Dealing with missing values

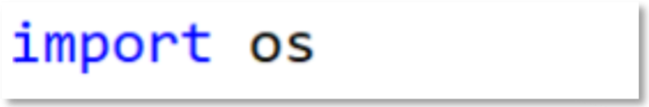
GITAA
Transforming careers

In this lecture

- Identifying missing values
- Approaches to fill the missing values

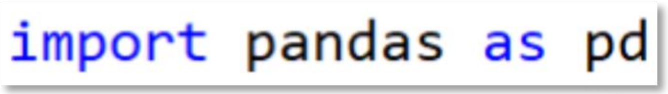
Importing data into Spyder

- Importing necessary libraries



```python
import os
```

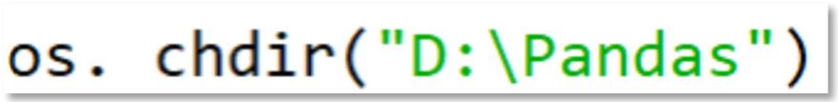
‘os’ library to change the working directory



```python
import pandas as pd
```

‘pandas’ library to work with dataframes

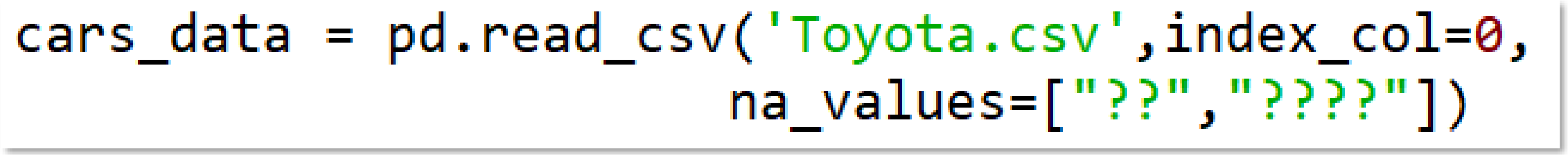
- Changing the working directory



```python
os. chdir("D:\Pandas")
```

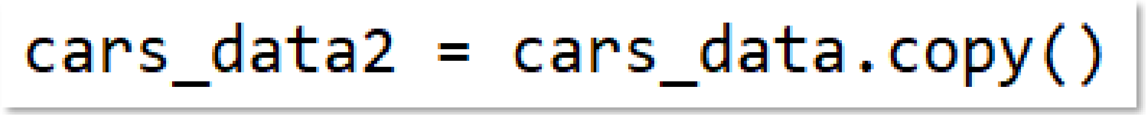
Importing data into Spyder

- Importing data

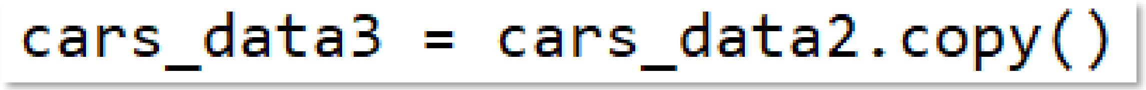


```
cars_data = pd.read_csv('Toyota.csv',index_col=0,
                        na_values=["??","????"])
```

- Creating copies of original data



```
cars_data2 = cars_data.copy()
```



```
cars_data3 = cars_data2.copy()
```

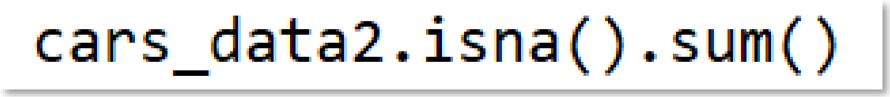
Identifying missing values

- In Pandas dataframes, missing data is represented by NaN (an acronym for Not a Number)
- To check null values in Pandas dataframes, `isnull()` and `isna()` are used
- These functions returns a dataframe of Boolean values which are True for NaN values

Identifying missing values

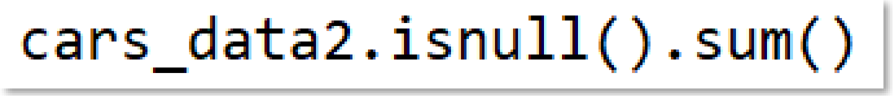
Dataframe.isna.sum(), Dataframe.isnull.sum()

