python A-Z بخش مربوط به Fundamentals of Data Science و همینطور ChatGPT گرفته شده.

## Data Science چیه؟

ساده‌ترین تعریف از علم داده یا **Data Science** تولید اطلاعات معنادار هست. در واقع **meaningful information** یه کلیدواژه واسه علم داده به حساب می‌آد. این داده‌ها تو پایگاه داده یا **Database** قرار میگیرن و میتونن متعلق به یه پروژه کوچیک واسه یه شرکت باشن یا یه پروژه بزرگ دولتی. اگه داده‌ها خیلی زیاد باشن بهشون میگن **Big Data**. علم داده به دانشمندان داده که بهشون **Data Scientist** میگن نیاز داره. این افراد با داده‌ها سروکار دارن و داده‌ها رو از منابع متفاوتی گردآوری میکنن و چیزی تولید میکنن به اسم **DOA (Data-oriented Applications)** که داده‌ها رو تجزیه و تحلیل میکنه. اصول پایه علم داده شامل سه مورد زیر هست:

ریاضیات

آمار

IT (Information Technology)

یه اصطلاح دیگه هم داریم به اسم **Data Literacy** یعنی سواد داده یا توانایی خوندن، درک کردن، خلق و ارتباط برقرار کردن با داده‌ها یا Data به عنوان اطلاعات یا Information. در واقع سواد داده یعنی توانایی هدایت اطلاعات معنادار. وقتی داده‌ها بزرگ و پیچیده میشن، شرکتها دانشمند داده استخدام میکنن که مهارتهای تجزیه و تحلیل داده رو داشته باشه. دلیلش اینه که موفقیت یه کسب و کار به داده‌ها مرتبط هست. واسه همین شرکتها از همه کارمنداشون انتظار دارن که تا حد مورد انتظاری سواد داده داشته باشن.

**مهارتهای سواد داده** که گاهی بهشون **Data Story Telling** یا داستان گویی داده هم گفته میشه شامل موارد زیر هست:

۱- شناخت و هدف استفاده از داده‌ها

۲- تفسیر داده‌های تصویری Data Visualizations مثل نمودارها و گرافها

۳- تفکر انتقادی یا Thinking Critically در مورد داده‌های تولید شده توسط تحلیلگر داده یا Data Analysis

۴- درک ابزارها و متدهای تجزیه و تحلیل و موارد استفاده از این ابزارها

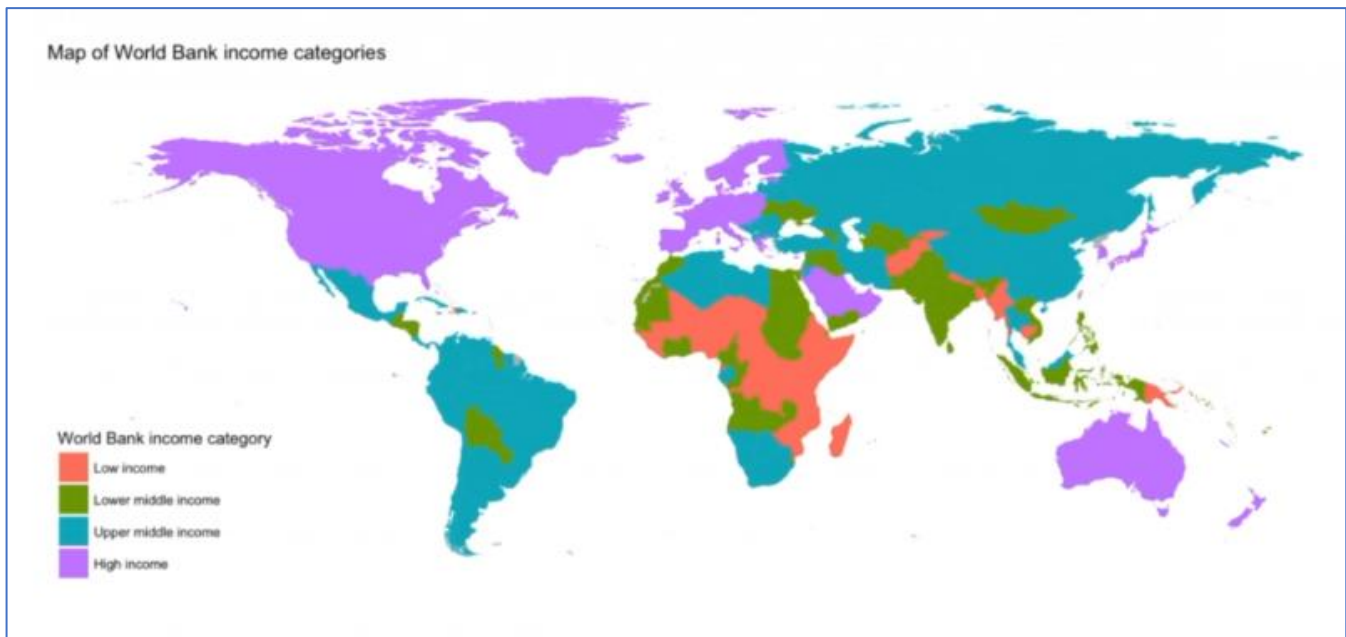
۵- تشخیص داده‌های گم شده یا کمبود داده‌ها یا داده‌هایی که گمراه کننده هستن.

۶- توانایی انتقال داده‌ها به افرادی که سواد داده ندارن.

یادت باشه که **Data Literacy** معادل **Statistical Literacy** یا سواد آماری **نیست**. سواد داده یعنی اینکه بدونیم معنی داده چیه و بتونیم گرافها و نمودارها رو بخونیم و بتونیم ازشون یه نتیجه‌گیری بدیم. در واقع سواد داده یعنی تفسیر خلاصه‌های آماری.

## زبانهای برنامه‌نویسی در کشورهای توسعه یافته و در حال توسعه

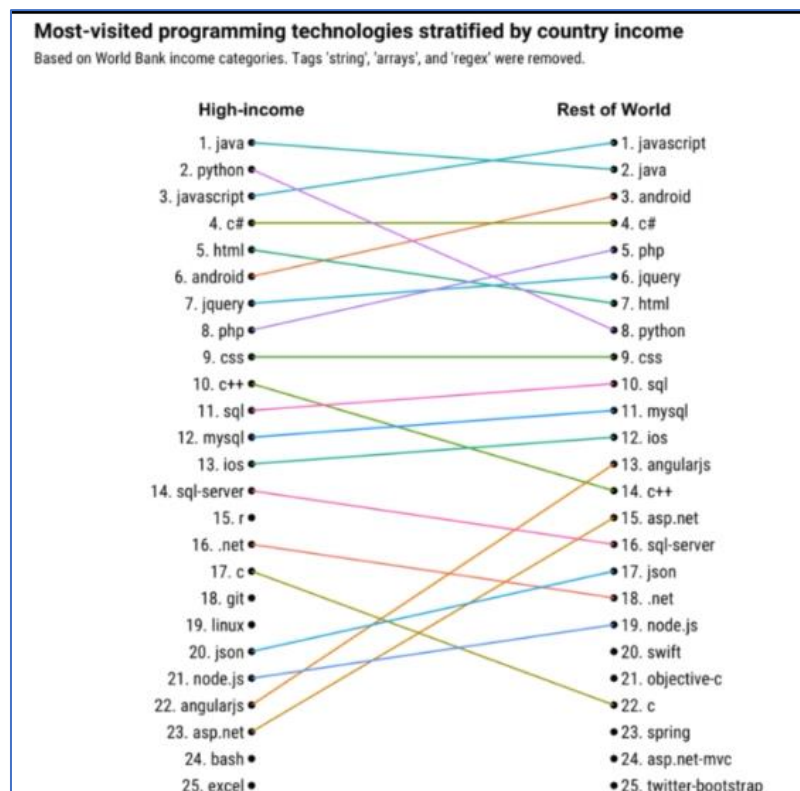
بر اساس داده‌های بانک جهانی در سال ۲۰۱۷، کشورهای در حال توسعه به زبانه‌های برنامه‌نویسی یا programming languages روی آوردن در حالی که از همون سال کشورهای توسعه یافته شروع به استفاده از Data Science و Big Data کردن. نقشه زیر کشورها رو بر اساس درآمد تو سال ۲۰۱۷ دسته‌بندی کرده.



بر اساس تحلیل ترافیک وبسایتها در سال ۲۰۱۷ مشخص شد که زبان **Java** تو کشورهای توسعه یافته اولین زبان برنامه‌نویسی بود که بیشترین تگ یا برچسب رو تو به خودش اختصاص داد و **پایتون** هم دومین زبانشون بود ولی تو کشورهای در حال توسعه پایتون رتبه ۸ رو بدست آورد.

نتیجه این تحقیق تو نمودار زیر اومده. **زبان برنامه‌نویسی R** هم اصلا تو کشورهای توسعه یافته استفاده

نمی‌شد.



برای علم داده نیاز داری که به زبان برنامه نویسی پایتون و کتابخانه‌های پایتون مربوط به علم داده تسلط داشته باشی.

## کتابخانه‌های پایتون

پایتون کتابخانه‌های بی‌نظیری برای Data Science علم داده، Machine learning یادگیری ماشین، Artificial Intelligence هوش مصنوعی، Deep learning یادگیری عمیق و ... داره. چند تا از مهمترین کتابخانه‌های پایتون که در سال ۲۰۲۳ معرفی شده که یادگیریشون ضروری هست رو اینجا آوردیم.

- ✚ Numpy
- ✚ Pandas
- ✚ Matplotlib
- ✚ Scikit\_Learn
- ✚ PyTorch
- ✚ Natural Language Toolkit (NLTK)
- ✚ Seaborn
- ✚ SciPy
- ✚ Plotly
- ✚ OpenCv
- ✚ LightGBM
- ✚ Eli5
- ✚ PyBrain
- ✚ Squirify
- ✚ Permetrics
- ✚ ...

تو جزوه‌های قبلی کتابخانه‌های زیر رو با هم یاد گرفتیم.

✚ برای کار با داده‌ها یا data:

✓ **numpy as np** (واسه محاسبات ریاضی و کار کردن با آرایه‌ها)

✓ **pandas as pd** (واسه تبدیل آرایه‌ها به جدول و یا کار با جدولهای اکسل و CSV)

✓ **matplotlib.pyplot as plt** (واسه ترسیم نمودار)

✚ برای زیبایی بیشتر نمودارها

✓ **seaborn as sns** (زیباسازی نمودارها)

✓ **squarify** (برای ایجاد نمودارهای Treemaps یا نمودارهای سلسله‌مراتبی مستطیل شکل)

✚ برای چک کردن داده‌های موجود در یک جدول

✓ **missingno as msno** (بررسی میزان و توزیع داده‌های از دست رفته در مجموعه داده‌ها)

میخوایم کاربرد این کتابخانه‌ها رو تو یه پروژه واقعی Data Science ببینیم. ولی قبل از شروع پروژه کاربردی، واسه اینکه کتابخانه‌هایی که تا اینجا یاد گرفتی واست مرور شه، یه دور دیگه دستورها و کدهای هر کدوم از این کتابخانه‌ها رو با هم دوره می‌کنیم.

