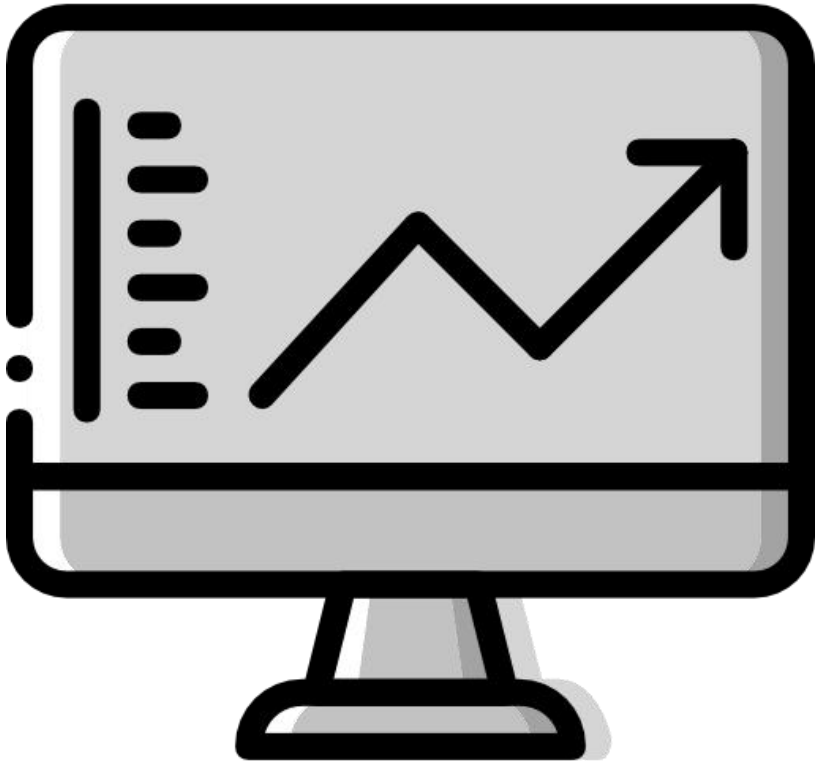
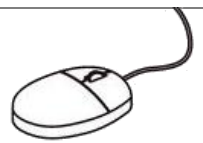
南通师范高等专科学校
Nantong Normal College



pandas
数据处理之一

执教：朱亚林

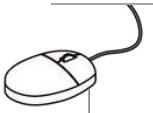
pandas简介



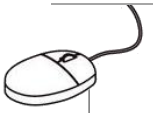
什么是pandas

- pandas是Python的一个数据分析包。是一个高性能，高效率，高水平的数据分析库。
- pandas是为了解决数据分析任务而创建的，纳入了大量的库和标准数据模型，提供了高效地操作大型数据集所需的工具。
- 通过带有标签的列和索引，Pandas 使用户可以以一种所有人都能理解的方式来处理数据。它可以让用户毫不费力地从诸如 csv 类型的文件中导入数据。我们可以用它快速地对数据进行复杂的转换和过滤等操作。

pandas的安装



```
pip install pandas
```



pandas中的数据结构

- Series
- DataFrame

	apples
0	3
1	2
2	0
3	1

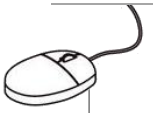
	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

Series

Series 是一种一维的数据类型，其中的每个元素都有各自的标签。(与数组和字典的功能类似)

	apples
0	3
1	2
2	0
3	1

创建Series



创建Series的方法如下。

Series（［数据1，数据2，…］，index［索引1，索引2，…］）

调用Series函数

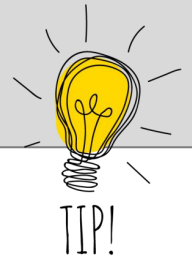
```
>>> import pandas as pd
>>> s = pd.Series([1,3,5,np.nan,6,8])
>>> s
0  1
1  3
2  5
3  NaN
4  6
5  8
dtype: float64
```

左侧为index，右侧为对应元素

创建Series

- 创建Series时，若不指定标签，pandas默认使用从0开始依次递增的数值作为标签。这种情况下，标签与Series对象中元素的索引（在数组中的位置）一致。
- 同时，也可以使用有意义的标签作为索引赋给数组

```
>>> import pandas as pd
>>> s = pd.Series( [ 15,-2,7,9 ] ,index= [ 'a','b','c','d' ] )
>>> s
a  15
b  -2
c   7
d   9
dtype: int64
```



左侧的index为自定义内容

创建Series

Series内容的查看

s.index →

	apples
0	3
1	2
2	0
3	1

← s.values

Series的索引

- Series的索引与数组的索引很相似，Series的元素需要使用索引值来访问

```
>>> import pandas as pd
>>> s = pd.Series([1,3,5,7,9],index=['a','b','c','d','e'])
>>> s[0]
--- what ?
>>> s['d']
--- what ?
>>> s[1:3]
--- what ?
>>> s[['b','d']]
--- what ?
```

