

A cluster of colorful geometric shapes, including triangles and squares in shades of blue, yellow, green, and orange, arranged in a complex, overlapping pattern in the top-left corner.

# Pandas数据处理

主讲：万永权



# 目录

## CONTENTS

1. Pandas概述
2. Pandas的数据类型
3. Pandas数据的查看与编辑
4. pandas读取文本文件和excel

# Pandas

- ◆ Pandas是基于NumPy的数据分析模块
- ◆ Pandas的名称来自于**面板数据**（**Panel data**）和python数据分析（**data analysis**）。
- ◆ 使用Pandas，无论数据来源如何，我们都可以完成数据处理和分析中的五个典型步骤：**加载，准备，操作，建模和分析**。
- ◆ Python与Pandas一起使用的领域广泛，包括学术和商业领域，包括金融，经济学，统计学，分析学等。
- ◆ 可以说Pandas是使得Python能够成为高效且强大的数据分析环境的重要因素之一。

# Pandas两个重要的数据结构

Pandas主要处理以下2种数据结构：

(1) **Series**：一维数组，与NumPy的一维ndarray类似。数据结构接近List列表，数据元素可以是不同的数据类型。

(2) **DataFrame**：二维数据结构。DataFrame可以理解成Series的容器，其内部的每项元素都可以看作一个Series。

◆ 这些数据结构都是构建在NumPy数组的基础之上，运算速度很快。

## Series对象

- ◆ Series由一组数据以及一组与之相关的数据标签（即索引）组成。
- ◆ 可通过索引来访问数组中的数据。



# 图解Series对象

	语文	数学	英语
0	110	105	99
1	105	88	115
2	109	120	130

图 3.9 原始数据（成绩表）



图 3.10 图解 Series

## 创建一个Series对象

- ◆ 使用Pandas的Series()函数方法，语法如下：
- ◆ `s=pd.Series(data,index=index)`
- ✓ data：表示数据，支持Python字典、多维数组、标量值
- ✓ index是索引，缺省情况下是0到N-1(N为数据的长度)的整数型索引。访问Series对象成员可以用索引编号，也可以按索引名。

# 1. Series的创建

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

【例4-1】通过列表创建Series

```
In[1]: import pandas as pd
      obj = pd.Series([1, -2, 3, -4]) #仅有一个数组构成
      print(obj)

Out[1]: 0    1
      1   -2
      2    3
      3   -4
      dtype: int64
```



# DataFrame对象

- DataFrame是一个表格型的数据结构。列索引（columns）对应字段名，行索引（index）对应行号，值（values）是一个二维数组。每一列表示一个独立的属性，各个列的数据类型（数值、字符串、布尔值等）可以不同。
- DataFrame既有行索引也有列索引，所以DataFrame也可以看成是Series的容器。



