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

In [2]:
```
#2-Main Data Types

series=pd.Series(["BMW","Honda","Toyota"])
```

In [11]:
```
series
```

Out[11]:
```
0       BMW
1     Honda
2    Toyota
dtype: object
```

In [12]:
```
#series 1-Dimensional
```

In [13]:
```
colors=pd.Series(["Red","Blue","White"])
colors
```

Out[13]:
```
0      Red
1     Blue
2    White
dtype: object
```

In [14]:
```
#DataFrame = 2 Dimensional
car_data=pd.DataFrame({"Car_Make":series,"Colors":colors})
car_data
```

Out[14]:

	Car_Make	Colors
0	BMW	Red
1	Honda	Blue
2	Toyota	White

In [15]:
```
# Since typoing data is a very tedious job ,thus we import data
```

In [16]:
```
#Import data
car_sales=pd.read_csv("car-sales.csv")
car_sales
```

Out[16]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
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 [24]:
```
#Exporting a DataFrame
car_sales.to_csv("exported-car-sales.to_csv",index=False) #here index function is
# car_sales.to_excel  #for exporting data to excel file
```

In [25]:
```
exported_car_sales=pd.read_csv("exported-car-sales.to_csv")
exported_car_sales
```

Out[25]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
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

Describing Data

In [30]:
```
#Attribute
car_sales.dtypes

#Functions
# car_sales.to_csv()    #:is a function that gonna perform some kind of operation
```

Out[30]:
```
Make              object
Colour            object
Odometer (KM)      int64
Doors              int64
Price             object
dtype: object
```

In [33]:
```
car_sales.columns
```

Out[33]:
```
Index(['Make', 'Colour', 'Odometer (KM)', 'Doors', 'Price'], dtype='object')
```

In [35]:
```
#another way of writing the above
car_columns=car_sales.columns
car_columns
```

Out[35]:
```
Index(['Make', 'Colour', 'Odometer (KM)', 'Doors', 'Price'], dtype='object')
```

In [36]:
```
car_sales.index

#from below we can seee that index starts from [0,10)
```

Out[36]:
```
RangeIndex(start=0, stop=10, step=1)
```

In [37]:
```
car_sales
```

Out[37]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
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 [38]:
```
car_sales.describe()
# Gives statistical information about our numeric columns
#it only works on numeric columns ,thus price is not inlucded here (because it is c
```

Out[38]:

	Odometer (KM)	Doors
count	10.000000	10.000000
mean	78601.400000	4.000000
std	61983.471735	0.471405
min	11179.000000	3.000000
25%	35836.250000	4.000000
50%	57369.000000	4.000000
75%	96384.500000	4.000000
max	213095.000000	5.000000

In [39]:
```
car_sales.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 [42]:
```
car_sales.mean()
```

```
C:\Users\Kushagra Gupta\AppData\Local\Temp\ipykernel_16572\4073448239.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  car_sales.mean()
```

Out[42]:
```
Odometer (KM)    78601.4
Doors                4.0
dtype: float64
```

In [44]:
```
#you can applyt these function on individual series too
car_prices=pd.Series([3000,1500,111250])
car_prices.mean()
```

Out[44]:
```
38583.333333333336
```

In [45]:
```
car_sales.sum()
#if object- then combine whatever is written there
# If int- then sums all the columns
```

Out[45]:
```
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 [48]:
```
# we can do it for single column as well
car_sales["Doors"].sum()
```

Out[48]:
```
40
```

In [49]:
```
car_sales["Doors"].mean()
```

Out[49]:
```
4.0
```

In [52]:
```
len(car_sales)
```

Out[52]:
```
10
```

Viewing and Selecting Data

In [53]:
```
car_sales.head()
'''
shows top 5 data, but why 5?
suppose we had been delaing with thousands of data ,then it had not been convinient
Thus it gives us a quick snapshot in a little space of what our data frame contains
so you may look upon the first 5 rows if it fits in.
'''
```

Out[53]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00

In [56]:
```
# Suppopse first 5 aren;t enough for you , you may call first 7 too
car_sales.head(7)
```

Out[56]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00

In [57]:
```
car_sales.tail()
# Similar operation but for the end part
'''It shows the bottom 5 rows, this operation comes in handy when you are doing alt
```

Out[57]:

	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 [59]:
```
#.loc and iloc
animals=pd.Series(["cat","dog","bird","panda","snake"])
animals
```

Out[59]:
```
0      cat
1      dog
2     bird
3    panda
4    snake
dtype: object
```

In [60]:
```
animals=pd.Series(["cat","dog","bird","panda","snake"],index=[3,8,5,2,5])
animals
```

Out[60]:
```
3      cat
8      dog
5     bird
2    panda
5    snake
dtype: object
```

In [63]:
```
animals.loc[5]  #have a bit fifferent type of syntax
# loc stands for location/index
```

