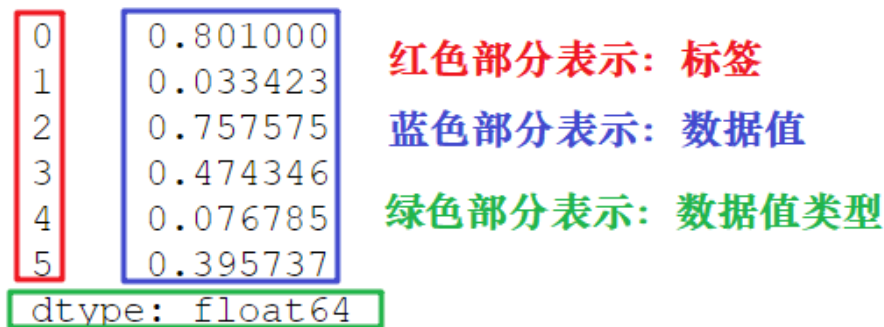


# Series 结构

Series 结构，也称 Series 序列，是 Pandas 常用的数据结构之一，它是一种类似于一维数组的结构，由一组数据值（value）和一组标签组成，其中标签与数据值具有对应关系。

标签不必是唯一的，但必须是可哈希类型。该对象既支持基于整数的索引，也支持基于标签的索引，并提供了许多方法来执行涉及索引的操作。ndarray 的统计方法已被覆盖，以自动排除缺失的数据（目前表示为 NaN）

Series 可以保存任何数据类型，比如整数、字符串、浮点数、Python 对象等，它的标签默认为整数，从 0 开始依次递增。Series 的结构图，如下所示：



通过标签我们可以更加直观地查看数据所在的索引位置。

```
1 # 引用numpy
2 import numpy as np
3 # 引入pandas
4 import pandas as pd
```

## 数据结构Series创建

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

- data 输入的数据，可以是列表、常量、ndarray 数组等,如果是字典,则保持参数顺序
- index 索引值,必须是可散列的(不可变数据类型 (str, bytes和数值类型)), 并且与数据具有相同的长度,允许使用非唯一索引值。如果未提供，将默认为RangeIndex (0, 1, 2, ..., n)
- dtype 输出系列的数据类型。如果未指定，将从数据中推断
- name 为Series定义一个名称

- copy 表示对 data 进行拷贝，默认为 False,仅影响Series和ndarray数组

## 1. 创建

### 1) 列表/数组作为数据源创建Series

```
1 # 列表作为数据创建Series
2 ar_list = [3,10,3,4,5]
3 print(type(ar_list))
4 # 使用列表创建Series
5 s1 = pd.Series(ar_list)
6 print(s1)
7 print(type(s1))
```

```
1 # 数组作为数据源
2 np_rand = np.arange(1,6)
3 # 使用数组创建Series
4 s1 = pd.Series(np_rand)
5 s1
```

- 通过index 和values属性取得对应的标签和值

```
1 # 默认为RangeIndex (0, 1, 2, ..., n)
2 s1.index
```

```
1 # 可以强制转化为列表输出
2 list(s1.index)
```

```
1 # 返回Series所有值,数据类型为ndarray
2 print(s1.values, type(s1.values))
```

- 通过标签取得对应的值,或者修改对应的值

```
1 s1[1] # 取得索引为1 的数据
```

```
1 s1[2] = 50 # 改变索引为2的数据值
2 s1
```

```
1 s1[-1]
```

```
1
```

- 和列表索引区别:
- 默认的索引RangeIndex,不能使用负值,来表示从后往前找元素,
- 获取不存在的索引值对应数据,会报错,但是可以赋值,相当于新增数据
- 可以新增不同类型索引的数据,新增不同类型索引的数据,索引的类型会发生自动变化

```
1 # ①.默认的索引RangeIndex,不能使用负值,来表示从后往前找元素,
2 s1[-1] = 20
3 s1
```

```
1 # ②当前索引为-1,不存在,不会报错,可以添加
2 s1[-1] = 20
3 print(s1)
4 print(s1.index)
```

```
1 # ③新增不同类型索引的数据,索引的类型会发生自动变化
2 s1["a"] = 40
3 s1.index
```

```
1 print(s1)
2 # 因为我标签存在-1
3 s1[-1]
```

```
1 将s1标签为3的值改为50
```