دستورها	توضیحات
Import numpy as np	وارد کردن numpy به نوت بوک ArcGIS Pro
np.array()	ساخت آرایه به همراه عضوهای آرایه
type()	تشخیص نوع آرایه
.ndim	مشخص کردن تعداد ابعاد آرایه
.shape	مشخص کردن تعداد سطر و ستون آرایه‌ها (خروجی رو به صورت Tuples برمیگرددونه (یعنی اعدادی داخل پرانتز که با ویرگول از هم جدا شدن: (۲,۴) یا (۶,۱))
.size	مشخص کردن تعداد اعضای آرایه
for / in	بررسی تک تک اعضا و خروجی گرفتن از همه اعضا
.flat	دیدن آرایه چندبعدی در یک بعد
np.zeros() np.ones((۳,۵), dtype = int) np.full()	تولید آرایه با اعضای ۰ تولید آرایه با اعضای ۱ تولید آرایه با اعضای خاص اگر بخوایم اعداد رو به صورت عدد صحیح دریافت کنیم باید از dtype استفاده کنیم.
print (i, end = " ")	end به پارامتر از تابع print() هست. به شکل عمومی، این تابع نتیجه رو تو خط‌های متفاوت نشون میده. ولی " end = " " کمک میکنه که نتیجه تو یه خط دیده شه.
np.arange()	این تابع هر آرگومانی که بهش بدی، از ۰ تا یکی مونده به اون عدد رو به صورت آرایه بهت خروجی میده. میتونی دو تا آرگومان بهش بدی که از عدد اول تا یکی مونده به آخر رو به صورت، رایج خروجی بده. با سه تا آرگومان هم میشه آرایه درستش کرد که آرگومان سوم گام رو تعیین میکنه که میتونه عدد منفی هم باشه یعنی از آخر به اول شمرده شه.
np.linspace(۰ , ۵, num =۵)	بهش اولین و آخرین آرگومان رو میدیم. یه سری عدد از عدد اول تا آخر که عدد آخر هم جزوش هست رو خروجی میده. یه کلیدواژه اختیاری به اسم num هم داره گام رو مشخص میکنه.
.reshape	تبدیل آرایه یک بعدی به آرایه چندبعدی
np.random.randint()	برای دریافت اعداد رندوم از یه محدوده
.sum() .max() .min() .mean() .std() .sqrt()	مجموع اعداد بزرگترین عدد آرایه کوچکترین عدد آرایه میانگین اعداد انحراف معیار اعداد جذر گرفتن از اعداد

<code>[]/ [۳:۷]/ [:۳]/ [۳,۰]/ [:,۲:۴]/[:,(۲,۳)]</code>	واسه دسترسی به اعضای آرایه با شماره ایندکس عضوها
<code>.view()</code>	از روی تابع اصلی به تابع فرعی با همون اعضا میسازیم که با انجام عملیات روی هر کدوم از توابع، <b>روی تابع دیگه هم اعمال میشه</b> .
<code>.copy()</code>	از روی تابع اصلی به تابع فرعی با همون اعضا میسازیم که با انجام عملیات روی تابع اصلی، تابع فرعی هم تغییر میکنه <b>ولی برعکسش صادق نیست</b> .
<code>np.argpartition(a, kth, axis=-۱, kind='introselect', order=None)</code>	این تابع با توجه به الگوریتمی که با کمک کلیدواژه kind بهش داده میشه به آرایه متناسب به شاخصهایی که بهش داده شده بهمون برمیگردونه. مثلا ایندکس ۵ تا از بزرگترین اعضای مجموعه رو بده. چند تا پارامتر داره که باید به ترتیب تو پرانتزش تعریف شه: <p><b>a</b> ➦ اشاره به نام آرایه داره. اینجا اسم آرایه اول رو بهش میدیم <code>Array_۱</code></p> <p><b>kth</b> ➦ به ایندکس عنصر که میخواد تو این بخش شرکت کنه اشاره داره. همه اعضای کوچکتر از این عدد باید قبل این عدد قرار بگیرن و اعداد مساوی یا بزرگتر از این عدد باید بیان بعدش. (این عدد باید از تعداد سطرهای آرایه چندبعدی کمتر باشه).</p> <p><b>axis</b> ➦ به محور اشاره داره که اعضا در امتدادش مرتب میشن. پیش فرضش ۱- هست. اگه وجود نداشت از آرایه مسطح استفاده میشه (برای آرایه چند بعدی عددش رو ۰ میدیم).</p> <p><b>kind</b> ➦ الگوریتم رو انتخاب میکنه که پیش فرض <code>introselect</code> هست.</p> <p><b>order</b> ➦ اگه برای آرایه a فیلد تعریف شده باشه، این آرگومان مشخص میکنه که کدوم فیلدها اول، دوم و ... بیان. یه تک فیلد به عنوان یه رشته یا string تعریف میشه و نیازی نیست که همه فیلدها رو به این شکل خاص و ویژه کنیم. ولی معنیش این نیست که از بقیه فیلدها استفاده نمیشه. واسه اینکه بگیریم بعد از این فیلد خاص چه فیلدهایی بیان از dtype استفاده میکنیم.</p>
<code>np.sort()</code>	واسه مرتب کردن اعضای آرایه از کوچکترین به بزرگترین
<code>.ravel()</code> <code>.unravel_index()</code>	واسه داشتن ایندکس بزرگترین اعداد تو آرایه چندبعدی. در این صورت نیازی نیست axis بدیم. عدد k هم نیازی نیست از تعداد سطرها کمتر باشه. واسه رفع خطای axis و سباز از <code>unravel_index()</code> استفاده میشه.
<code>np.allclose(a, b, rtol=۱e-۰۵, atol=۱e-۰۸, equal_nan=False)</code>	واسه تطبیق دادن دو تا آرایه با هم استفاده میشه که نتیجه مقدار <b>بولینی</b> هست:



	<p><b>a و b:</b> اسم دو تا آرایه هستن که میخوایم با هم مقایسه‌شون کنیم.</p> <p><b>rtol:</b> بهش پارامتر تحمل نسبی یا the relative tolerance parameter میگن که عددش برابر هست با <math>1e-05</math></p> <p><b>atol:</b> بهش پارامتر تحمل مطلق یا the absolute tolerance parameter میگن که عددش برابر هست با <math>1e-08</math></p> <p><b>equival_nan:</b> nan مخفف not a number هست یعنی عدد نیست. وقتی بخوایم nan ها رو تو دو تا آرایه مقایسه کنیم. اگه nan تو آرایه a برابر nan تو آرایه b باشه True برمیگرده در غیر اینصورت False.</p> <p>منظور از عدد تحمل، اختلاف اعداد متناظر دو تا آرایه هست.</p>
<p><b>np.clip(a, a_min, a_max, out=None, **kwargs)</b></p>	<p>مقادیر آرایه رو برش میزنه یا کمترشون میکنه:</p> <p><b>a:</b> اسم آرایه اصلی که میخوایم روش clip یا برش انجام بدیم.</p> <p><b>a_min, a_max:</b> یه عدد به عنوان بزرگترین و کوچکترین عدد آرایه بهش میدیم و تو خروجی اگه عدد از مینیمم کمتر بود خود عدد مینیمم برمیگرده و اگه بیشتر بود خود عدد ماکزیمم</p> <p><b>out:</b> آرایه خروجی باید تو این بشینه باید shape یا شکلی با آرایه اول یکی باشه. البته نوشتنش اختیاری هست.</p> <p><b>**kwargs:</b> به بقیه کلیدواژه‌های پارامتر اشاره داره.</p> <p>عدد ماکزیمم و مینیمم نشون میده که اگه عددی زیر یا بالای این دو تا باشه، خود عدد ماکزیمم و مینیمم براشون خروجی گرفته شه.</p>
<p><b>np.where(condition, [x, y, ]/)</b></p>	<p>یه عنصر رو که از x یا y انتخاب شده بر اساس شرطی که بهش میدیم بهمون برمیگردونه.</p>
<p><b>np.pad(array, pad_width, mode='constant', **kwargs)</b></p>	<p>واسه اضافه کردن <b>پدینگ</b> به آرایه استفاده میشه. پدینگ یعنی اضافه کردن یه مقدار به لبه‌های آرایه.</p> <p><b>array:</b> آرایه‌ای که میده با آرایه اول رتبه برابر داره.</p> <p><b>pad_width:</b> هم مشخص میکنه که عددی که به لبه‌های هر محور آرایه اضافه شده چی هست.</p> <p>بخش <b>mode</b> شامل توابع constant, edge, linear_ramp, wrap, symmetric و reflect هست که پیش فرض روی constant هست و وقتی انتخابش کنیم باید constant_values هم براش بنویسیم.</p>
<p><b>np.put(a, ind, v, mode='raise')</b></p>	<p>عناصر خاصی از یه آرایه رو با مقادیر داده شده جایگزین میکنه.</p>



	<p><b>a</b>: آرایه هدف هست که میخوایم عناصرش رو تغییر بدیم.</p> <p><b>ind</b>: ایندکسهای هدف که به صورت عدد صحیح هستن.</p> <p><b>v</b>: مقداری هست که میخوایم در آرایه هدف با ایندکس مشخص قرارش بدیم. اگه <b>v</b> از <b>ind</b> کوچیکتر باشه به صورت تکراری پر میشه.</p> <p><b>mode</b>: یه گزینه هست که رفتار ایندکسهای خارج از محدوده رو تعریف میکنه و سه تا مقدار <b>raise</b>, <b>wrap</b>, <b>clip</b> قبول میکنه. <b>raise</b> پیش فرض هست و معنیش میشه اضافه کردن خطا. <b>wrap</b> به معنی پیچیدن دور چیزی و <b>clip</b> به معنی برش دادن به بازه هست.</p>
<code>np.random.choice(a, size=None, replace=True, p=None)</code>	<p>یه نمونه تصادفی از یه آرایه تک بعدی ایجاد میکنه.</p> <p><b>a</b>: یه آرایه تک بعدی اولیه هست.</p> <p><b>size</b>: مقدار پیش فرضش <b>None</b> هست. ولی میتونی مقداری به صورت <b>tuples</b> داشته باشه.</p> <p><b>replace</b>: یه عدد بولینی قبول میکنه. پیش فرضش <b>True</b> هست. یعنی مقادیر <b>a</b> میتونن چندین بار انتخاب بشن.</p> <p><b>p</b>: احتمال هر ورودی در <b>a</b> رو بررسی میکنه. اگه این بخش رو تعریف نکنیم فرض رو بر این گذاشتیم که همه ورودیهای <b>a</b> توزیع یکنواختی دارن.</p>
<code>np.random.rand(d0, d1, ..., dn)</code>	<p>اعداد تصادفی اعشاری بین ۰ تا ۱ تولید میکنه.</p>
<code>.astype()</code>	<p>تبدیل اعداد به هم مثلاً از اعشاری به عدد صحیح</p>

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>۱- <b>data</b>: شامل constants یا یه عدد ثابت، lists یا لیستها، ndarray آرایه چندبعدی</p> <p>۲- <b>index</b>: مقدار ایندکس هست</p> <p>۳- <b>dtype</b>: همون data type هست. نوع داده سریهای خروجی رو تعیین میکنه.</p> <p>۴- <b>name</b>: نام سری</p> <p>۵- <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 چسبوند. 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" = ۲)	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 = ۱ بنویسیم ستونها رو بر اساس اسمایشون 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 = ۰ باشه سطرها پاک میشن و وقتی axis = ۱ 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 مجموعه‌ای 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 = ۰</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=۰, 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 <code>۰</code> هست میتونیم بهش <code>۱</code> 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>

