Pandas Template

In [1]:

```
import pandas as pd
```

In [2]:

```
# series: 1-dimentional use []#
series = pd.Series(["Cat","Dog","Donkey"])
hehavior=pd.Series(["Meow","Bark","Squeek"])
color=pd.Series(["Black","White","Grey"])
```

In [3]:

```
# dataframe: 2-dimentional use {}#
animal = pd.DataFrame({"Animal":series, "Behavior":hehavior})
animal.head()
```

Out[3]:

	Animal	Behavior
0	Cat	Meow
1	Dog	Bark
2	Donkey	Squeek

Import Data

In [4]:

```
#read data from static file#
carsales=pd.read_csv("car-sales-Copy1.csv")
carsales.to_csv ("car-sales.csv", index=False) # so the index column will not be a column in the exported file
```

In [5]:

```
#read data from GitHub, make sure GitHub in "raw" status#
heart_disease = pd.read_csv("https://raw.githubusercontent.com/mrdbourke/zero-to-mastery-ml/master/data/heart-disease.csv")
```

Describe Data

In [6]:

```
#if a function, use (); if an attribute, don't use ()#
carsales.dtypes
```

Out[6]:

```
Make                object
Colour              object
Odometer (KM)        int64
Doors                int64
Price               object
dtype: object
```

In [7]:

```
carsales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Make           10 non-null     object
 1   Colour         10 non-null     object
 2   Odometer (KM)  10 non-null     int64 
 3   Doors          10 non-null     int64 
 4   Price          10 non-null     object
dtypes: int64(2), object(3)
memory usage: 528.0+ bytes
```

In [8]:

```
#will show mean of numeric columns#
carsales.mean()
```

Out[8]:

```
Odometer (KM)    78601.4
Doors                4.0
dtype: float64
```

In [9]:

```
#will sum all columns#
carsales.sum()
```

Out[9]:

```
Make             ToyotaHondaToyotaBMWNissanToyotaHondaHondaToyo...
Colour               WhiteRedBlueBlackWhiteGreenBlueBlueWhiteWhite
Odometer (KM)                                               786014
Doors                                                           40
Price            $4,000.00$5,000.00$7,000.00$22,000.00$3,500.00...
dtype: object
```

In [10]:

```
#query just one column entire column date using []#
carsales["Odometer (KM)"].sum()
```

Out[10]:

```
786014
```

In [11]:

```
len(carsales)
```

Out[11]:

```
10
```

View and Select Data

In [12]:
```
#show top and bottom 5 only#
carsales.head()
carsales.tail()
```

Out[12]:

	Make	Colour	Odometer (KM)	Doors	Price
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [13]:
```
#.loc &.iloc
animal2 = pd.Series(["cat","dog","panda"],index=[0,3,5])
animal2
```

Out[13]:
```
0      cat
3      dog
5    panda
dtype: object
```

In [14]:
```
#.loc refers to index#
animal2.loc[5]
```

Out[14]:
```
'panda'
```

In [15]:
```
#.iloc refers to position 0,1,2#
animal2.iloc[2]
```

Out[15]:
```
'panda'
```

In [16]:
```
#.iloc slicing giving first 2 colums#
animal2.iloc[:2]
```

Out[16]:
```
0    cat
3    dog
dtype: object
```

In [17]:
```
#.loc slicing giving index up to 2#
animal2.loc[:2]
```

Out[17]:
```
0    cat
dtype: object
```

In [18]:
```
#select a column: two ways#
carsales["Make"]
carsales.Make
```

Out[18]:
```
0    Toyota
1     Honda
2    Toyota
3       BMW
4    Nissan
5    Toyota
6     Honda
7     Honda
8    Toyota
9    Nissan
Name: Make, dtype: object
```

In [19]:
```
#filter with boolean criteria#
carsales[carsales["Make"]=="Toyota"]
carsales[carsales["Odometer (KM)"]>100000]
```

Out[19]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
4	Nissan	White	213095	4	$3,500.00

In [20]:
```
#!!! easy way to make a 2-dimentional pivot!!!#
pd.crosstab(carsales["Make"],carsales["Doors"])
```

Out[20]:

Doors	3	4	5
Make			
BMW	0	0	1
Honda	0	3	0
Nissan	0	2	0
Toyota	1	3	0

In [21]:
```
#groupby function, returns mean of all the numerical fields#
carsales.groupby(["Make"]).mean()
```

Out[21]:

	Odometer (KM)	Doors
Make		
BMW	11179.000000	5.00
Honda	62778.333333	4.00
Nissan	122347.500000	4.00
Toyota	85451.250000	3.75

Visualization

In [22]:

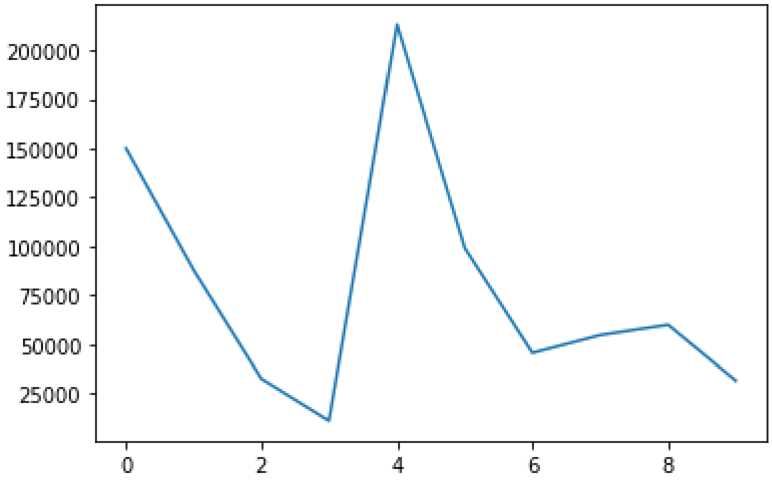
```
#pip install matplotlib#
import matplotlib
import matplotlib.pyplot as plt
```

In [23]:

```
carsales["Odometer (KM)"].plot()
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a370350>
```

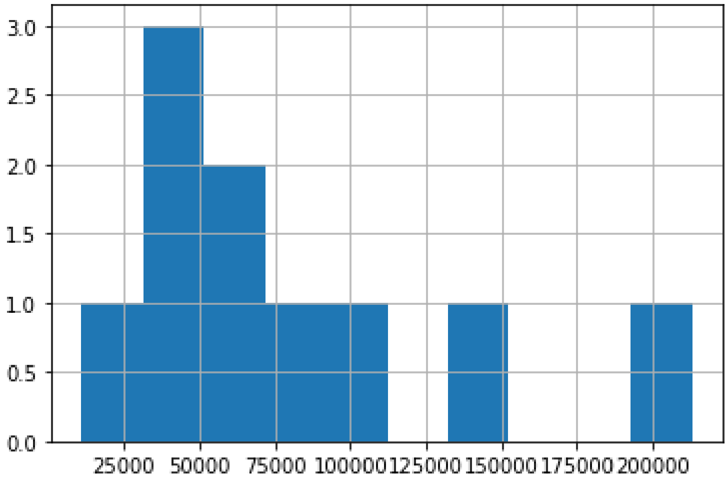


In [24]:

```
#run distribution#
carsales["Odometer (KM)"].hist()
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a6fc9d0>
```



In [25]:

```
# Price is a string column #
# Convert string to integer#
# Use str.replace for a regex, lost the cents#
carsales["Price"]=carsales["Price"].str.replace('[\$\,\.]','').astype(in
t)
```

In [26]:
```
carsales["Price"].dtype
```

Out[26]:
```
dtype('int64')
```

Manipulating Data

In [27]:
```
# This won't save to carsales#
carsales["Make"].str.lower()
```

Out[27]:
```
0    toyota
1     honda
2    toyota
3       bmw
4    nissan
5    toyota
6     honda
7     honda
8    toyota
9    nissan
Name: Make, dtype: object
```

In [28]:
```
# Save the format change
carsales["Make"] = carsales["Make"].str.lower()
carsales.head()
```

Out[28]:

	Make	Colour	Odometer (KM)	Doors	Price
0	toyota	White	150043	4	400000
1	honda	Red	87899	4	500000
2	toyota	Blue	32549	3	700000
3	bmw	Black	11179	5	2200000
4	nissan	White	213095	4	350000

In [29]:
```
carmissingdata = pd.read_csv("car-sales-missing-data.csv")
carmissingdata
```