## 2) 字典作为数据源创建Series

```
1 d = {'a': 1, 'b': 2, 'c': 3}
2 ser = pd.Series(data=d)
3 ser
```

- 通过index 和values属性取得对应的标签和值

```
1 # 标签索引
2 ser.index
```

```
1 # Series值
2 ser.values
```

- 通过标签取得对应的值,或者修改对应的值

```
1 ser['a']
```

```
1 ser["s"] = 50
2 ser
```

```
1 ser[0]
2 # 如果标签非数值型,既可以用标签获取值,也可以用标签的下标获取值
```

```
1 ser.index
```

```
1 ser["a"]
```

```
1 ser[1]
```

```
1 print(ser)
2 ser[-1]
```

```
1 d = {'a': 1, 5: 2, 'c': 3}
2 ser1 = pd.Series(data=d)
3 ser1
```

```

1 #ser1["a"]
2 # 标签如果存在数值型的数据,就不能使用标签的下标获取值
3 # Series通过索引取值,优先使用是标签索引
4 ser1[0]
5 #ser1[1]

```

- 取得数据时,先进行标签的检查,如果标签中没有,再进行索引的检查,都不存在则报错

```

1 d = {'a': 1, 'b': 2, 'c': 3}
2 ser = pd.Series(data=d)
3 print(ser)
4 print("=====")
5 # 取得第一个元素
6 print('ser["a"]:%s'% ser["a"], ' ser[0]:%s'% ser[0])
7 # 取得最后一个元素
8 print('ser["c"]:%s'% ser["c"], ' ser[-1]:%s'% ser[-1])

```

```
1
```

### 3) 通过标量创建

```

1 s = pd.Series(100,index=range(5))
2 s

```

## 2. 参数说明

- a. index 参数

索引值,必须是可散列的(不可变数据类型 (str, bytes和数值类型)), 并且与数据具有相同的长度,允许使用非唯一索引值。如果未提供,将默认为RangeIndex (0, 1, 2, ..., n)

- 使用“显式索引”的方法定义索引标签

```

1 data = np.array(['a', 'b', 'c', 'd'])
2 #自定义索引标签 (即显示索引) , 需要和数据长度一致
3 s = pd.Series(data,index=[100,101,102,103])
4 s

```

- 从指定索引的字典构造序列

```

1 d = {'a': 1, 'b': 2, 'c': 3}
2 ser = pd.Series(d, index=['a', 'b', 'c'])
3 ser

```

- 当传递的索引值未匹配对应的字典键时,使用 NaN (非数字) 填充。

```

1 d = {'a': 1, 'b': 2, 'c': 3}
2 ser = pd.Series(data=d, index=['x', 'b', 'z'])
3 ser

```

请注意,索引是首先使用字典中的键构建的。在此之后,用给定的索引值对序列重新编制索引,因此我们得到所有NaN。

- 通过匹配的索引值,改变创建Series数据的顺序

```
1 d = {'a': 1, 'b': 2, 'c': 3}
2 ser = pd.Series(data=d, index=['c', 'b', 'a'])
3 ser
```

- b. name参数

我们可以给一个Series对象命名,也可以给一个Series数组中的索引列起一个名字,pandas为我们设计好了对象的属性,并在设置了name属性值用来进行名字的设定。以下程序可以用来完成该操作。

```
1 dict_data1 = {
2     "Beijing":2200,
3     "Shanghai":2500,
4     "Shenzhen":1700
5 }
6 data1 = pd.Series(dict_data1)
7 data1
```

```
1 data1 = pd.Series(dict_data1)
2 data1.name = "City_Data"
3 data1.index.name = "City_Name"
4 print(data1)
```

```
1 data1.name
```

```
1 #data1.index.name
```

```
1 data1
```

