学习途径：Bilibili Max极客菌

官方文档： https://pandas.pydata.org/

Series 结构

由一组数值和标签组成的类似一维数组的结构。

标签：不唯一，可哈希类型。支持基于整数的索引，也支持基于标签的索引。

Series：可以保存任何数据类型。标签默认为整数，从0开始递增。

标签　数值
显式索引　　隐式索引
0　0.80213
1　0.01245
2　2.15486
3　6.54891
dtype：float64 数据类型

创建Series

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

data: 输入的数据
index：索引值，不可变数据类型，且个数要与data一至。默认为RangeIndex（0,1，...n）
dtype：数据类型
name：为Series定义一个名字
copy 表示对data进行拷贝。默认为False

From ndarray 引用对象
list 副本
dict
scalar value：data为一个常量或变量。必须指定索引，每个索引的值都为指定的scalar value

**1）列表/数组作为数据源

```
import numpy as np
import pandas as pd


a = [3,25,3,9,7]
s1 = pd.Series(a)
print(s1)
```

```
0     3
1    25
2     3
3     9
4     7
dtype: int64
```

获取index

series.index

```
s1.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

```
# 转化为index
print(list(s1.index))
```

```
[0, 1, 2, 3, 4]
```

获取值/修改值

获取所有值 series.values

通过标签 series[index]

```
print(s1[2])
s1[2] = -1
print(s1[2])
print(s1[-1])
print(s1[20])
```

```
3
-1
```

```
---------------------------------------------------------------
-------------------

KeyError                                  Traceback
(most recent call last)

<ipython-input-4-25c2be8bdb94> in <module>()
      2 s1[2] = -1
      3 print(s1[2])
----> 4 print(s1[-1])
      5 print(s1[20])
```

```
D:\Anaconda3\lib\site-packages\pandas\core\series.py in
__getitem__(self, key)
    621         key = com._apply_if_callable(key, self)
    622         try:
--> 623             result = self.index.get_value(self,
key)
    624
    625             if not is_scalar(result):
```

```
D:\Anaconda3\lib\site-
packages\pandas\core\indexes\base.py in get_value(self,
series, key)
   2558         try:
   2559             return self._engine.get_value(s, k,
-> 2560
tz=getattr(series.dtype, 'tz', None))
   2561         except KeyError as e1:
   2562             if len(self) > 0 and
self.inferred_type in ['integer', 'boolean']:
```

```
pandas/_libs/index.pyx in
pandas._libs.index.IndexEngine.get_value()
```

```
pandas/_libs/index.pyx in
pandas._libs.index.IndexEngine.get_value()
```

```
pandas/_libs/index.pyx in
pandas._libs.index.IndexEngine.get_loc()
```

```
pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.Int64HashTable.get_item()
```

```
pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.Int64HashTable.get_item()
```

```
KeyError: -1
```

和列表索引的区别：

1. 不能用负数表示索引
2. 获取不存在的索引会报错，但可以用不存在的索引来新增数据
3. 可以新增不同索引类型的数据，Series的索引数据类型也会一起变化

```
s1[-1] = 88
s1['a'] = 66
print(s1)
s1.index

s1.values
```

**2) 字典作为数据源

```
d = {'a':1,'b':2,'c':3}
s2 = pd.Series(d)
print(s2.index)
print(s2.values)
```

```
s2['a']

#s2['d'] #报错 KeyError: 'd'
s2['d'] = 9
s2['d']
```

通过标签的下标取值

```
s2[0]
```

```
s2[0] = 6
s2['0'] = 7
print(s2[0])
print(s2)
```

顺序：如果标签全为非数值型，先通过标签找值，如果没有再找标签的下标。但是如果标签的存在整型的标签，就不会继续找标签的下标，只会在标签里面找。

```
d = {'a':1,5:2,'c':3}
s3 = pd.Series(data=d)
print(s3.index)
print(s3.values)
```