توضیحات	توابع و متدها
یه تابع هست که باعث میشه نمودارهایی که تو نوت بوکمون ترسیم میکنیم همینجا بتونیم ببینیمشون و تو همین نوت بوک هم ذخیره شن	<code>%matplotlib inline</code>
واسه نمایش نمودارهای ساخته شده استفاده میشه.	<code>plt.show()</code>
واسه ساخت نمودار خطی با دو مقدار X و Y استفاده میشه. تو پرانتزش میشه با <code>color =</code> به نمودار رنگ داد میشه. ضخامت و نوع خط هم میشه براش تعریف کرد. میشه رنگ و طرح رو با هم ترکیب کرد مثلاً: 'r' یا 'g'. با <code>marker</code> هم میشه روی خط علائم نشوند و به خودش و حاشیه‌هاش اندازه و رنگ داد.	<code>plt.plot()</code> <code>color =</code> <code>linewidth = / lw =</code> <code>linestyle = / ls =</code> <code>marker =</code> <code>markersize =</code> <code>markerfacecolor =</code> <code>markeredgecolor =</code> <code>markeredgewidth =</code>
برای لیبل زدن به محورهای X و Y استفاده میشه.	<code>plt.xlabel()/plt.ylabel ()</code>
برای عنوان دادن به نمودار استفاده میشه.	<code>plt.title()</code>
برای ترسیم نمودارهای سینوسی و کسینوسی استفاده میشه.	<code>.sin()</code> و <code>.cos()</code>
ترسیم نمودار میله ای استفاده میشه. ارتفاع و عرض میله‌ها رو میشه بهش داد.	<code>plt.bar()</code> <code>.get_height()/ .get_width</code>
نمودار پراکندگی داده‌ها با دو تا متغیر X و Y هست.	<code>plt.scatter()</code>
واسه ترسیم نمودار قطبی استفاده میشه که واسه نشون دادن رابطه بین دو یا چند متغیر هست. با <code>fill()</code> میشه داخل نمودار رو رنگی کرد و بهش شفافیت هم داد.	<code>plt.polar()</code> <code>.fill()</code>
واسه ترسیم نمودار پله‌ای استفاده میشه.	<code>plt.step()</code>
واسه ساخت فیگور یا شکل استفاده میشه. تعیین اندازه شکل رنگ پشت زمینه به نمودار	<code>plt.figure()</code> <code>figsize = ( , )/ dpi = ()</code> <code>facecolor = " "</code>
هیستوگرام میده. در این نمودار، محور افقی به مقادیر مختلف متغیر پیوسته اختصاص داده میشه و محور عمودی نشان دهنده فراوانی هر بازه هست. n یه لیست هست که تعداد آیتمها رو تو هر bin مشخص میکنه. bins نقطه شروع bin یا میله رو مشخص میکنه. patches هم یه لیست آبجکت هست برای هر bin. در واقع مستطیلهایی روی نمودار هستن که رنگ پیش فرض آبی دارن. <code>axvline()</code> یه خط عمودی در سرتاسر میله‌ها اضافه میکنه.	<code>plt.hist()</code> <code>bins, n, patches</code> <code>.axvline()</code>

plt.bar()	<p>نمودار جعبه‌ای با استفاده از ۵ تا عدد زیر وضعیت گروهی از داده‌ها رو به تصویر میکشه.</p> <p>✚ <b>Min یا حداقل:</b> کمترین مقدار در دسته داده‌ها (بدون در نظر گرفتن داده‌های پرت).</p> <p>✚ <b>چارک اول:</b> ۲۵ درصد داده‌ها کمتر از این مقدار هستن.</p> <p>✚ <b>چارک دوم یا میانگین:</b> مقدار وسط دسته داده‌ها هست.</p> <p>نصف مقادیر کمتر و نصف مقادیر بیشتر از اون مقدار قرار دارن.</p> <p>✚ <b>چارک سوم:</b> ۷۵ درصد داده‌ها کمتر از اون مقدار هستن.</p> <p>✚ <b>Max یا حداکثر:</b> بزرگترین مقدار در دسته داده‌ها (بدون در نظر گرفتن داده‌های پرت).</p>
plt.violinplot() showmeans = True	<p>نمودار ویولنی از نمودار جعبه‌ای ساده اطلاعات بیشتری رو منتقل میکنه. برای مقایسهٔ داده‌های آماری به صورت خلاصه (مثل بازه‌ها و چارکها) کاربرد داره ولی امکان مشاهدهٔ تغییرات و اختلافات در داده رو نمیده.</p>
.corr() plt.imshow() plt.colorbar()	<p>هیت مپ Heat map یک روش دیداری برای نمایش داده‌های دو بعدی هست. در این روش، هر مقدار داده با یک رنگ متفاوت نمایش داده میشه.</p> <p>✚ از <b>corr()</b> واسه ماتریس همبستگی استفاده میکنیم. اعدادش بین -۱ تا +۱ هستن. یعنی از منفی‌ترین تا مثبت‌ترین. از قرمز تیره میده تا قرمز روشن.</p> <p>✚ از <b>imshow()</b> برای نمایش تصویر استفاده میکنیم. بهمون تصاویر مربعی میده.</p> <p>✚ <b>colorbar()</b> تعریف میکنیم که کنار تصویر بهمون یه میله رنگی هم بده.</p>
plt.stackplot()	<p>واسه نمایش توالی زمانی استفاده میشه که به صورت پشته‌ای از منحنی‌ها رسم میشه.</p>
plt.pie() plt.axis()	<p>نمودار pie برامون میسازه. یه autopct داره که نشون دهنده میزان عدد اعشار هر نقطه داده هست. باید تو پرانتز plt.axis() کلمه 'equal' وارد شه که x و y برابر بگیره و دایره خروجی بده.</p>
plt.subplot()	<p>ترسیم نمودارهای مشابه به صورت همزمان.</p> <p>سه تا آرگومان داره اولی عدد <b>سطر</b>، دومی عدد <b>ستون</b> و سومی <b>شماره نمودار</b> هست. رنگ هم میشه بهشون داد.</p>
.add_axes()	<p>واسه اضافه کردن محور یا axes به فیگور استفاده میشه.</p> <p>۴ تا عدد میگیره <u>دو تا اول نقاط شروع و دو تا عدد دوم نقاط پایان</u> محورها هستن.</p>

<code>.legend()</code>	واسه راهنما زدن استفاده میشه. با <code>loc =</code> مشخص میکنیم که راهنما کجا بیاد.
<code>plt.tight_layout()</code>	وقتی چند تا نمودار کنار هم میذاریم و میخوایم اعداد محورهایشون تو هم نرن استفاده میشه.
<code>type()</code> <code>.set_title()</code>	نوع داده محور رو مشخص میکنه. واسه عنوان دادن به هر نمودار موقع ترسیم نمودارهای چندتایی استفاده میشه.
<code>.savefig()</code>	واسه ذخیره کردن نمودارها استفاده میشه.
<code>.set_xlim()/ .set_ylim()</code>	محدوده محورها رو میشه تغییر داد. اولین عدد داخل پرانتز <b>شروع</b> و دومین عدد <b>پایانش</b> هست.
<code>set.xticks()/ set.yticks()</code> <code>set_xticklabels()</code>	واسه تغییر اعداد و عبارات روی محور <code>x</code> و <code>y</code> استفاده میشه. با <code>label</code> میشه اعداد رو به صورت نوشتاری روی محور نشون داد.
<code>.spins[].set_visible()</code>	خطوط <code>+</code> طرف محور هستن که میشه برشون داشت. پیش فرضشون <code>True</code> هست.
<code>.grid()</code> <code>ls = / lw =</code>	واسه شبکه بندی نمودار استفاده میشه. تو پرانتزش باید <code>True</code> بنویسیم. میشه بهش ضخامت و نوع خط هم داد.

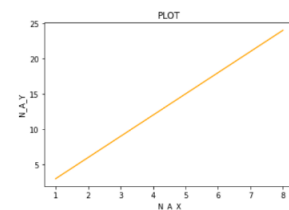
## نمودارهای ترسیم شده در Matplotlib

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import calendar
%matplotlib inline
```

### ترسیم چند نمودار

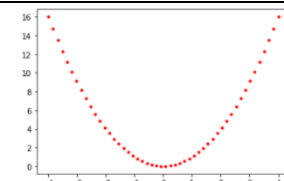
#### تمرین ۱

```
N_A_X = np.arange(۱,۹)
N_A_Y = np.arange(۳,۲۷,۳)
plt.xlabel("N_A_X")
plt.ylabel("N_A_Y")
plt.title("PLOT")
plt.plot(N_A_X, N_A_Y, color = "orange")
plt.show()
```



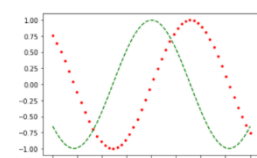
#### تمرین ۲

```
LS_X = np.linspace(-۴,۴,۵۰)
LS_Y = LS_X ** ۲
plt.plot(LS_X, LS_Y, "r.")
plt.show()
```



#### تمرین ۳

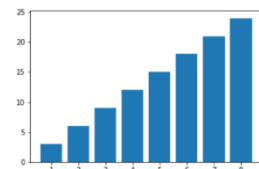
```
plt.plot(LS_X, np.sin(LS_X), "r.")
plt.plot(LS_X, np.cos(LS_X), "g--")
plt.show()
```





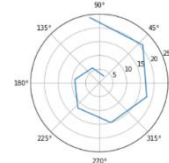
### تمرین ۴

```
N_A_X = np.arange(۱,۹)
N_A_Y = np.arange(۳,۲۷,۳)
plt.bar(N_A_X, N_A_Y)
plt.show()
```



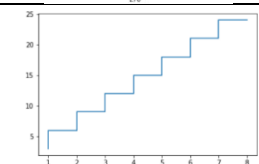
### تمرین ۵

```
plt.polar(N_A_X, N_A_Y)
plt.show()
```



### تمرین ۶

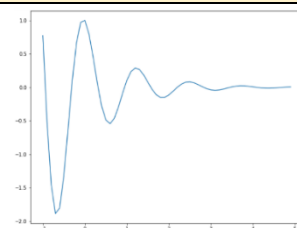
```
plt.step(N_A_X, N_A_Y)
plt.show()
```



## Figure, Subplot and Axes

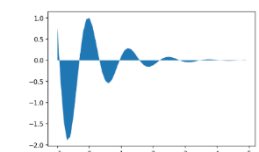
### تمرین ۱

```
Figure = plt.figure(figsize = (۱۰,۸))
X = np.arange(-۱,۵,۰,۱)
Y = np.exp(-X)*np.cos(۵*X)
plt.plot(X,Y)
plt.show()
```



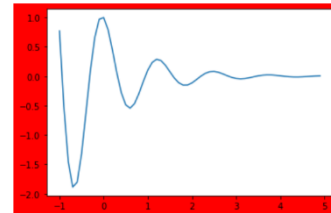
### تمرین ۲