**i** 序列的名称,如果是DataFrame的一部分,还包括列名

- 如果用于形成数据帧,序列的名称将成为其索引或列名。每当使用解释器显示序列时,也会使用它。

```
1 # 使用Series创建DataFrame类型
2 df = pd.DataFrame(data1)
3 print(df,type(df))
4 print("="*20)
5 # 输出City_Data列的数据和类型
6 print(df['City_Data'],type(df['City_Data']))
```

- c. copy参数

copy 表示对 data 进行拷贝,默认为 False,仅影响Series和ndarray数组

```
1 # 数组作为数据源
2 np_rand = np.arange(1,6)
3 # 使用数组创建Series
4 s1 = pd.Series(np_rand)
5 s1
```

```
1 # 改变Series标签为1的值
2 s1[1] = 50
3
4 # 输出Series对象s1
5 print("s1:", s1)
6
7 # 输出数组对象np_rand
8 print("np_rand:", np_rand)
```

```
1 # 当源数据非Series和ndarray类型时,
2 # 数组作为数据源
3 my_list = [1, 2, 3, 4, 5, 6]
4 # 使用数组创建Series
5 s2 = pd.Series(my_list)
6 s2
```

```
1 # 改变Series标签为1的值
2 s2[1] = 50
3
4 # 输出Series对象s2
5 print("s2:", s2)
6
7 # 输出列表对象my_list
8 print("my_list:", my_list)
```

```
1
```

```
1
```

## Series的索引/切片

### 1. 下标索引

类似于 列表索引

```
1 s = pd.Series(np.random.rand(5))
2 print(s)
3 print(s[3], type(s[3]), s[3].dtype)
```

上面的位置索引和标签索引刚好一致,会使用标签索引

当使用负值时,实际并不存在负数的标签索引

### 2. 标签索引

当索引为object类型时,既可以使用标签索引也可以使用位置索引

Series 类似于固定大小的 dict，把 index 中的索引标签当做 key，而把 Series 序列中的元素值当做 value，然后通过 index 索引标签来访问或者修改元素值。

使用索引标签访问单个元素值：

```
1 s = pd.Series(np.random.rand(5), index=list("abcde"))
2 print(s["b"], type(s["b"]), s["b"].dtype)
```

使用索引标签访问多个元素值

```
1 s = pd.Series([6,7,8,9,10], index = ['a', 'b', 'c', 'd', 'e'])
2 print(s)
3 # 注意需要选择多个标签的值,用[]来表示(相当于[]中包含一个列表)
4 print(s[['a', 'c', 'd']]) # s['a', 'c', 'd']
```

```
1 s1
```

多标签会创建一个新的数组

```
1 s1 = s[["b", "a", "e"]]
2 s1["b"] = 10
3 print("s1:", s1)
4 print("s源数据:", s)
```

### 3. 切片

- Series使用标签切片运算与普通的Python切片运算不同：Series使用标签切片时，其末端是包含的。
- Series使用python切片运算即使用位置数值切片，其末端是不包含。

通过下标切片的方式访问 Series 序列中的数据，示例如下：

```
1 s = pd.Series(np.random.rand(10))
2 s
```

```
1 # 位置索引和标签索引刚好一致,使用切片时,如果是数值会认为是python切片运算,不包含末端
2 s[1:5]
```

```
1 s = pd.Series([1,2,3,4,5], index = ['a', 'b', 'c', 'd', 'e'])
2 print(s)
3 print(s[1:4])
4 # print(s[0]) #位置下标
5 # print(s['a']) #标签
```

```
1 s = pd.Series([1,2,3,4,5], index = ['a', 'b', 'c', 'd', 'e'])
2 print(s[:3])
```

```
1 # 如果想要获取最后三个元素,也可以使用下面的方式:
2 s = pd.Series([1,2,3,4,5], index = ['a', 'b', 'c', 'd', 'e'])
3 print(s[-3:])
```

