Getting familiarity with pandas

Series

```
import pandas as pd
```

```
ser=pd.Series([3, 7, 14, 18, 21, 14])
ser
```

```
0     3
1     7
2    14
3    18
4    21
5    14
dtype: int64
```

```
ser=pd.Series([3, 7, 14, 18, 21, 14], index=['a', 'b', 'c', 'd', 'e',
'f'])
ser
```

```
a     3
b     7
c    14
d    18
e    21
f    14
dtype: int64
```

```
ser.index
```

```
Index(['a', 'b', 'c', 'd', 'e', 'f'], dtype='object')
```

```
ser.values
```

```
array([ 3,  7, 14, 18, 21, 14], dtype=int64)
```

```
ser.dtype
```

```
dtype('int64')
```

```
ser.head(1)
```

```
a    3
dtype: int64
```

```
ser.tail(1)
```

```
f    14
dtype: int64
```

Data Frame

```
data={
    'Name':['Jimmy', 'Kim', 'Gus', 'Mike', 'Victor'],
    'Age':[40, 35, 55, 58, 35],
    'Rank':['A', 'C', 'B', 'C', 'D']
}

df=pd.DataFrame(data)
df
```

```
     Name  Age Rank
0   Jimmy   40    A
1     Kim   35    C
2     Gus   55    B
3    Mike   58    C
4  Victor   35    D
```

```
df.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

```
df.columns
```

```
Index(['Name', 'Age', 'Rank'], dtype='object')
```

```
df.values
```

```
array([['Jimmy', 40, 'A'],
       ['Kim', 35, 'C'],
       ['Gus', 55, 'B'],
       ['Mike', 58, 'C'],
       ['Victor', 35, 'D']], dtype=object)
```

```
df.dtypes
```

```
Name    object
Age      int64
Rank    object
dtype: object
```

```
df.head(1)
```

```
    Name  Age Rank
0  Jimmy   40    A
```

```
df.tail(1)
```

```
     Name  Age Rank
4  Victor   35    D
```

```
df.shape
```

```
(5, 3)

df.size

15
```

Creating Series and DF's from different forms

```
arr=[13, 67, 18, 30, 10]
#This is a normal array

ser=pd.Series(arr)
ser

0    13
1    67
2    18
3    30
4    10
dtype: int64

data={
    'Name':['Jimmy', 'Kim', 'Gus', 'Mike', 'Victor'],
    'Age':[40, 35, 55, 58, 35],
    'Rank':['A', 'C', 'B', 'C', 'D']
}
df=pd.DataFrame(data)
df

     Name  Age Rank
0   Jimmy   40    A
1     Kim   35    C
2     Gus   55    B
3    Mike   58    C
4  Victor   35    D

ser1=pd.Series(df['Name'])
ser1

0     Jimmy
1       Kim
2       Gus
3      Mike
4    Victor
Name: Name, dtype: object

df1=pd.read_csv('students.csv')
df1

  Students Grades
0    Jimmy      A
```

```
1  Chandler      C
2    Kerion      B
```

```
ser1=pd.Series(df1['Grades'])
ser1
```

```
0    A
1    C
2    B
Name: Grades, dtype: object
```

Common operations on data frames

```
df['Age']
```

```
0    40
1    35
2    55
3    58
4    35
Name: Age, dtype: int64
```

```
df[['Name', 'Age']]
```

```
     Name  Age
0   Jimmy   40
1     Kim   35
2     Gus   55
3    Mike   58
4  Victor   35
```

```
df.loc[0]
```

```
Name    Jimmy
Age        40
Rank        A
Name: 0, dtype: object
```

```
df.loc[0:3]
```

```
    Name  Age Rank
0  Jimmy   40    A
1    Kim   35    C
2    Gus   55    B
3   Mike   58    C
```

```
#This can also be done using iloc
#iloc is used to locate the integer value of index if we dont the what
the index is
df.iloc[2]
```

```
Name     Gus
Age       55
Rank       B
Name: 2, dtype: object
```

```
#multiple rows can be accessed using iloc also
df.iloc[0:3]
```

```
    Name  Age Rank
0  Jimmy   40    A
1    Kim   35    C
2    Gus   55    B
```