```
plt.figure(dpi = (۱۵۰))
plt.stackplot(X,Y)
plt.show()
```



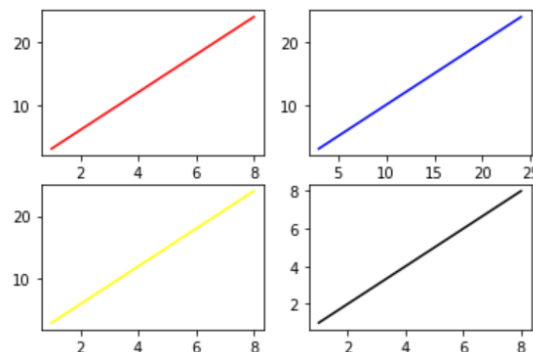
### تمرین ۳: اجرا شده در نوت بوک ژوپیتتر

```
Figure = plt.figure(facecolor = 'red')
X = np.arange(-۱,۵,۰,۱)
Y = np.exp(-X)*np.cos(۵*X)
plt.plot(X,Y)
plt.show()
```



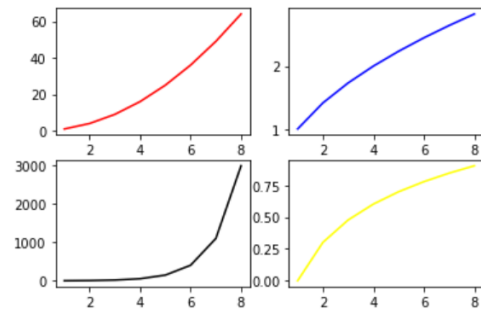
### تمرین ۴

```
X۱ = np.arange(۱,۹)
Y۱ = np.arange(۳,۲۷,۳)
plt.subplot(۲,۲,۱)
plt.plot(X۱,Y۱,"red")
plt.subplot(۲,۲,۲)
plt.plot(Y۱,Y۱,"blue")
plt.subplot(۲,۲,۳)
plt.plot(X۱,Y۱,"yellow")
plt.subplot(۲,۲,۴)
plt.plot(X۱,X۱,"black")
plt.show()
```



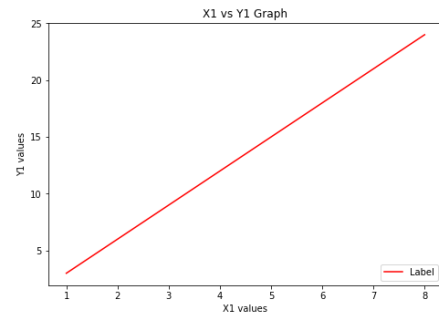
### تمرین #۵

```
plt.subplot(۲,۲,۱)
plt.plot(X',X'*X',"red")
plt.subplot(۲,۲,۲)
plt.plot(X',np.sqrt(X'),"blue")
plt.subplot(۲,۲,۳)
plt.plot(X',np.exp(X'),"black")
plt.subplot(۲,۲,۴)
plt.plot(X',np.log'(X'),"yellow")
plt.show()
```



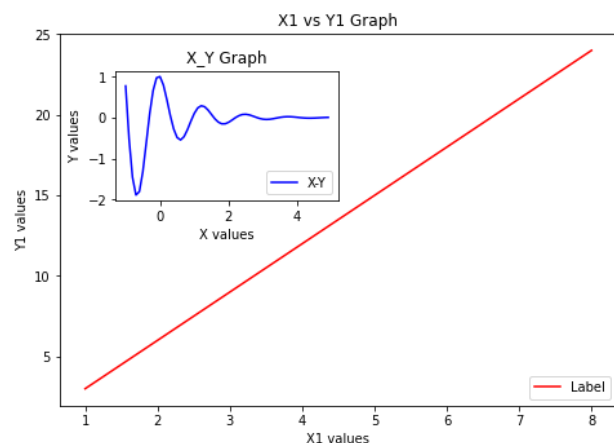
### تمرین #۶

```
Figure = plt.figure()
Axes = Figure.add_axes([۰,۰,۱,۱])
Axes.set_xlabel("X' values")
Axes.set_ylabel("Y' values")
Axes.set_title("X' vs Y' Graph")
Axes.plot(X',Y',"red")
Axes.legend(labels = ["Label"], loc = "lower right")
plt.show()
```



### تمرین #۷

```
Figure = plt.figure()
Axes = Figure.add_axes([۰,۰,۱,۱])
Axes.set_xlabel("X' values")
Axes.set_ylabel("Y' values")
Axes.set_title("X' vs Y' Graph")
Axes.plot(X',Y',"red")
Axes.legend(labels = ["Label"], loc = "lower right")
Axes_۲ = Figure.add_axes([۰,۱,۰,۵۵,۰,۴,۰,۳۵])
Axes_۲.set_xlabel("X values")
Axes_۲.set_ylabel("Y values")
Axes_۲.set_title("X_Y Graph")
Axes_۲.plot(X,Y,"blue")
Axes_۲.legend(labels = ["X-Y"], loc = "lower right")
plt.show()
```

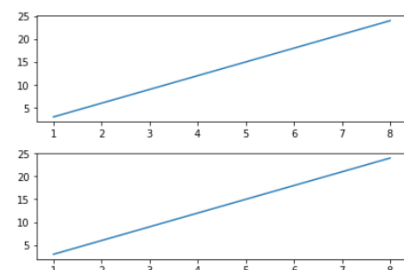


## سفرسی سازی شکل‌ها یا Figure Customization

### تمرین #۱

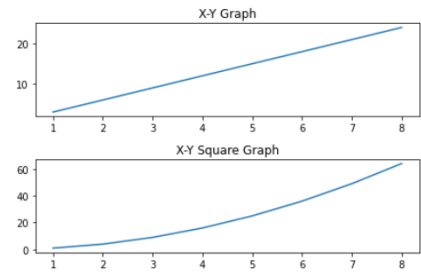
```
X = np.arange(۱,۹)
Y = np.arange(۳,۲۷,۳)
Figure, Axes = plt.subplots(nrows=۲, ncols=۱)
plt.tight_layout()
print(type(Axes))
print(Axes[۰])
print(Axes[۱])
for ax in Axes:
    print(ax.plot(X,Y))
plt.show()
```

```
<class 'numpy.ndarray'>
AxesSubplot(0.0929977,0.577778;0.863484x0.370833)
AxesSubplot(0.0929977,0.0965278;0.863484x0.370833)
[<matplotlib.lines.Line2D object at 0x000001E0C7E94820>]
[<matplotlib.lines.Line2D object at 0x000001E0C7E79640>]
```



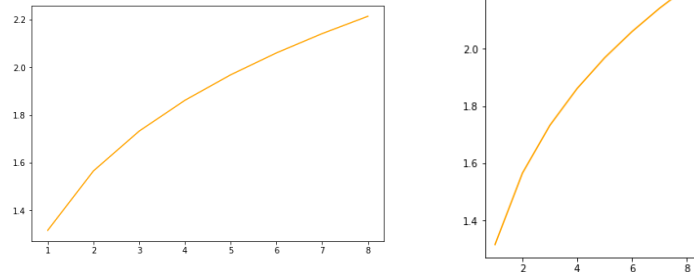
## تمرین ۲

```
Figure, Axes = plt.subplots(nrows=۲, ncols=۱)
Axes[۰].plot(X,Y)
Axes[۰].set_title("X-Y Graph")
Axes[۱].plot(X,X**۲)
Axes[۱].set_title("X-Y Square Graph")
plt.tight_layout()
plt.show()
```



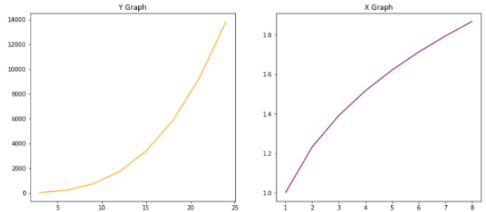
## تمرین ۳

```
Figure_S = plt.figure(figsize=(۶,۴))
Axes_S = Figure_S.add_axes([۰,۰,۱,۱])
Axes_S.plot(X,Y**۰.۲۵, color="orange")
plt.show()
Figure_S = plt.figure(figsize=(۳,۴))
Axes_S = Figure_S.add_axes([۰,۰,۱,۱])
Axes_S.plot(X,Y**۰.۲۵, color="orange")
plt.show()
```



## تمرین ۴

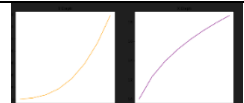
```
Figure_۲, Axes_۲ = plt.subplots(nrows=۱, ncols=۲,
figsize=(۱۴,۶))
Axes_۲[۰].plot(Y,Y**۳, color="orange")
Axes_۲[۰].set_title("Y Graph")
Axes_۲[۱].plot(X,X**۰.۳, color="purple")
Axes_۲[۱].set_title("X Graph")
plt.show()
```



## تمرین ۵: اجرا شده در نوت بوک ژوپیتر

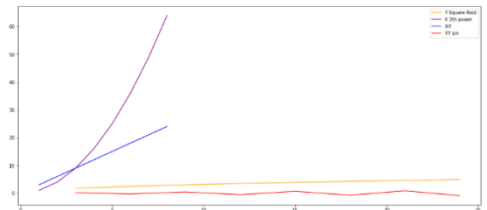
```
Figure_۲.savefig("Figure_۲.png")
```

Figure\_2



## تمرین ۶

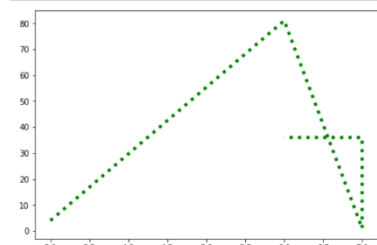
```
Figure_۳, Axes_۳ = plt.subplots(figsize=(۱۴,۶))
Axes_۳ = Figure_۳.add_axes([۰,۰,۱,۱])
Axes_۳.plot(Y,Y**۰.۵, label="Y Square Root", color="orange")
Axes_۳.plot(X,X**۲, label="X ۲th power", color="purple")
Axes_۳.plot(X,Y, label="X-Y", color="blue")
Axes_۳.plot(Y,np.sin(Y), label="Y-Y sin", color="red")
Axes_۳.legend()
plt.show()
```



## سفارشی سازی گرافها در matplotlib

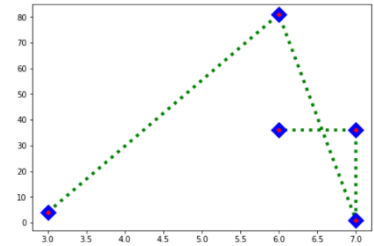
## تمرین ۱

```
X = np.random.randint(۱,۱۲,۵)
Y = np.random.randint(۱,۱۰,۵)
Figure = plt.figure()
Axes = Figure.add_axes([۰,۰,۱,۱])
Axes.plot(X,Y**۲,color="green", lw=۴, ls=":")
plt.show()
```



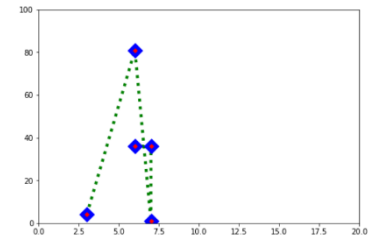
### تمرین ۲