Out[29]:

	Make	Colour	Odometer	Doors	Price
0	Toyota	White	150043.0	4.0	$4,000
1	Honda	Red	87899.0	4.0	$5,000
2	Toyota	Blue	NaN	3.0	$7,000
3	BMW	Black	11179.0	5.0	$22,000
4	Nissan	White	213095.0	4.0	$3,500
5	Toyota	Green	NaN	4.0	$4,500
6	Honda	NaN	NaN	4.0	$7,500
7	Honda	Blue	NaN	4.0	NaN
8	Toyota	White	60000.0	NaN	NaN
9	NaN	White	31600.0	4.0	$9,700

In [30]:
```
#fill N/A value with mean#
carmissingdata["Odometer"].fillna(carmissingdata["Odometer"].mean())
carmissingdata# see no change, have to use reasignment
```

Out[30]:

	Make	Colour	Odometer	Doors	Price
0	Toyota	White	150043.0	4.0	$4,000
1	Honda	Red	87899.0	4.0	$5,000
2	Toyota	Blue	NaN	3.0	$7,000
3	BMW	Black	11179.0	5.0	$22,000
4	Nissan	White	213095.0	4.0	$3,500
5	Toyota	Green	NaN	4.0	$4,500
6	Honda	NaN	NaN	4.0	$7,500
7	Honda	Blue	NaN	4.0	NaN
8	Toyota	White	60000.0	NaN	NaN
9	NaN	White	31600.0	4.0	$9,700

In [31]:
```
#use reasignment
carmissingdata["Odometer"]=carmissingdata["Odometer"].fillna(carmissingdata["Odometer"].mean())
```

In [32]:
```
# use inplace=True to avoid reassginment#
carmissingdata["Odometer"].fillna(carmissingdata["Odometer"].mean(),inplace=True)
```

In [33]:

```
# use dropna to remove the rows that has NaN#
carmissingdata.dropna()
```

Out[33]:

	Make	Colour	Odometer	Doors	Price
0	Toyota	White	150043.000000	4.0	$4,000
1	Honda	Red	87899.000000	4.0	$5,000
2	Toyota	Blue	92302.666667	3.0	$7,000
3	BMW	Black	11179.000000	5.0	$22,000
4	Nissan	White	213095.000000	4.0	$3,500
5	Toyota	Green	92302.666667	4.0	$4,500

Creating Data

In [34]:

```
#Add a Column from Series (not list)
#always append at the end
seats_column=pd.Series([5,5,5,5,5])
carsales["Seats"] = seats_column
carsales
```

Out[34]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats
0	toyota	White	150043	4	400000	5.0
1	honda	Red	87899	4	500000	5.0
2	toyota	Blue	32549	3	700000	5.0
3	bmw	Black	11179	5	2200000	5.0
4	nissan	White	213095	4	350000	5.0
5	toyota	Green	99213	4	450000	NaN
6	honda	Blue	45698	4	750000	NaN
7	honda	Blue	54738	4	700000	NaN
8	toyota	White	60000	4	625000	NaN
9	nissan	White	31600	4	970000	NaN

In [35]:

```
#fillna with an exact value#
carsales["Seats"].fillna(5,inplace=True)
carsales
```

Out[35]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats
0	toyota	White	150043	4	400000	5.0
1	honda	Red	87899	4	500000	5.0
2	toyota	Blue	32549	3	700000	5.0
3	bmw	Black	11179	5	2200000	5.0
4	nissan	White	213095	4	350000	5.0
5	toyota	Green	99213	4	450000	5.0
6	honda	Blue	45698	4	750000	5.0
7	honda	Blue	54738	4	700000	5.0
8	toyota	White	60000	4	625000	5.0
9	nissan	White	31600	4	970000	5.0

In [36]:

```
# Add a Column from List (not Series), need exact same amount of elements in the new list#
fuel_economy=[3.2,2.3,4.4,5.0,6.0,2.4,7.7,8.1,6.3,5.4] # need to fill all the spaces, otherwise, error
carsales["fuel_economy"] = fuel_economy
carsales
```