Out[63]:
```
5     bird
5    snake
dtype: object
```

In [64]:
```
animals.loc[8]
```

Out[64]:
```
'dog'
```

In [65]:
```
car_sales.loc[3]
```

Out[65]:
```
Make                  BMW
Colour              Black
Odometer (KM)       11179
Doors                   5
Price            $22,000.00
Name: 3, dtype: object
```

In [67]:
```
animals
```

Out[67]:
```
3      cat
8      dog
5     bird
2    panda
5    snake
dtype: object
```

In [68]:
```
#iloc
animals.iloc[3]
#iloc refers to position  i.e. panda is at 3rd position
```

Out[68]:
```
'panda'
```

In [69]:
```
# you may uise concept of slicing
animals.iloc[:3]
# Thus prints from [0,3) or [0,2]
```

Out[69]:
```
3     cat
8     dog
5    bird
dtype: object
```

In [72]:
```
car_sales.loc[:3]
#gives us data inlcuding index 3
#will be same as car_sales.head(4)
```

Out[72]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00

In [74]:
```
car_sales["Make"]
```

Out[74]:
```
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 [76]:
```
car_sales["Colour"]
```

Out[76]:
```
0    White
1      Red
2     Blue
3    Black
4    White
5    Green
6     Blue
7     Blue
8    White
9    White
Name: Colour, dtype: object
```

In [79]:
```
car_sales["Make"]
```

Out[79]:
```
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 [82]:
```
# the below syntax will also print the same thing
car_sales.Make
# The only difference between the two is the syntax

#NOTE: Remember,if you tried this with column name having space it wont work
```

Out[82]:
```
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 [83]:
```
#for exmaple
car_sales.Odometer (KM)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [83], in <cell line: 2>()
      1 #for exmaple
----> 2 car_sales.Odometer (KM)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:5575, in NDFrame.__getat
tr__(self, name)
   5568 if (
   5569     name not in self._internal_names_set
   5570     and name not in self._metadata
   5571     and name not in self._accessors
   5572     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5573 ):
   5574     return self[name]
-> 5575 return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'Odometer'
```

In [86]:
```
# But if you had done it with
car_sales["Odometer (KM)"]
#It would work
```

Out[86]:
```
0    150043
1     87899
2     32549
3     11179
4    213095
5     99213
6     45698
7     54738
8     60000
9     31600
Name: Odometer (KM), dtype: int64
```

In [87]:
```
car_sales[car_sales["Make"]=="Toyota"]
```

Out[87]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
2	Toyota	Blue	32549	3	$7,000.00
5	Toyota	Green	99213	4	$4,500.00
8	Toyota	White	60000	4	$6,250.00

In [90]:
```
car_sales[car_sales["Odometer (KM)"]>100000]
```

Out[90]:

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

In [91]:
```
pd.crosstab(car_sales["Make"],car_sales["Doors"])
# Thus aggregating two columns together
```

Out[91]:

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

In [92]:
```
# But what if we want to take in account of more columns at once

#groupby
car_sales.groupby(["Make"]).mean()
# It will give mean values of all the unique make values i.e. mean of all BMW cars
```

Out[92]:

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

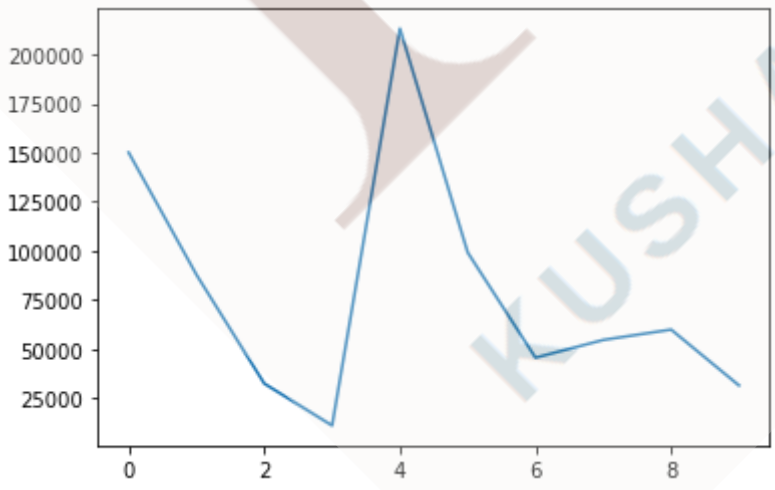
In [94]:
```
# In case the matplotlib library isn't installed then