```
df[df['Age']>50]
```

```
   Name  Age Rank
2   Gus   55    B
3  Mike   58    C
```

```
df[df['Rank']=='C']
```

```
   Name  Age Rank
1   Kim   35    C
3  Mike   58    C
```

```
df[(df['Age']>52) | (df['Rank']=='B')]
```

```
   Name  Age Rank
2   Gus   55    B
3  Mike   58    C
```

```
#we can add a new column to our data frame
df['DOB']=2024-df['Age']
df
```

```
     Name  Age Rank   DOB
0   Jimmy   40    A  1984
1     Kim   35    C  1989
2     Gus   55    B  1969
3    Mike   58    C  1966
4  Victor   35    D  1989
```

```
df['Name']=df['Name'].str.upper()
df
```

```
     Name  Age Rank   DOB
0   JIMMY   40    A  1984
1     KIM   35    C  1989
2     GUS   55    B  1969
3    MIKE   58    C  1966
4  VICTOR   35    D  1989
```

```
df=df.drop(columns=['DOB'])
df

     Name  Age Rank
0   JIMMY   40    A
1     KIM   35    C
2     GUS   55    B
3    MIKE   58    C
4  VICTOR   35    D

#To modify the excisting data
df.loc[2, 'Rank']='A'
df

     Name  Age Rank
0   JIMMY   40    A
1     KIM   35    C
2     GUS   55    A
3    MIKE   58    C
4  VICTOR   35    D
```

Data Handling

```
import pandas as pd
import numpy as np

# Sample DataFrame with missing values
data = {
    'Name': ['Ram', 'Bheem', 'Laxman', np.nan],
    'Age': [25, np.nan, 35, 40],
    'City': ['New York', 'Los Angeles', np.nan, 'Chicago']
}
df = pd.DataFrame(data)

# Check for missing values
missing_values = df.isna()

print(missing_values)

    Name    Age   City
0  False  False  False
1  False   True  False
2  False  False   True
3   True  False  False

count_missing_val=df.isna().sum()
count_missing_val

Name    1
Age     1
```

```
City    1
dtype: int64
```

```
# Drop rows with any missing values
df_dropped_rows = df.dropna()

# Drop columns with any missing values
df_dropped_cols = df.dropna(axis=1)

print(df_dropped_rows)
print(df_dropped_cols)
```

```
  Name   Age      City
0  Ram  25.0  New York
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3]
```

```
df_filled_value = df.fillna({'Name': 'Ravan', 'Age': df['Age'].mean(),
'City': 'Colomba'})

df_filled_value
```

```
     Name        Age         City
0     Ram  25.000000     New York
1   Bheem  33.333333  Los Angeles
2  Laxman  35.000000      Colomba
3   Ravan  40.000000      Chicago
```

Data transformation

```
# Sample DataFrame
df = pd.DataFrame({
    'Value': ['1', '2', '3', '4']
})
# Convert 'Value' to integer
df['Value'] = df['Value'].astype(int)
df
```

```
   Value
0      1
1      2
2      3
3      4
```

```
# Sample DataFrames
df1 = pd.DataFrame({'A': [1, 2], 'B': [3, 4]})
df2 = pd.DataFrame({'A': [5, 6], 'B': [7, 8]})

# Concatenate DataFrames vertically
```

```
df_concat = pd.concat([df1, df2])
df_concat
```

```
   A  B
0  1  3
1  2  4
0  5  7
1  6  8
```

```
# Sample DataFrames
df1 = pd.DataFrame({'ID': [1, 2, 3], 'Value': ['A', 'B', 'C']})
df2 = pd.DataFrame({'ID': [1, 2, 4], 'Description': ['X', 'Y', 'Z']})

# Merge DataFrames on 'ID'
df_merged = pd.merge(df1, df2, on='ID', how='inner')
df_merged
```

```
   ID Value Description
0   1     A           X
1   2     B           Y
```

Conclusion

Applications