Machine Learning

Lab 2

S Shyam Sundaram

19BCE1560

9th August, 2021

EXERCISE 1: NUMPY

CREATING ARRAYS

Code

```
import numpy as np
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)
# Creating a rank 1 Array
arr = np.array([1, 2, 3])
print("Array with Rank 1: \n",arr)
# Creating a rank 2 Array
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
print("Array with Rank 2: \n", arr)
# Creating an array from tuple
arr = np.array((1, 3, 2))
print("\nArray created using "
      "passed tuple:\n", arr)
#Using 'arange'
arr8 = np.arange(1,10,3)
print("Array created with arange: \t\n",arr8)
#Using linspace
arr9 = np.linspace(1,10,50)
print("Array created using linspace requesting 50 elements within 1 to 10:\t\n",arr9)
```

Output

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

In [2]: 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 [3]: # 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 [4]: # 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 [5]: # 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

Code

```
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)
```

Output

```
In [8]: 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        : int32
        Shape           : (3, 3)
        Size            : 9
        itemsize        : 4
        No. of dim      : 2
        No. of bytes    : 36
```

SPECIAL ARRAYS

Code

```
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)
```

Output

```
In [9]: 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
 [[0.78050112 0.94526287]
 [0.90809064 0.48067856]
 [0.77292278 0.71222528]]
Random integer array    :
 [[5 3 6 1 2 1]
 [3 4 3 4 3 0]]
```

SLICING

Code

```
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])
```

Output

```
In [12]: 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

Code

```
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])
```

Output

```
In [13]: 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

Code

```
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)
```

Output

```
In [14]: 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)

         Summation:      [5 7 9]
         Difference:     [-3 -3 -3]

In [15]: arr16+=5
         print("Previous output after adding 5:",arr16)

         Previous output after adding 5: [2 2 2]
```

TRIGONOMETRIC OPERATIONS

Code

```
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)
```

Output

```
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]
```

EXERCISE 2: PANDAS

CREATING DATAFRAMES FROM LIST

Code

```
import pandas as pd
# list of strings
lst = ['Soon', 'For', 'Good', 'is',
        'portal', 'for', 'Good']
# Calling DataFrame constructor on list
df = pd.DataFrame(lst)
print(df)
```

Output

```
In [18]: # list of strings
         lst = ['Soon', 'For', 'Good', 'is',
                     'portal', 'for', 'Good']

In [19]: # Calling DataFrame constructor on list
         df = pd.DataFrame(lst)
         print(df)

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

CREATING DATAFRAME FROM DICTIONARY

Code

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

# Create DataFrame
df = pd.DataFrame(data)
print(df)
# 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)

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

Output

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

         # Create DataFrame
         df = pd.DataFrame(data)
         print(df)
```

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

```
In [22]: # 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)

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

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

CREATE DATAFRAME FROM CSV FILE

Code

```
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)
```

Output

```
In [25]: # making data frame from csv file
         data = pd.read_csv("nba.csv", index_col ="Name")
```

```
In [26]: # 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
```

RETRIEVAL OF DATA FROM DATAFRAME

Code

```
# retrieving columns by indexing operator
first = data["Age"]
print(first)
# retrieving rows by iloc method
row2 = data.iloc[3]
print(row2)
```

Output

```
In [27]: # 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 [29]: # 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
```

HANDLING MISSING VALUES

Code

```
# 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()
```

```
# filling missing value using fillna()
df.fillna(0)
# 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() #drop rows with NA
```

Output

In order to check missing values in Pandas DataFrame, we use a function isnull() and notnull(). Both function help in checking whether a value is NaN or not. These function can also be used in Pandas Series in order to find null values in a series.

In [30]:
```
# 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)
```

In [31]:
```
# using isnull() function
df.isnull()
```

Out[31]:

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

In order to fill null values in a datasets, we use fillna(), replace() and interpolate() function these function replace NaN values with some value of their own. All these function help in filling a null values in datasets of a DataFrame. Interpolate() function is basically used to fill NA values in the dataframe but it uses various interpolation technique to fill the missing values rather than hard-coding the value.

In [32]:
```
# filling missing value using fillna()
df.fillna(0)
```

Out[32]:

	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 [33]:
```
# 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)
```

In [35]:
```
df.dropna() #drop rows with NA
```

Out[35]:

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

ITERATING OVER ROWS

Code

```
# 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)

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

Output

In order to iterate over rows, we can use three function iteritems(), iterrows(), itertuples() . These three function will help in iteration over rows.

