**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)

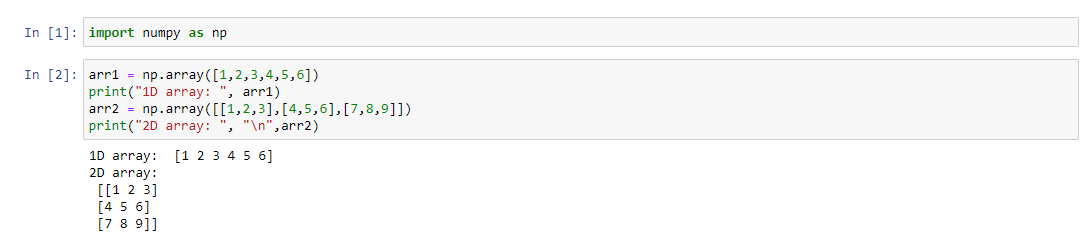
print("Array created with arange: \t\n",arr8)

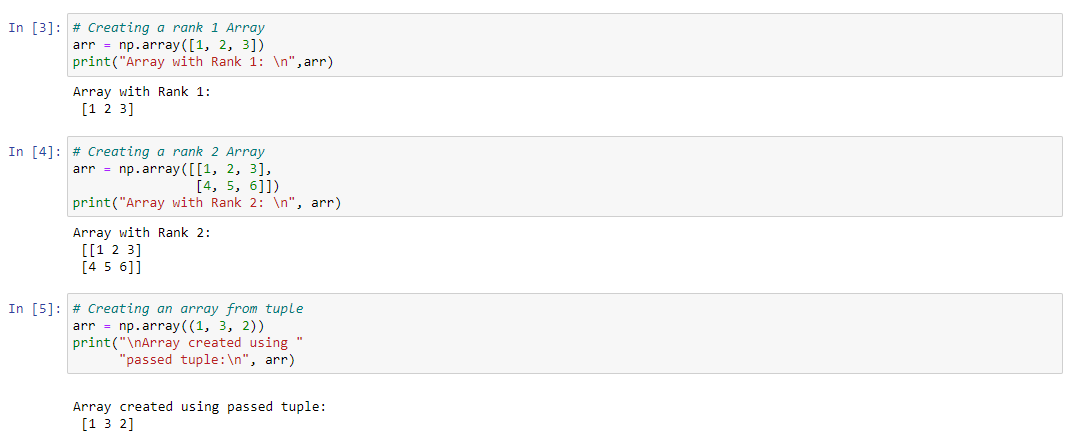
#Using linspace

arr9 = np.linspace(1,10,50)

print("Array created using linspace requesting 50 elements within 1 to 10:\t\n",arr9)

**Output**

****

****

**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)

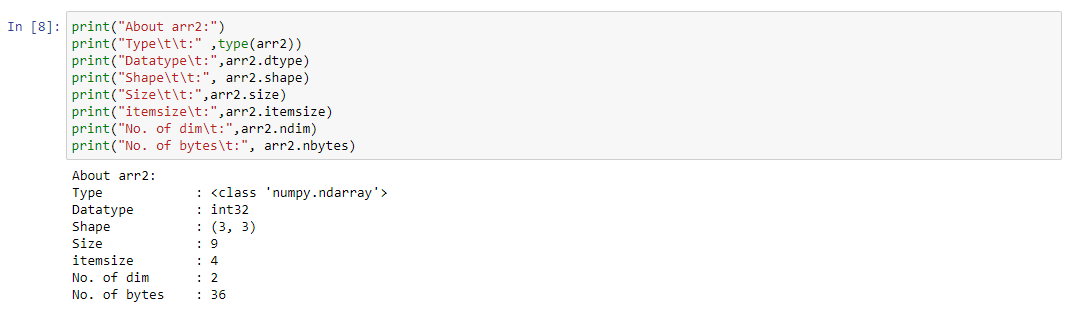
print("Size\t\t:",arr2.size)

print("itemsize\t:",arr2.itemsize)

print("No. of dim\t:",arr2.ndim)

print("No. of bytes\t:", arr2.nbytes)

**Output**

****

**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)

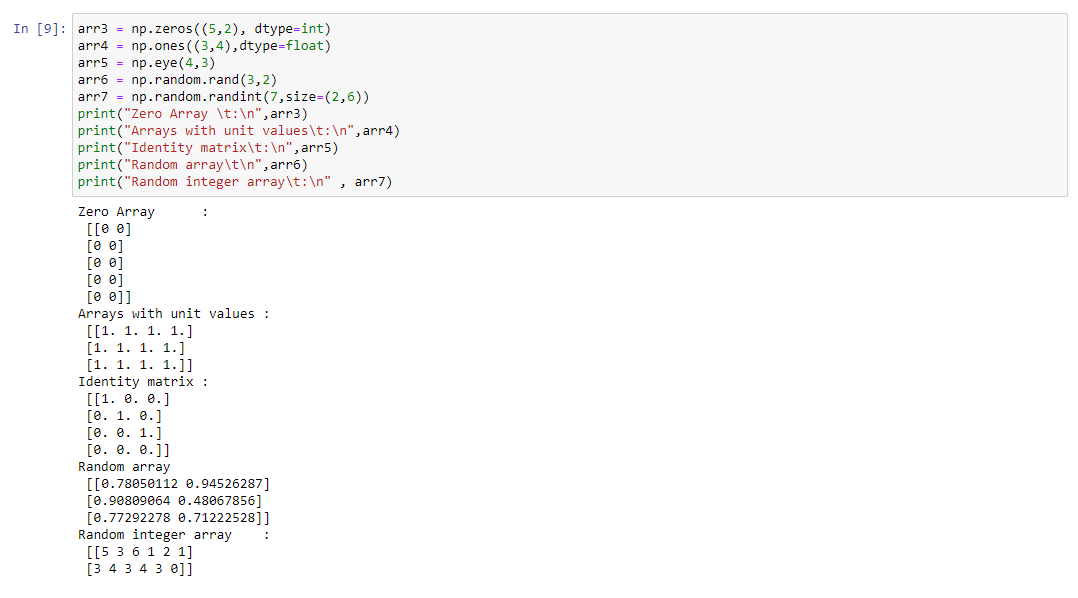
print("Arrays with unit values\t:\n",arr4)

print("Identity matrix\t:\n",arr5)

print("Random array\t\n",arr6)

print("Random integer array\t:\n" , arr7)