- To check the count of missing values present in each column

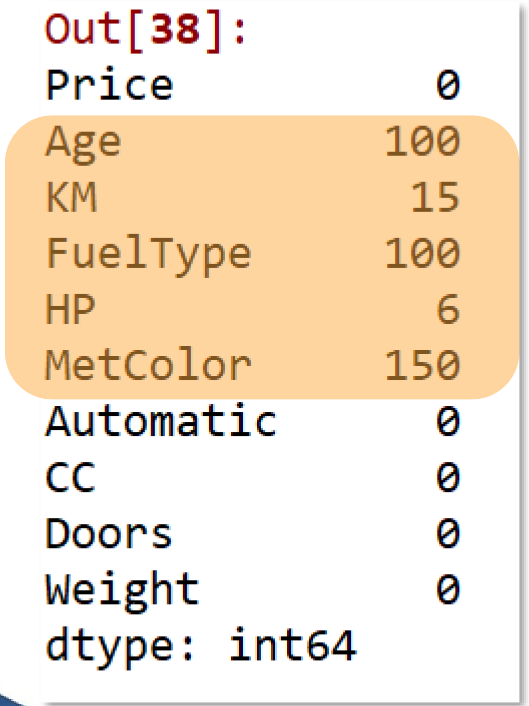


```
cars_data2.isna().sum()
```

(or)



```
cars_data2.isnull().sum()
```

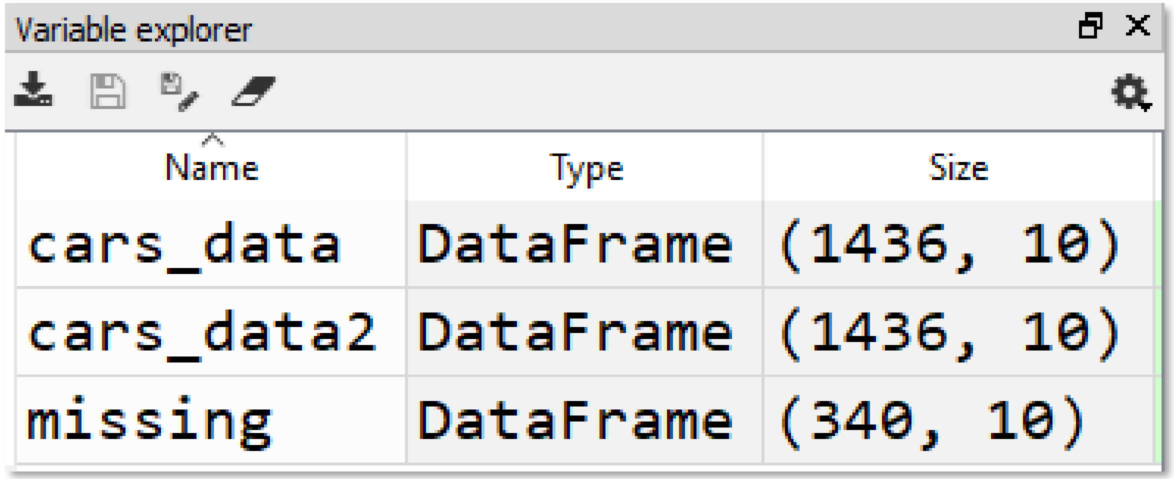


```
Out[38]:
Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

Identifying missing values

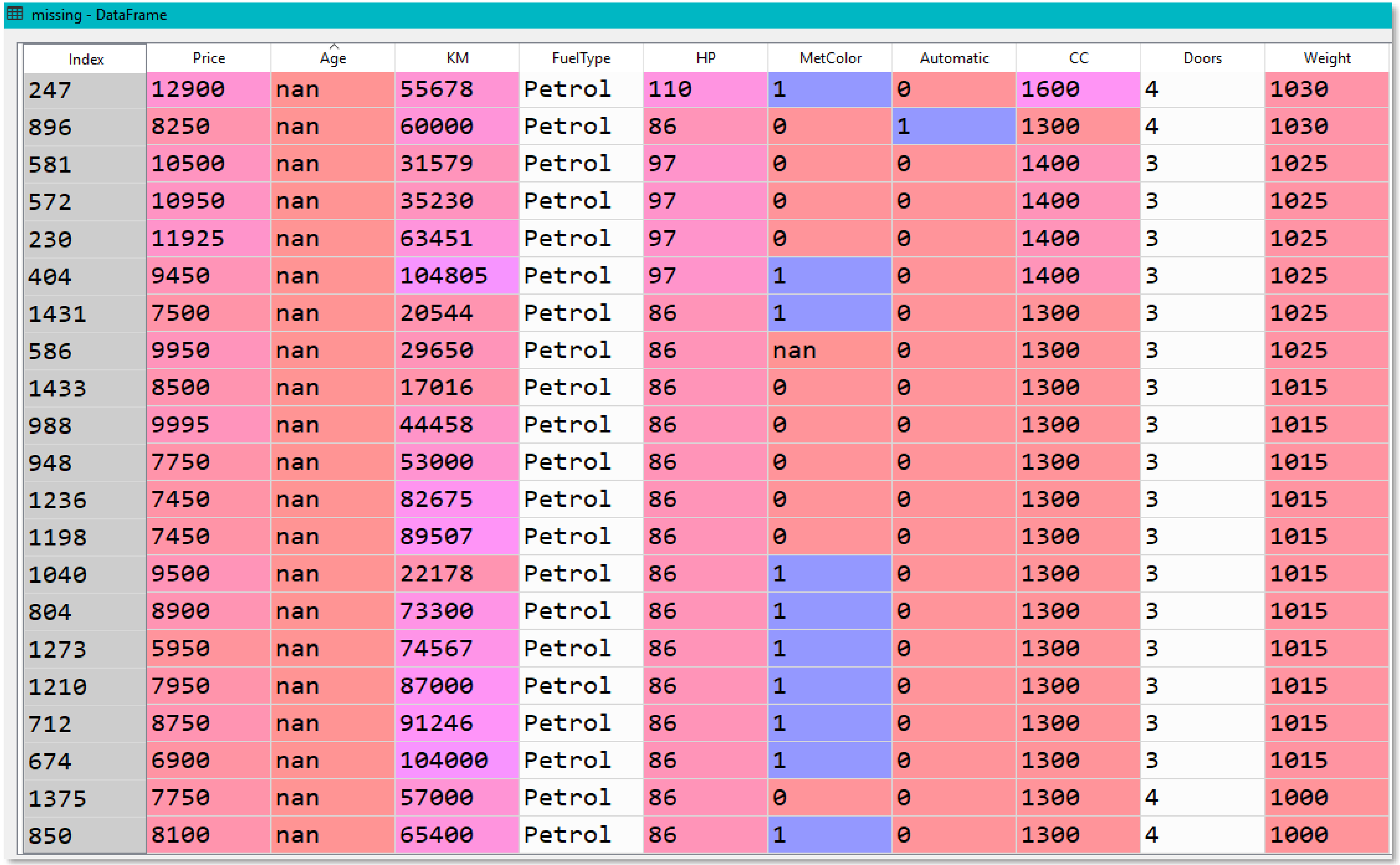
- Subsetting the rows that have one or more missing values

