CSE4020

Lab
Session 1

TOPIC: 3 exercises on fundamentals of Machine Learning

Name: Makesh Srinivasan
Registration number: 19BCE1717
Slot: L31 + L32
Date: 14 Aug, 2021 Monday
Faculty: Prof. Abdul Quadir

Exercises:

1) Numpy
2) Pandas
3) Pandas (with try it yourself)

Exercise 1: Numpy

Fundamentals of Numpy and arrays

In [1]:
```
import numpy as np
```

In [3]:
```
arr1 = np.array([1,2,3,4,5,6])
print("1D array: ", arr1)
arr2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print("2D array: ", "\n",arr2)
```

```
1D array:  [1 2 3 4 5 6]
2D array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [4]:
```
arr1 = np.array([1,2,3,4,5,6])
print("1D array: ", arr1)
arr2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print("2D array: ", "\n",arr2)
```

```
1D array:  [1 2 3 4 5 6]
2D array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [5]:
```
# Creating a rank 1 Array
arr = np.array([1, 2, 3])
print("Array with Rank 1: \n",arr)
```

```
Array with Rank 1:
 [1 2 3]
```

In [7]:
```
# Creating a rank 2 Array
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
print("Array with Rank 2: \n", arr)
```

```
Array with Rank 2:
 [[1 2 3]
 [4 5 6]]
```

In [8]:
```
# Creating an array from tuple
arr = np.array((1, 3, 2))
print("\nArray created using "
      "passed tuple:\n", arr)
```

```
Array created using passed tuple:
 [1 3 2]
```

Understanding attributes of arrays

In [9]:
```
print("About arr2:")
print("Type\t\t:" ,type(arr2))
print("Datatype\t:",arr2.dtype)
print("Shape\t\t:", arr2.shape)
print("Size\t\t:",arr2.size)
print("itemsize\t:",arr2.itemsize)
print("No. of dim\t:",arr2.ndim)
print("No. of bytes\t:", arr2.nbytes)
```

```
About arr2:
Type            : <class 'numpy.ndarray'>
Datatype        : int64
Shape           : (3, 3)
Size            : 9
itemsize        : 8
No. of dim      : 2
No. of bytes    : 72
```

Creating Special arrays

In [10]:
```
arr3 = np.zeros((5,2), dtype=int)
arr4 = np.ones((3,4),dtype=float)
arr5 = np.eye(4,3)
arr6 = np.random.rand(3,2)
arr7 = np.random.randint(7,size=(2,6))
print("Zero Array \t:\n",arr3)
print("Arrays with unit values\t:\n",arr4)
print("Identity matrix\t:\n",arr5)
print("Random array\t\n",arr6)
print("Random integer array\t:\n" , arr7)
```

```
Zero Array      :
 [[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]
Arrays with unit values :
 [[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
Identity matrix :
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 0.]]
Random array
 [[0.83199921 0.38730768]
 [0.31777386 0.7129062 ]
 [0.44464553 0.92039782]]
Random integer array    :
 [[5 5 1 6 1 5]
 [6 0 3 2 5 3]]
```

Creating arrays using "arange"

In [11]:
```
arr8 = np.arange(1,10,3)
print("Array created with arange: \t\n",arr8)
```

```
Array created with arange:
 [1 4 7]
```

Creating arrays using "linspace"

In [12]:
```
arr9 = np.linspace(1,10,50)
print("Array created using linspace requesting 50 elements within 1 to 10:\t\n",arr9)
```

```
Array created using linspace requesting 50 elements within 1 to 10:
 [ 1.          1.18367347  1.36734694  1.55102041  1.73469388  1.91836735
  2.10204082  2.28571429  2.46938776  2.65306122  2.83673469  3.02040816
  3.20408163  3.3877551   3.57142857  3.75510204  3.93877551  4.12244898
  4.30612245  4.48979592  4.67346939  4.85714286  5.04081633  5.2244898
  5.40816327  5.59183673  5.7755102   5.95918367  6.14285714  6.32653061
  6.51020408  6.69387755  6.87755102  7.06122449  7.24489796  7.42857143
  7.6122449   7.79591837  7.97959184  8.16326531  8.34693878  8.53061224
  8.71428571  8.89795918  9.08163265  9.26530612  9.44897959  9.63265306
  9.81632653 10.        ]
```

In [13]:
```
arr10 = np.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20]])
print(arr10)
print("Row 2:",arr10[1,:])
print("Column 2:",arr10[:,3])
print("Elements 7,8,12,13 :\n", arr10[1:3,1:3])
print("All elements: \n",arr10[::])
print("Strides:\n",arr10[0::2,1:3])
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
Row 2: [ 6  7  8  9 10]
Column 2: [ 4  9 14 19]
Elements 7,8,12,13 :
 [[ 7  8]
 [12 13]]
All elements:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
Strides:
 [[ 2  3]
 [12 13]]
```

