Арифметические операции

Со структурами данных *pandas* можно поэлементно проводить операции сложения, вычитания, умножения, деления.

Давайте создадим два объекта `Series` и на их примере рассмотрим арифметические операции.

```
srs1 = pd.Series([1, 2, 3])
srs2 = pd.Series([10, 20, 30, 40])
print(srs1, srs2, sep='\n\n')
```

Результат:

```
0    1
1    2
2    3
dtype: int64

0    10
1    20
2    30
3    40
dtype: int64
```

Сложение. Метод `add()`

Складывать объекты `Series` можно как с помощью оператора сложения `'+'`, так и при помощи метода **`add()`**.

Сложим две серии, используя оператор сложения `'+'`:

```
srs = srs1 + srs2

print(srs)
```

Результат:

```
0    11.0
1    22.0
2    33.0
3     NaN
dtype: float64
```

Арифметические операции двух и более объектов `Series` происходят поэлементно по индексам. В нашем примере объекты `Series` имеют разную размерность, поэтому элемент с индексом `3` получил значение `NaN`. Это произошло из-за того, что *pandas* в одном из объектов не нашел значения с индексом `3`.

Решить проблему с пропущенными значениями, при сложении структур `Series` разных размерностей, позволяет

использование метода `add()`, который для подобных случаев имеет в своем составе параметр `fill_value`.

Воспользуемся методом `add()`, пока без использования параметра `fill_value`:

```
srs1.add(srs2)
```

Результат:

```
0    11.0
1    22.0
2    33.0
3     NaN
dtype: float64
```

Параметру `fill_value` передается значение, которое "заполняет" пропуски, тем самым выравнивая ряды.

```
srs1.add(srs2, fill_value=0)
```

Результат:

```
0    11.0
1    22.0
2    33.0
3    40.0
dtype: float64
```

К элементам структуры также можно добавить скалярное значение:

```
srs1 + 100
```

Результат:

```
0    101
1    102
2    103
dtype: int64
```

Или, используя метод `add()`:

```
srs1.add(100)
```

Результат:

```
0    101
1    102
2    103
dtype: int64
```

Вычитание. Метод `sub()`

Вычитать один объект `Series` из другого можно как с помощью оператора вычитания `'-'`, так и с использованием метода **`sub()`**.

```
srs1 = pd.Series([1, 2, 3])
srs2 = pd.Series([10, 20, 30, 40])
print(srs1, srs2, sep='\n\n')
```

```
0    1
1    2
2    3
dtype: int64

0    10
1    20
2    30
3    40
dtype: int64
```

Вычтем серию `srs2` из `srs1`:

```
srs = srs1 - srs2

print(srs)
```

Результат:

```
0    -9.0
1   -18.0
2   -27.0
3     NaN
dtype: float64
```

Теперь проделаем ту же самую манипуляцию с использованием метода `sub()` с параметром `fill_value=0`:

```
srs1.sub(srs2, fill_value=0)
```

Результат:

```
0    -9.0
1   -18.0
2   -27.0
3   -40.0
dtype: float64
```

Умножение. Метод `mul()`

Произведение структур `Series` осуществляется или с использованием оператора умножения `'*'`, или при помощи метода **`mul()`**.

```
srs1 = pd.Series([1, 2, 3])
srs2 = pd.Series([10, 20, 30, 40])
print(srs1, srs2, sep='\n\
```

```
0    1
1    2
2    3
dtype: int64

0    10
1    20
2    30
3    40
dtype: int64
```

Произведение двух объектов `Series`:

```
srs = srs1 * srs2
print(srs)
```

Результат:

```
0    10.0
1    40.0
2    90.0
3     NaN
dtype: float64
```

С использованием метода `mul()` с параметром `fill_value=1`:

```
srs1.mul(srs2, fill_value=1)
```

Результат:

```
0    10.0
1    40.0
2    90.0
3    40.0
dtype: float64
```

Для **возведения в степень** используется оператор `'**'`:

```
srs1 ** 2
```

Результат:

```
0    1
1    4
2    9
dtype: int64
```

Деление. Метод `div()`

Разделить один объект `Series` на другой можно или при помощи оператора деления `'/'`, или с использованием метода **`div()`**.

```
srs1 = pd.Series([1, 2, 3])
srs2 = pd.Series([10, 20, 30, 40])
print(srs1, srs2, sep='\n\n')
```

```
0    1
1    2
2    3
dtype: int64

0    10
1    20
2    30
3    40
dtype: int64
```

Разделим серию `srs2` на `srs1`:

```
srs = srs2 / srs1
print(srs)
```

Результат:

```
0    10.0
1    10.0
2    10.0
3     NaN
dtype: float64
```

Проведем ту же самую манипуляцию, но с использованием метода `div()` с параметром `fill_value=1`:

```
srs2.div(srs1, fill_value=1)
```

Результат:

```
0    10.0
1    10.0
2    10.0
3    40.0
dtype: float64
```