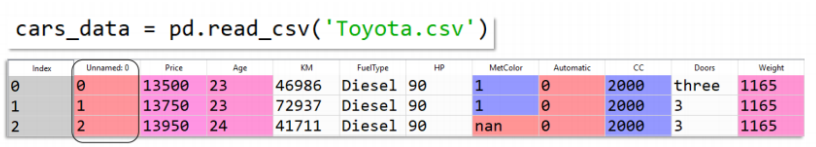
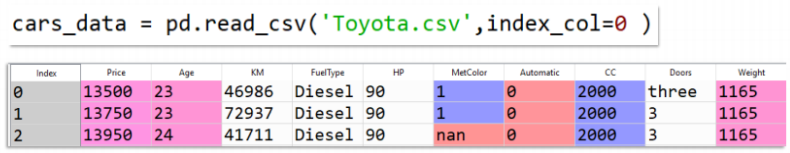
Pandas

Introduction:

- Name pandas is derived from the word Panel Data – an econometrics term for multidimensional data
- It is an Open-source Python library that provides high performance data manipulation and analysis using its powerful data structures
- It has its own s high-performance, easy-to-use data structures and analysis tools for the Python programming language
- It deals with 2D data called as *Data Frame*
- Usually, it requires following import files:
  import pandas as pd
  import numpy as np
  import os
- We need to set path of directory where input files are kept using:
  os.chdir('D:/Pandas')

Sr No	Method	Example and Output	Explanation
1.	Dataframe=pd.read_csv('file_name.csv')	a) Importing Data  b) Importing Data by setting first column as index column 	Imports the data of .csv file into dataframe

```
cars_data = pd.read_csv('Toyota.csv')
```

Index	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23	46986	Diesel	90	1	0	2000	three	1165
1	1	13750	23	72937	Diesel	90	1	0	2000	3	1165
2	2	13950	24	41711	Diesel	90	nan	0	2000	3	1165

```
cars_data = pd.read_csv('Toyota.csv',index_col=0 )
```

Index	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23	46986	Diesel	90	1	0	2000	three	1165
1	13750	23	72937	Diesel	90	1	0	2000	3	1165
2	13950	24	41711	Diesel	90	nan	0	2000	3	1165

2.	DataFrame.index	In [14]: ``` import os import pandas as pd import numpy as np os.chdir("Z:/Pandas") car=pd.read_csv('Toyota.csv',index_col=0) ``` In [13]: ``` car.index ``` Out[13]: ``` Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ... 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435], dtype='int64', length=1436) ```	To get the index (row labels) of the dataframe
3.	Dataframe.columns	In [12]: ``` car.columns ``` Out[12]: ``` Index(['Price', 'Age', 'KM', 'FuelType', 'HP', 'MetColor', 'Automatic', 'CC', 'Doors', 'Weight'], dtype='object') ```	To get all the column labels
4.	Dataframe.size	In [15]: ``` car.size ``` Out[15]: ``` 14360 ```	To get the total number of elements from the data frame
5.	Dataframe.shape	In [16]: ``` car.shape ``` Out[16]: ``` (1436, 10) ```	To get the shape of the data frame
6.	Dataframe.memory_usage	``` car.memory_usage() ``` Out[17]: ``` Index 11488 Price 11488 Age 11488 KM 11488 FuelType 11488 HP 11488 MetColor 11488 Automatic 11488 CC 11488 Doors 11488 Weight 11488 dtype: int64 ```	T get Memory occupied by each column in bytes
7.	Dataframe.ndim	In [18]: ``` car.ndim ``` Out[18]: ``` 2 ```	To get the Dimension of data frame

8.	Dataframe.head([n])	`car.head(5)` Out[19]: Price Age KM FuelType HP MetColor Automatic CC Doors Weight 0 13500 23.0 46986 Diesel 90 1.0 0 2000 three 1165 1 13750 23.0 72937 Diesel 90 1.0 0 2000 3 1165 2 13950 24.0 41711 Diesel 90 NaN 0 2000 3 1165 3 14950 26.0 48000 Diesel 90 0.0 0 2000 3 1165 4 13750 30.0 38500 Diesel 90 0.0 0 2000 3 1170	The function head returns the first n rows from the dataframe
9.	Dataframe.tail(n)	`car.tail(5)` Out[20]: Price Age KM FuelType HP MetColor Automatic CC Doors Weight 1431 7500 NaN 20544 Petrol 86 1.0 0 1300 3 1025 1432 10845 72.0 ?? Petrol 86 0.0 0 1300 3 1015 1433 8500 NaN 17016 Petrol 86 0.0 0 1300 3 1015 1434 7250 70.0 ?? NaN 86 1.0 0 1300 3 1015 1435 6950 76.0 1 Petrol 110 0.0 0 1600 5 1114	Returns Last n Rows
10	Dataframe.at[index, Label]	`car.at[4,'KM']` Out[22]: '38500'	Label based Scalar Lookup
11.	Dataframe.iat[index, index]	`car.iat[4,9]` Out[24]: 1170	Index based Lookup
12.	Dataframe.loc[Row : Row, Col : Col]	a) `car.loc[:,'FuelType']` Out[26]: 0 Diesel 1 Diesel 2 Diesel 3 Diesel 4 Diesel ... 1431 Petrol 1432 Petrol 1433 Petrol 1434 NaN 1435 Petrol Name: FuelType, Length: 1436, dtype: object	Accessing Group of rows and Columns by Labels