Masking - Important concept in Image processing

In [14]:
```
arr11 = np.array([1,2,3,4,5,6,7,8,9])
mask = np.array([0,1,1,0,1,0,1,0,0],dtype=bool)
print(arr11[mask])
```

```
[2 3 5 7]
```

Scalar Operations - Arithmetic

In [15]:
```
arr13 = np.array([1,2,3])
arr14 = np.array([4,5,6])
arr15 = arr13+arr14
arr16 = arr13 - arr14
print("Summation:\t",arr15)
print("Difference:\t",arr16)

arr16+=5
print("Previous output after adding 5:",arr16)
```

```
Summation:      [5 7 9]
Difference:     [-3 -3 -3]
Previous output after adding 5: [2 2 2]
```

Trignometric & Logarithmic operations

In [16]:
```
arr17 = np.array([15,30,45,90])
result7 = np.sin(arr17)
print("Sin values:\t",result7)
result8 = np.log(arr17)
print("Log value:\t",result8)
```

```
Sin values:	 [ 0.65028784 -0.98803162  0.85090352  0.89399666]
Log value:	 [2.7080502  3.40119738 3.80666249 4.49980967]
```

In [ ]:
```

```

Exercise 2: Pandas

Pandas

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

In [2]:
```
lst = ['Soon', 'For', 'Good', 'is', 'portal', 'for', 'Good']
```

In [3]:
```
df = pd.DataFrame(lst)
print(df)
```

```
         0
0     Soon
1      For
2     Good
3       is
4   portal
5      for
6     Good
```

In [4]:
```
# intialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}
```

In [5]:
```
# Create DataFrame
df = pd.DataFrame(data)
```

In [6]:
```
# Print the output.
print(df)
```

```
    Name  Age
0    Tom   20
1   nick   21
2  krish   19
3   jack   18
```

Pandas

In [7]:
```
# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
```

In [8]:
```
# select two columns
print(df[['Name', 'Qualification']])
```

```
     Name Qualification
0     Jai           Msc
1  Princi            MA
2  Gaurav           MCA
3    Anuj           Phd
```

In [10]:
```
# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving row by loc method
first = data.loc["Avery Bradley"]
second = data.loc["R.J. Hunter"]


print(first, "\n\n\n", second)
```

```
Team        Boston Celtics
Number                   0
Position                PG
Age                     25
Height                2-Jun
Weight                  180
College               Texas
Salary           7730337.0
Name: Avery Bradley, dtype: object


 Team        Boston Celtics
Number                   28
Position                 SG
Age                      22
Height                5-Jun
Weight                  185
College       Georgia State
Salary           1148640.0
Name: R.J. Hunter, dtype: object
```

In [11]:
```
# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving columns by indexing operator
first = data["Age"]
print(first)
```

```
Name
Avery Bradley    25
Jae Crowder      25
John Holland     27
R.J. Hunter      22
Jonas Jerebko    29
                 ..
Trey Lyles       20
Shelvin Mack     26
Raul Neto        24
Tibor Pleiss     26
Jeff Withey      26
Name: Age, Length: 457, dtype: int64
```

In [12]:
```
# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")
# retrieving rows by iloc method
row2 = data.iloc[3]
print(row2)
```

```
Team         Boston Celtics
Number                   28
Position                 SG
Age                      22
Height                5-Jun
Weight                  185
College       Georgia State
Salary           1148640.0
Name: R.J. Hunter, dtype: object
```

In [13]:
```
# In order to check missing values in Pandas DataFrame, we use a function isnull() and notnull(). Both fu
```

In [14]:
```
# importing numpy as np
import numpy as np
```

In [15]:
```
# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from list
df = pd.DataFrame(dict)

# using isnull() function
df.isnull()
```

Out[15]:

	First Score	Second Score	Third Score
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

In [16]:
```
# Filling missing values using fillna(), replace() and interpolate() :
# In order to fill null values in a datasets, we use fillna(), replace() and interpolate() funct
```

In [17]:
```
# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

# filling missing value using fillna()
df.fillna(0)
```

Out[17]:

	First Score	Second Score	Third Score
0	100.0	30.0	0.0
1	90.0	45.0	40.0
2	0.0	56.0	80.0
3	95.0	0.0	98.0

In [18]:
```
# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, np.nan, 45, 56],
        'Third Score':[52, 40, 80, 98],
        'Fourth Score':[np.nan, np.nan, np.nan, 65]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)
df.dropna()
```

Out[18]:

	First Score	Second Score	Third Score	Fourth Score
3	95.0	56.0	98	65.0

In [19]:
```
# Iterating over rows :
# In order to iterate over rows, we can use three function iteritems(), iterrows(), itertuples(
```

