

Создание объекта Series из NumPy-массива

Для начала создадим массив NumPy:

```
import numpy as np

arr = np.array([52.50, 1386.20, 214.20])

print(arr, type(arr), sep='\n\n')
```

Результат:

```
[ 52.5 1386.2 214.2]

<class 'numpy.ndarray'>
```

И передадим его в конструктор класса `Series`:

```
srs = pd.Series(arr)
print(srs)
```

Результат:

```
0      52.5
1    1386.2
2     214.2
dtype: float64
```

В итоге получаем объект `Series`.

Обращаемся к конкретным значениям объекта Series

Теперь давайте разберемся с тем, как обращаться к конкретному элементу серии. Для того, чтобы получить значение конкретного элемента серии, нужно в квадратных скобках указать метку индекса этого элемента.

Например, чтобы получить значение первого элемента в серии из предыдущего примера, нужно написать следующий код:

```
print(srs[0])
```

Результат:

```
52.5
```

Для получения значений нескольких элементов, используются срезы:

```
print(srs[0:2])
```

Результат:

```
0      52.5
1    1386.2
dtype: float64
```

Если объект *Series* имеет нечисловую индексацию, то получить конкретное значение можно как по числовой метке индекса, так и передав строку с именем метки индекса в квадратных скобках.

Создадим серию с именованными индексами:

```
arr = np.array([52.50, 1386.20, 214.20])

srs = pd.Series(arr, index=['ММК', 'Северсталь', 'НЛМК'])

print(srs)
```

Результат:

```
ММК      52.5
Северсталь 1386.2
НЛМК     214.2
dtype: float64
```

Давайте получим значение по метке 'НЛМК':

```
print(srs['НЛМК'])
```

Результат:

```
214.2
```

Но если укажем числовую индексацию, то результат будет аналогичным:

```
print(srs[2])
```

Результат:

```
214.2
```

Меняем значение

Для изменения значений в объектах *Series* используется оператор присваивания '='.

Давайте в серии из предыдущего примера изменим значение по индексу 'ММК' с 52.5 на 50:

```
srs['ММК'] = 50

print(srs)
```

Результат:

```
ММК          50.0
Северсталь   1386.2
НЛМК         214.2
dtype: float64
```

Как видим, значение поменялось с 52.5 на 50.0.

Но если мы теперь посмотрим на массив NumPy, используя который мы создали серию `srs`, то увидим, что и в нём произошло изменение значения с 52.5 на 50.0:

```
print(arr)
```

Результат:

```
[ 50.  1386.2  214.2]
```

Параметр `copy` конструктора класса `Series`

Вернемся к параметрам конструктора класса `Series` и освежим в памяти его синтаксис:

```
class pandas.Series(data=None, index=None, dtype=None, name=None, copy=False)
```

Чтобы в массиве, на базе которого создается объект `Series`, не происходили изменения, когда меняются значения в серии, для этого в конструкторе `Series` предусмотрен параметр `copy`. Если этому параметру передать значение `True`, то серия создастся на копии массива, и любые последующие изменения в объекте `Series` никак не отразятся на исходном массиве.

Давайте создадим серию с параметром `copy=True`, произведем в ней изменения и посмотрим на исходный массив.

```
arr = np.array([52.50, 1386.20, 214.20])
print(arr)
```

Результат:

```
[ 52.5 1386.2  214.2]
```

Создадим серию:

```
srs = pd.Series(arr, index=['ММК', 'Северсталь', 'НЛМК'], copy=True)
print(srs)
```

Результат:

```
ММК          52.5
Северсталь   1386.2
НЛМК          214.2
dtype: float64
```

Поменяем значение для 'ММК':

```
srs['ММК'] = 50
print(srs)
```

Результат:

```
ММК          50.0
Северсталь   1386.2
НЛМК          214.2
dtype: float64
```

Посмотрим на исходный массив:

```
print(arr)
```

Результат:

```
[ 52.5 1386.2 214.2]
```

Как можем убедиться, изменения, произведенные в объекте `Series`, никак не отразились на исходном массиве.

Параметр `name` конструктора класса `Series`

Параметр `name` используется для создания имени у объекта `Series`.

Создадим серию `srs` из массива `arr` и назовем ее 'Shares':

```
srs = pd.Series(arr, index=['ММК', 'Северсталь', 'НЛМК'], name='Shares')
print(srs)
```

Результат:

```
ММК          52.5
Северсталь   1386.2
НЛМК         214.2
Name: Shares, dtype: float64
```

Как видим, создан объект `Series` с именем `Shares`.

Создание объекта `Series` из константы

Мы уже рассмотрели три варианта создания структуры `Series`: из списка, из словаря и с использованием массива `NumPy`. Еще один вариант создания объектов `Series` - это использование скалярной величины.

Например:

```
srs = pd.Series(170, index=['Газпром', 'Газпром', 'Газпром'])

print(srs)
```

Результат:

```
Газпром    170
Газпром    170
Газпром    170
dtype: int64
```

Добавление нового элемента в структуру `Series`

Для того, чтобы добавить новый элемент в структуру `Series`, достаточно указать новый индекс для объекта и задать значение элементу.

Добавим в серию `srs` из предыдущего примера новый элемент с индексом `'Новатэк'` и значением `1750`:

```
srs['Новатэк'] = 1750
print(srs)
```

Результат:

```
Газпром    170
Газпром    170
Газпром    170
Новатэк    1750
dtype: int64
```

Или рассмотрим пример добавления нового элемента в `Series`, имеющий числовую индексацию. Для этого создадим серию:

```
srs = pd.Series(data=[20, 30, 40])
print(srs)
```

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

и добавим новый элемент со значением 50:

```
srs[3] = 50
print(srs)
```

Результат:

```
0    20
1    30
2    40
3    50
dtype: int64
```

Удаление элементов из структуры Series

Для удаления элементов из объекта `Series` используется метод `drop()`, которому передается список удаляемых меток.

Например:

```
print(srs)
```

```
0    20
1    30
2    40
3    50
dtype: int64
```

```
srs.drop([1, 2])
```

Результат:

```
0    20
3    50
dtype: int64
```

Стоит отметить, что при использовании метода `drop()`, по умолчанию, текущая структура не изменяется, а возвращается новый объект `Series`, в котором будут отсутствовать выбранные метки.

Если вывести на печать серию `srs`, то увидим, что в ней не произошли изменения:

```
print(srs)
```

```
0    20
1    30
2    40
3    50
dtype: int64
```

Если нужно изменить непосредственно саму структуру `Series`, то методу `drop()` дополнительно нужно передать параметр `inplace=True`:

```
srs.drop([1, 2], inplace=True)

print(srs)
```

Результат:

```
0    20
3    50
dtype: int64
```