# DataFrame结构

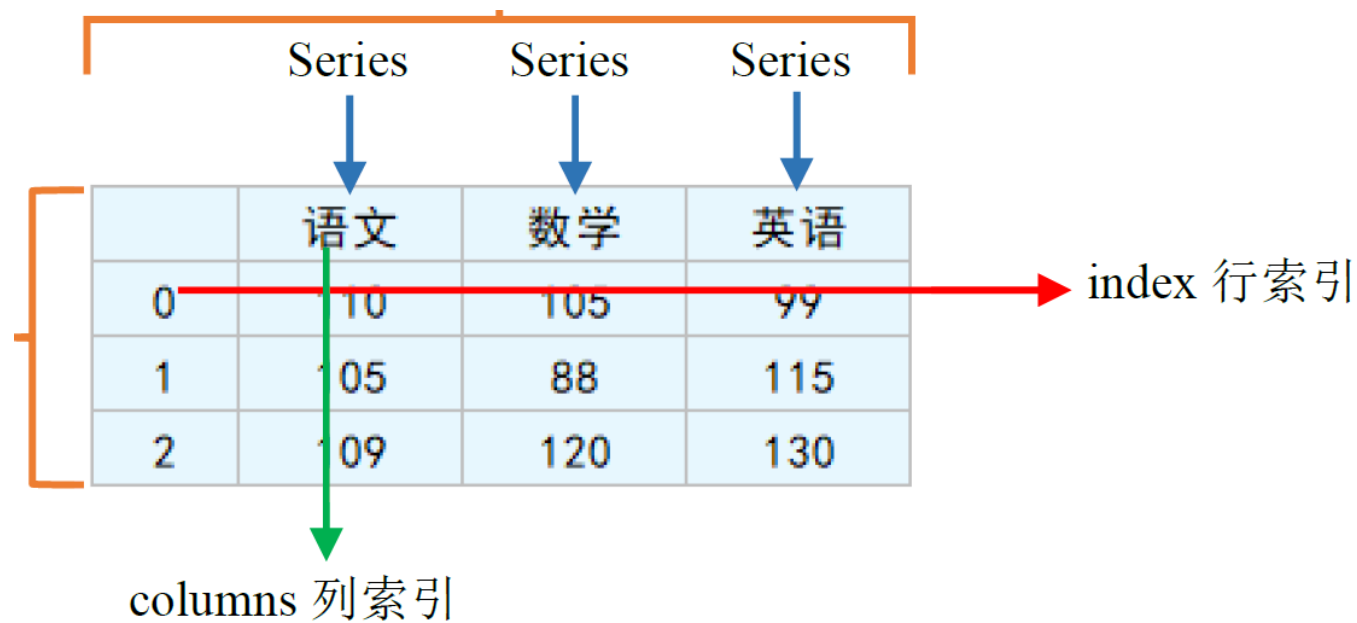


图 3.11 DataFrame 结构



## 1. 创建DataFrame对象

构建DataFrame的办法有很多，基本方法是使用DataFrame（）函数构造，格式如下：

```
DataFrame([data, index, columns, dtype, copy])
```

### **【例】：**

- 1) 从字典构建DataFrame
- 2) 从数组创建Dataframe
- 3) 从csv文件中读取数据到DataFrame**

# DataFrame的创建

格式： `pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)`

```
In[9]: data = {  
        'name':['张三', '李四', '王五', '小明'],  
        'sex':['female', 'female', 'male', 'male'],  
        'year':[2001, 2001, 2003, 2002],  
        'city':['北京', '上海', '广州', '北京']  
    }  
    df = pd.DataFrame(data)  
    print(df)
```

```
Out[9]:
```

	name	sex	year	city
0	张三	female	2001	北京
1	李四	female	2001	上海
2	王五	male	2003	广州
3	小明	male	2002	北京

DataFrame会自动加上索引（跟Series一样），且全部列会被有序排列。如果指定了列序列，则DataFrame的列就会按照指定顺序进行排列。



## 创建一个DataFrame对象

参数说明：

- ✓ data: 表示数据，可以是ndarray数组、Series对象、列表、字典等。
- ✓ index: 表示行标签（索引）。
- ✓ columns: 列标签（索引）。
- ✓ dtype: 每一列数据的数据类型，其与Python数据类型有所不同，如object数据类型对应的是Python的字符型。
- ✓ copy: 用于复制数据。
- ✓ 返回值: DataFrame。



## 4.汇总和描述性统计计算

Pandas的Serise对象和DataFrame对象都继承了NumPy的统计函数，拥有常用的数学和统计方法，可以对一系列或多列数据进行统计分析。

**【例】** 见示例。

常用的描述和汇总统计函数

函数名	功能说明
count	统计数据值的数量，不包括NA值。
describe	对Series、DataFrame的列计算汇总统计。
min,max	计算最小值、最大值
argmin,argmax	计算最小值、最大值的索引位置
idxmin,idxmax	计算最小值、最大值的索引值
sum	计算总和
mean	计算平均值
median	返回中位数
var	计算样本值的方差
std	计算样本值的标准差
cumsum	计算样本值的累计和
diff	计算一阶差分

# Pandas中的数据结构

## 2. DataFrame的属性

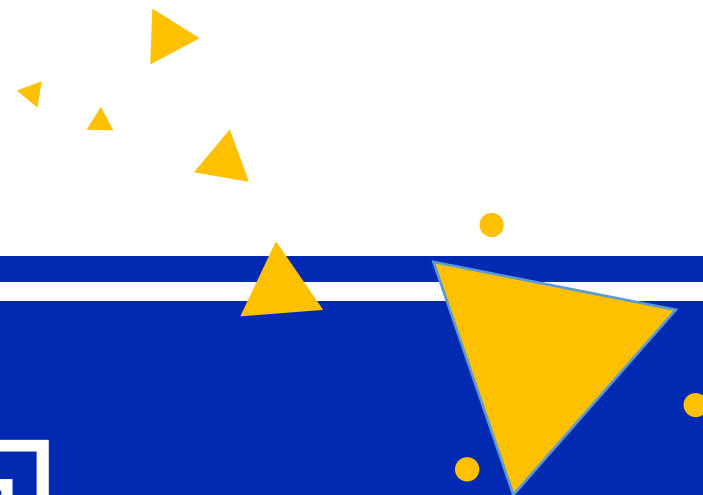
函数	返回值
values	元素
index	索引
columns	列名
dtypes	类型
size	元素个数
ndim	维度数
shape	数据形状（行列数目）