In [20]:
```
# dictionary of lists
dict = {'name':["aparna", "pankaj", "sudhir", "Geeku"],
        'degree': ["MBA", "BCA", "M.Tech", "MBA"],
        'score':[90, 40, 80, 98]}

# creating a dataframe from a dictionary
df = pd.DataFrame(dict)

print(df)
```

```
     name  degree  score
0  aparna     MBA     90
1  pankaj     BCA     40
2  sudhir  M.Tech     80
3   Geeku     MBA     98
```

In [21]:
```
# dictionary of lists
dict = {'name':["aparna", "pankaj", "sudhir", "Geeku"],
        'degree': ["MBA", "BCA", "M.Tech", "MBA"],
        'score':[90, 40, 80, 98]}

# creating a dataframe from a dictionary
df = pd.DataFrame(dict)

# iterating over rows using iterrows() function
for i, j in df.iterrows():
    print(i, j)
    print()
```

```
0 name      aparna
degree       MBA
score         90
Name: 0, dtype: object

1 name      pankaj
degree       BCA
score         40
Name: 1, dtype: object

2 name      sudhir
degree    M.Tech
score         80
Name: 2, dtype: object

3 name      Geeku
degree       MBA
score         98
Name: 3, dtype: object
```

In [22]:
```
# Iterating over Columns :
# In order to iterate over columns, we need to create a list of dataframe columns and then iterating
```

In [23]:
```
# dictionary of lists
dict = {'name':["aparna", "pankaj", "sudhir", "Geeku"],
        'degree': ["MBA", "BCA", "M.Tech", "MBA"],
        'score':[90, 40, 80, 98]}

# creating a dataframe from a dictionary
df = pd.DataFrame(dict)

print(df)
```

```
     name  degree  score
0  aparna     MBA     90
1  pankaj     BCA     40
2  sudhir  M.Tech     80
3   Geeku     MBA     98
```

Pandas

In [24]:
```
# creating a list of dataframe columns
columns = list(df)

for i in columns:

    # printing the third element of the column
    print (df[i][2])
```

```
sudhir
M.Tech
80
```

In [25]:
```
df2 = pd.read_csv("nba.csv")
df2.head()
```

Out[25]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	2-Jun	180	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6-Jun	235	Marquette	6796117.0
2	John Holland	Boston Celtics	30	SG	27	5-Jun	205	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	5-Jun	185	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	10-Jun	231	NaN	5000000.0

In [26]:
```
df2.tail()
```

Out[26]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
452	Trey Lyles	Utah Jazz	41	PF	20	10-Jun	234	Kentucky	2239800.0
453	Shelvin Mack	Utah Jazz	8	PG	26	3-Jun	203	Butler	2433333.0
454	Raul Neto	Utah Jazz	25	PG	24	1-Jun	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21	C	26	3-Jul	256	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24	C	26	Jul-00	231	Kansas	947276.0

Pandas

In [27]:
```
print(df2.head)
```

```
<bound method NDFrame.head of                Name            Team  Number Position  Age  Height  Weight  \
0    Avery Bradley  Boston Celtics       0       PG   25   2-Jun     180
1      Jae Crowder  Boston Celtics      99       SF   25   6-Jun     235
2     John Holland  Boston Celtics      30       SG   27   5-Jun     205
3      R.J. Hunter  Boston Celtics      28       SG   22   5-Jun     185
4    Jonas Jerebko  Boston Celtics       8       PF   29  10-Jun     231
..             ...             ...     ...      ...  ...     ...     ...
452     Trey Lyles       Utah Jazz      41       PF   20  10-Jun     234
453   Shelvin Mack       Utah Jazz       8       PG   26   3-Jun     203
454      Raul Neto       Utah Jazz      25       PG   24   1-Jun     179
455   Tibor Pleiss       Utah Jazz      21        C   26   3-Jul     256
456    Jeff Withey       Utah Jazz      24        C   26  Jul-00     231

                College     Salary
0                 Texas  7730337.0
1             Marquette  6796117.0
2    Boston University        NaN
3          Georgia State  1148640.0
4                   NaN  5000000.0
..                  ...        ...
452           Kentucky  2239800.0
453             Butler  2433333.0
454                NaN   900000.0
455                NaN  2900000.0
456             Kansas   947276.0

[457 rows x 9 columns]>
```