```
print(s3[0])
print(s3[1])
print(s3[5])
```

指定索引

Series属性

name：返回Series对象的名称
shape:返回Series对象的形状
size:返回Series对象的元素个数
index:返回显示索引
values:返回Series对象的值

```
s4 = pd.Series([1,2,3,4,5],name="ccc")
print(s4.name)
print(s4.shape)
print(s4.size)
print(s4.index)
print(s4.values)
```

Series数学运算

```
s5 = pd.Series([1,2,3,4,5])
s5
```

**1）与非Pandas对象, 广播机制

```
s5 + 1
```

**2）Numpy fun

**3）Series之间

```
-索引对齐原则
-对不齐补空值，使用add/sub/mul/div函数处理空值
```

```
#-索引对齐原则
S6 = pd.Series([1,2,3],index=list('ABC'))
S7 = pd.Series([6,1,1],index=list('BAC'))
print(S6)
print(S7)
S6+S7
```

```
A    1
B    2
C    3
dtype: int64
B    6
A    1
C    1
dtype: int64
```

```
A    2
B    8
C    4
dtype: int64
```

```
#-对不齐补空值，使用add/sub/mul/div函数处理空值
S8 = pd.Series([1,2,3],index=list('ABC'))
S9 = pd.Series([6,1,1],index=list('BBC'))
print(S8)
print(S9)
S8+S9
```

```
A    1
B    2
C    3
dtype: int64
B    6
B    1
C    1
dtype: int64
```

```
A    NaN
B    8.0
B    3.0
C    4.0
dtype: float64
```

DataFrame 结构

-二维的表格型数据结构。可看作由Series组成的字典（共用一个索引）
-由一定顺序排列（多列）数据组成，每一列的数据类型可能不同。

```
   行索引   S1     S2     S3 列索引
    K1    1      A     4
    K2    2      B     5
    K3    3      C     6    value
```

构造DataFrame

import numpy as np
import pandas as pd

from pandas import Series,DataFrame

```
# data=None
# index=None, column=None
#dtype=None
#copy=False

data=np.random.randint(0,100,size=(3,4))
index = ['cara','jim','Amy']
columns = ['语文','数学','英语','化学']
d1 = pd.DataFrame(data,index,columns)
d1
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	语文	数学	英语	化学
cara	99	47	55	91
jim	23	94	3	48
Amy	97	8	7	48

1）From dict of Series or dicts 使用一个由Series构造的字典或一个字典构造

```
names = ['cara','jim','Amy']
chinese = [100,99,98]
english = [95,96,97]
math = [93,94,97]
score = {
    'aname':names,
    'chinese':chinese,
    'math':math,
    'english':english
}
d2 = pd.DataFrame(score)
d2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	aname	chinese	english	math
0	cara	100	95	93
1	jim	99	96	94
2	Amy	98	97	97

```
names = ['cara','jim','Amy']
chinese = [100,99,98]
english = [95,96,97]
```

```
math = [93,94,97]
snames = pd.Series(names,index=list('abc'))
schinese = pd.Series(chinese,index=list('abc'))
senglish = pd.Series(english,index=list('abc'))
smath = pd.Series(math,index=list('abc'))

sscore = {
    'aname':snames,
    'chinese':schinese,
    'math':smath,
    'english':senglish
}
d3 = pd.DataFrame(sscore)
d3
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	aname	chinese	english	math
a	cara	100	95	93
b	jim	99	96	94
c	Amy	98	97	97

2）From dict of ndarray / lists 使用一个由列表或ndarray构造的字典构造
ndarray 长度必须保持一致

3）From a list of dicts 使用一个由字典构成的列表构造

4）使用DataFrame from dict()函数构造

DataFrame属性

·dtypes
·values ndarray类型<class 'numpy.ndarray'>
·index
·columns

DataFrame运算

1）与非pandas对象运算，广播机制

2）与Series对象

3）与DataFrame对象

-索引对齐原则
-对不齐补空值，使用add/sub/mul/div函数处理空值

4）Numpy fun

5）转置运算