>Question

更改Series元素

- 指定Series索引位置，即可以修改其对应的值。

```
>>> import pandas as pd
>>> s = pd.Series([1,3,5,7,9],index=['a','b','c','d','e'])
>>> s[1]=2
>>> s['d']=8
```

EXAMPLE

追加Series元素

给Series追加元素，可以使用append方法。

```
>>> import pandas as pd
>>> s = pd.Series([1,3,5,7,9],index=['a','b','c','d','e'])
>>> n = pd.Series([11],index='f')
>>> s.append(n)
```

删除Series元素

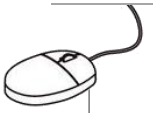
给Series删除元素，可以使用drop方法指定索引，也可以使用指定值删除。

```
>>> import pandas as pd
>>> s = pd.Series([1,3,5,7,9],index=['a','b','c','d','e'])
>>> s.drop('a')
>>> s[7!=s.values]
>>> s.drop(s.index[3])

#以上删除不修改Series本身
```

EXAMPLE

Series元素的排序



- 使用sort_index()方法对索引进行排序
- 使用sort_values()方法对值进行排序
- 以上两个方法都包含ascending参数，其值为True为升序，其值为False为降序
- 在Series上调用reindex()方法重排数据，使得它符合新的索引，如果索引的值不存在，就引入缺失数据值。

Series元素的排序

排序案例

```
>>> import pandas as pd
>>> s = pd.Series([11,3,15,7,9],index=['a','b','c','d','e'])
>>> s.sort_index()
>>> s.sort_values()
>>> s1 = s.reindex(['d','b','e','f'])
# 对于新出现的索引会默认赋给NaN的值，可使用以下两种方法给NaN填充值
>>> s1 = s.reindex(['d','b','e','f'],fill_value=0)
>>> s1.fillna(0)
```

DataFrame

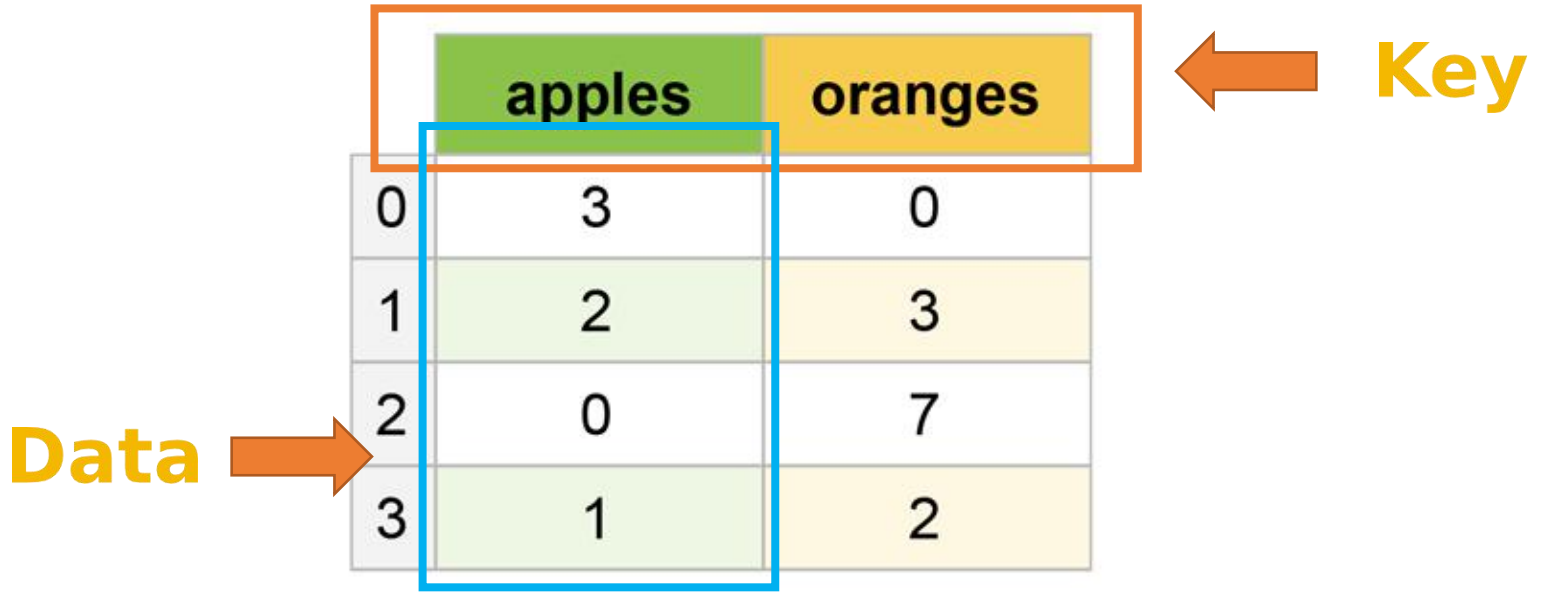
- DataFrame是一种列表式的数据结构。与Excel的电子表格或者关系型数据库的数据表非常相似。其设计初衷是实现多维的Series。
- DataFrame由按照一定顺序排列的多列数据组成，各列的数据类型可以有所不同。

columns

index

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

DataFrame还可以理解为一个由Series组成的字典。其中每一列的名称为字典的键，形成DataFrame的列Series作为字典的值。每个Series的所有元素映射到称为index的标签数组。