```
Figure = plt.figure()
Axes = Figure.add_axes([0,0,1,1])
Axes.plot(X,Y**۲,color="green", lw=۴, ls=":", marker= "D",
          markersize = ۱۰, markerfacecolor = "red",
          markeredgewidth = "blue", markeredgewidth = ۵)
plt.show()
```



### تمرین ۳

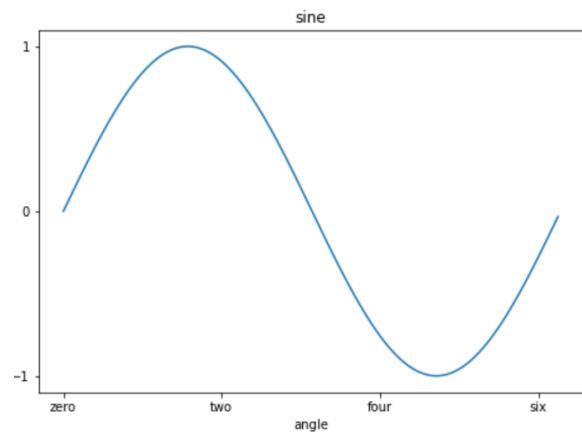
```
Figure = plt.figure()
Axes = Figure.add_axes([0,0,1,1])
Axes.plot(X,Y**۲,color="green", lw=۴, ls=":", marker= "D",
          markersize = ۱۰, markerfacecolor = "red",
          markeredgewidth = "blue", markeredgewidth = ۵)
Axes.set_xlim(0,۲۰)
Axes.set_ylim(0,۱۰۰)
plt.show()
```



## Grid, Spines, Ticks

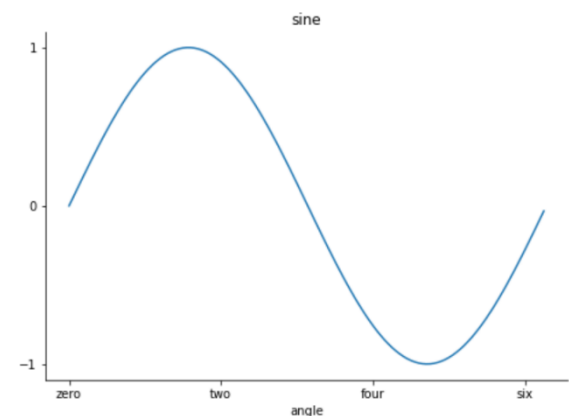
### تمرین ۱

```
X = np.arange(0,math.pi*۲,0.۵)
Figure = plt.figure()
Axes = Figure.add_axes([0,0,1,1])
Y = np.sin(X)
Axes.plot(X,Y)
Axes.set_xlabel("angle")
Axes.set_title("sine")
Axes.set_xticks([0,۲,۴,۶])
Axes.set_xticklabels(["zero","two","four","six"])
Axes.set_yticks([-1,0,1])
plt.show()
```



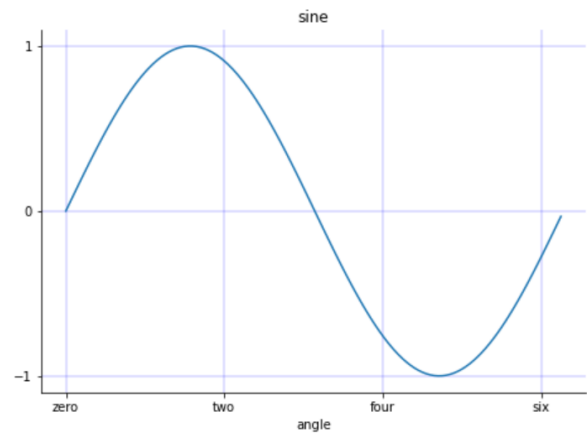
### تمرین ۲

```
X = np.arange(0,math.pi*۲,0.۵)
Figure = plt.figure()
Axes = Figure.add_axes([0,0,1,1])
Y = np.sin(X)
Axes.plot(X,Y)
Axes.set_xlabel("angle")
Axes.set_title("sine")
Axes.set_xticks([0,۲,۴,۶])
Axes.set_xticklabels(["zero","two","four","six"])
Axes.set_yticks([-1,0,1])
Axes.spines["right"].set_visible(False)
Axes.spines["top"].set_visible(False)
plt.show()
```



### تمرین ۳

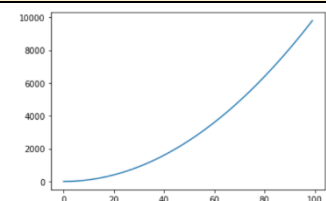
```
X = np.arange(0,math.pi*۲,0.۰۵)
Figure = plt.figure()
Axes = Figure.add_axes([0,0,1,1])
Y = np.sin(X)
Axes.plot(X,Y)
Axes.set_xlabel("angle")
Axes.set_title("sine")
Axes.set_xticks([0,۲,۴,6])
Axes.set_xticklabels(["zero","two","four","six"])
Axes.set_yticks([-1,0,1])
Axes.spines["right"].set_visible(False)
Axes.spines["top"].set_visible(False)
Axes.grid(color="b", ls="-", lw=0.۲۵)
plt.show()
```



### نمودارهای پایه matplotlib

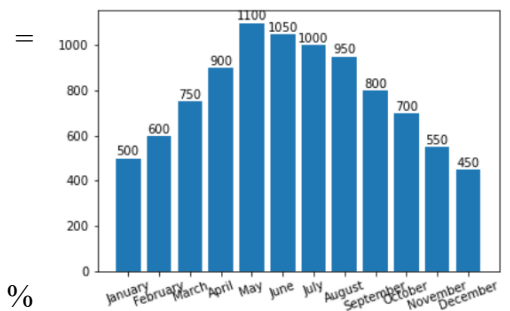
#### #Line plot

```
X = range(100)
Y = [value ** ۲ for value in X]
plt.plot(X,Y)
plt.show()
```



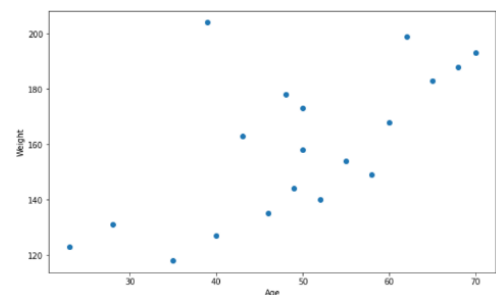
#### #bar plot

```
month_name = [1,2,3,4,5,6,7,8,9,10,11,12]
units_sold
[500,600,750,900,1100,1050,1000,950,800,700,550,450]
fig, ax = plt.subplots()
plt.xticks(month_name, calendar.month_name[1:13], rotation=۲۰)
plot = ax.bar(month_name,units_sold)
for rect in plot:
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/۲., 1.۰۰۲*height, '%d' %
int(height), ha = 'center', va = 'bottom')
plt.show()
```



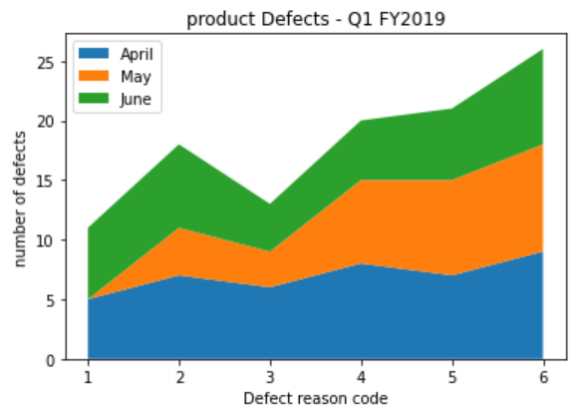
#### #scatter bar

```
plt.figure(figsize=(10,6))
age_weight = pd.read_excel('scatter_ex.xlsx', 'age_weight')
X = age_weight['age']
Y = age_weight['weight']
plt.scatter(X,Y)
plt.xlabel('Age')
plt.ylabel('Weight')
plt.show()
```



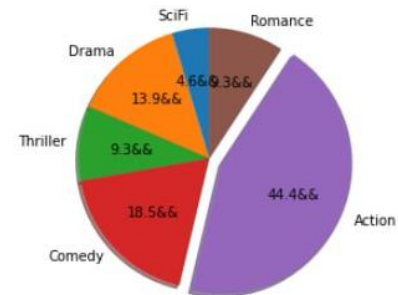
### #Stack plot

```
x = np.array([1,2,3,4,5,6], dtype=np.int32)
Apr = [5,7,6,8,7,9]
May = [0,4,3,7,8,9]
June = [6,7,4,0,6,8]
labels = ["April","May","June"]
fig, ax = plt.subplots()
ax.stackplot(x, Apr, May, June, labels=labels)
ax.legend(loc=2)
plt.xlabel('Defect reason code')
plt.ylabel('number of defects')
plt.title('product Defects - Q1 FY2019')
plt.show()
```



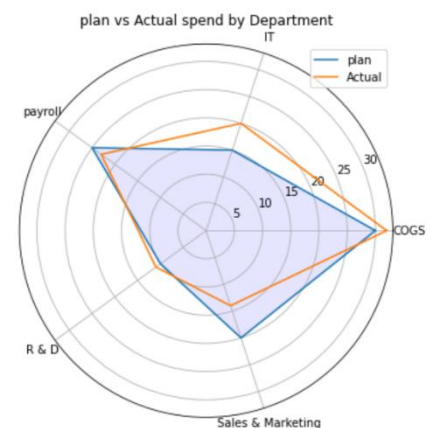
### #Pie plot: with Jupyter Notebook

```
labels = ["SciFi", "Drama", "Thriller", "Comedy", "Action", "Romance"]
sizes = [5, 15, 10, 20, 48, 10]
explode = (0, 0, 0, 0, 1, 0)
plt.pie(sizes, labels=labels, explode=explode, autopct='%1.1f%%',
        shadow=True, startangle=90)
plt.axis('equal')
plt.show()
```



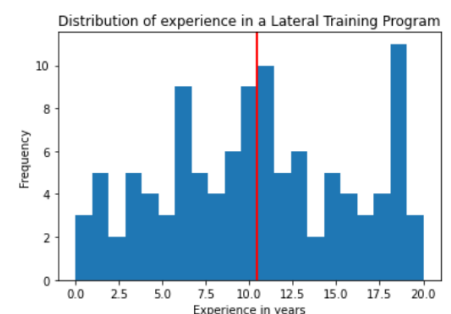
### #polar plot

```
Depts= ["COGS", "IT", "payroll", "R & D", "Sales & Marketing"]
rp = [30, 15, 20, 10, 20, 30]
ra = [32, 20, 23, 11, 14, 32]
theta = np.linspace(0, 2*np.pi, len(rp))
plt.figure(figsize=(10, 6))
plt.subplot(polar=True)
(lines, labels) = plt.thetagrids(range(0, 360, int(360/len(Depts))),
(Depts))
plt.plot(theta, rp)
plt.fill(theta, rp, 'b', alpha=0.1)
plt.plot(theta, ra)
plt.legend(labels=('plan', 'Actual'), loc=1)
plt.title("plan vs Actual spend by Department")
plt.show()
```



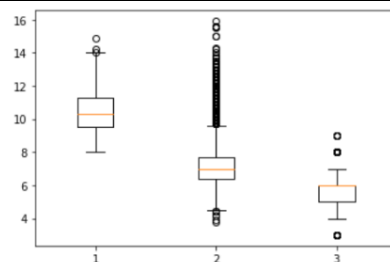
### #Hist plot