**Output**

****

**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,:])

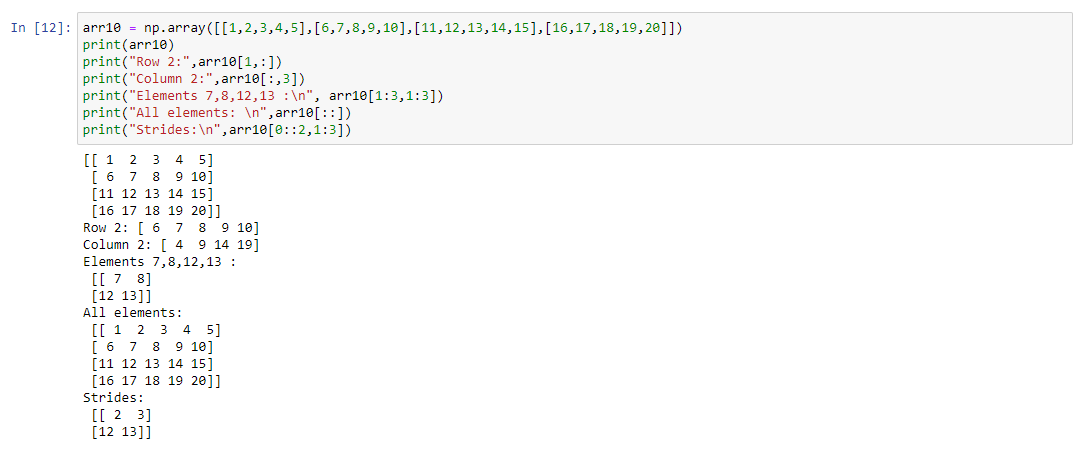
print("Column 2:",arr10[:,3])

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**

****

**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**

****

**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**

****

**TRIGONOMETRIC OPERATIONS**

**Code**

arr17 = np.array([15,30,45,90])

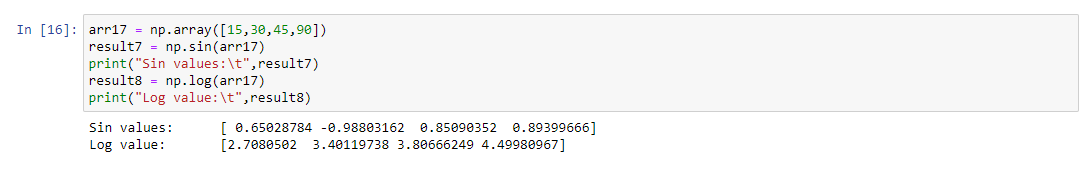
result7 = np.sin(arr17)

print("Sin values:\t",result7)

result8 = np.log(arr17)

print("Log value:\t",result8)

**Output**

****

**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**



**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']}

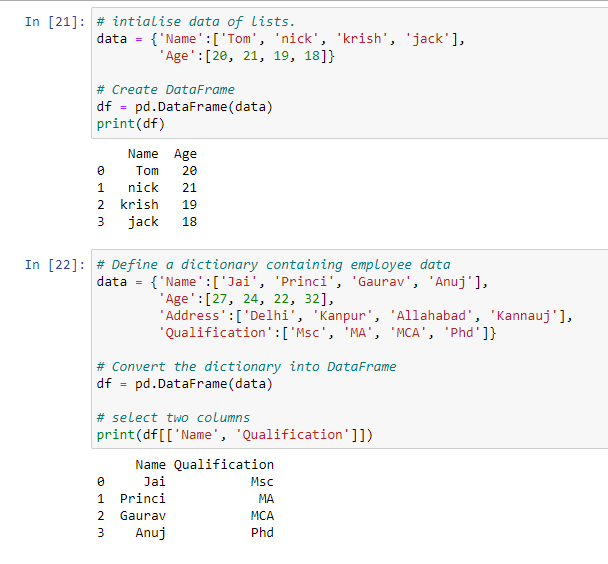
# Convert the dictionary into DataFrame

df = pd.DataFrame(data)

# select two columns

print(df[['Name', 'Qualification']])

**Output**

****

**CREATE DATAFRAME FROM CSV FILE**

**Code**

data = pd.read\_csv("nba.csv", index\_col ="Name")

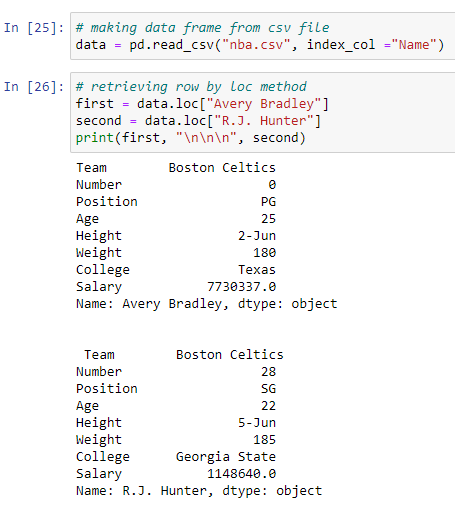
# retrieving row by loc method

first = data.loc["Avery Bradley"]

second = data.loc["R.J. Hunter"]