In [36]:
```
# 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 [37]:
```
# 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
```

ITERATING OVER COLUMNS

Code

```
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)
# creating a list of dataframe columns
columns = list(df)

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

Output

In order to iterate over columns, we need to create a list of dataframe columns and then iterating through that list to pull out the dataframe columns.

In [39]:
```
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 [40]:
```
# 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
```

WORKING WITH NBA DATASET

READING THE CSV AND DISPLAY THE HEAD AND TAIL

Code

```
df2 = pd.read_csv("nba.csv")
df2.head()
df2.tail()
```

Output

```
In [8]: df2 = pd.read_csv("nba.csv")
        df2.head()
```

Out[8]:

	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 [9]: df2.tail()
```

Out[9]:

	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

CHECKING OUT THE TYPE AND INFORMATION REGARDING THE DATA FRAME

Code

```
print("Type:\n",type(df2))
print("Information about the dataframe:\n")
df2.info(verbose=True)
print("Shape of dataframe:",df2.shape)
df2.drop_duplicates()
print(df2.shape)
print("Columns of dataframe:\t",df2.columns)
print(df2.describe())
```

Output

```
In [19]: print("Type:\n",type(df2))
         print("Information about the dataframe:\n")
         df2.info(verbose=True)

Type:
 <class 'pandas.core.frame.DataFrame'>
Information about the dataframe:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 457 entries, 0 to 456
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      457 non-null    object
 1   Team      457 non-null    object
 2   Number    457 non-null    int64
 3   Position  457 non-null    object
 4   Age       457 non-null    int64
 5   Height    457 non-null    object
 6   Weight    457 non-null    int64
 7   College   373 non-null    object
 8   Salary    446 non-null    float64
dtypes: float64(1), int64(3), object(5)
memory usage: 32.3+ KB
```

DROPPING DUPLICATE ROWS AND DISPLAYING A DESCRIPTION OF THE DATASET

Code

```
print("Shape of dataframe:",df2.shape)
df2.drop_duplicates()
print(df2.shape)
print("Columns of dataframe:\t",df2.columns)
print(df2.describe())
```

Output

```
In [11]: print("Shape of dataframe:",df2.shape)
         df2.drop_duplicates()
         print(df2.shape)

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

```
In [45]: print("Columns of dataframe:\t",df2.columns)

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

```
In [46]: 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 (CHARACTER DEATH DATA)

REMOVING THE HEADER

Code

```
df2 = pd.read_csv('character-deaths.csv',header=None)
df2.head()
```

Output

Reading the CSV without using the 1st row as header or column names.

```
In [4]: df2 = pd.read_csv('character-deaths.csv',header=None)
        df2.head()
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	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

ADDING THE HEADER BACK

Code

```
header=df2.iloc[0]
df2=df2[1:]
df2.columns=header
df2.head()
```

Output

Using the first row of the dataframe as column names

```
In [5]: header=df2.iloc[0]
        df2=df2[1:]
        df2.columns=header
        df2.head()
```

Out[5]:

	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

DISPLAYING THE FIRST 10 ROWS

Code

df2[:10]

Output

```
In [7]: df2[:10]
```

Out[7]:

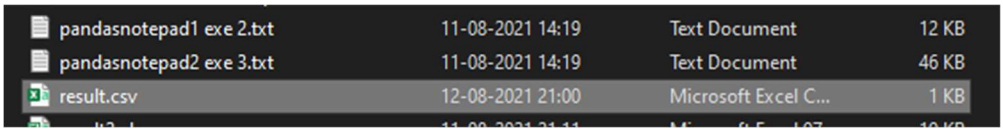
	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
1	Addam Marbrand	Lannister	NaN	NaN	NaN	56	1	1	1	1	1	1	0
2	Aegon Frey (Jinglebell)	None	299	3	51	49	1	1	0	0	1	0	0
3	Aegon Targaryen	House Targaryen	NaN	NaN	NaN	5	1	1	0	0	0	0	1
4	Adrack Humble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

SAVING THE MODIFIED DATAFRAME AS CSV (SAVING THE FIRST 10 ROWS RETRIEVED)

Code

```
df_temp=df2[:10]
df_temp.to_csv("result.csv",index=False)
```

Output



OPEN AN XLS FILE AND SAVE IT AS CSV

Code

```
df3=pd.read_excel("result2.xls")
df3
df3.to_csv("result3.csv",index=False)
```

Output

Open an xls file and save it as CSV

In [9]:
```
df3=pd.read_excel("result2.xls")
df3
```

Out[9]:

	Unnamed: 0	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
0	1	Addam Marbrand	Lannister	NaN	NaN	NaN	56.0	1	1	1	1	1	1	0
1	2	Aegon Frey (Jinglebell)	None	299.0	3.0	51.0	49.0	1	1	0	0	1	0	0
2	3	Aegon Targaryen	House Targaryen	NaN	NaN	NaN	5.0	1	1	0	0	0	0	1
3	4	Adrack Humble	House Greyjoy	300.0	5.0	20.0	20.0	1	1	0	0	0	0	1
4	5	Aemon Costayne	Lannister	NaN	NaN	NaN	NaN	1	1	0	0	1	0	0
5	6	Aemon Estermont	Baratheon	NaN	NaN	NaN	NaN	1	1	0	1	1	0	0
6	7	Aemon Targaryen (son of Maekar I)	Night's Watch	300.0	4.0	35.0	21.0	1	1	1	0	1	1	0
7	8	Aenys Frey	None	300.0	5.0	NaN	59.0	0	1	1	1	1	0	1
8	9	Aeron Greyjoy	House Greyjoy	NaN	NaN	NaN	11.0	1	1	0	1	0	1	0
9	10	Aethan	Night's Watch	NaN	NaN	NaN	0.0	1	0	0	0	1	0	0

In [10]:
```
df3.to_csv("result3.csv",index=False)
```

SETTING COLUMN AS AN INDEX

Code

```
df=df2.copy(deep=True)
df.set_index("Gender",inplace=True)
df
```

Output

In [11]:
```
df=df2.copy(deep=True)
df.set_index("Gender",inplace=True)
df
```

Out[11]:

Gender	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Nobility	GoT	CoK	SoS	FfC	DwD
1	Addam Marbrand	Lannister	NaN	NaN	NaN	56	1	1	1	1	1	0
1	Aegon Frey (Jinglebell)	None	299	3	51	49	1	0	0	1	0	0
1	Aegon Targaryen	House Targaryen	NaN	NaN	NaN	5	1	0	0	0	0	1
1	Adrack Humble	House Greyjoy	300	5	20	20	1	0	0	0	0	1
1	Aemon Costayne	Lannister	NaN	NaN	NaN	NaN	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1	Zollo	None	NaN	NaN	NaN	21	0	0	0	1	0	0
1	Yurkhaz zo Yunzak	None	300	5	59	47	0	0	0	0	0	1
1	Yezzan Zo Qaggaz	None	300	5	57	25	1	0	0	0	0	1
1	Torwynd the Tame	Wildling	300	5	73	73	0	0	0	1	0	0
1	Talbert Serry	Tyrell	300	4	29	29	1	0	0	0	1	0

917 rows × 12 columns

INFORMATION ABOUT THE DATAFRAME

Code

df2.info
type(df2)