```
gre_exp = np.array([12, 15, 13, 20, 19, 20, 11, 19, 11, 12, 19, 13, 12, 10, 6, 19, 3, 1, 1, 0, 4, 4, 6, 0, 3, 7, 12, 7, 9, 8, 12, 11, 11, 18, 19, 18, 19, 3, 6, 0, 6, 9, 11, 10, 14, 14, 16, 17, 17, 19, 0, 2, 0, 3, 1, 4, 6, 6, 8, 7, 7, 6, 7, 11, 11, 10, 11, 10, 13, 13, 10, 18, 20, 19, 1, 0, 8, 16, 19, 19, 17, 16, 11, 1, 10, 13, 10, 3, 8, 6, 9, 10, 10, 19, 2, 4, 0, 6, 9, 11, 10, 9, 10, 9, 10, 16, 18, 13])
nbins = 21
n, bins, patches = plt.hist(gre_exp, bins = nbins)
plt.xlabel("Experience in years")
plt.ylabel("Frequency")
plt.title("Distribution of experience in a Lateral Training Program")
plt.axvline(x=gre_exp.mean(), linewidth=2, color='r')
plt.show()
```



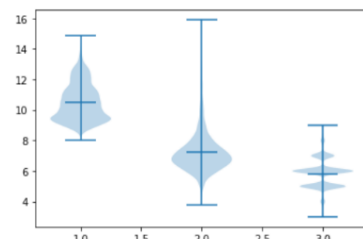
### #box plot

```
wine_quality = pd.read_csv('winequality.csv')
data = [wine_quality['alcohol'], wine_quality['fixed acidity'],
        wine_quality['quality']]
plt.boxplot(data)
plt.show()
```



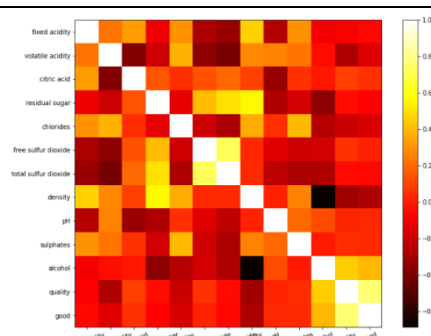
### #violin plot

```
wine_quality = pd.read_csv('winequality.csv')
data = [wine_quality['alcohol'], wine_quality['fixed acidity'],
        wine_quality['quality']]
plt.violinplot(data, showmeans = True)
plt.show()
```



### #Heat map with Jupyter Notebook

```
wine_quality = pd.read_csv('winequality.csv')
corr = wine_quality.corr()
plt.figure(figsize=(12,9))
plt.imshow(corr, cmap = 'hot')
plt.colorbar()
plt.xticks(range(len(corr)), corr.columns, rotation = 20)
plt.yticks(range(len(corr)), corr.columns)
plt.show()
```



### خلاصه دستورهایی seaborn

توابع و متدها	توضیحات
<code>sns.set()</code>	تنظیمات ترسیم رو به تنظیمات پیش فرض seaborn تغییر میده.
<code>set_style(style, [rc])</code> <code>axes_style(style, [rc])</code>	پارامترهایی رو تنظیم میکنه که سبک کلی نمودارها رو کنترل کنه. پارامترهایی رو دریافت میکنه که سبک کلی نمودارها رو کنترل میکنه. ✓ <b>Style</b> یه دیکشنری از پارامترها یا یکی از استایل‌های مقابل رو میگیره: <code>{darkgrid, whitegrid, dark, white, ticks}</code> ✓ <b>rc</b> دیکشنریهای استایل پیش فرض رو لغو میکنه. آوردنش تو پراپرتی اختیاری هست. پارامترهای سبک یا <code>style parameters</code> ویژگی‌هایی مثل رنگ پس زمینه و فعال بودن یا نبودن شبکه یا <code>grid</code> به طور پیش فرض رو کنترل می‌کنن. این کار با استفاده از سیستم <code>matplotlib rcParams</code> انجام میشه.



Seaborn.set_context(context, [font_scale]. [rc])	<p>پارامترهایی رو تنظیم میکنه که مقیاس بندی عناصر نمودار رو کنترل میکنه مثل اندازه برچسبها، خطوط و ....</p> <p>✓ <b>context</b> چهارتا پارامتر دیکشنری قبول میکنه: <b>paper, notebook, talk, poster</b></p> <p>✓ <b>font_scale</b> اندازه فونت عناصر رو مشخص میکنه.</p> <p><b>rc</b> واسه لغو زمینه های پیش فرض استفاده میشه. مثل متدهای بالا آوردنش تو پرانتز اختیاری هست.</p>
.tolist()	اطلاعات ستونهای جدول رو به لیست تبدیل میکنه.
sns.boxplot()	نمودار جعبه ای
sns.despine(True)	برای پاک کردن <u>اسپینهای</u> اطراف نمودار
seaborn.color_palette([palette],[n_colors],[desat])	<p>واسه اختصاص دادن <u>رنگ</u> به عناصر</p> <p>✓ <b>palette</b> اسم پالت رنگی هست. پیش فرضش none هست یعنی هیچی انتخاب نشه. یه پارامتر اختیاری هست.</p> <p>✓ <b>n_colors</b> تعداد رنگ هست. اگه تعداد رنگی که میدیم از تعداد رنگهای خودش بیشتر باشه حالت دایره عمل میکنه و برمیگرده به اولین رنگ. اینم یه پارامتر اختیاری هست.</p> <p><b>Desat</b> میزان ترکیبات هر رنگ رو مشخص میکنه. این هم اختیاری هست.</p>
sns.palplot()	واسه خروجی گرفتن از پلتهای رنگی
sns.light_palette()	پالتهای رنگی تک رنگ هستن و از تیره به روشن یا برعکس همون رنگ رو بهت خروجی میدن.
sns.color_palette() sns.diverging_palette()	<p>با دادن عدد یا اسم پالت ساخته میشه. متداولترین پالتهای رنگی diverging شامل موارد زیر هستن:</p> <p>🌈 <b>پالت رنگهای گرم و سرد یا coolwarm:</b> این نوع پالت شامل ترکیب رنگهای گرم مثل نارنجی یا قرمز و رنگهای سرد مثل آبی یا سبز هست. این ترکیبها تبدیل های جذابی از نظر رنگی ایجاد می کنن.</p> <p>🌈 <b>پالت رنگهای روشن و تاریک:</b> در این نوع پالت، یک رنگ روشن با یک رنگ تاریک ترکیب</p>

	<p>می‌شه. این تضاد نوری تبدیل‌های شدیدی از نظر رنگی به وجود می‌آرن.</p> <p>🌈 <b>پالت رنگ‌های متقابل:</b> در این نوع پالت، دو رنگ متقابل از دو قسمت مختلف پالت برای تاکید بر تقابل رنگی استفاده میشن.</p> <p><b>پالت رنگ‌های تاریک و روشن در کنار یک رنگ وسطی:</b> در این نوع، دو رنگ تاریک و روشن در کنار یک رنگ وسطی ترکیب می‌شن تا تبدیل‌های متعادلی از نظر رنگی ایجاد کنن.</p>
<b>sns.load_data()</b>	برای اضافه کردن فایل‌های اکسل و CSV
<b>sns.barplot</b> (x=None, y=None, hue=None, data=None, order=None, hue_order=None, estimator=<function mean at ۰x۰۰۰۰۰۰۲BC۳EB۵C۴CA>, ci=۹۵, n_boot=۱۰۰۰, units=None, orient=None, color=None, palette=None, saturation=۰٫۷۵, errcolor='۲۶', errwidth=None, capsize=None, dodge=True, ax=None, **kwargs)	یک تابع در کتابخانه Seaborn در پایتون هست که برای رسم نمودارهای میله‌ای (Bar Plot) استفاده می‌شه. باید بهش دو تا ستون x و y معرفی شه. اندازه هر ستونش میانگین اعداد همون ستون در جدول هستن.
<b>seaborn.kdeplot</b> (data=None, *, x=None, y=None, hue=None, weights=None, palette=None, hue_order=None, hue_norm=None, color=None, fill=None, multiple='layer', common_norm=True, common_grid=False, cumulative=False, bw_method='scott', bw_adjust=۱, warn_singular=True, log_scale=None, levels=۱۰, thresh=۰٫۰۵, gridsize=۲۰۰, cut=۳, clip=None, legend=True, cbar=False, cbar_ax=None, cbar_kws=None, ax=None, **kwargs)	<b>KDE</b> مخفف Kernel Density Estimation یا تخمین چگالی احتمال بر اساس داده‌های مشاهده شده هست. نشون میده که <u>چقدر احتمال وقوع مقادیر مختلف در داده‌ها وجود داره</u> . یه نمودار اسموتینگ یا Smoothed histogram بهمون خروجی میده که همه مقادیر داده رو پوشش میده.
<b>sns.jointplot</b> (x, y, data=None, kind='scatter', stat_func=None, color=None, height=۶, ratio=۵, space=۰٫۲, dropna=True, xlim=None, ylim=None, joint_kws=None, marginal_kws=None, annot_kws=None, **kwargs)	این نمودار بهمون کمک میکنه که <u>رابطه بین دو تا متغیر</u> رو بررسی کنیم و توزیع هر متغیر رو جداگونه نمایش بدیم. تو قسمت kind که نوع نمودار رو مشخص میکنه میتونی از kde, reg, hist و ... هم استفاده کنی که نمودارهای متعدد داشته باشی.
<b>seaborn.pairplot</b> (data, *, hue=None, hue_order=None, palette=None, vars=None,	واسه ترسیم <u>نمودارهای جفتی</u> بین متغیرهای مختلف استفاده میشه. این نمودار به صورت پیش فرض یه

<pre>x_vars=None, y_vars=None, kind='scatter', diag_kind='auto', markers=None, height=۲/۵, aspect=۱, corner=False, dropna=False, plot_kws=None, diag_kws=None, grid_kws=None, size=None)</pre>	<p>شبکه از محورها رو ایجاد میکنه طوری که هر متغیر عددی در داده‌ها با متغیرهای دیگه در محور y به اشتراک گذاشته میشه. ازش واسه دیدن تعاملات بین متغیرهای مختلف در یه مجموعه داده استفاده میشه.</p>
<pre>seaborn.violinplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=۰.۷۵, fill=True, inner='box', split=False, width=۰.۸, dodge='auto', gap=۰, linewidth=None, linecolor='auto', cut=۲, gridsize=۱۰۰, bw_method='scott', bw_adjust=۱, density_norm='area', common_norm=False, hue_norm=None, formatter=None, log_scale=None, native_scale=False, legend='auto', scale=&lt;deprecated&gt;, scale_hue=&lt;deprecated&gt;, bw=&lt;deprecated&gt;, inner_kws=None, ax=None, **kwargs)</pre>	<p>. نمودار ویولن یک ابزار مفید در تجسم توزیع داده‌های عددی و احتمالی هست. این نمودار با ترکیب ویژگی‌های نمودار جعبه‌ای (box plot) و نمودار چگالی هسته (kernel density plot)، اطلاعات زیادی از توزیع داده رو نمایش می‌ده. برای نمایش توزیع داده‌ها و تجزیه و تحلیل آماری استفاده میشه. عرض نمودار ویولنی نشون دهنده توزیع داده‌های احتمالی هست که همیشه توزیع داده‌های مختلف رو هم با هم مقایسه کرد. در قسمت میانی نمودار همیشه نمایش پیکها و توزیع چگالی داده‌ها رو دید.</p>
<pre>seaborn.FacetGrid(data, *, row=None, col=None, hue=None, col_wrap=None, sharex=True, sharey=True, height=۳, aspect=۱, palette=None, row_order=None, col_order=None, hue_order=None, hue_kws=None, dropna=False, legend_out=True, despine=True, margin_titles=False, xlim=None, ylim=None, subplot_kws=None, gridspec_kws=None) FacitGrid.map(func, *args, **kwaegs)</pre>	<p>. FacetGrid واسه مصورسازی متغیرهای چندگانه به صورت مجزا عالیه.</p> <p>❖ از دستور <b>FacetGrid.map(func, *args, **kwargs)</b> هم همیشه واسه ترسیم نمودارهای متعدد در یه شبکه تعریف شده استفاده کرد. هر تابع یا func رو واسه هر زیرمجموعه داده در هر شبکه فراخونی میکنه. اگه در حالت hue باشه باید یه آرگومان label هم قبول کنه. بعد داده‌های هر متغیر به ترتیب مشخص شده در فراخوانی به func منتقل میشن.</p> <p>❖ از <b>seaborn.FacetGrid.map_dataframe</b> که به صورت <b>FacetGrid.map_dataframe(func, *args, **kwargs)</b> نوشته میشه واسه تطبیق دیتافریمها با نمودارها استفاده میشه. این متد واسه ترسیم مالتی پلاتها بر اساس داده‌های موجود در یه دیتافریم به صورت هزمان بکار میره. با کمک <b>map_DataFrame</b> میتونی یه تابع</p>

سفارشی رو تعریف کنی و این تابع رو روی داده‌های DataFrame اعمال کنی و نمودارهایی با توجه به این تابع ایجاد کنی.