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

**Output**



**RETRIEVAL OF DATA FROM DATAFRAME**

**Code**

# retrieving columns by indexing operator

first = data["Age"]

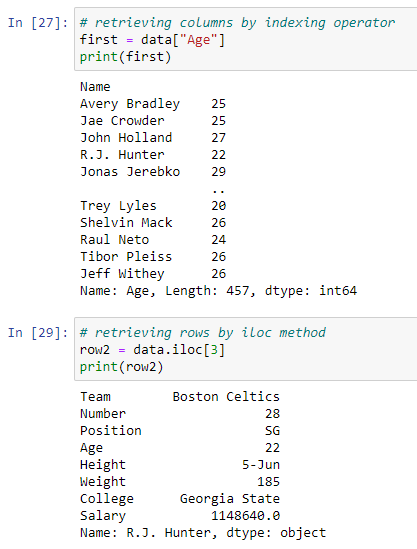
print(first)

# retrieving rows by iloc method

row2 = data.iloc[3]

print(row2)

**Output**



**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],

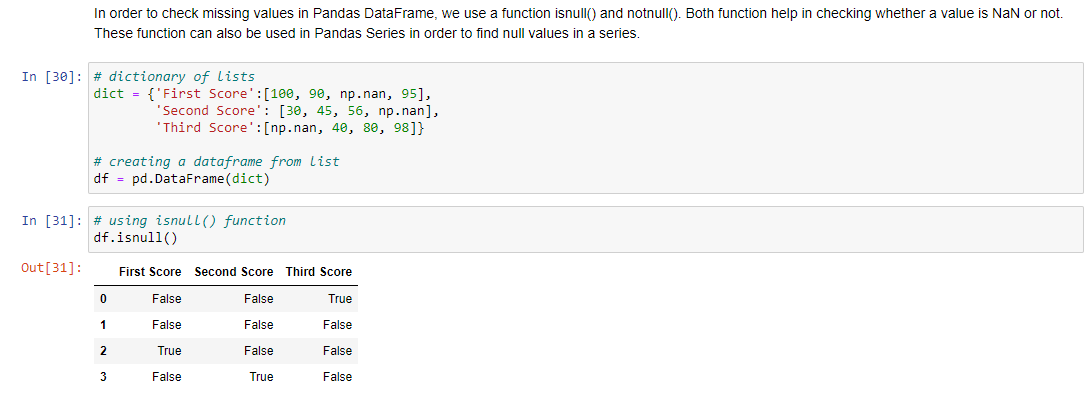
'Fourth Score':[np.nan, np.nan, np.nan, 65]}

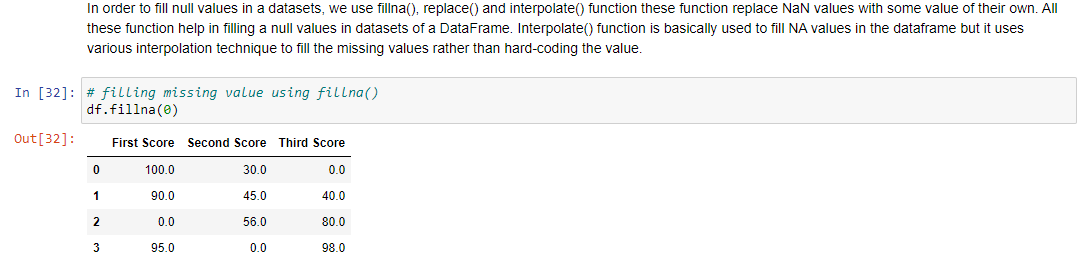
# creating a dataframe from dictionary

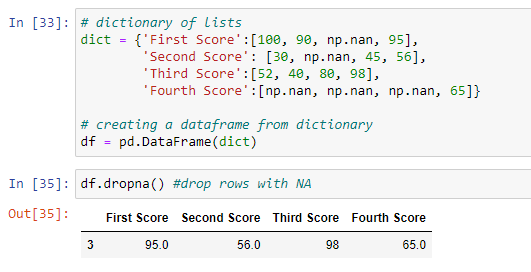
df = pd.DataFrame(dict)

df.dropna() #drop rows with NA

**Output**

****

****

****

**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)

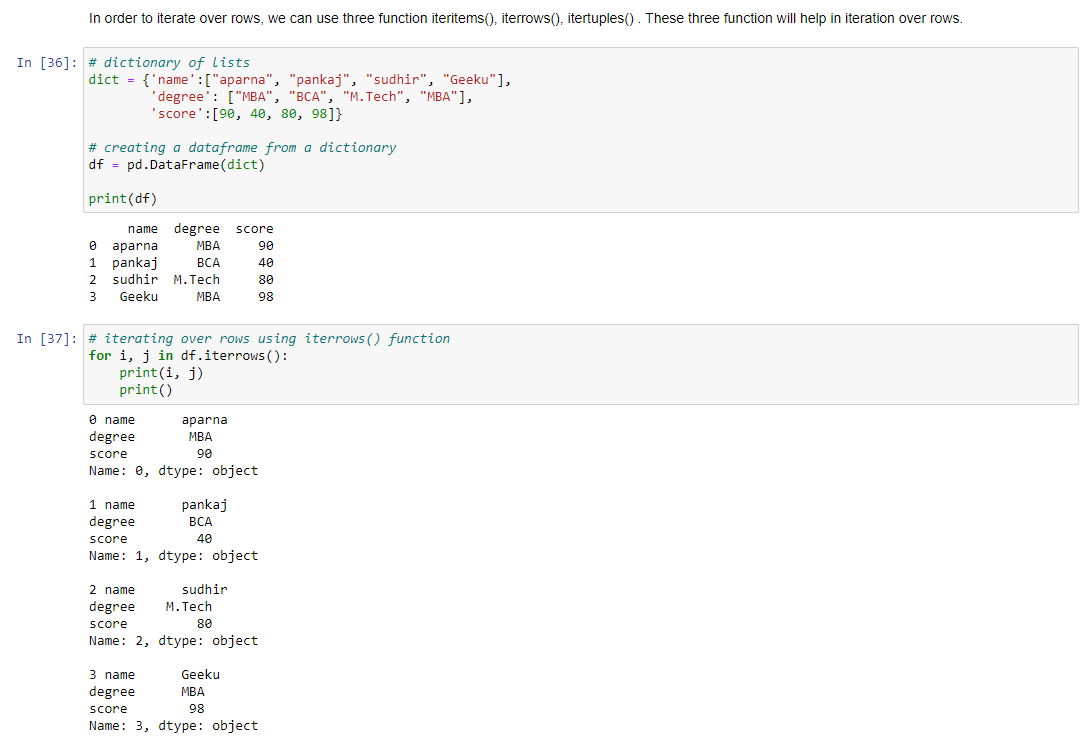
# iterating over rows using iterrows() function