```
missing = cars_data2[cars_data2.isnull().any(axis=1)]
```



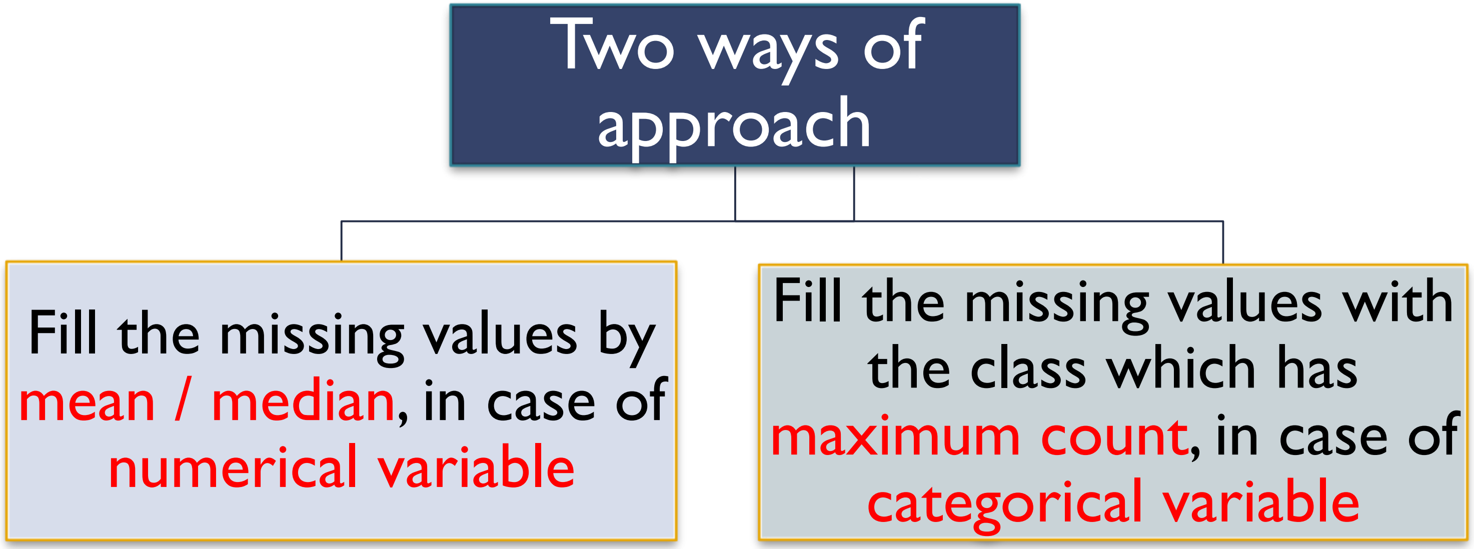
Name	Type	Size
cars_data	DataFrame	(1436, 10)
cars_data2	DataFrame	(1436, 10)
missing	DataFrame	(340, 10)

Identifying missing values



Index	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
247	12900	nan	55678	Petrol	110	1	0	1600	4	1030
896	8250	nan	60000	Petrol	86	0	1	1300	4	1030
581	10500	nan	31579	Petrol	97	0	0	1400	3	1025
572	10950	nan	35230	Petrol	97	0	0	1400	3	1025
230	11925	nan	63451	Petrol	97	0	0	1400	3	1025
404	9450	nan	104805	Petrol	97	1	0	1400	3	1025
1431	7500	nan	20544	Petrol	86	1	0	1300	3	1025
586	9950	nan	29650	Petrol	86	nan	0	1300	3	1025
1433	8500	nan	17016	Petrol	86	0	0	1300	3	1015
988	9995	nan	44458	Petrol	86	0	0	1300	3	1015
948	7750	nan	53000	Petrol	86	0	0	1300	3	1015
1236	7450	nan	82675	Petrol	86	0	0	1300	3	1015
1198	7450	nan	89507	Petrol	86	0	0	1300	3	1015
1040	9500	nan	22178	Petrol	86	1	0	1300	3	1015
804	8900	nan	73300	Petrol	86	1	0	1300	3	1015
1273	5950	nan	74567	Petrol	86	1	0	1300	3	1015
1210	7950	nan	87000	Petrol	86	1	0	1300	3	1015
712	8750	nan	91246	Petrol	86	1	0	1300	3	1015
674	6900	nan	104000	Petrol	86	1	0	1300	3	1015
1375	7750	nan	57000	Petrol	86	0	0	1300	4	1000
850	8100	nan	65400	Petrol	86	1	0	1300	4	1000

Approached to fill the missing values



Two ways of approach

Fill the missing values by mean / median, in case of numerical variable

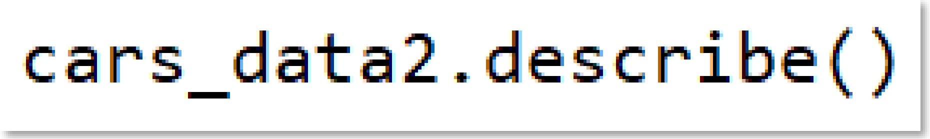
Fill the missing values with the class which has maximum count, in case of categorical variable

Imputing missing values

- Look at the description to know whether numerical variables should be imputed with mean or median

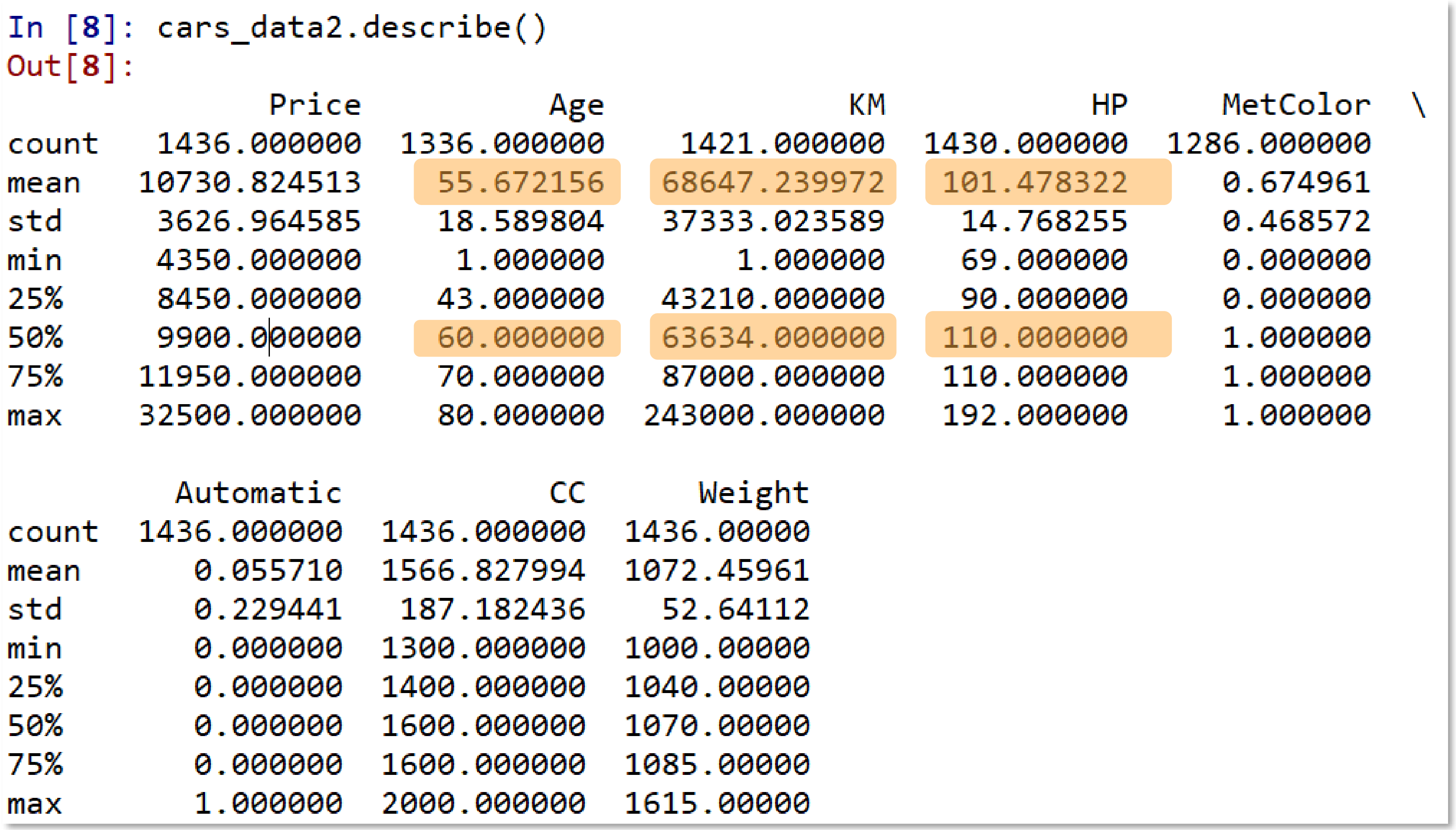
```
DataFrame.describe()
```

- Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset’s distribution, excluding NaN values