Out[36]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	fuel_economy
0	toyota	White	150043	4	400000	5.0	3.2
1	honda	Red	87899	4	500000	5.0	2.3
2	toyota	Blue	32549	3	700000	5.0	4.4
3	bmw	Black	11179	5	2200000	5.0	5.0
4	nissan	White	213095	4	350000	5.0	6.0
5	toyota	Green	99213	4	450000	5.0	2.4
6	honda	Blue	45698	4	750000	5.0	7.7
7	honda	Blue	54738	4	700000	5.0	8.1
8	toyota	White	60000	4	625000	5.0	6.3
9	nissan	White	31600	4	970000	5.0	5.4

In [37]:

```
# Create a column with other columns' operation#
carsales["total_fuel_used"]=carsales["Odometer (KM)"]/100*carsales["fuel_economy"]
carsales.head()
```

Out[37]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	fuel_economy	total_fuel_used
0	toyota	White	150043	4	400000	5.0	3.2	4801.376
1	honda	Red	87899	4	500000	5.0	2.3	2021.677
2	toyota	Blue	32549	3	700000	5.0	4.4	1432.156
3	bmw	Black	11179	5	2200000	5.0	5.0	558.950
4	nissan	White	213095	4	350000	5.0	6.0	12785.700

In [38]:

```
# Create a column with one single value
carsales["wheels"]=4  # can put "True" any type in here
carsales.head()
```

Out[38]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	fuel_economy	total_fuel_used	wheels
0	toyota	White	150043	4	400000	5.0	3.2	4801.376	4
1	honda	Red	87899	4	500000	5.0	2.3	2021.677	4
2	toyota	Blue	32549	3	700000	5.0	4.4	1432.156	4
3	bmw	Black	11179	5	2200000	5.0	5.0	558.950	4
4	nissan	White	213095	4	350000	5.0	6.0	12785.700	4

In [39]:

```
# Remove the column, if you are talking about a column, axis=1
carsales=carsales.drop("wheels",axis=1)
carsales.head()
```

Out[39]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	fuel_economy	total_fuel_used
0	toyota	White	150043	4	400000	5.0	3.2	4801.376
1	honda	Red	87899	4	500000	5.0	2.3	2021.677
2	toyota	Blue	32549	3	700000	5.0	4.4	1432.156
3	bmw	Black	11179	5	2200000	5.0	5.0	558.950
4	nissan	White	213095	4	350000	5.0	6.0	12785.700

Select Sample

In [40]:

```
# shuffle and select sample =50% of the data if 0.5, 100% of the data if
frac=1
carsales.sample(frac=0.5)
carsales_shuffle=carsales.sample(frac=1)
carsales_shuffle.head()
```

Out[40]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	fuel_economy	total_fuel_used
1	honda	Red	87899	4	500000	5.0	2.3	2021.677
2	toyota	Blue	32549	3	700000	5.0	4.4	1432.156
3	bmw	Black	11179	5	2200000	5.0	5.0	558.950
8	toyota	White	60000	4	625000	5.0	6.3	3780.000
6	honda	Blue	45698	4	750000	5.0	7.7	3518.746

In [41]:

```
# put it back to order "reset_index"
# that this, there would be an index column, to remove the index column,
use drop=True
carsales_shuffle.reset_index(drop=True,inplace=True)
carsales_shuffle.head()
```

Out[41]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	fuel_economy	total_fuel_used
0	honda	Red	87899	4	500000	5.0	2.3	2021.677
1	toyota	Blue	32549	3	700000	5.0	4.4	1432.156
2	bmw	Black	11179	5	2200000	5.0	5.0	558.950
3	toyota	White	60000	4	625000	5.0	6.3	3780.000
4	honda	Blue	45698	4	750000	5.0	7.7	3518.746

In [42]:

```
#change a column with apply and lambda
carsales["Odometer (KM)"]=carsales["Odometer (KM)"].apply(lambda x: x/1.
6)
carsales.head()
```

Out[42]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	fuel_economy	total_fuel_used
0	toyota	White	93776.875	4	400000	5.0	3.2	4801.376
1	honda	Red	54936.875	4	500000	5.0	2.3	2021.677
2	toyota	Blue	20343.125	3	700000	5.0	4.4	1432.156
3	bmw	Black	6986.875	5	2200000	5.0	5.0	558.950
4	nissan	White	133184.375	4	350000	5.0	6.0	12785.700

##