```
seaborn.regplot(data=None, *, x=None, y=None,
x_estimator=None, x_bins=None, x_ci='ci',
scatter=True, fit_reg=True, ci=95, n_boot=1000,
units=None, seed=None, order=1, logistic=False,
lowess=False, robust=False, logx=False,
x_partial=None, y_partial=None, truncate=True,
dropna=True, x_jitter=None, y_jitter=None,
label=None, color=None, marker='o',
scatter_kws=None, line_kws=None, ax=None)
```

نمودار رگرسیون یا Regression Plot، واسه نشون دادن رابطه بین دو تا متغیر استفاده میشه. در این نمودار، محور X به یک متغیر مستقل و محور Y به یک متغیر وابسته اختصاص داده میشه. این نمودار به صورت خطی یا غیرخطی رسم میشه و خط رسم شده، بهترین تقریب برای رابطه بین دو تا متغیر هست. از این نمودار در آمار و همچنین در یادگیری ماشین واسه پیدا کردن رابطه بین دو تا متغیر استفاده میشه.

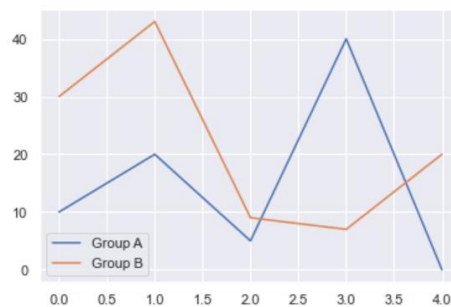
## نمودارهای ترسیم شده در seaborn

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

### کنترل زیبایی فیکورها

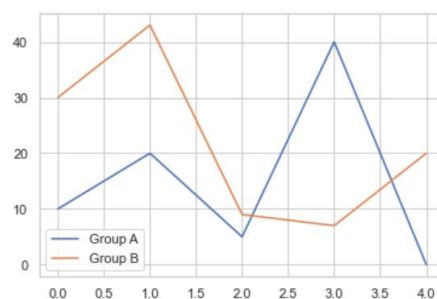
#### تمرین #۱

```
sns.set()
plt.figure()
X1 = [10, 20, 50, 40, 0]
X2 = [30, 43, 9, 7, 20]
plt.plot(X1, label = "Group A")
plt.plot(X2, label = "Group B")
plt.legend()
plt.show()
```



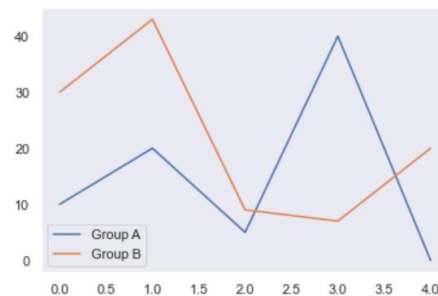
#### تمرین #۲

```
sns.set_style("whitegrid")
plt.figure()
X1 = [10, 20, 50, 40, 0]
X2 = [30, 43, 9, 7, 20]
plt.plot(X1, label = "Group A")
plt.plot(X2, label = "Group B")
plt.legend()
plt.show()
```



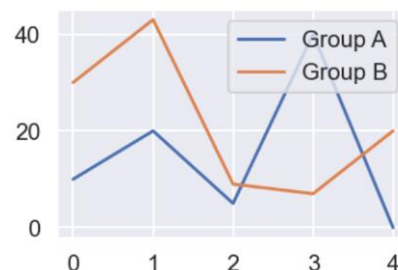
#### تمرین #۳

```
sns.set()
plt.figure()
X1 = [10, 20, 5, 40, 0]
X2 = [30, 43, 9, 7, 20]
with sns.axes_style('dark'):
    plt.plot(X1, label = "Group A")
    plt.plot(X2, label = "Group B")
plt.legend()
plt.show()
```



### تمرین ۴

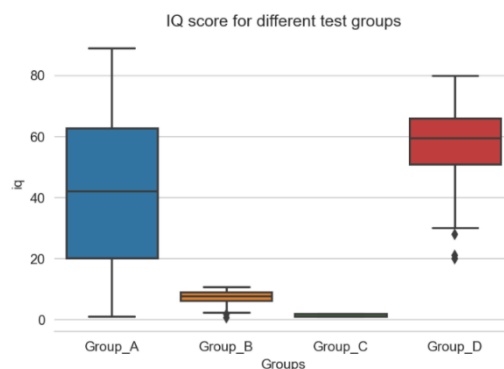
```
sns.set_context("poster")
plt.figure()
X1 = [10, 20, 5, 40, 0]
X2 = [30, 43, 9, 7, 20]
plt.plot(X1, label = "Group A")
plt.plot(X2, label = "Group B")
plt.legend()
plt.show()
```



### تمرین ۵

```
My_data = pd.read_csv('gpa_iq.csv')
Group_A = My_data[My_data.columns[0]].tolist()
Group_B = My_data[My_data.columns[1]].tolist()
Group_C = My_data[My_data.columns[2]].tolist()
Group_D = My_data[My_data.columns[3]].tolist()
data = pd.DataFrame({'Groups': ['Group_A'] * len(Group_A) +
                                ['Group_B'] * len(Group_B) +
                                ['Group_C'] * len(Group_C) +
                                ['Group_D'] * len(Group_D),
                    'iq': Group_A + Group_B + Group_C +
                        Group_D})
```











```
plt.figure(dpi=100)
sns.set_style('whitegrid')
sns.boxplot(data=data, x='Groups', y='iq')
sns.despine(left=True, right=True, top=True)
plt.title('IQ score for different test groups')
plt.show()
```



### پالت‌های رنگی در seaborn

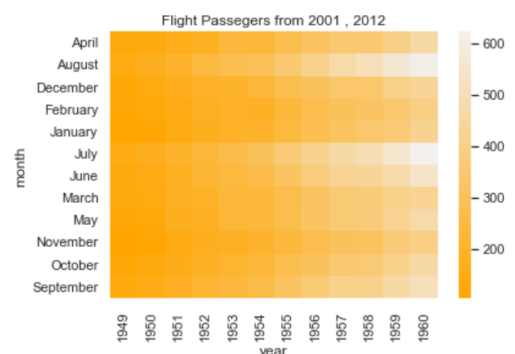
### پالت‌های رنگی categorical

خروجی	کد	نوع پالت
	palette_deep = sns.color_palette("deep") sns.palplot(palette_deep) palette_deep	deep
	palette_muted = sns.color_palette("muted") sns.palplot(palette_muted) palette_muted	muted
	palette_bright = sns.color_palette("bright") sns.palplot(palette_bright) palette_bright	bright
	palette_pastel = sns.color_palette("pastel") sns.palplot(palette_pastel) palette_pastel	pastel

	<pre>palette_dark = sns.color_palette("dark") sns.palplot(palette_dark) palette_dark</pre>	dark
	<pre>palette_colorblind = sns.color_palette("colorblind") sns.palplot(palette_colorblind) palette_colorblind</pre>	colorblind
<b>پالت‌های رنگی sequential</b>		
<b>خروجی</b>	<b>کد</b>	<b>نوع پالت</b>
	<pre>palette_seq\ = sns.light_palette("brown") sns.palplot(palette_seq\)</pre>	brown
	<pre>palette_seq\ = sns.light_palette("brown", reverse = True) sns.palplot(palette_seq\)</pre>	brown / reverse
	<pre>palette_seq\ = sns.light_palette("blue") sns.palplot(palette_seq\)</pre>	blue
	<pre>palette_seq\ = sns.light_palette("blue", reverse = True) sns.palplot(palette_seq\)</pre>	blue/reverse
	<pre>palette_dark = sns.color_palette("dark") sns.palplot(palette_dark)</pre>	purple
	<pre>palette_colorblind = sns.color_palette("colorblind") sns.palplot(palette_colorblind)</pre>	Purple/ reverse
<b>پالت‌های رنگی diverging</b>		
<b>خروجی</b>	<b>کد</b>	<b>نوع پالت</b>
	<pre>palette_div = sns.color_palette("coolwarm", 7) sns.palplot(palette_div)</pre>	coolwarm
	<pre>palette_div_custom = sns.diverging_palette(250, 250, n=7) sns.palplot(palette_div_custom)</pre>	با شماره

### تمرین #۱

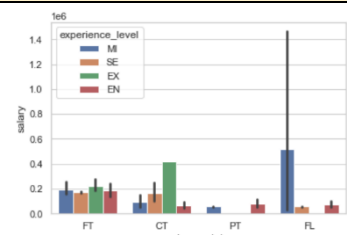
```
Data = pd.read_csv('flights.csv')
Data_Frame = Data.pivot_table(index='month', columns='year',
values='passengers')
sns.set()
plt.figure(dpi=100)
sns.heatmap(Data_Frame, cmap = sns.light_palette('orange',
as_cmap = True, reverse = True))
plt.title('Flight Passengers from ۲۰۰۱ , ۲۰۱۲')
plt.show()
```



### نمودارهای پایه در seaborn

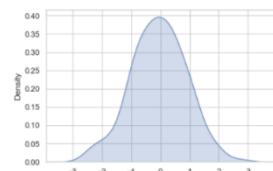
### تمرین #۱

```
Data = pd.read_csv('salaries.csv')
sns.set(style='whitegrid')
sns.barplot(x='employment_type', y='salary',
hue='experience_level', data=Data)
plt.show()
```



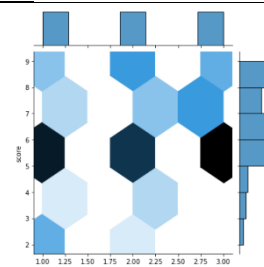
## تمرین ۲