```
cars_data2.describe()
```

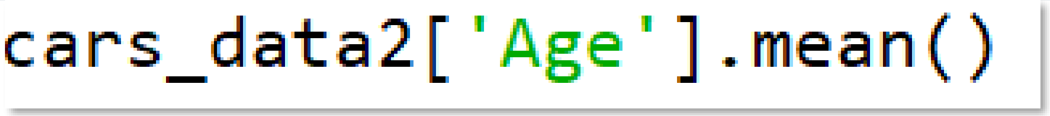
Statistical summary of data



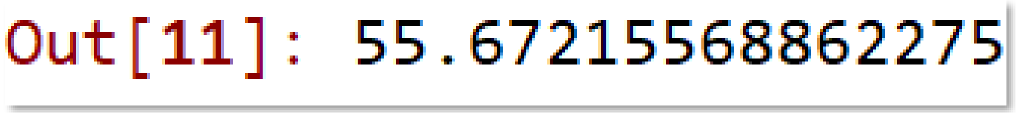
```
In [8]: cars_data2.describe()
Out[8]:
              Price          Age             KM           HP     MetColor  \
count   1436.000000  1336.000000    1421.000000  1430.000000  1286.000000
mean   10730.824513    55.672156   68647.239972   101.478322     0.674961
std     3626.964585    18.589804   37333.023589    14.768255     0.468572
min     4350.000000     1.000000       1.000000    69.000000     0.000000
25%     8450.000000    43.000000   43210.000000    90.000000     0.000000
50%     9900.000000    60.000000   63634.000000   110.000000     1.000000
75%    11950.000000    70.000000   87000.000000   110.000000     1.000000
max    32500.000000    80.000000  243000.000000   192.000000     1.000000