'''
%matplotlib inline
import matplotlib.pyplot as plt
'''
#Write these lines of code to follow
#Here '%' sign is a magic function , In Jupyter notebook it means it does specific
#Thus all its saying is plot our map inside the notebook
```

Out[94]:
```
'\n%matplotlib inline\nimport matplotlib.pyplot as plt\n'
```

In [93]:
```
car_sales["Odometer (KM)"].plot()
```

Out[93]:
```
<AxesSubplot:>
```

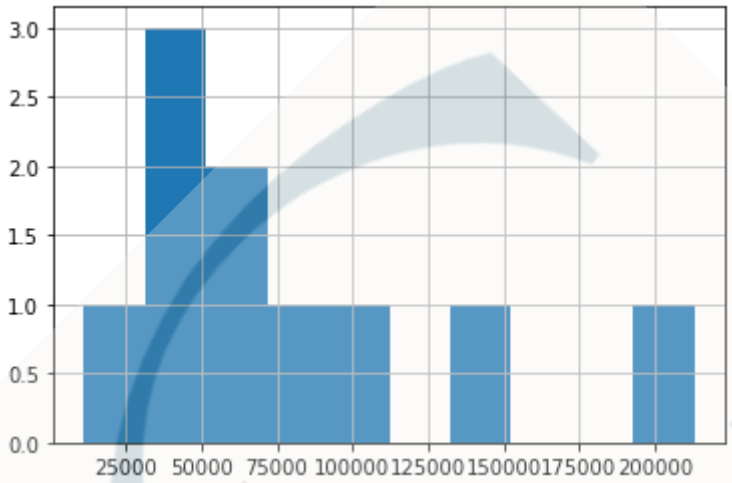


In [95]:
```
car_sales["Odometer (KM)"].hist()
```

```
# To print histogram - basically a fancy word of spread of data
```

Out[95]:

```
<AxesSubplot:>
```

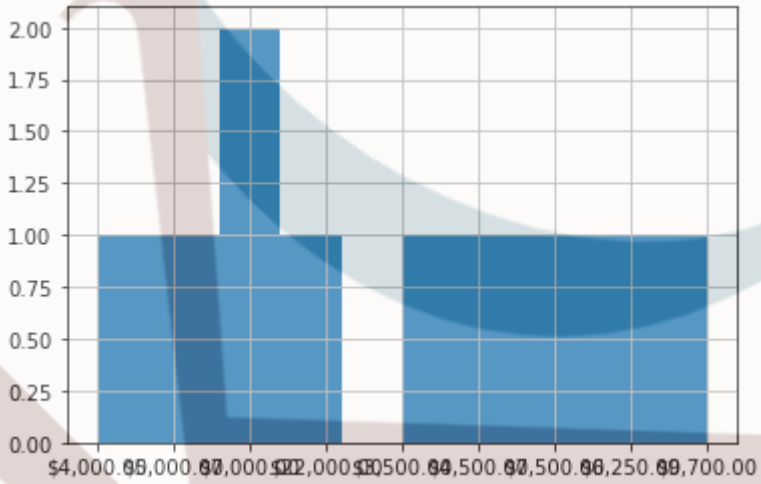


In [97]:

```
car_sales["Price"].hist()
```

Out[97]:

```
<AxesSubplot:>
```



In [102…

```
car_sales
```

Out[102]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000.0
1	Honda	Red	87899	4	5000.0
2	Toyota	Blue	32549	3	7000.0
3	BMW	Black	11179	5	22000.0
4	Nissan	White	213095	4	3500.0
5	Toyota	Green	99213	4	4500.0
6	Honda	Blue	45698	4	7500.0
7	Honda	Blue	54738	4	7000.0
8	Toyota	White	60000	4	6250.0
9	Nissan	White	31600	4	9700.0

In [101…

```
# since you remember price is object and not int, lets comvert object to int with t
```

```
# car_sales["Price"]=car_sales["Price"].str.replace('[\$\,\.]','').astype(int)
car_sales["Price"] = car_sales["Price"].replace('[\$\,]', '', regex=True).astype(fl
car_sales
```

Out[101]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000.0
1	Honda	Red	87899	4	5000.0
2	Toyota	Blue	32549	3	7000.0
3	BMW	Black	11179	5	22000.0
4	Nissan	White	213095	4	3500.0
5	Toyota	Green	99213	4	4500.0
6	Honda	Blue	45698	4	7500.0
7	Honda	Blue	54738	4	7000.0
8	Toyota	White	60000	4	6250.0
9	Nissan	White	31600	4	9700.0

Manipulating Data

In [3]:
```
car_sales["Make"].str.lower()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [3], in <cell line: 1>()
----> 1 car_sales["Make"].str.lower()

NameError: name 'car_sales' is not defined
```

In [105…
```
car_sales
```

Out[105]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000.0
1	Honda	Red	87899	4	5000.0
2	Toyota	Blue	32549	3	7000.0
3	BMW	Black	11179	5	22000.0
4	Nissan	White	213095	4	3500.0
5	Toyota	Green	99213	4	4500.0
6	Honda	Blue	45698	4	7500.0
7	Honda	Blue	54738	4	7000.0
8	Toyota	White	60000	4	6250.0
9	Nissan	White	31600	4	9700.0

In [2]:
```
# #Here you can check that it is not updtaed, thus to save your changes to columns
# car_sales["Make"]=car_sales["Make"].str.lower()
```