通过标签切片的方式访问 Series 序列中的数据，示例如下：

- Series使用标签切片时，其末端是包含的

```
1 s1= pd.Series([6,7,8,9,10],index = ['a','b','c','d','e'])
2 s1["b":"d"]
```

```
1 s1= pd.Series([6,7,8,9,10],index = ['e','d','c','b','a'])
2 s1["c":"a"]
```

```
1 s1["c":"c"]
```

```
1 s1= pd.Series([6,7,8,9,10],index = ['e','d','a','b','a'])
2 s1
```

注意：

在上面的索引方式，我们知道了位置索引和标签索引在index为数值类型时候的不同，

- 当index为数值类型的时候，使用位置索引会抛出keyerror的异常，也就是说当index为数值类型的时候，索引使用的是名称索引。
- 但是在切片的时候，有很大的不同，如果index为数值类型的时候，切片使用的是位置切片。总的来说，当index为数值类型的时候：
- 进行索引的时候，相当于使用的是名称索引；
- 进行切片的时候，相当于使用的是位置切片；

## Series数据结构 基本技巧

### 1. 查看前几条和后几条数据

```
1 s = pd.Series(np.random.rand(15))
2 s
```

```
1
2 print(s.head()) # 默认查看前5条数据
3 print(s.head(1)) # 默认查看前1条数据
```

```
1 print(s.tail()) # 默认查看后5条数据
```

### 2. 重新索引: `reindex`

使用可选填充逻辑，使Series符合新索引

将NaN放在上一个索引中没有值的位置。除非新索引等同于当前索引,并且生成新对象。



```

1 s = pd.Series(np.random.rand(5), index=list("abcde"))
2 print("====s====")
3 #print(s)
4
5 # 新索引在上一个索引中不存在,生成新对象时,对应的值,设置为NaN
6 s1 = s.reindex(list("cde"))
7 print("====s1====")
8 print(s1)
9 print("====s====")
10 print(s)

```

```

1 # 设置填充值
2 s2 = s.reindex(list("cde12"), fill_value=0)
3 print(s2)

```

### 3.对齐运算

是数据清洗的重要过程，可以按索引对齐进行运算，如果没对齐的位置则补NaN，最后也可以填充NaN

```

1
2 s1 = pd.Series(np.random.rand(3), index=["Kelly", "Anne", "T-C"])
3
4 s2 = pd.Series(np.random.rand(3), index=["Anne", "Kelly", "LiLy"])
5
6 print("====s1====")
7 print(s1)
8 print("====s2====")
9 print(s2)
10 print("====s1+s2====")
11 print(s1+s2)

```

### 4.删除和添加

#### ○ 删除

```

1 s = pd.Series(np.random.rand(5), index=list("abcde"))
2 print(s)
3 s1 = s.drop("a") # 返回删除后的值,原值不改变 ,默认inplace=False
4 print(s1)
5 print(s)

```

```

1 s = pd.Series(np.random.rand(5), index=list("abcde"))
2 s1 = s.drop("a", inplace=True) # 原值发生变化,返回None
3 #s = s.drop("a")
4 print(s1)
5 print(s)
6
7 # inplace默认为True,返回None

```

#### ○ 添加

```

1 import pandas as pd
2 # 添加
3 s1 = pd.Series(np.random.rand(5), index=list("abcde"))
4 print(s1)
5
6 s1["s"] = 100 # 对应的标签没有就是添加,,有就是修改
7 print(s1)
8

```

```
1
```

```
1
```

## \* 课堂作业

1.分别通过字典/数组的方式,创建以下要求的Series

Zhangsan 90.0

wangwu 89.5

lilei 68.0

Name:作业1, dtype:float64

```

1 2.创建一个Series,包含1组元素,且每个值为0-100的均匀分布随机值,index为a-j,请分别筛选出
2 - ① 标签为b,c的值为多少
3 - ②.Series中第4到第6个值是那些
4 - ③ Series中大于50 的值有那些

```

3.创建以下Series 并按照规定要求修改得到的结果

创建s:

a 0

b 1

c 2

d 3

e 4

f 5

g 6

h 7

i 8

j 9

dtype:int32

////////////////////////////////////

s修改后:

a 100

c 2

d 3

e 100

f 100

g 6

i 8

j 9

dtype:int32