## 重要属性

属 性	描 述	举 例
values	查看所有元素的值	df.values
dtypes	查看所有元素的类型	df.dtypes
index	查看所有行名、重命名行名	df.index df.index=[1,2,3]
columns	查看所有列名、重命名列名	df.columns df.columns=['语','数','外']
T	行列数据转换	df.T
head	查看前n条数据，默认5条	df.head() df.head(10)
tail	查看后n条数据，默认5条	df.tail() df.tail(10)
shape	查看行数和列数，[0]表示行，[1]表示列	df.shape[0] df.shape[1]
info	查看索引，数据类型和内存信息	df.info





# Pandas数据访问

03



# 基于Pandas的Index对象的访问操作

## ◆ Pandas的Index对象

- Series中的index属性
- DataFrame中的index属性和columns属性

## ◆ Index对象的特征

- 不可修改、有序及可切片的

# 基于Pandas的Index对象的访问操作

## ■ 索引的不同访问方式

### ● 类似dict方式的访问

- 可以将Series和DataFrame看作一个dict，而DataFrame相当于每一个元素是Series的dict，所以可以用类似dict访问的方式来访问Series和DataFrame。
- 通过[]访问

### ● 对象属性方式的访问

- 接受参数类型包括单个变量，数组形式（list或者NumPy的ndarray），布尔数组或者回调函数。
- 通过.loc[]和.iloc[]访问

## 2. 访问DataFrame对象

可以通过索引对DataFrame进行访问， 可以获取其中的一个或多个行和/或列。

选取类型	选取方法	说明
索引名选取	<code>obj[ col ]</code>	选取某列
	<code>obj[ colList ]</code>	选取某几列
	<code>obj.loc[ index, col ]</code>	选取某行某列
	<code>obj.loc[ indexList, colList ]</code>	选取多行多列
位置序号选取	<code>obj.iloc[ iloc, cloc]</code>	选取某行某列
	<code>obj.iloc[ ilocList, clocList ]</code>	选取多行多列
	<code>obj.iloc[ a:b, c:d ]</code>	选取a~(b-1)行, c~(d-1)列
条件筛选	<code>obj.loc[ condition, colList ]</code>	使用索引构造条件表达式 选取满足条件的行
	<code>obj.iloc[ condition, clocList ]</code>	使用位置序号构造条件表达式 选取满足条件的行



# 通过列索引访问数据

通过列索引或以属性的方式可以单独获取DataFrame的列数据，返回的数据类型为Series。

```
data1 = df[['Team', 'Previous \ntitles']]  
print(data1)
```

	Team	Previous \ntitles
0	Russia	0
1	Saudi Arabia	0
2	Egypt	0
3	Uruguay	2
4	Porugal	0
5	Spain	1
6	Morocco	0
7	IRAN	0
8	France	1
9	Australia	0
10	Peru	0

# 以行索引访问数据

## 2. 选取行

通过切片形式可以选取一行或多行数据。

```
In[26]: print(df)
print('显示前 2 行:\n',df[:2])
print('显示 2-3 两行:\n',df[1:3])
```

```
Out[26]: name  year  sex city
0   张三  2001  female  北京
1   李四  2001  female  上海
2   王五  2003   male  广州
3   小明  2002   male  北京
```

显示前 2 行:

```
name  year  sex city
0   张三  2001  female  北京
1   李四  2001  female  上海
```

显示 2-3 两行:

```
name  year  sex city
1   李四  2001  female  上海
2   王五  2003   male  广州
```

# 基于Pandas的Index对象的访问操作

## ■ 索引的不同访问方式

### ● loc方式

- Pandas的loc函数的输入主要关注index的label，筛选条件与label相关，接收index的label作为参数输入。
- 包括单个的label，label的数组或者label的分片（slice）表达形式；可以接收一个布尔数组作为参数输入；可以接收参数为调用loc函数的对象（Series或者DataFrame类型）的回调函数作为参数输入。

### ● iloc方式

- 而iloc函数与loc函数不同，关注的则是index的position。index的position作为参数输入，包括
  - 表示position的单个整数，position的数组或者position的分片（slice）表达形式；
  - 可以接收一个布尔数组作为参数输入；
  - 可以接收参数为调用loc函数的对象（Series或者DataFrame类型）的回调函数作为参数输入。