for i, j in df.iterrows():

print(i, j)

print()

**Output**

****

**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

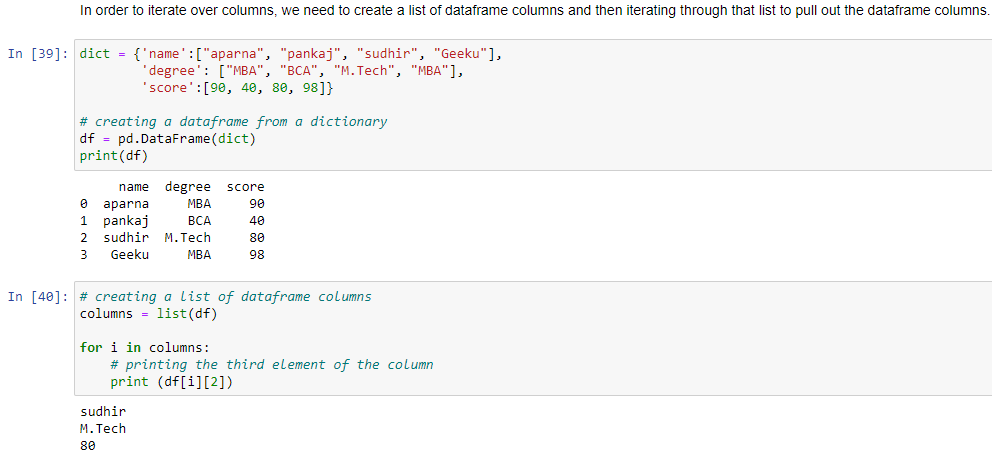
columns = list(df)

for i in columns:

# printing the third element of the column

print (df[i][2])

**Output**



**WORKING WITH NBA DATASET**

**READING THE CSV AND DISPLAY THE HEAD AND TAIL**

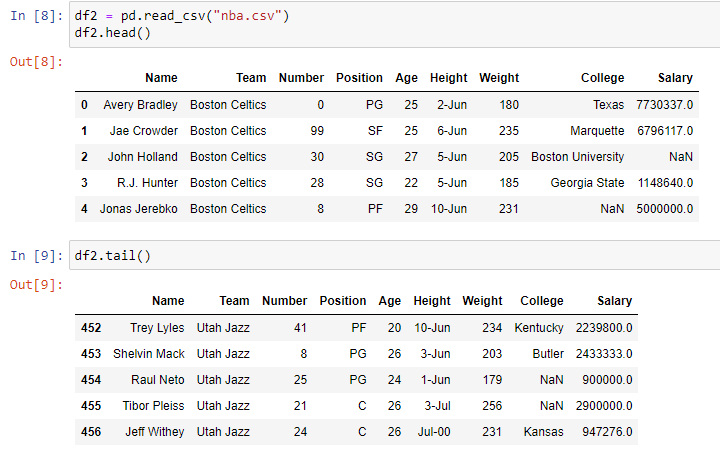
**Code**

df2 = pd.read\_csv("nba.csv")

df2.head()

df2.tail()

**Output**



**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)

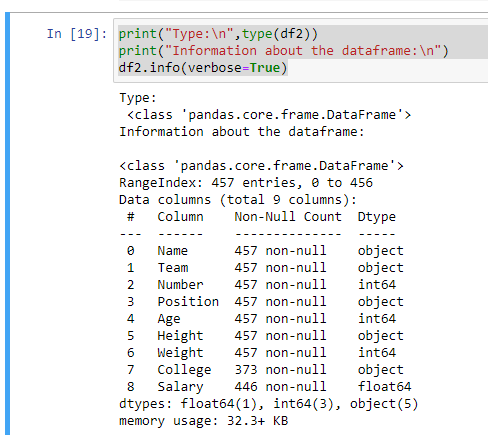
df2.drop\_duplicates()

print(df2.shape)

print("Columns of dataframe:\t",df2.columns)

print(df2.describe())

**Output**



**DROPPING DUPLICATE ROWS AND DISPLAYING A DESCRIPTION OF THE DATASET**

**Code**

print("Shape of dataframe:",df2.shape)