In [28]:
```
print("Type:\n",type(df2))
print("Information about the dataframe:",df2.info)
```

```
Type:
 <class 'pandas.core.frame.DataFrame'>
Information about the dataframe: <bound method DataFrame.info of                Name            Team  Number Position
Age  Height  Weight  \
0    Avery Bradley  Boston Celtics       0       PG   25   2-Jun     180
1      Jae Crowder  Boston Celtics      99       SF   25   6-Jun     235
2     John Holland  Boston Celtics      30       SG   27   5-Jun     205
3      R.J. Hunter  Boston Celtics      28       SG   22   5-Jun     185
4    Jonas Jerebko  Boston Celtics       8       PF   29  10-Jun     231
..             ...             ...     ...      ...  ...     ...     ...
452     Trey Lyles       Utah Jazz      41       PF   20  10-Jun     234
453   Shelvin Mack       Utah Jazz       8       PG   26   3-Jun     203
454      Raul Neto       Utah Jazz      25       PG   24   1-Jun     179
455   Tibor Pleiss       Utah Jazz      21        C   26   3-Jul     256
456    Jeff Withey       Utah Jazz      24        C   26  Jul-00     231

                College     Salary
0                 Texas  7730337.0
1             Marquette  6796117.0
2    Boston University        NaN
3          Georgia State  1148640.0
4                   NaN  5000000.0
..                  ...        ...
452           Kentucky  2239800.0
453             Butler  2433333.0
454                NaN   900000.0
455                NaN  2900000.0
456             Kansas   947276.0

[457 rows x 9 columns]>
```

Pandas

In [29]:
```
print("Shape of datafframe:",df2.shape)
df2.drop_duplicates()
print(df2.shape)
```

```
Shape of datafframe: (457, 9)
(457, 9)
```

In [30]:
```
print("Columns of dataframe:\t",df2.columns)
```

```
Columns of dataframe:    Index(['Name', 'Team', 'Number', 'Position', 'Age', 'Height', 'Weight',
       'College', 'Salary'],
      dtype='object')
```

In [31]:
```
print(df2.describe())
```

```
            Number         Age      Weight        Salary
count  457.000000  457.000000  457.000000  4.460000e+02
mean    17.678337   26.938731  221.522976  4.842684e+06
std     15.966090    4.404016   26.368343  5.229238e+06
min      0.000000   19.000000  161.000000  3.088800e+04
25%      5.000000   24.000000  200.000000  1.044792e+06
50%     13.000000   26.000000  220.000000  2.839073e+06
75%     25.000000   30.000000  240.000000  6.500000e+06
max     99.000000   40.000000  307.000000  2.500000e+07
```

Exercise 3: Pandas (with try it yourself)

The try it yourself parts are in green tables

In [2]:
```
import pandas as pd
# Building DataFrames with Pandas
# From Python Dictionaries
```

In [4]:
```
list1 = ['S1','S2','S3','S4']
English = [89,90,78,98]
Maths = [67,90,100,100]
Science = [78,89,90,89]

titles = ['ID', 'English','Maths','Science']
values = [list1,English,Maths,Science]
result = list(zip(titles,values))
result
```

Out[4]:
```
[('ID', ['S1', 'S2', 'S3', 'S4']),
 ('English', [89, 90, 78, 98]),
 ('Maths', [67, 90, 100, 100]),
 ('Science', [78, 89, 90, 89])]
```

In [5]:
```
result1 = dict(result)
df1 = pd.DataFrame(result1)
print(df1)
```

```
   ID  English  Maths  Science
0  S1       89     67       78
1  S2       90     90       89
2  S3       78    100       90
3  S4       98    100       89
```

In [6]:
```
# Importing data
# Dataframes from CSV files
```

In [9]:
```
df2 = pd.read_csv('character-deaths.csv')
df2.head()
```

Out[9]:

	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
0	Addam Marbrand	Lannister	NaN	NaN	NaN	56.0	1	1	1	1	1	1	0
1	Aegon Frey (Jinglebell)	None	299.0	3.0	51.0	49.0	1	1	0	0	1	0	0
2	Aegon Targaryen	House Targaryen	NaN	NaN	NaN	5.0	1	1	0	0	0	0	1
3	Adrack Humble	House Greyjoy	300.0	5.0	20.0	20.0	1	1	0	0	0	0	1
4	Aemon Costayne	Lannister	NaN	NaN	NaN	NaN	1	1	0	0	1	0	0

In [10]:
```
df2.tail()
```

Out[10]:

	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
912	Zollo	None	NaN	NaN	NaN	21.0	1	0	0	0	1	0	0
913	Yurkhaz zo Yunzak	None	300.0	5.0	59.0	47.0	1	0	0	0	0	0	1
914	Yezzan Zo Qaggaz	None	300.0	5.0	57.0	25.0	1	1	0	0	0	0	1
915	Torwynd the Tame	Wildling	300.0	5.0	73.0	73.0	1	0	0	0	1	0	0
916	Talbert Serry	Tyrell	300.0	4.0	29.0	29.0	1	1	0	0	0	1	0