         Automatic           CC      Weight
count  1436.000000  1436.000000  1436.00000
mean      0.055710  1566.827994  1072.45961
std       0.229441   187.182436    52.64112
min       0.000000  1300.000000  1000.00000
25%       0.000000  1400.000000  1040.00000
50%       0.000000  1600.000000  1070.00000
75%       0.000000  1600.000000  1085.00000
max       1.000000  2000.000000  1615.00000
```

Imputing missing values of 'Age'

- Calculating the mean value of the ***Age*** variable



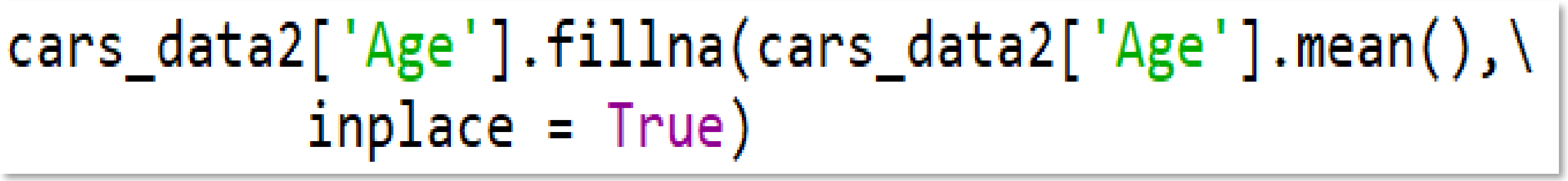
```
cars_data2['Age'].mean()
```



```
Out[11]: 55.67215568862275
```

- To fill NA/NaN values using the specified value

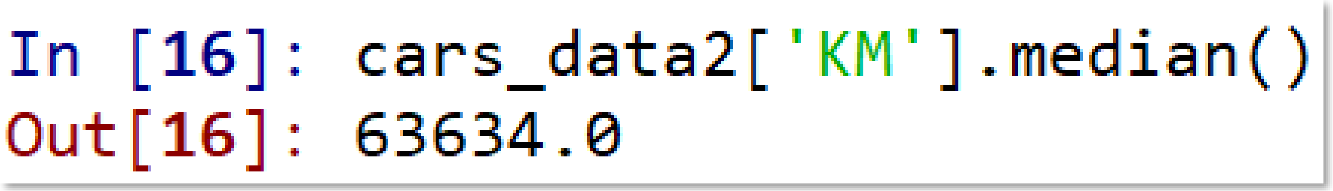
DataFrame.fillna()



```
cars_data2['Age'].fillna(cars_data2['Age'].mean(),\
                          inplace = True)
```

Imputing missing values of ‘KM’

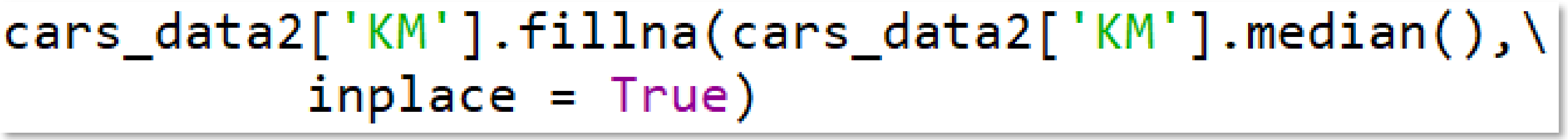
- Calculating the median value of the ***KM*** variable



```
In [16]: cars_data2['KM'].median()
Out[16]: 63634.0
```

- To fill NA/NaN values using the specified value

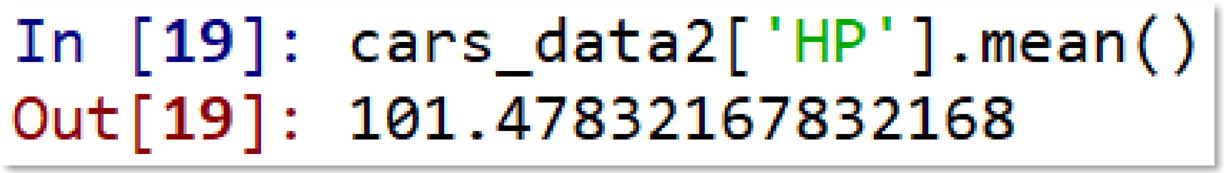
DataFrame.fillna()