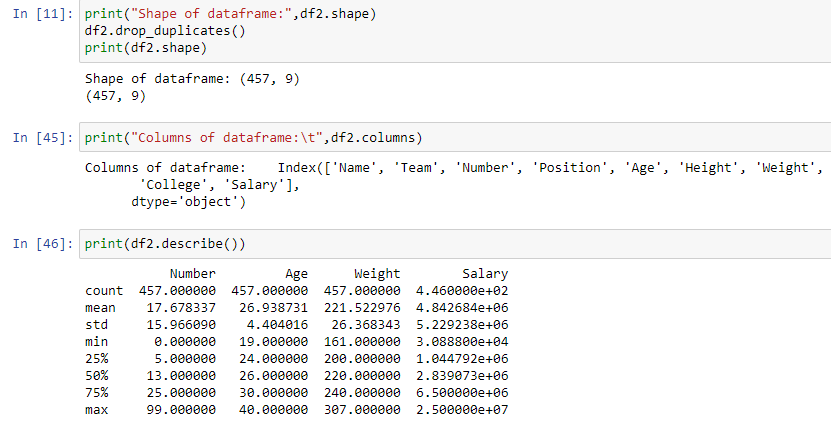
df2.drop\_duplicates()

print(df2.shape)

print("Columns of dataframe:\t",df2.columns)

print(df2.describe())

**Output**



**EXERCISE 3: PANDAS (CHARACTER DEATH DATA)**

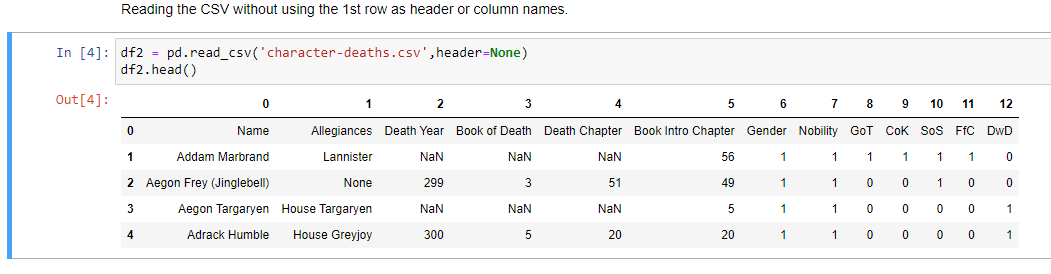
**REMOVING THE HEADER**

**Code**

df2 = pd.read\_csv('character-deaths.csv',header=None)

df2.head()

**Output**



**ADDING THE HEADER BACK**

**Code**

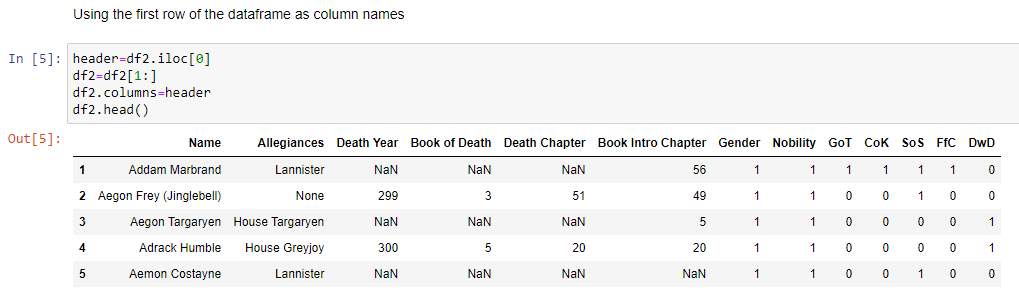
header=df2.iloc[0]

df2=df2[1:]

df2.columns=header

df2.head()

**Output**

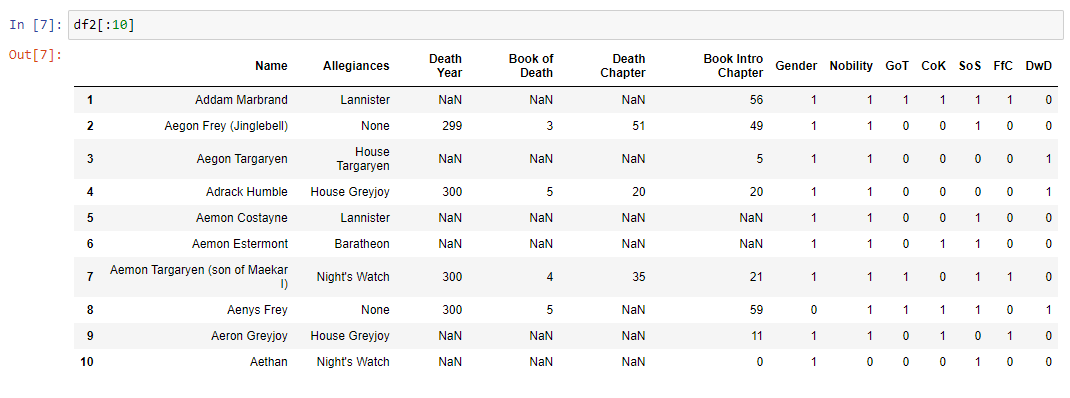


**DISPLAYING THE FIRST 10 ROWS**

**Code**

df2[:10]

**Output**



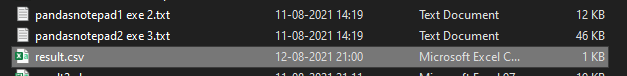
**SAVING THE MODIFIED DATAFRAME AS CSV (SAVING THE FIRST 10 ROWS RETRIEVED)**

**Code**

df\_temp=df2[:10]

df\_temp.to\_csv("result.csv",index=False)