TRY IT YOURSELF

Remove headers from the csv file and import

Then,add header to it

In [60]:
```
df=pd.read_csv('character-deaths.csv',header=None)
new_header = df.iloc[0]
df2 = df[1:]
df2.columns = new_header
df2.head()
```

Out[60]:

	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
1	Addam Marbrand	Lannister	NaN	NaN	NaN	56	1	1	1	1	1	1	0
2	Aegon Frey (Jinglebell)	None	299	3	51	49	1	1	0	0	1	0	0
3	Aegon Targaryen	House Targaryen	NaN	NaN	NaN	5	1	1	0	0	0	0	1
4	Adrack Humble	House Greyjoy	300	5	20	20	1	1	0	0	0	0	1
5	Aemon Costayne	Lannister	NaN	NaN	NaN	NaN	1	1	0	0	1	0	0

Display first 10 lines of the file, and Save the processed data frames to CSV File

In [62]:
```
df2.head(10)
```

Out[62]:

	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
1	Addam Marbrand	Lannister	NaN	NaN	NaN	56	1	1	1	1	1	1	0
2	Aegon Frey (Jinglebell)	None	299	3	51	49	1	1	0	0	1	0	0
3	Aegon Targaryen	House Targaryen	NaN	NaN	NaN	5	1	1	0	0	0	0	1
4	Adrack Humble	House Greyjoy	300	5	20	20	1	1	0	0	0	0	1
5	Aemon Costayne	Lannister	NaN	NaN	NaN	NaN	1	1	0	0	1	0	0
6	Aemon Estermont	Baratheon	NaN	NaN	NaN	NaN	1	1	0	1	1	0	0
7	Aemon Targaryen (son of Maekar I)	Night's Watch	300	4	35	21	1	1	1	0	1	1	0
8	Aenys Frey	None	300	5	NaN	59	0	1	1	1	1	0	1
9	Aeron Greyjoy	House Greyjoy	NaN	NaN	NaN	11	1	1	0	1	0	1	0
10	Aethan	Night's Watch	NaN	NaN	NaN	0	1	0	0	0	1	0	0

In [15]:
```
df2.to_csv("result.csv",index=False)
```

TRY IT YOURSELF

Open a file with xls as extension and save it back as a csv file

In [17]:
```
df=pd.read_excel('file_example_XLS_10.xls')
df
```

Out[17]:

	0	First Name	Last Name	Gender	Country	Age	Date	Id
0	1	Dulce	Abril	Female	United States	32	15/10/2017	1562
1	2	Mara	Hashimoto	Female	Great Britain	25	16/08/2016	1582
2	3	Philip	Gent	Male	France	36	21/05/2015	2587
3	4	Kathleen	Hanner	Female	United States	25	15/10/2017	3549
4	5	Nereida	Magwood	Female	United States	58	16/08/2016	2468
5	6	Gaston	Brumm	Male	United States	24	21/05/2015	2554
6	7	Etta	Hurn	Female	Great Britain	56	15/10/2017	3598
7	8	Earlean	Melgar	Female	United States	27	16/08/2016	2456
8	9	Vincenza	Weiland	Female	United States	40	21/05/2015	6548

In [18]:
```
df.to_csv("xls_result.csv",index=False)
```

result.csv	an hour ago	38.9 kB
xls_result.csv	an hour ago	543 B

Pandas - Operations: Creating index for entities

In [19]:
```
df2.set_index("Gender",inplace=True)
print(df2.head)
```

```
<bound method NDFrame.head of 0                          Name      Allegiances Death Year Book of Death  \
Gender
1                Addam Marbrand        Lannister        NaN           NaN
1       Aegon Frey (Jinglebell)             None        299             3
1               Aegon Targaryen  House Targaryen        NaN           NaN
1                 Adrack Humble    House Greyjoy        300             5
1                 Aemon Costayne        Lannister        NaN           NaN
...                          ...              ...        ...           ...
1                         Zollo             None        NaN           NaN
1             Yurkhaz zo Yunzak             None        300             5
1              Yezzan Zo Qaggaz             None        300             5
1              Torwynd the Tame         Wildling        300             5
1                 Talbert Serry           Tyrell        300             4

0      Death Chapter Book Intro Chapter Nobility GoT CoK SoS FfC DwD
Gender
1                NaN                56        1   1   1   1   1   0
1                 51                49        1   0   0   1   0   0
1                NaN                 5        1   0   0   0   0   1
1                 20                20        1   0   0   0   0   1
1                NaN               NaN        1   0   0   1   0   0
...              ...               ...      ...  ..  ..  ..  ..  ..
1                NaN                21        0   0   0   1   0   0
1                 59                47        0   0   0   0   0   1
1                 57                25        1   0   0   0   0   1
1                 73                73        0   0   0   1   0   0
1                 29                29        1   0   0   0   1   0

[917 rows x 12 columns]>
```