```
cars_data2['KM'].fillna(cars_data2['KM'].median(),\
         inplace = True)
```

Imputing missing values of ‘HP’

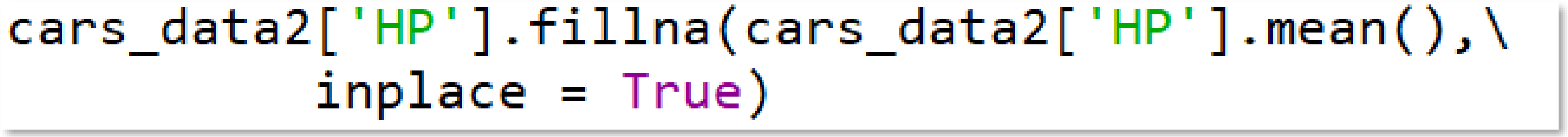
- Calculating the mean value of the ***HP*** variable



```
In [19]: cars_data2['HP'].mean()
Out[19]: 101.47832167832168
```

- To fill NA/NaN values using the specified value

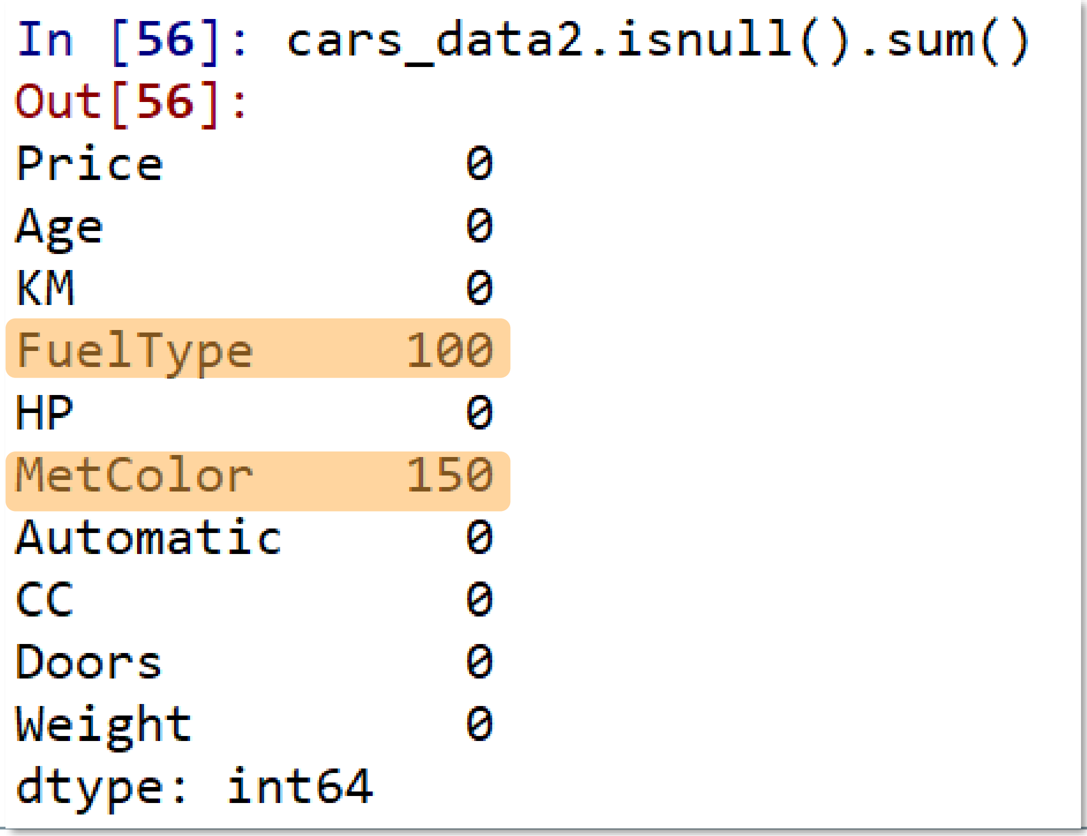
DataFrame.fillna()



```
cars_data2['HP'].fillna(cars_data2['HP'].mean(),\
          inplace = True)
```

Imputing missing values of ‘HP’

- Check for missing data after filling values

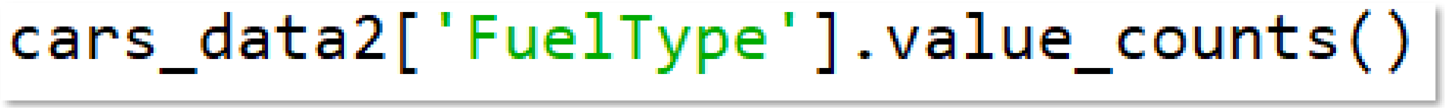


```
In [56]: cars_data2.isnull().sum()
Out[56]:
Price          0
Age            0
KM             0
FuelType     100
HP             0
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

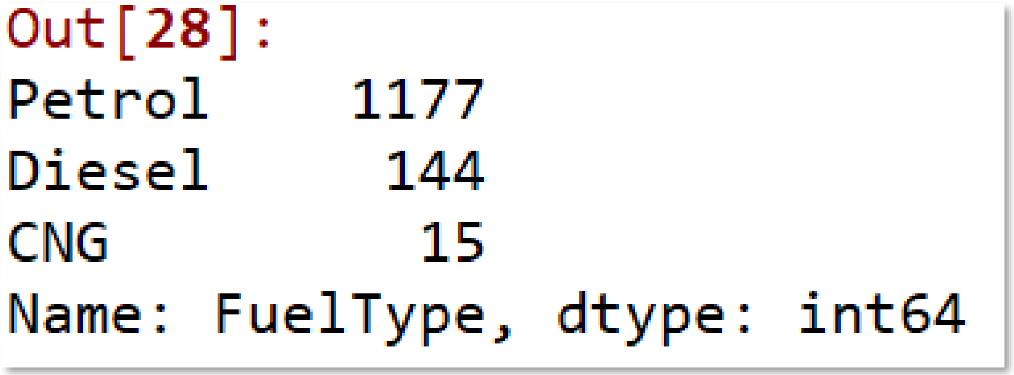
Imputing missing values of ‘FuelType’

