Binary operations on Dataframes

Introduction

- In mathematics a binary operator or a dyadic operator is a function that combines two values to produce a new value. The binary operator function could perform addition, subtraction and so on to return the new value. Pandas DataFrame has several binary operator functions defined for combining two DataFrames. The binary operator functions return a new DataFrame as a result of combining the two DataFrames.
- The basic rule to be remembered is: the dataframe should have equal rows and columns.

Addition binary operation

- **Two DataFrames can be added** by using the **add()** method or the **radd()** pandas DataFrame class.
- Calling **add()** method is similar to calling the operator **+**. However, the **add()** method can be passed a fill value, which will be used for **NaN** values in the **DataFrame**.

```
import pandas as pd
 dataSet1 = [(10, 20, 30),(40, 50, 60),(70, 80, 90)];
dataFrame1 = pd.DataFrame(data=dataSet1)
dataSet2 = [(5, 15, 25),(35, 45, 55),(65, 75, 85)];
dataFrame2 = pd.DataFrame(data=dataSet2)
 print("DataFrame1:");
print(dataFrame1);
print("DataFrame2:");
print(dataFrame2);
 result = dataFrame1.add(dataFrame2);
print("Result of adding two pandas dataframes:");
print(result)
```

```
DataFrame1:
   0   1   2
0  10  20  30
1  40  50  60
2  70  80  90
DataFrame2:
   0   1   2
0   5  15  25
1  35  45  55
2  65  75  85
Result of adding two pandas dataframes:
    0    1    2
0   15   35   55
1   75   95  115
2  135  155  175
```

Subtracting the Dataframe

- The **sub()** method or **rsub()**of pandas **DataFrame** subtracts the elements of one **DataFrame** from the elements of another DataFrame.
- Invoking **sub()** method on a **DataFrame** object is equivalent to calling the binary subtraction operator(-).
- The **sub()** method supports passing a parameter for missing values(np.nan, None).

```
import pandas as pd
 data1 = [(2, 4, 6, 8),(1, 3, 5, 7),(5, 0, 0, 9)];
data2 = [(1, 1, 0, 1),(1, 0, 1, 1),(0, 1, 1, 0)];
dataFrame1 = pd.DataFrame(data=data1);
print("DataFrame1:");
print(dataFrame1);
dataFrame2 = pd.DataFrame(data=data2);
print("DataFrame2:");
print(dataFrame2);
 # Subtracting DataFrame2 from DataFrame1
subtractionResults = dataFrame1 - dataFrame2;
print("Result of subtracting dataFrame1 from dataFrame2:");
print(subtractionResults);
```

```
DataFrame1:
   0  1  2  3
0  2  4  6  8
1  1  3  5  7
2  5  0  0  9
DataFrame2:
   0  1  2  3
0  1  1  0  1
1  1  0  1  1
2  0  1  1  0
Result of subtracting dataFrame1 from
dataFrame2:
   0  1  2  3
0  1  3  6  7
1  0  3  4  6
2  5 -1 -1  9
```

Multiplication of Dataframes

- The **mul()** method of DataFrame object multiplies the elements of a DataFrame object with another DataFrame object, series or any other Python sequence.
- **mul()** does an elementwise multiplication of a DataFrame with another DataFrame, a pandas Series or a Python Sequence.
- Calling the **mul()** method or **rmul()** is similar to using the binary multiplication operator(*).
- The **mul()** method provides a parameter fill_value using which values can be passed to replace the np.nan, None values present in the data.

```
import pandas as pd
dataSet1 =  [(2, 0, 1, 0), (1, 0, 3, 0), (4, 3, 2, 0)];
dataSet2 =  [(1, 1, 2, 1), (1, 2, 1, 1), (3, 1, 1, 3)];
dataFrame1 = pd.DataFrame(data=dataSet1);
dataFrame2 = pd.DataFrame(data=dataSet2);
 print("Elements present in DataFrame1:");
print(dataFrame1);
print("Elements present in DataFrame2:");
print(dataFrame2);
multiplicationResults = dataFrame1.mul(dataFrame2);
print("Result of element-wise multiplication of two Data Frames:");
print(multiplicationResults);
```

```
Elements present in DataFrame1:
   0  1  2  3
0  2  0  1  0
1  1  0  3  0
2  4  3  2  0
Elements present in DataFrame2:
   0  1  2  3
0  1  1  2  1
1  1  2  1  1
2  3  1  1  3
Result of element-wise multiplication of
two Data Frames:
    0  1  2  3
0   2  0  2  0
1   1  0  3  0
2  12  3  2  0
```

Division of DataFrames

- **div()** method or **rdiv()**divides element-wise division of one pandas DataFrame by another.
- DataFrame elements can be divided by a pandas series or by a Python sequence as well.
- Calling **div()** on a DataFrame instance is equivalent to invoking the division operator (/).
- The **div()** method provides the fill_value parameter which is used for replacing the np.nan and None values present in the DataFrame or in the resultant value with any other value.

```
import pandas as pd
data1 = [(20, 40, 60, 80), (10, 30, 50, 70), (15, 25, 35, 45)];
 data2 = [(2, 3, 4, 6),(5, 1, 0, 3),(3, 6, 9, 1)];
dataFrame1 = pd.DataFrame(data=data1);
dataFrame2 = pd.DataFrame(data=data2);
divisionResults = dataFrame1.div(dataFrame2);
print("Elements of DataFrame1:")
print(dataFrame1);
print("Elements of DataFrame2:")
print(dataFrame2);
print("DataFrame1 elements divided by DataFrame2 elements:")
print(divisionResults);
```

```
Elements of DataFrame1:
    0   1   2   3
0  20  40  60  80
1  10  30  50  70
2  15  25  35  45
Elements of DataFrame2:
   0  1  2  3
0  2  3  4  6
1  5  1  0  3
2  3  6  9  1
DataFrame1 elements diveded by DataFrame2 elements:
      0          1          2          3
0  10.0  13.333333  15.000000  13.333333
1   2.0  30.000000        inf  23.333333
2   5.0   4.166667   3.888889  45.000000
```