In [20]:
```
df2=pd.read_csv("result.csv")
print(df2.head)
```

```
<bound method NDFrame.head of                          Name      Allegiances  Death Year  Book of Death  \
0             Addam Marbrand        Lannister         NaN            NaN
1    Aegon Frey (Jinglebell)             None       299.0            3.0
2            Aegon Targaryen  House Targaryen         NaN            NaN
3              Adrack Humble    House Greyjoy       300.0            5.0
4              Aemon Costayne        Lannister         NaN            NaN
..                        ...              ...         ...            ...
912                    Zollo             None         NaN            NaN
913        Yurkhaz zo Yunzak             None       300.0            5.0
914         Yezzan Zo Qaggaz             None       300.0            5.0
915         Torwynd the Tame         Wildling       300.0            5.0
916             Talbert Serry           Tyrell       300.0            4.0

     Death Chapter  Book Intro Chapter  Gender  Nobility  GoT  CoK  SoS  FfC  \
0              NaN                56.0       1         1    1    1    1    1
1             51.0                49.0       1         1    0    0    1    0
2              NaN                 5.0       1         1    0    0    0    0
3             20.0                20.0       1         1    0    0    0    0
4              NaN                 NaN       1         1    0    0    1    0
..             ...                 ...     ...       ...  ...  ...  ...  ...
912            NaN                21.0       1         0    0    0    1    0
913           59.0                47.0       1         0    0    0    0    0
914           57.0                25.0       1         1    0    0    0    0
915           73.0                73.0       1         0    0    0    1    0
916           29.0                29.0       1         1    0    0    0    1

     DwD
0      0
1      0
2      1
3      1
4      0
..   ...
912    0
913    1
914    1
915    0
916    0

[917 rows x 13 columns]>
```

click to scroll output; double click to hide

Type and information about dataframe

In [21]:
```
print("Type:\n",type(df2))
print("Information about the dataframe:",df2.info)
```

```
Type:
 <class 'pandas.core.frame.DataFrame'>
Information about the dataframe: <bound method DataFrame.info of                          Name      Allegiances  Deat
h Year  Book of Death  \
0              Addam Marbrand        Lannister         NaN             NaN
1    Aegon Frey (Jinglebell)             None       299.0             3.0
2            Aegon Targaryen  House Targaryen         NaN             NaN
3              Adrack Humble    House Greyjoy       300.0             5.0
4              Aemon Costayne        Lannister         NaN             NaN
..                        ...              ...         ...             ...
912                    Zollo             None         NaN             NaN
913        Yurkhaz zo Yunzak             None       300.0             5.0
914         Yezzan Zo Qaggaz             None       300.0             5.0
915         Torwynd the Tame         Wildling       300.0             5.0
916            Talbert Serry           Tyrell       300.0             4.0

     Death Chapter  Book Intro Chapter  Gender  Nobility  GoT  CoK  SoS  FfC  \
0               NaN                56.0       1         1    1    1    1    1
1              51.0                49.0       1         1    0    0    1    0
2               NaN                 5.0       1         1    0    0    0    0
3              20.0                20.0       1         1    0    0    0    0
4               NaN                 NaN       1         1    0    0    1    0
..              ...                 ...     ...       ...  ...  ...  ...  ...
912             NaN                21.0       1         0    0    0    1    0
913            59.0                47.0       1         0    0    0    0    0
914            57.0                25.0       1         1    0    0    0    0
915            73.0                73.0       1         0    0    0    1    0
916            29.0                29.0       1         1    0    0    0    1

     DwD
0      0
1      0
2      1
3      1
4      0
..   ...
912    0
913    1
914    1
915    0
916    0

[917 rows x 13 columns]>
```

In [22]:
```
print("Shape of datafframe:",df2.shape)
df2.drop_duplicates()
print(df2.shape)
```

```
Shape of datafframe: (917, 13)
(917, 13)
```

TRY IT YOURSELF

In [23]:
```
## Append same data in df2 to df2 itself using append function, print its shape.
## Use drop_duplicates to remove duplicate
## print shape again
```

In [24]:
```
df2=df2.append(df2)
print(df2.shape)
```

```
(1834, 13)
```

In [25]:
```
df2=df2.drop_duplicates()
print(df2.shape)
```

```
(917, 13)
```

In [26]:
```
print("Columns of dataframe:\t",df2.columns)
```

```
Columns of dataframe:    Index(['Name', 'Allegiances', 'Death Year', 'Book of Death', 'Death Chapter',
       'Book Intro Chapter', 'Gender', 'Nobility', 'GoT', 'CoK', 'SoS', 'FfC',
       'DwD'],
      dtype='object')
```