b)

```
car.loc[0:10,'FuelType']
```

```
Out[27]: 0     Diesel
         1     Diesel
         2     Diesel
         3     Diesel
         4     Diesel
         5     Diesel
         6     Diesel
         7        NaN
         8     Petrol
         9     Diesel
         10    Petrol
         Name: FuelType, dtype: object
```

c)

```
In [28]: #Accessing all the elements of row 1

         car.loc[1,:]
```

```
Out[28]: Price         13750
         Age              23
         KM            72937
         FuelType     Diesel
         HP               90
         MetColor          1
         Automatic         0
         CC             2000
         Doors             3
         Weight         1165
         Name: 1, dtype: object
```

d)

```
In [34]: #row 0 column Price to HP

         car.loc[0,'Price':'HP']
```

```
Out[34]: Price        13500
         Age             23
         KM           46986
         FuelType    Diesel
         HP              90
         Name: 0, dtype: object
```

Pandas Data Types and Data Cleansing:

- The way information gets stored in a dataframe or a python object affects the analysis and outputs of calculations
- There are two main types of data
    - numeric

Python data type	Pandas data type	Description
int	int64	Numeric characters
float	float64	Numeric characters with decimals

    - character types
        - category & object

category	object
A string variable consisting of only a few different values. Converting such a string variable to a categorical variable will save some memory	The column will be assigned as object data type when it has mixed types (numbers and strings). If a column contains 'nan'(blank cells), pandas will default to object datatype.
A categorical variable takes on a limited, fixed number of possible values	For strings, the length is not fixed

- Numeric data types includes integers and floats
    - For example: integer – 10, float – 10.53
- Strings are known as objects in pandas which can store values that contain numbers and / or characters
    - For example:'category1'

Learning Outcome of Point 13-17:

- Data types ◦ Numeric ◦ Character
- Checked data types of each column
- Count of unique data types
- Selected data based on data types
- Concise summary of dataframe
- Checked format of each column
- Got unique elements of each column

Sr No	Method	Example and Output	Explanation
13.	DataFrame.dtypes	`car.dtypes` Out[4]: Price int64 Age float64 KM object FuelType object HP object MetColor float64 Automatic int64 CC int64 Doors object Weight int64 dtype: object	dtypes returns a series with the data type of each column

14.	DataFrame.get_dtype_counts()	In [15]:
		`#3) count of unique data types`
		`car.dtypes.value_counts()`
		
		Out[15]:
		object 4
		int64 4
		float64 2
		dtype: int64
			It returns counts of unique data types in the dataframe

15.	DataFrame.select_dtypes(include=None, exclude=None)	In [17]:
		`#4) selecting data based on data type`
		`car.select_dtypes(exclude=[object])`
		
		Out[17]:
		Price Age MetColor Automatic CC Weight
		0 13500 23.0 1.0 0 2000 1165
		1 13750 23.0 1.0 0 2000 1165
		2 13950 24.0 NaN 0 2000 1165
		3 14950 26.0 0.0 0 2000 1165
		4 13750 30.0 0.0 0 2000 1170
		... ...
		1431 7500 NaN 1.0 0 1300 1025
		1432 10845 72.0 0.0 0 1300 1015
		1433 8500 NaN 0.0 0 1300 1015
		1434 7250 70.0 1.0 0 1300 1015
		1435 6950 76.0 0.0 0 1600 1114
		
		1436 rows × 6 columns
			It returns a subset of the columns from dataframe based on the column dtypes

<table>
<tr>
<td>16.</td>
<td>info()</td>
<td><pre>
In [18]: #5) Concise Summary of dataframe and 6) Checking Format of Each Column

         car.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column   Non-Null Count Dtype
--- ------   -------------- -----
 0   Price   1436 non-null   int64
 1   Age   1336 non-null   float64
 2   KM   1436 non-null   object
 3   FuelType   1336 non-null   object
 4   HP   1436 non-null   object
 5   MetColor   1286 non-null   float64
 6   Automatic 1436 non-null   int64
 7   CC   1436 non-null   int64
 8   Doors   1436 non-null   object
 9   Weight   1436 non-null   int64
dtypes: float64(2), int64(4), object(4)
memory usage: 123.4+ KB
</pre></td>
<td>returns a concise summary of a dataframe
• data type of index
• data type of columns
• count of non-null values
• memory usage</td>
</tr>
<tr>
<td>17.</td>
<td>numpy.unique(array)</td>
<td><pre>
In [19]: #7) Getting Unique Elements of each Column

         print(np.unique(car['KM']))

['1' '10000' '100123' ... '99865' '99971' '??']