Output

```
In [12]: df2.info

Out[12]: <bound method DataFrame.info of 0                    Name      Allegiances Death Year Book of Death  \
1            Addam Marbrand        Lannister        NaN           NaN
2    Aegon Frey (Jinglebell)             None        299             3
3          Aegon Targaryen  House Targaryen        NaN           NaN
4            Adrack Humble    House Greyjoy        300             5
5            Aemon Costayne        Lannister        NaN           NaN
..                      ...              ...        ...           ...
913                  Zollo             None        NaN           NaN
914       Yurkhaz zo Yunzak             None        300             5
915        Yezzan Zo Qaggaz             None        300             5
916        Torwynd the Tame         Wildling        300             5
917           Talbert Serry           Tyrell        300             4

0   Death Chapter Book Intro Chapter Gender Nobility GoT CoK SoS FfC DwD
1             NaN                56      1        1   1   1   1   1   0
2              51                49      1        1   0   0   1   0   0
3             NaN                 5      1        1   0   0   0   0   1
4              20                20      1        1   0   0   0   0   1
5             NaN                NaN     1        1   0   0   1   0   0
..            ...                ...    ...      ...  ..  ..  ..  ..  ..
913           NaN                21      1        0   0   0   1   0   0
914            59                47      1        0   0   0   0   0   1
915            57                25      1        1   0   0   0   0   1
916            73                73      1        0   0   0   1   0   0
917            29                29      1        1   0   0   0   1   0

[917 rows x 13 columns]>

In [13]: type(df2)

Out[13]: pandas.core.frame.DataFrame
```

APPENDING DATA TO DATAFRAME

Code

df=df2[-10:]
df2=df2.append(df) #appending the last 10 rows of the dataframe to itself, thus reatng duplicate rows
df2[-20:]
df2.shape

Output

```
In [14]: df=df2[-10:]
         df2=df2.append(df) #appending the last 10 rows of the dataframe to itself, thus reatng duplicate rows
         df2[-20:]
```

Out[14]:

	Name	Allegiances	Death Year	Book of Death	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
908	Yohn Royce	Arryn	NaN	NaN	NaN	29	1	1	1	0	0	1	0
909	Yoren	Night's Watch	299	2	19	13	1	0	1	1	0	0	0
910	Young Henly	Night's Watch	299	3	55	55	1	0	0	0	1	0	0
911	Ysilla	House Targaryen	NaN	NaN	NaN	8	1	0	0	0	0	0	1
912	Zei	Stark	NaN	NaN	NaN	64	0	0	0	0	1	0	0
913	Zollo	None	NaN	NaN	NaN	21	1	0	0	0	1	0	0
914	Yurkhaz zo Yunzak	None	300	5	59	47	1	0	0	0	0	0	1
915	Yezzan Zo Qaggaz	None	300	5	57	25	1	1	0	0	0	0	1
916	Torwynd the Tame	Wildling	300	5	73	73	1	0	0	0	1	0	0
917	Talbert Serry	Tyrell	300	4	29	29	1	1	0	0	0	1	0
908	Yohn Royce	Arryn	NaN	NaN	NaN	29	1	1	1	0	0	1	0
909	Yoren	Night's Watch	299	2	19	13	1	0	1	1	0	0	0
910	Young Henly	Night's Watch	299	3	55	55	1	0	0	0	1	0	0
911	Ysilla	House Targaryen	NaN	NaN	NaN	8	1	0	0	0	0	0	1
912	Zei	Stark	NaN	NaN	NaN	64	0	0	0	0	1	0	0
913	Zollo	None	NaN	NaN	NaN	21	1	0	0	0	1	0	0
914	Yurkhaz zo Yunzak	None	300	5	59	47	1	0	0	0	0	0	1
915	Yezzan Zo Qaggaz	None	300	5	57	25	1	1	0	0	0	0	1
916	Torwynd the Tame	Wildling	300	5	73	73	1	0	0	0	1	0	0
917	Talbert Serry	Tyrell	300	4	29	29	1	1	0	0	0	1	0

```
In [15]: df2.shape
Out[15]: (927, 13)
```