`Series.value_counts()`

- Returns a Series containing counts of unique values
- The values will be in descending order so that the first element is the most frequently-occurring element
- Excludes NA values by default



```
cars_data2['FuelType'].value_counts()
```

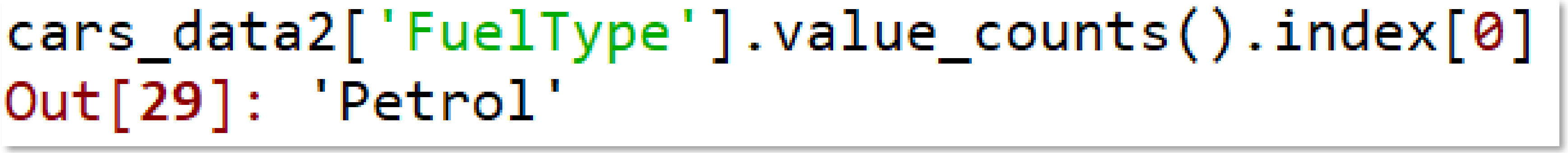


```
Out[28]:
Petrol    1177
Diesel     144
CNG         15
Name: FuelType, dtype: int64
```

Imputing missing values of ‘FuelType’

`Series.value_counts()`

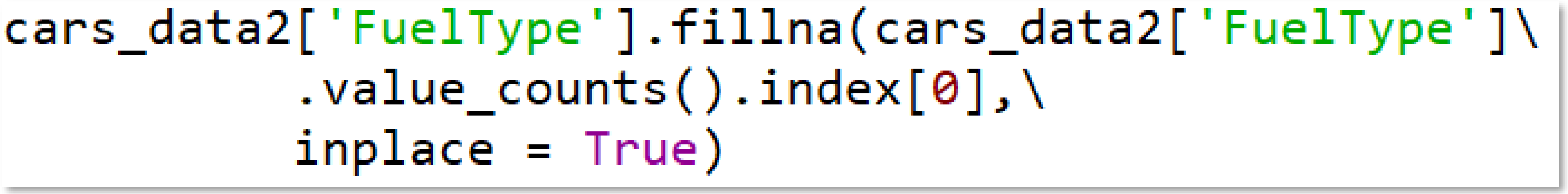
- To get the mode value of ***FuelType***



```
cars_data2['FuelType'].value_counts().index[0]
Out[29]: 'Petrol'
```

- To fill NA/NaN values using the specified value

`DataFrame.fillna()`

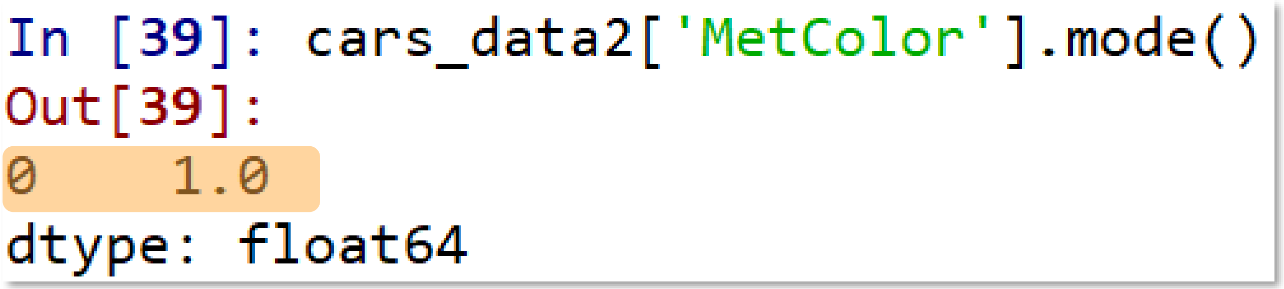


```
cars_data2['FuelType'].fillna(cars_data2['FuelType']\
          .value_counts().index[0],\
          inplace = True)
```

Imputing missing values of ‘MetColor’

Series.value_counts()

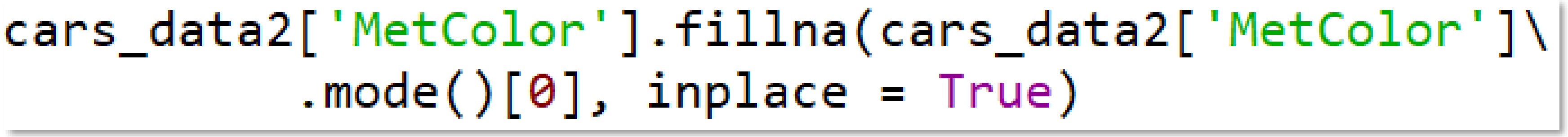
- To get the mode value of ***MetColor***



```
In [39]: cars_data2['MetColor'].mode()
Out[39]:
0     1.0
dtype: float64
```