**Output**

****

**OPEN AN XLS FILE AND SAVE IT AS CSV**

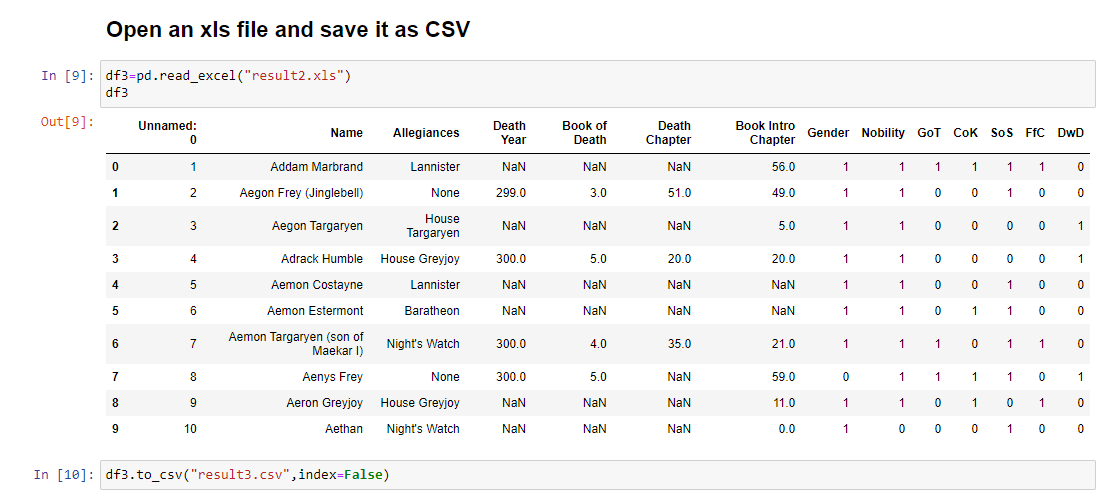
**Code**

df3=pd.read\_excel("result2.xls")

df3

df3.to\_csv("result3.csv",index=False)

**Output**



**SETTING COLUMN AS AN INDEX**

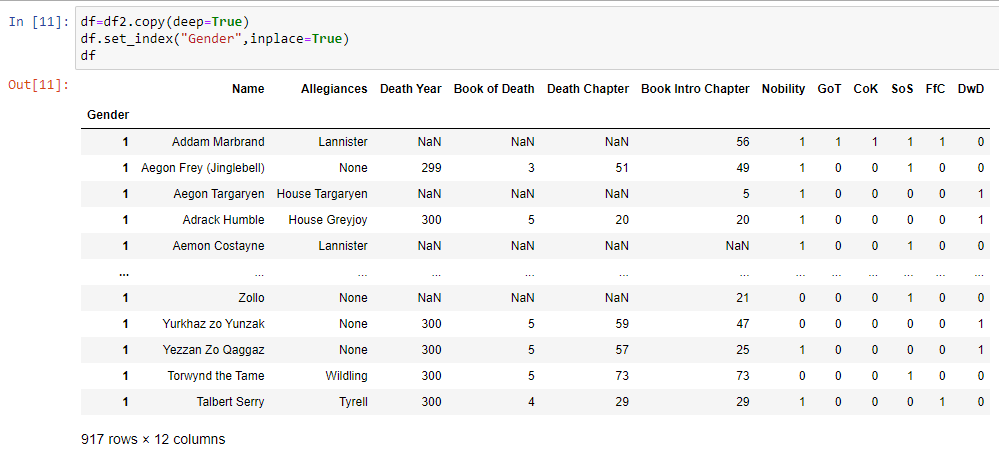
**Code**

df=df2.copy(deep=True)

df.set\_index("Gender",inplace=True)

df

**Output**



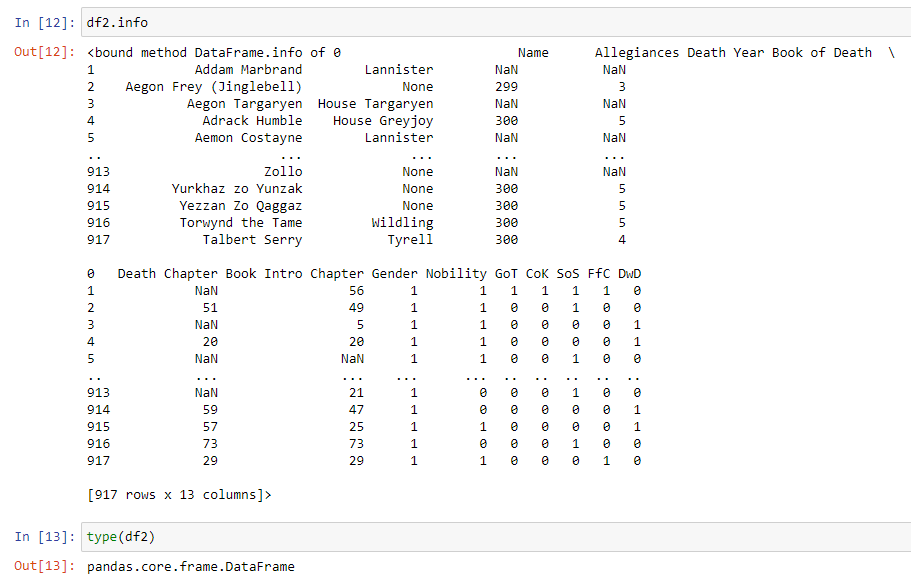
**INFORMATION ABOUT THE DATAFRAME**

**Code**

df2.info

type(df2)