In [20]: print(np.unique(car['HP']))

['107' '110' '116' '192' '69' '71' '72' '73' '86' '90' '97' '98' '????']

In [21]: print(np.unique(car['MetColor']))

[0. 1. nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan]

In [22]: print(np.unique(car['Automatic']))

[0 1]

In [23]: print(np.unique(car['Doors']))

['2' '3' '4' '5' 'five' 'four' 'three']
</pre></td>
<td>unique() is used to find the unique elements of a column

‘HP’ has special character to it – Hence, it has been read as object instead of int64

‘Doors’ has been read as object instead of int64 because of values ‘five’ ‘four’ ‘three’ which are strings</td>
</tr>
</table>

Missing Values:

We need to know how missing values are represented in the dataset in order to make reasonable decisions

The missing values exist in the form of ‘nan’ ‘??’, ‘????’

Python, by default replace blank values with ‘nan’

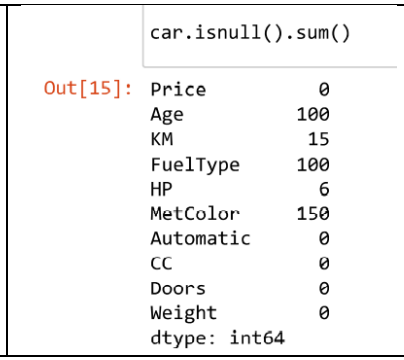
Now, importing the data considering other forms of missing values in a dataframe

 Importing data Concise summary of dataframe Converting variable’s data types Category vs Object data type Cleaning column ‘Doors Getting count of missing values

Sr No	Method	Example and Output	Explanation

18	Dataframe=pd.read_csv(‘file_name.csv’, na_values=[‘??’,’????’])	`car=pd.read_csv('Toyota.csv', index_col=0, na_values=["??","????"])`
		`car.info()`
		
		`<class 'pandas.core.frame.DataFrame'>`
		`Int64Index: 1436 entries, 0 to 1435`
		`Data columns (total 10 columns):`
		` #  Column  Non-Null Count  Dtype`
		`---  ------  --------------  -----`
		` 0  Price  1436 non-null  int64`
		` 1  Age  1336 non-null  float64`
		` 2  KM  1421 non-null  float64`
		` 3  FuelType  1336 non-null  object`
		` 4  HP  1430 non-null  float64`
		` 5  MetColor  1286 non-null  float64`
		` 6  Automatic  1436 non-null  int64`
		` 7  CC  1436 non-null  int64`
		` 8  Doors  1436 non-null  object`
		` 9  Weight  1436 non-null  int64`
		`dtypes: float64(4), int64(4), object(2)`
		`memory usage: 123.4+ KB`
			Imports the data of .csv file into dataframe by considering other forms of data

19	Dataframe.astype( ) Dataframe.nbytes( )	``` car['MetColor']=car['MetColor'].astype('object') car['Automatic']=car['Automatic'].astype('object') car.info() ``` ``` <class 'pandas.core.frame.DataFrame'> Int64Index: 1436 entries, 0 to 1435 Data columns (total 10 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 Price 1436 non-null int64 1 Age 1336 non-null float64 2 KM 1421 non-null float64 3 FuelType 1336 non-null object 4 HP 1430 non-null float64 5 MetColor 1286 non-null object 6 Automatic 1436 non-null object 7 CC 1436 non-null int64 8 Doors 1436 non-null object 9 Weight 1436 non-null int64 dtypes: float64(3), int64(3), object(4) memory usage: 123.4+ KB ```	astype( ) method is used to explicitly convert data types from one to another nbytes() is used to get the total bytes consumed by the elements of the columns
20	Dataframe.replace( )	In [10]: ``` print('Memory when data is of category Type') car['FuelType'].astype('category').nbytes ``` ``` Memory when data is of category Type ``` Out[10]: ``` 1460 ``` In [12]: ``` #5) Cleaning column 'Doors #use of replace() print(np.unique(car['Doors'])) car['Doors'].replace('three',3,inplace=True) car['Doors'].replace('four',4,inplace=True) car['Doors'].replace('five',5,inplace=True) ``` ``` ['2' '3' '4' '5' 'five' 'four' 'three'] ``` In [14]: ``` car['Doors']=car['Doors'].astype('int64') print(np.unique(car['Doors'])) ``` ``` [2 3 4 5] ```	replace() is used to replace a value with the desired value

21	Dataframe.isnull().sum()		To check the count of missing values present in each column Dataframe.isnull.sum() is used