In [27]:
```
df2.rename(columns={'Book of Death':'BoD'},inplace=True)
print("Columns of dataframe:\t",df2.columns)
```

```
Columns of dataframe:    Index(['Name', 'Allegiances', 'Death Year', 'BoD', 'Death Chapter',
       'Book Intro Chapter', 'Gender', 'Nobility', 'GoT', 'CoK', 'SoS', 'FfC',
       'DwD'],
      dtype='object')
```

In [28]:
```
print(df2.describe())
```

```
       Death Year          BoD  Death Chapter  Book Intro Chapter      Gender  \
count  305.000000  307.000000     299.000000          905.000000  917.000000
mean   299.157377    2.928339      40.070234           28.861878    0.828790
std      0.703483    1.326482      20.470270           20.165788    0.376898
min    297.000000    1.000000       0.000000            0.000000    0.000000
25%    299.000000    2.000000      25.500000           11.000000    1.000000
50%    299.000000    3.000000      39.000000           27.000000    1.000000
75%    300.000000    4.000000      57.000000           43.000000    1.000000
max    300.000000    5.000000      80.000000           80.000000    1.000000

         Nobility         GoT         CoK         SoS         FfC         DwD
count  917.000000  917.000000  917.000000  917.000000  917.000000  917.000000
mean     0.468920    0.272628    0.353326    0.424209    0.272628    0.284624
std      0.499305    0.445554    0.478264    0.494492    0.445554    0.451481
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
75%      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
```

In [29]:
```
print("No. of chapters talking about death:\t",df2['Death Chapter'].count())
print("No. of unique chapters talking about death:\t",df2['Death Chapter'].nunique())
print("Unique chapters talking about death:\t",df2['Death Chapter'].unique())
```

```
No. of chapters talking about death:     299
No. of unique chapters talking about death:      71
Unique chapters talking about death:     [nan 51. 20. 35. 56.  4. 46. 10. 34. 47. 41. 21. 33. 39. 52. 37. 24. 31.
 27. 62. 65.  7. 49. 50. 42. 30. 66. 64. 29.  0. 53. 58. 76. 63. 55. 14.
 61. 12. 68. 69. 80. 36.  1. 19. 72. 59. 43. 70. 75. 11.  3. 60. 26. 44.
 67. 45. 18. 23. 57. 25. 16. 17. 48.  9.  2.  6. 32. 77. 74. 40. 38. 73.]
```

TRY IT YOURSELF

In [29]:
```
print("No. of chapters talking about death:\t",df2['Death Chapter'].count())
print("No. of unique chapters talking about death:\t",df2['Death Chapter'].nunique())
print("Unique chapters talking about death:\t",df2['Death Chapter'].unique())
```

```
No. of chapters talking about death:     299
No. of unique chapters talking about death:      71
Unique chapters talking about death:     [nan 51. 20. 35. 56.  4. 46. 10. 34. 47. 41. 21. 33. 39. 52. 37. 24. 31.
 27. 62. 65.  7. 49. 50. 42. 30. 66. 64. 29.  0. 53. 58. 76. 63. 55. 14.
 61. 12. 68. 69. 80. 36.  1. 19. 72. 59. 43. 70. 75. 11.  3. 60. 26. 44.
 67. 45. 18. 23. 57. 25. 16. 17. 48.  9.  2.  6. 32. 77. 74. 40. 38. 73.]
```

In [30]:
```
print("First five values of 'Allegiances' column\t:\n",df2.iloc[0:5,0])
```

```
First five values of 'Allegiances' column       :
 0              Addam Marbrand
1    Aegon Frey (Jinglebell)
2           Aegon Targaryen
3             Adrack Humble
4             Aemon Costayne
Name: Name, dtype: object
```

Accessing using location - .loc[ ]