创建DataFrame

- 最常用的方法即是参照上述示意图，传递一个dict对象给DataFrame构造函数。dict对象以每一列的名称作为键，每个键都有一个数组作为值。

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2]}
>>> df=pd.DataFrame(data)
>>> print(df)
```

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

EXAMPLE

创建DataFrame

创建DataFrame时可以使用columns来指定需要的列。创建的DataFrame各列的顺序与指定的列顺序是一致的，与原字典中的顺序无关。

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'])
>>> print(df)
```

arrays must all be same length

EXAMPLE

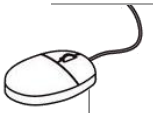
创建DataFrame

- DataFrame对象与Series一样，如果index数组没有明确指定标签，pandas也会自动为其添加一列从0开始的递增数值作为索引。如果想使用指定标签作为索引，则将标签放入数组中，赋给index即可。

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'],index=['1st','2nd','3rd','4th'])
>>> print(df)
```

EXAMPLE

DataFrame值的增加

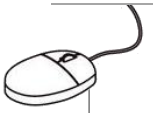


按列增加

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'],index=['1st','2nd','3rd','4th'])
>>> pears=[5,3,2,1]
>>> df.insert(0,'pears',pears) #在第0列，加上columns名称为pears，值为pears中的数值
>>> tomatos=[1,1,2,3]
>>> df['tomatos']=tomatos #默认在df最后一列添加内容
```

EXAMPLE

DataFrame值的增加



按行增加

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'],index=['1st','2nd','3rd','4th'])
>>> df.loc['5th']=[8,9] # 如有5th，则会修改原值，如无则新增
>>> df_insert = pd.DataFrame({'banana':[18,19],'apples':[22,11]},index =
['6th','7th'])
>>> ndf = df.append(df_insert,ignore_index = True) #返回添加后的值，并不会修改df的值。ignore_index默认为False，意思是不忽略index值，即生成的新的ndf的index采用df_insert中的index值。若为True，则新的ndf的index值不使用df_insert中的index值，而是自己默认生成
```

EXAMPLE

DataFrame值的查询

使用df['column_name'] 和df[row_start_index, row_end_index]

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'],index=['1st','2nd','3rd','4th'])
>>> df['banana']
>>> df[1:3]
>>> df[-3:-1]
```

EXAMPLE

DataFrame值的查询

使用df.loc[index,column]

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'],index=['1st','2nd','3rd','4th'])
>>> df.loc['1st','apples']
>>> ndf.loc[0:2,'apples']
>>> df.loc[df['apples']==3,'banana'] #选取apples列中值为3的banana列对应的值
```

EXAMPLE

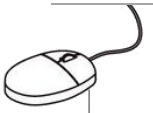
DataFrame值的查询

使用iloc[row_index, column_index]

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples','oranges'])
>>> df.iloc[1,2]
>>> df.iloc[[1,3],0:2] #第1行和第3行，从第0列到第2列（不包含第2列）的数据
```

EXAMPLE

DataFrame值的修改

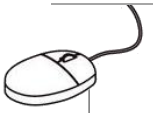


改标题

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'],index=['1st','2nd','3rd','4th'])
>>> df.columns = ['XJ','PG'] #要把所有的列全写上，否则报错
>>> df.rename(columns = {'XJ':'banana','PG':'apple'},inplace = True) #inplace若为
True，直接修改df，否则，不修改df，只是返回一个修改后的数据。
```

EXAMPLE

DataFrame值的修改



改数值

```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples','oranges'])
>>> df.loc[1,'apples'] = 9 #修改index为1，column为'apples'的那一个值为9
>>> df.loc[1] = [19,21] #修改index为'1'的那一行的所有值。
>>> df.iloc[1,2] = 19 #修改某一元素
>>> df.iloc[:,2] = [11,22,33] #修改一整列
>>> df.iloc[0,:] = [12,23,15] #修改一整行
```

EXAMPLE

DataFrame值的删除


```
>>> import pandas as pd
>>> import numpy as np
>>> data={'apples':[3,2,0,1],'oranges':[0,3,7,2],'banana':[2,4,6,8]}
>>> df=pd.DataFrame(data,columns=['banana','apples'])
>>> df.drop([1,3],axis = 0,inplace = False)#删除index值为1和3的两行
>>> df.drop(['apples'],axis = 1,inplace = False) #删除apples列
>>>  del df['apples']
```

EXAMPLE