DROPPING DUPLICATE ROWS (THE REPEATING ROWS ADDED ABOVE)

Code

df2.drop_duplicates(inplace=True)
df2.shape

Output

```
In [16]: df2.drop_duplicates(inplace=True)
         df2.shape
Out[16]: (917, 13)
```

RENAMING COLUMN NAMES (RENAMING 'BOOK OF DEATH' AS 'BOD')

Code

```
print("Column names: ",df2.columns)
df2.rename(columns={'Book of Death':'BoD'},inplace=True)
print(df2.columns)
```

Output

```
In [17]: print("Column names: ",df2.columns)

Column names:  Index(['Name', 'Allegiances', 'Death Year', 'Book of Death', 'Death Chapter',
       'Book Intro Chapter', 'Gender', 'Nobility', 'GoT', 'CoK', 'SoS', 'FfC',
       'DwD'],
      dtype='object', name=0)

In [18]: df2.rename(columns={'Book of Death':'BoD'},inplace=True)
         print(df2.columns)

Index(['Name', 'Allegiances', 'Death Year', 'BoD', 'Death Chapter',
       'Book Intro Chapter', 'Gender', 'Nobility', 'GoT', 'CoK', 'SoS', 'FfC',
       'DwD'],
      dtype='object', name=0)
```

DESCRIBING THE DATAFRAME

Code

```
df2.describe()
```

Output

```
In [19]: df2.describe()
```

Out[19]:

	Name	Allegiances	Death Year	BoD	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
count	917	917	305	307	299	905	917	917	917	917	917	917	917
unique	916	21	4	5	71	79	2	2	2	2	2	2	2
top	Myles	None	299	3	34	0	1	0	0	0	0	0	0
freq	2	253	156	97	16	41	760	487	667	593	528	667	656

COUNTING UNIQUE VALUES

Code

```
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())
```

Output

```
In [20]: 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']
```

ACCESSING ROWS AND COLUMNS (USING LOC AND ILOC)

Code

```
print("Row 0 with iloc[0]: \n",df2.iloc[0]) #Row 0 with iloc[0]
print("Rows 4 and 5 with columns 1,2 and 3\n",df2.iloc[[4,5],[1,2,3]]) #columns 1,2,3 of rows 4 and 5

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

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

Output

iloc

```
In [21]: print("Row 0 with iloc[0]: \n",df2.iloc[0]) #Row 0 with iloc[0]
         print("Rows 4 and 5 with columns 1,2 and 3\n",df2.iloc[[4,5],[1,2,3]]) #columns 1,2,3 of rows 4 and 5
```

```
Row 0 with iloc[0]:
 0
Name                   Addam Marbrand
Allegiances                 Lannister
Death Year                        NaN
BoD                               NaN
Death Chapter                     NaN
Book Intro Chapter                 56
Gender                              1
Nobility                            1
GoT                                 1
CoK                                 1
SoS                                 1
FfC                                 1
DwD                                 0
Name: 1, dtype: object
Rows 4 and 5 with columns 1,2 and 3
 0 Allegiances Death Year  BoD
5   Lannister        NaN  NaN
6   Baratheon        NaN  NaN
```

```
In [22]: print("First five values of 'Allegiances' column\t:\n",df2.iloc[0:5,0])
```

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

loc

```
In [23]: df2.set_index('Name',inplace=True)
```

```
In [24]: print("\n\nUsing loc method:\t",df2.loc['Adrack Humble','Death Year'])
```

```
Using loc method:        300
```

CREATING SUBSETS USING RELATIONAL OPERATORS (RETRIEVING THOSE WITH DEATH YEAR < 400)

Code

```
df3=df2.copy(deep=True)
df3['Death Year']=df3['Death Year'].astype(float)
df3 = df3[df3['Death Year']<400]
print(df3)
```

Output

```
In [25]: df3=df2.copy(deep=True)
         df3['Death Year']=df3['Death Year'].astype(float)

