Arithmetic Operations on Pandas Series

Just like with NumPy ndarrays, we can perform element-wise arithmetic operations on Pandas Series. In this lesson we will look at arithmetic operations between Pandas Series and single numbers. Let's create a new Pandas Series that will hold a grocery list of just fruits.

```
# We create a Pandas Series that stores a grocery list of just fruits

fruits= pd.Series(data = [10, 6, 3,], index = ['apples', 'oranges', 'bananas'])
```

```
# We display the fruits Pandas Series

fruits
```

```
apples      10
oranges      6
bananas      3
dtype: int64
```

We can now modify the data in fruits by performing basic arithmetic operations. Let's see some examples

Example 1. Element-wise basic arithmetic operations

```
# We print fruits for reference

print('Original grocery list of fruits:\n ', fruits)
```

```
# We perform basic element-wise operations using arithmetic symbols

print()

print('fruits + 2:\n', fruits + 2) # We add 2 to each item in fruits

print()

print('fruits - 2:\n', fruits - 2) # We subtract 2 to each item in fruits

print()

print('fruits * 2:\n', fruits * 2) # We multiply each item in fruits by 2

print()

print('fruits / 2:\n', fruits / 2) # We divide each item in fruits by 2

print()
```

```
Original grocery list of fruits:
apples      10
```

```
oranges       6
bananas       3
dtype: int64

fruits + 2:
apples        12
oranges       8
bananas       5
dtype: int64

fruits - 2:
apples         8
oranges       4
bananas       1
dtype: int64

fruits * 2:
apples        20
oranges      12
bananas       6
dtype: int64

fruits / 2:
apples         5.0
oranges       3.0
bananas       1.5
dtype: float64
```

You can also apply mathematical functions from NumPy, such assqrt(x), to all elements of a Pandas Series.

Example 2. Use mathematical functions from NumPy to operate on Series

```
# We import NumPy as np to be able to use the mathematical functions

import numpy as np


# We print fruits for reference

print('Original grocery list of fruits:\n', fruits)


# We apply different mathematical functions to all elements of fruits

print()

print('EXP(X) = \n', np.exp(fruits))
```

```
print()

print('SQRT(X) =\n', np.sqrt(fruits))

print()

print('POW(X,2) =\n',np.power(fruits,2)) # We raise all elements of fruits to the power of 2
```

```
Original grocery list of fruits:
apples        10
oranges        6
bananas        3
dtype: int64

EXP(X) =
apples       22026.465795
oranges         403.428793
bananas          20.085537
dtype: float64

SQRT(X) =
apples           3.162278
oranges          2.449490
bananas          1.732051
dtype: float64

POW(X,2) =
apples        100
oranges        36
bananas         9
dtype: int64
```

Pandas also allows us to only apply arithmetic operations on selected items in our fruits grocery list. Let's see some examples

Example 3. Perform arithmetic operations on selected elements

```
# We print fruits for reference

print('Original grocery list of fruits:\n ', fruits)

print()


# We add 2 only to the bananas

print('Amount of bananas + 2 = ', fruits['bananas'] + 2)

print()
```

```
# We subtract 2 from apples

print('Amount of apples - 2 = ', fruits.iloc[0] - 2)

print()


# We multiply apples and oranges by 2

print('We double the amount of apples and oranges:\n', fruits[['apples', 'oranges']] * 2)

print()


# We divide apples and oranges by 2

print('We half the amount of apples and oranges:\n', fruits.loc[['apples', 'oranges']] / 2)
```

```
Original grocery list of fruits:
apples        10
oranges        6
bananas        3
dtype: int64

Amount of bananas + 2 = 5

Amount of apples - 2 = 8

We double the amount of apples and oranges:
apples        20
oranges       12
dtype: int64

We half the amount of apples and oranges:
apples        5.0
oranges       3.0
dtype: float64
```

You can also apply arithmetic operations on Pandas Series of mixed data type provided that the arithmetic operation is defined for *all* data types in the Series, otherwise, you will get an error. Let's see what happens when we multiply our grocery list by 2

Example 4. Perform multiplication on a Series having integer and string elements

```
# We multiply our grocery list by 2

groceries * 2
```

```
eggs              60
apples            12
milk        YesYes
```

```
bread       NoNo
dtype: object
```

As we can see, in this case, since we multiplied by 2, Pandas doubles the data of each item including the strings. Pandas can do this because the multiplication operation * is defined both for numbers and strings. If you were to apply an operation that was valid for numbers but not strings, say for instance, / you will get an error. So when you have mixed data types in your Pandas Series make sure the arithmetic operations are valid on *all* the data types of your elements.