# .loc() 和 .iloc() 函数

## ■ 4.3.1 DataFrame数据的查询

### 3. 选取行和列

- ✓ DataFrame.loc(行索引名称或条件, 列索引名称)
- ✓ DataFrame.iloc(行索引位置, 列索引位置)

```
#显示 name 和 year 两列  
display(df5.loc[:,'name','year'])  
#显示北京和上海行中的 name 和 year 两列  
display(df5.loc[['北京','上海'],'name','year'])  
#显示 year 大于等于 2002 的 name 和 year 信息  
display(df5.loc[df5['year']>=2002,'name','year'])
```



## 4.3 DataFrame的数据查询与编辑

### ◆ 4.3.1 DataFrame数据的查询

【例 4-28】利用 `iloc` 选取行和列。↵

```
In[28]:↵ print(df5.iloc[:,2]) ↵  
        #显示前两列↵  
        print(df5.iloc[[1,3]]) ↵  
        #显示第1 和第3 行↵  
        print(df5.iloc[[1,3],[1,2]])↵
```



# 基于Pandas的Index对象的访问操作

## ◆ 调用方式间的区别

### ➤ loc函数和iloc函数的区别

- loc函数和iloc函数都是对index的访问（Series的index和DataFrame的index），对于DataFrame也可以实现对于某个index下的某个column的访问。接收的数据类型相同但是含义不同，loc函数接收Index对象（index和columns）的label，而iloc函数接收Index对象（index和columns）的position。

### ➤ 通过loc访问和通过[]访问的区别

- loc函数和[]都是接收Index对象（index和columns）的label作为参数，但是loc函数是对index的访问（Series的index和DataFrame的index），[]在DataFrame中则是对columns的访问，在Series中无差别。



### 3. 修改DataFrame数据

#### 1) 修改数据

通过赋值语句修改数据，可以修改指定行、列的数据，还可以把要修改的数据查询筛选出来，或重新赋值。

#### 2) 增加列

DataFrame对象可以添加新的列，通过赋值语句赋值时，只要列索引名不存在，就添加新列，否则就修改列值，这与字典的特性相似。

#### 3) 合并添加数据

DataFrame对象可以增加新列，如果需要增加几行数据，需要将数据存入一个新DataFrame对象，然后将两个DataFrame对象进行合并。

**【例】**见示例。



## 4.3 DataFrame的数据查询与编辑

### ◆4.3.2 DataFrame数据的编辑

#### 1. 增加数据

增加一行直接通过append方法传入字典结构数据即可。

```
In[31]: data1 = {'city':'兰州','name':'李红','year':2005,'sex':'female'}  
df.append(data1,ignore_index = True)
```

```
Out[31]:
```

	name	year	sex	city
0	张三	2001	female	NaN
1	李四	2001	female	NaN
2	王五	2003	male	NaN
3	小明	2002	male	NaN
4	李红	2005	female	兰州



## 4.3 DataFrame的数据查询与编辑

### ◆4.3.2 DataFrame数据的编辑

#### 1. 增加数据

**增加一列**时，只需为要增加的列赋值即可创建一个新的列。若要指定新增列的位置，可以用insert函数。

【例 4-32】增加一列并赋值。

```
In[32]: df['score']=[85,78,96,80]
df.insert(1,'No',['001','002','003','004'])
df
```

```
Out[32]:
```

	name	No	sex	year	city	score
0	张三	001	female	2001	北京	85
1	李四	002	female	2001	上海	78
2	王五	003	male	2003	广州	96
3	小明	004	male	2002	北京	80



#### 4) 删除DataFrame对象的数据

Drop函数可以按行列删除数据，drop函数基本格式：

`<DataFrame对象>.drop(索引值或索引列表, axis=0, inplace=False……)`

主要参数：

- ◆ **axis**：默认为0，为行索引值或列索引列表；值为0表示删除行，值为1表示删除列。
- ◆ **inplace**：逻辑型，表示操作是否对原数据生效。默认为False，产生新对象，原DataFrame对象内容不变。