In [26]: df3 = df3[df3['Death Year']<400]
         print(df3)
```

```
0                                  Allegiances  Death Year BoD  \
Name
Aegon Frey (Jinglebell)                   None       299.0   3
Adrack Humble                   House Greyjoy       300.0   5
Aemon Targaryen (son of Maekar I)  Night's Watch       300.0   4
Aenys Frey                                None       300.0   5
Aggar                           House Greyjoy       299.0   2
...                                       ...         ...  ..
Young Henly                     Night's Watch       299.0   3
Yurkhaz zo Yunzak                         None       300.0   5
Yezzan Zo Qaggaz                          None       300.0   5
Torwynd the Tame                     Wildling       300.0   5
Talbert Serry                          Tyrell       300.0   4

0                                  Death Chapter Book Intro Chapter Gender  \
Name
Aegon Frey (Jinglebell)                      51                 49      1
Adrack Humble                                20                 20      1
Aemon Targaryen (son of Maekar I)            35                 21      1
Aenys Frey                                  NaN                 59      0
Aggar                                        56                 50      1
...                                         ...                ...    ...
Young Henly                                  55                 55      1
Yurkhaz zo Yunzak                            59                 47      1
Yezzan Zo Qaggaz                             57                 25      1
Torwynd the Tame                             73                 73      1
Talbert Serry                                29                 29      1

0                                  Nobility GoT CoK SoS FfC DwD
Name
Aegon Frey (Jinglebell)                   1   0   0   1   0   0
Adrack Humble                             1   0   0   0   0   1
Aemon Targaryen (son of Maekar I)         1   1   0   1   1   0
Aenys Frey                                1   1   1   1   0   1
Aggar                                     0   0   1   0   0   0
...                                     ...  ..  ..  ..  ..  ..
Young Henly                               0   0   0   1   0   0
Yurkhaz zo Yunzak                         0   0   0   0   0   1
Yezzan Zo Qaggaz                          1   0   0   0   0   1
Torwynd the Tame                          0   0   0   1   0   0
Talbert Serry                             1   0   0   0   1   0

[305 rows x 12 columns]
```

USING GROUPBY (GROUPING DATA BY 'DEATH YEAR')

Code

```
import numpy as np
df5 = df2.groupby('Death Year').count()
df5
```

Output

```
In [28]: import numpy as np

In [29]: df5 = df2.groupby('Death Year').count()
         df5
```

Out[29]:

Death Year	Allegiances	BoD	Death Chapter	Book Intro Chapter	Gender	Nobility	GoT	CoK	SoS	FfC	DwD
297	3	3	3	3	3	3	3	3	3	3	3
298	46	46	46	46	46	46	46	46	46	46	46
299	156	156	154	154	156	156	156	156	156	156	156
300	100	100	92	97	100	100	100	100	100	100	100

USING AGGREGATE WITH GROUPBY

Code

```
df3=df2.copy(deep=True)
df3.fillna(0)
df3['Gender']=df3['Gender'].astype(int)
df3['Death Year']=df3['Death Year'].astype(float)
df5 = df3.groupby("Death Year").agg({"Death Year":np.sum,"Gender":np.sum})
df5
```

Output

```
In [30]: df3=df2.copy(deep=True)
         df3.fillna(0)
         df3['Gender']=df3['Gender'].astype(int)
         df3['Death Year']=df3['Death Year'].astype(float)
         df5 = df3.groupby("Death Year").agg({"Death Year":np.sum,"Gender":np.sum})
         df5
```

Out[30]:

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

VISUALIZING THE DATA

Code

```
import matplotlib.pyplot as plt
df5.plot(x='Death Year',y='Gender',kind='scatter')
plt.show()
```

Output

```
In [35]: import matplotlib.pyplot as plt
         df5.plot(x='Death Year',y='Gender',kind='scatter')
Out[35]: <AxesSubplot:xlabel='Death Year', ylabel='Gender'>

In [36]: plt.show()
```