**Output**

****

**APPENDING DATA TO DATAFRAME**

**Code**

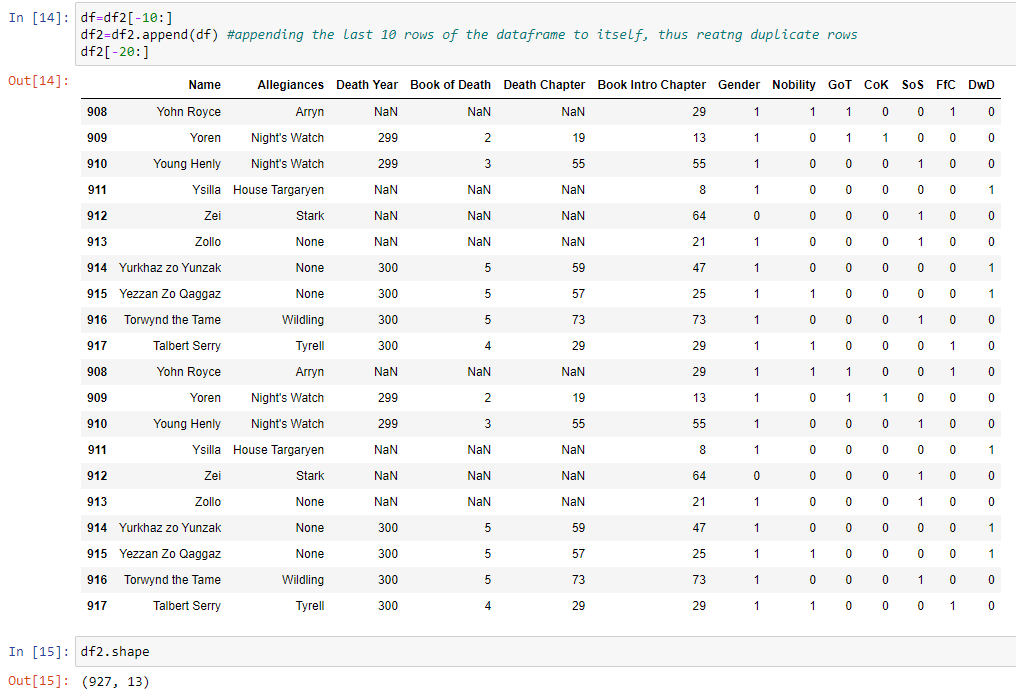
df=df2[-10:]

df2=df2.append(df) #appending the last 10 rows of the dataframe to itself, thus reatng duplicate rows

df2[-20:]

df2.shape

**Output**



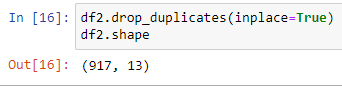
**DROPPING DUPLICATE ROWS (THE REPEATING ROWS ADDED ABOVE)**

**Code**

df2.drop\_duplicates(inplace=True)

df2.shape

**Output**



**RENAMING COLUMN NAMES (RENAMING ‘BOOK OF DEATH’ AS ‘BOD’)**

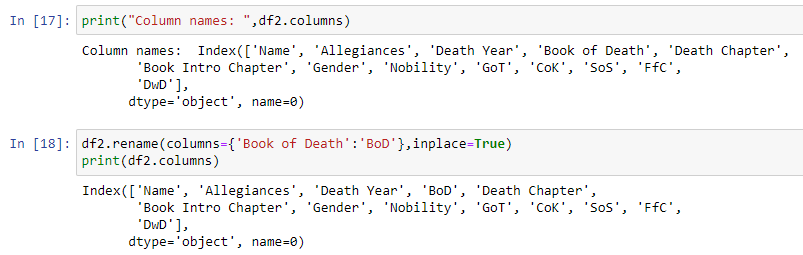
**Code**

print("Column names: ",df2.columns)

df2.rename(columns={'Book of Death':'BoD'},inplace=True)

print(df2.columns)

**Output**

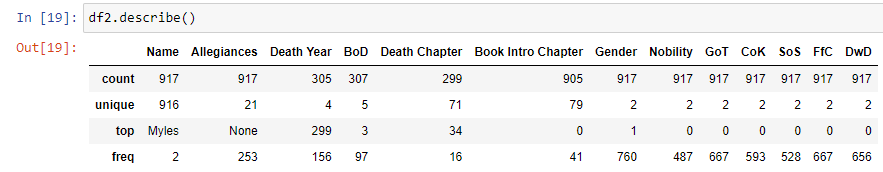


**DESCRIBING THE DATAFRAME**

**Code**

df2.describe()

**Output**



**COUNTING UNIQUE VALUES**

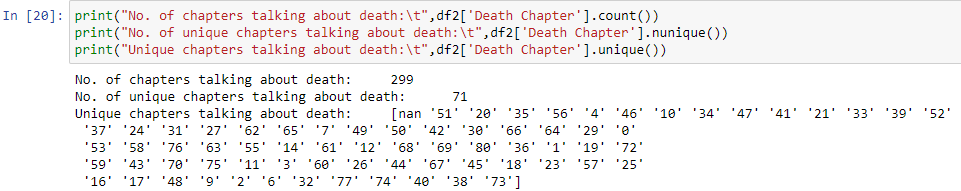
**Code**

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**

****

**ACCESSING ROWS AND COLUMNS (USING LOC AND ILOC)**

**Code**

print("Row 0 with iloc[0]: \n",df2.iloc[0]) #Row 0 with iloc[0]

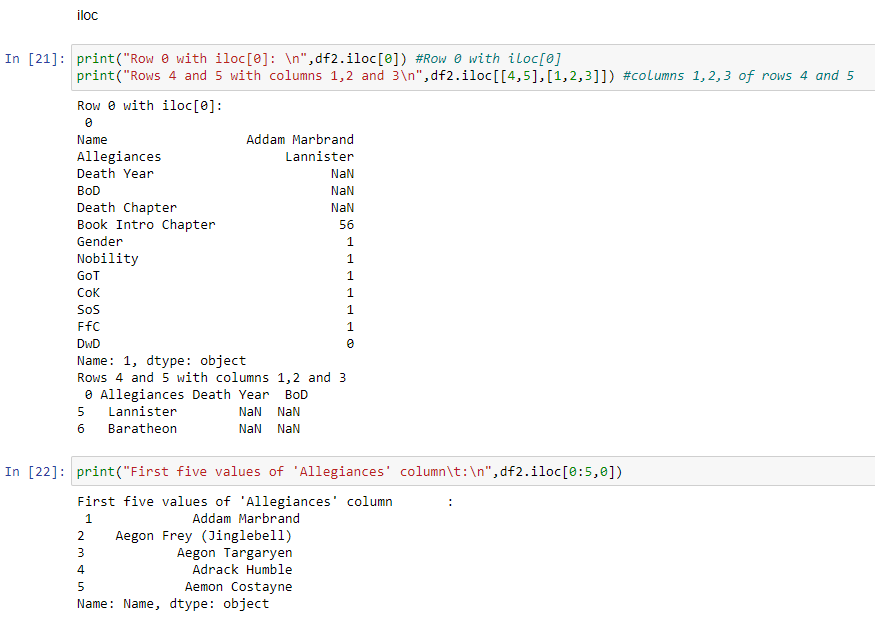
print("Rows 4 and 5 with columns 1,2 and 3\n",df2.iloc[[4,5],[1,2,3]]) #columns 1,2,3 of rows 4 and 5