```
X = np.random.randn(۲۰۰)
sns.kdeplot(X, shade=True)
plt.show()
```



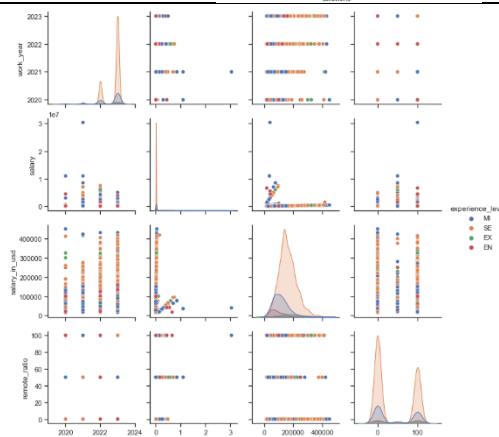
## تمرین ۳

```
data = pd.read_csv('attention.csv')
sns.jointplot(x="solutions", y="score",
              kind="hex", data=data)
plt.show()
```



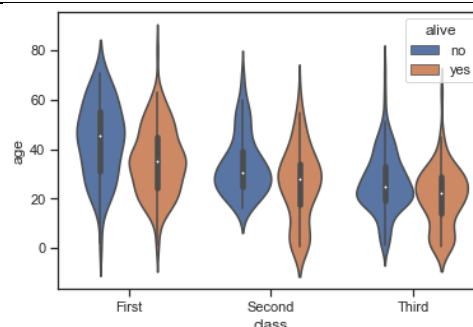
## تمرین ۴

```
Data = pd.read_csv('salaries.csv')
sns.set(style='ticks', color_codes=True)
g = sns.pairplot(Data, hue='experience_level')
plt.show()
```



## تمرین ۵

```
df = sns.load_dataset("titanic")
sns.violinplot(data=df, x="class", y="age", hue="alive")
plt.show()
```

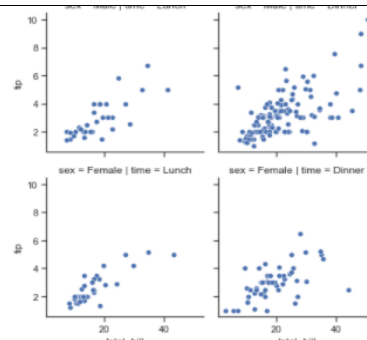


## ساخت نمودارهای چندگانه یا Multiplot در seaborn

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

## تمرین ۱

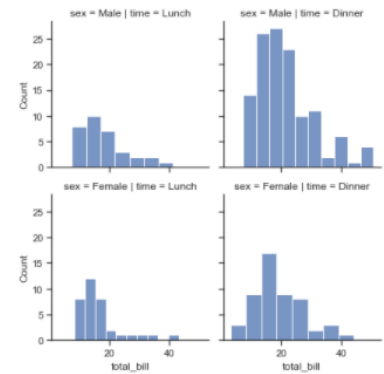
```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time", row="sex")
g.map(sns.scatterplot, "total_bill", "tip")
plt.show()
```





## تمرین ۲

```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time", row="sex")
g.map_dataframe(sns.histplot, x="total_bill")
plt.show()
```

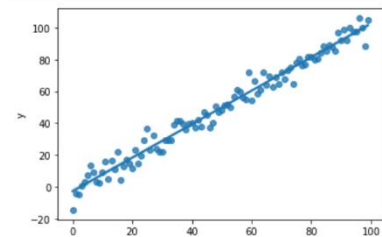


## Regression plot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

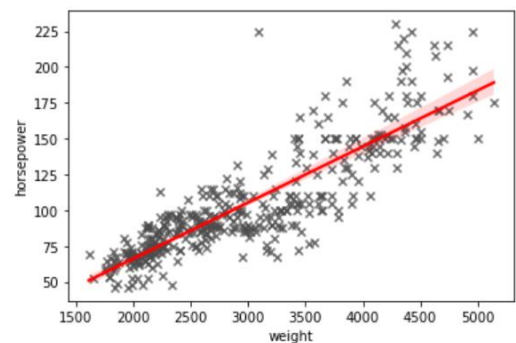
## تمرین ۱

```
x = np.arange(100)
y = x + np.random.normal(0, 5, size=100)
data = pd.DataFrame({'x': x, 'y': y})
sns.regplot(data=data, x='x', y='y')
plt.show()
```



## تمرین ۲

```
mpg = sns.load_dataset('mpg')
sns.regplot(data=mpg, x="weight", y="horsepower",
            ci=99, marker="x", color=".3",
            line_kws=dict(color="r"))
plt.show()
```



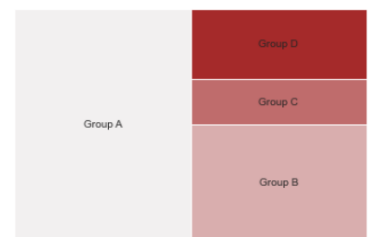
توضیحات	توابع و متدها
برای ایجاد نمودارهای Treemaps یا نمودارهای سلسله‌مراتبی مستطیل شکل. همیشه بهش اندازه، رنگ و برچسب داد و همینطور با pad فاصله بین مستطیلهای رو هم مشخص کرد.	squarify.plot()

### نمودارهای ترسیم شده در squarify

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

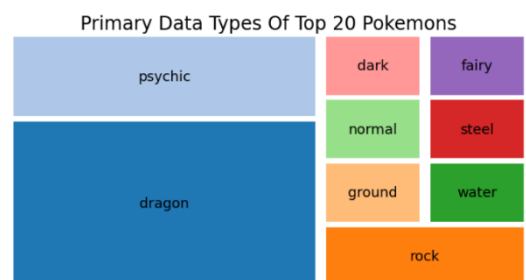
#### تمرین ۱

```
colors = sns.light_palette('brown', 4)
squarify.plot(sizes = [50, 25, 10, 15], label = ["Group A", "Group B",
        "Group C", "Group D"], color = colors)
plt.axis('off')
plt.show()
```



#### تمرین ۲

```
dataset = pd.read_csv("pokemon.csv")
df = pd.DataFrame(dataset)
top_20_pokemon = df.loc[:, ["name",
        "base_total", 'type']].sort_values(by="base_total",
        ascending=False)[:20]
plt.figure(figsize=(12, 6))
plt.axis("off")
axis = squarify.plot(top_20_pokemon['type'].value_counts(),
        label=top_20_pokemon['type'].value_counts().index,
        color=sns.color_palette("tab20", len(top_20_pokemon['type'].value_counts()))), pad=1,
        text_kwargs={'fontsize': 14})
axis.set_title("Primary Data Types Of Top 20 Pokemons", fontsize=24)
plt.show()
```



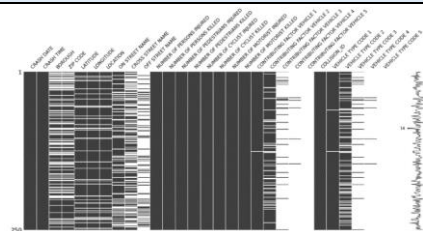
توضیحات	توابع و متدها
<p>تابخونه missingno ۴ تا نمودار داره:</p> <p>۱- <b>Bar chart</b>: تعداد مقادیر موجود در هر ستون رو با نادیده گرفتن مقادیر از دست رفته نمایش میده. همچنین درصدهایی رو در محور Y نشون میده که بهت این امکان رو میده که مقدار مقادیر مناسب/فقدان مقادیر در هر ستون رو درک کنی.</p> <p>۲- <b>Matrix</b>: نمودار ماتریس پوچ یا nullity بهت این امکان رو میده که توزیع داده‌ها رو در کل مجموعه داده در همه ستونها به طور همزمان درک کنی. علاوه بر اون در درک بهتر توزیع مقادیر از دست رفته در داده‌ها کمک کننده است. یه sparkline رو هم نمایش میده که ردیف‌هایی رو با حداکثر و حداقل بی اعتباری یا فقدان در یک مجموعه داده برجسته می‌کنه.</p> <p>۳- <b>Heatmap</b>: نمودار همبستگی بی اعتباری بین ستون‌های مجموعه داده رو نشون میده. بهت کمک میکنه که بفهمی که چطوری مقدار از دست رفته یک ستون با مقادیر از دست رفته در سایر ستونها مرتبط هست. sparkline بهت کمک میکنه تا بهتر بفهمی که در کجای مجموعه داده ما مقادیر زیادی از دست رفته وجود داره چون حرارت بالایی رو در اون بخش نشون میده.</p> <p>۴- <b>Dendrogram</b>: دندروگرام مثل نقشه حرارتی ستون‌ها رو بر اساس رابطه پوچ بین اونها گروه بندی میکنه. در واقع ستون‌هایی رو گروه‌بندی میکنه که در اونها رابطه nullity بیشتری وجود داره. تقریباً مثل خوشه‌بندی سلسله مراتبی عمل می‌کنه، ولی از همبستگی nullity برای خوشه‌بندی استفاده می‌کنه که ستون‌هایی رو با همون توزیع مقادیر گمشده در یک خوشه نگه می‌داره.</p>	<p>۱- msno.bar()</p> <p>۲- msno.matrix()</p> <p>۳- msno.heatmap()</p> <p>۴- msno.dendrogram()</p>

## نمودارهای ترسیم شده در missingno

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
%matplotlib inline
```

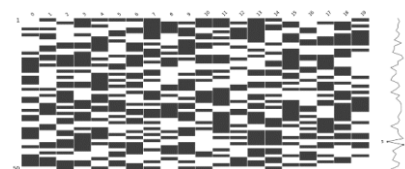
## تمرین ۱

```
collisions = pd.read_csv("Motor_Vehicle.csv")
msno.matrix(collisions.sample(۲۵۰))
plt.show()
```



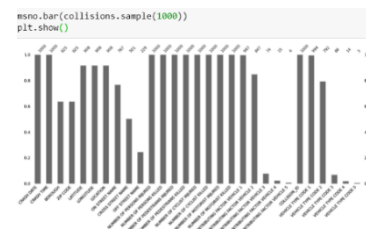
## تمرین ۲

```
null_pattern = (np.random.random(۱۰۰۰).reshape((۵۰, ۲۰)) >
۰,۵).astype(bool)
null_pattern = pd.DataFrame(null_pattern).replace({False: None})
msno.matrix(null_pattern.set_index(pd.period_range('۱/۱/۲۰۱۱',
'۲/۱/۲۰۱۵', freq='M'))))
plt.show()
```



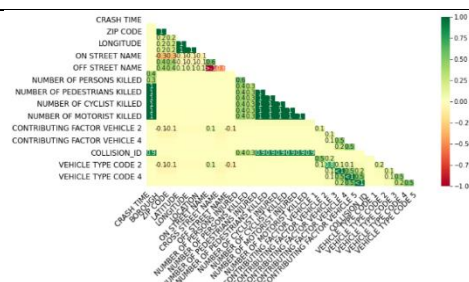
## تمرین ۳

```
msno.bar(collisions.sample(۱۰۰۰))
plt.show()
```



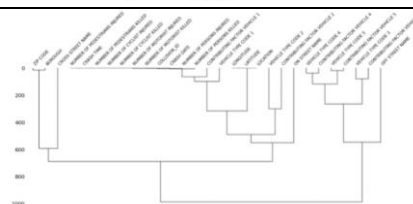
## تمرین ۴

```
msno.heatmap(collisions, cmap="RdYlGn", figsize=(۱۰,۵),
fontsize=۱۲)
plt.show()
```



## تمرین ۵

```
msno.dendrogram(collisions)
plt.show()
```



تو جزوه بعدی با هم به پروژه کاربردی انجام میدیم که دستورات این کتابخونه‌ها کامل برامون جا بیفته.