- To fill NA/NaN values using the specified value

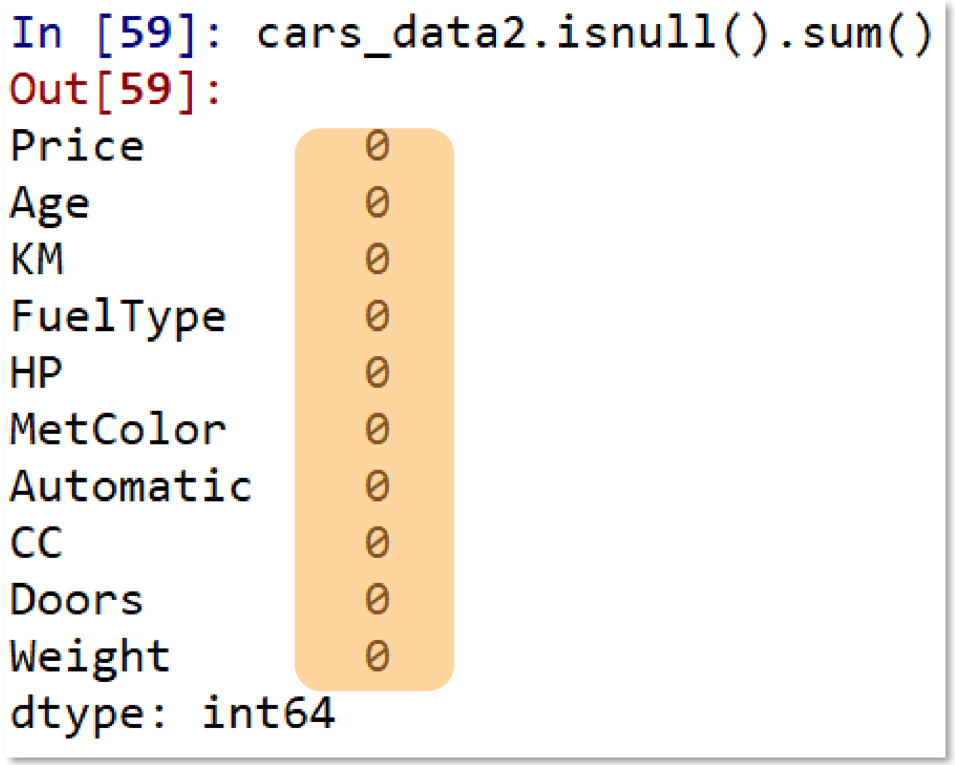
DataFrame.fillna()



```
cars_data2['MetColor'].fillna(cars_data2['MetColor']\
                .mode()[0], inplace = True)
```

Checking for missing values

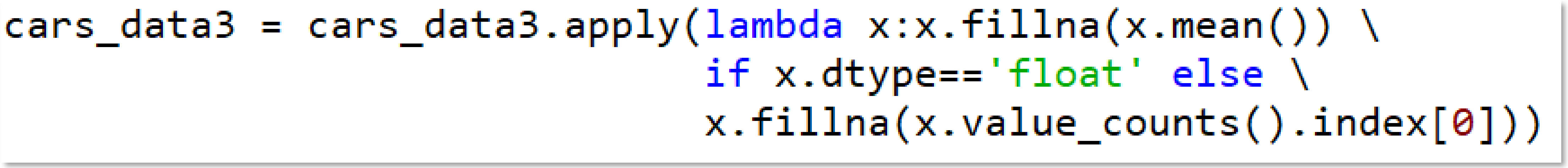
- Check for missing data after filling values



```
In [59]: cars_data2.isnull().sum()
Out[59]:
Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

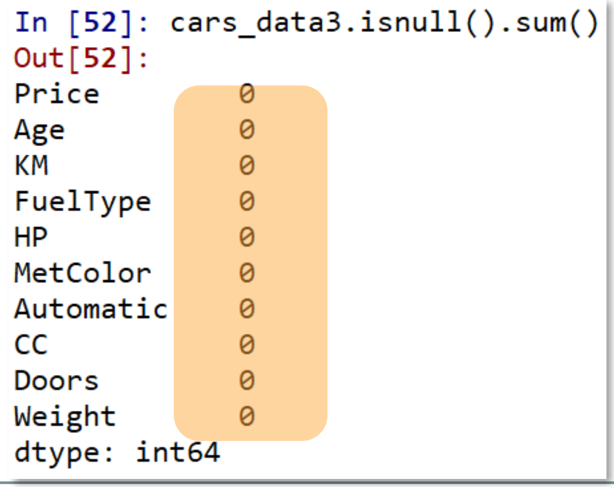
Imputing missing values using lambda functions

- To fill the NA/ NaN values in both numerical and categorial variables at one stretch



```
cars_data3 = cars_data3.apply(lambda x:x.fillna(x.mean()) \
                              if x.dtype=='float' else \
                              x.fillna(x.value_counts().index[0]))
```

- Check for missing data after filling values



```
In [52]: cars_data3.isnull().sum()
Out[52]:
Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

Summary

- Identifying missing values
- Approaches to fill the missing values

```
rror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
 _operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
 _operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

selection at the end -add
_ob.select= 1
ier_ob.select=1
ntext.scene.objects.active
("Selected" + str(modifier
irror_ob.select = 0
 = bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactl
```

THANK YOU