In [31]:
```
print(df2.head())
df2.set_index('Name',inplace=True)
print(df2.head())
print("\n\nUsing loc method:\t",df2.loc['Adrack Humble','Death Year'])
```

```
                      Name      Allegiances  Death Year  BoD  Death Chapter  \
0           Addam Marbrand        Lannister         NaN  NaN            NaN
1  Aegon Frey (Jinglebell)             None       299.0  3.0           51.0
2          Aegon Targaryen  House Targaryen         NaN  NaN            NaN
3            Adrack Humble    House Greyjoy       300.0  5.0           20.0
4            Aemon Costayne        Lannister         NaN  NaN            NaN

   Book Intro Chapter  Gender  Nobility  GoT  CoK  SoS  FfC  DwD
0                56.0       1         1    1    1    1    1    0
1                49.0       1         1    0    0    1    0    0
2                 5.0       1         1    0    0    0    0    1
3                20.0       1         1    0    0    0    0    1
4                 NaN       1         1    0    0    1    0    0
                             Allegiances  Death Year  BoD  Death Chapter  \
Name
Addam Marbrand                 Lannister         NaN  NaN            NaN
Aegon Frey (Jinglebell)             None       299.0  3.0           51.0
Aegon Targaryen          House Targaryen         NaN  NaN            NaN
Adrack Humble              House Greyjoy       300.0  5.0           20.0
Aemon Costayne                 Lannister         NaN  NaN            NaN

                         Book Intro Chapter  Gender  Nobility  GoT  CoK  SoS  \
Name
Addam Marbrand                         56.0       1         1    1    1    1
Aegon Frey (Jinglebell)                49.0       1         1    0    0    1
Aegon Targaryen                         5.0       1         1    0    0    0
Adrack Humble                          20.0       1         1    0    0    0
Aemon Costayne                          NaN       1         1    0    0    1

                         FfC  DwD
Name
Addam Marbrand             1    0
Aegon Frey (Jinglebell)    0    0
Aegon Targaryen            0    1
Adrack Humble              0    1
Aemon Costayne             0    0


Using loc method:        300.0
```

Creating subsets using relational operators

In [32]:
```
df3 = df2['Death Year']<400
print(df3)
df4 = df2[df3]
print(df4)
```

```
Name
Addam Marbrand             False
Aegon Frey (Jinglebell)     True
Aegon Targaryen            False
Adrack Humble               True
Aemon Costayne             False
                           ...
Zollo                      False
Yurkhaz zo Yunzak           True
Yezzan Zo Qaggaz            True
Torwynd the Tame            True
Talbert Serry               True
Name: Death Year, Length: 917, dtype: bool
                                    Allegiances  Death Year  BoD  \
Name
Aegon Frey (Jinglebell)                    None       299.0  3.0
Adrack Humble                    House Greyjoy       300.0  5.0
Aemon Targaryen (son of Maekar I)  Night's Watch       300.0  4.0
Aenys Frey                                 None       300.0  5.0
Aggar                            House Greyjoy       299.0  2.0
...                                        ...         ...  ...
Young Henly                      Night's Watch       299.0  3.0
Yurkhaz zo Yunzak                          None       300.0  5.0
Yezzan Zo Qaggaz                           None       300.0  5.0
Torwynd the Tame                     Wildling       300.0  5.0
Talbert Serry                          Tyrell       300.0  4.0

                                    Death Chapter  Book Intro Chapter  Gender  \
Name
Aegon Frey (Jinglebell)                      51.0                49.0       1
Adrack Humble                                20.0                20.0       1
Aemon Targaryen (son of Maekar I)            35.0                21.0       1
Aenys Frey                                    NaN                59.0       0
Aggar                                        56.0                50.0       1
...                                           ...                 ...     ...
Young Henly                                  55.0                55.0       1
Yurkhaz zo Yunzak                            59.0                47.0       1
Yezzan Zo Qaggaz                             57.0                25.0       1
Torwynd the Tame                             73.0                73.0       1
Talbert Serry                                29.0                29.0       1

                                    Nobility  GoT  CoK  SoS  FfC  DwD
Name
Aegon Frey (Jinglebell)                    1    0    0    1    0    0
```

using groupby

In [33]:
```
print(df2['Death Year'].nunique)
df5 = df2.groupby('Death Year').count()
```

```
<bound method IndexOpsMixin.nunique of Name
Addam Marbrand                NaN
Aegon Frey (Jinglebell)     299.0
Aegon Targaryen               NaN
Adrack Humble               300.0
Aemon Costayne                NaN
                            ...
Zollo                         NaN
Yurkhaz zo Yunzak           300.0
Yezzan Zo Qaggaz            300.0
Torwynd the Tame            300.0
Talbert Serry               300.0
Name: Death Year, Length: 917, dtype: float64>
```

In [34]:
```
df5
```

Out[34]:

Death Year	Allegiances	BoD	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
297.0	3	3	3	3	3	3	3	3	3	3	3
298.0	46	46	46	46	46	46	46	46	46	46	46
299.0	156	156	154	154	156	156	156	156	156	156	156
300.0	100	100	92	97	100	100	100	100	100	100	100

In [45]:
```
import numpy as np
```

In [44]:
```
df5 = df2.groupby("Death Year").agg({"Death Year":np.sum,"Gender":np.sum})
print(df5)
```

```
             Death Year  Gender
Death Year
297.0            891.0       3
298.0          13708.0      43
299.0          46644.0     141
300.0          30000.0      82
```

Visualization

In [49]:
```
%matplotlib inline
```

In [50]:
```
df5.plot(x='Death Year',y='Gender',kind='scatter')
```

Out[50]:
```
<AxesSubplot:xlabel='Death Year', ylabel='Gender'>
```