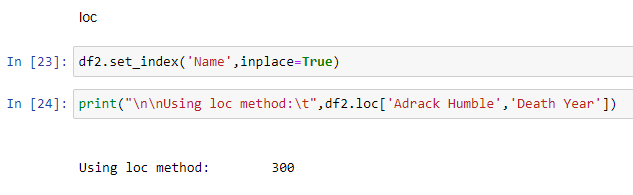
print("First five values of 'Allegiances' column\t:\n",df2.iloc[0:5,0])

df2.set\_index('Name',inplace=True)

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

**Output**





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

**Code**

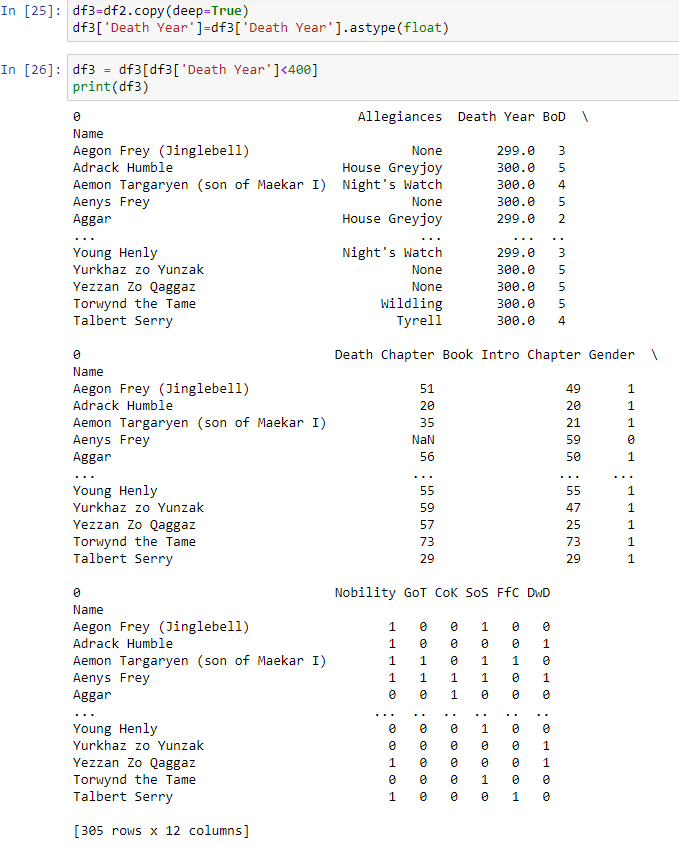
df3=df2.copy(deep=True)

df3['Death Year']=df3['Death Year'].astype(float)

df3 = df3[df3['Death Year']<400]

print(df3)

**Output**

****

**USING GROUPBY (GROUPING DATA BY ‘DEATH YEAR’)**

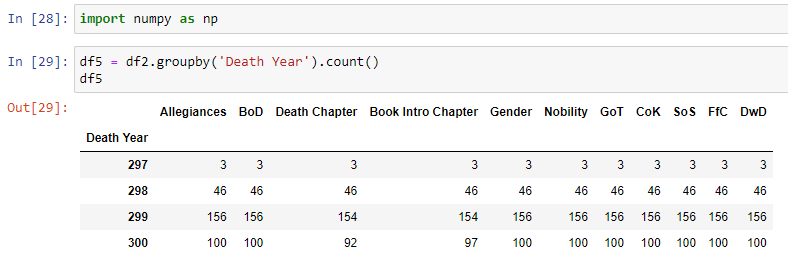
**Code**

import numpy as np

df5 = df2.groupby('Death Year').count()

df5

**Output**

****

**USING AGGREGATE WITH GROUPBY**

**Code**

df3=df2.copy(deep=True)

df3.fillna(0)

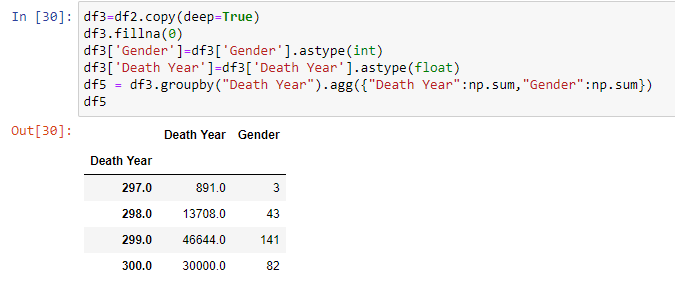
df3['Gender']=df3['Gender'].astype(int)

df3['Death Year']=df3['Death Year'].astype(float)

df5 = df3.groupby("Death Year").agg({"Death Year":np.sum,"Gender":np.sum})

df5

**Output**

****

**VISUALIZING THE DATA**

**Code**

import matplotlib.pyplot as plt

df5.plot(x='Death Year',y='Gender',kind='scatter')

plt.show()

**Output**