**【例】**见示例。



## 4.3 DataFrame的数据查询与编辑

### ◆4.3.2 DataFrame数据的编辑

#### 2. 删除数据

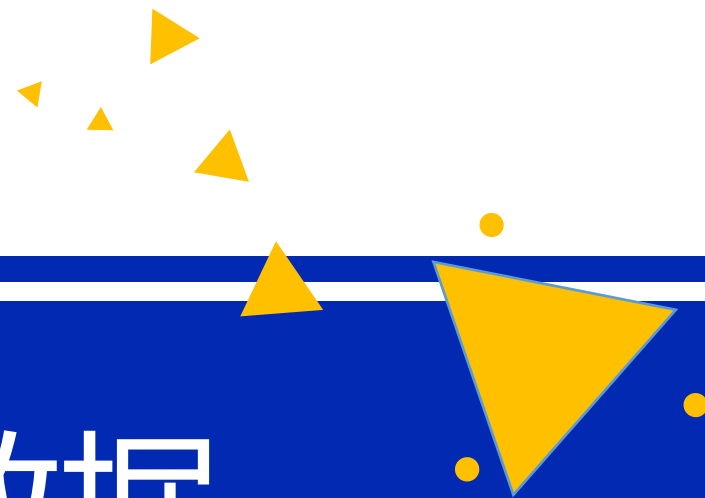
**删除数据**直接用drop方法，通过**axis参数**确定是删除的是行还是列。  
默认数据删除不修改原数据，需要在原数据删除行列需要设置参数  
inplace = True。

【例 4-33】删除数据行。

```
In[33]: print(df5.drop('广州'))
```

```
Out[33]:
```

	name	sex	year	age	C
city					
北京	张三	female	2001	20	85
上海	李四	female	2001	20	78
北京	小明	male	2002	20	80



# Pandas读取外部数据

02



# 读取文本文件

## 1. 文本文件读取

- 文本文件是一种由若干行字符构成的计算机文件，它是一种典型的顺序文件。
- csv是一种逗号分隔的文件格式，因为其分隔符不一定是逗号，又被称为字符分隔文件，文件以纯文本形式存储表格数据（数字和文本）。

# 读取文本文件

## 1.文本文件读取

```
pandas.read_csv(filepath_or_buffer, sep=',', header='infer', names=None, index_col=None, dtype=None, engine=None, nrows=None)
```

参数名称	说明
filepath	接收string。代表文件路径。无默认。
sep	接收string。代表分隔符。read_csv默认为 “,” ， read_table默认为制表符 “[Tab]” 。
header	接收int或sequence。表示将某行数据作为列名。默认为infer，表示自动识别。
names	接收array。表示列名。默认为None。
index_col	接收int、sequence或False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values） 。默认为None。
engine	接收c或者python。代表数据解析引擎。默认为c。

# 读取文本文件

## 1. 文本文件读取

- sep参数是指定文本的分隔符的，如果分隔符指定错误，在读取数据的时候，每一行数据将连成一片。
- header参数是用来指定列名的，如果是None则会添加一个默认的列名。
- encoding代表文件的编码格式，常用的编码有utf-8、utf-16、gbk、gb2312、gb18030等。如果编码指定错误数据将无法读取，IPython解释器会报解析错误。

# 读取excel

## 1.Excel文件读取

- pandas提供了read\_excel函数来读取“xls”“xlsx”两种Excel文件。
- *pandas.read\_excel(io, sheetname=0, header=0, index\_col=None, names=None, dtype=None)*

参数名称	说明
io	接收string。表示文件路径。无默认。
sheet_name	接收string、int。代表excel表内数据的分表位置。默认为0。
header	接收int或sequence。表示将某行数据作为列名。默认为infer，表示自动识别。
names	接收int、sequence或者False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
index_col	接收int、sequence或者False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values）。默认为None。

## 导入.xls或.xlsx文件

常用参数说明：

- ✓ io：字符串，.xls或.xlsx文件路径或类文件对象。
- ✓ sheet\_name：None、字符串、整数、字符串列表或整数列表，默认值为0。字符串用于工作表名称，整数为索引表示工作表位置，字符串列表或整数列表用于请求多个工作表，为None时获取所有工作表。

## 2. Excel文件储存

- 将文件存储为Excel文件，可以使用to\_excel方法。其语法格式如下。

*DataFrame.to\_excel(excel\_writer=None, sheetname=None, na\_rep="", header=True, index=True, index\_label=None, mode='w', encoding=None)*

- to\_csv方法的常用参数基本一致，区别之处在于指定存储文件的文件路径参数名称为excel\_writer，并且没有sep参数，增加了一个sheetnames参数用来指定存储的Excel sheet的名称，默认为sheet1。

# sheet\_name参数值

值	说 明
sheet_name=0	第一个Sheet页中的数据作为DataFrame
sheet_name=1	第二个Sheet页中的数据作为DataFrame
sheet_name="Sheet1"	名为Sheet1的Sheet页中的数据作为DataFrame
sheet_name=[0,1,'Sheet3']	第一个、第二个和名为Sheet3的Sheet页中的数据作为DataFrame



【示例12】 导入Excel文件。

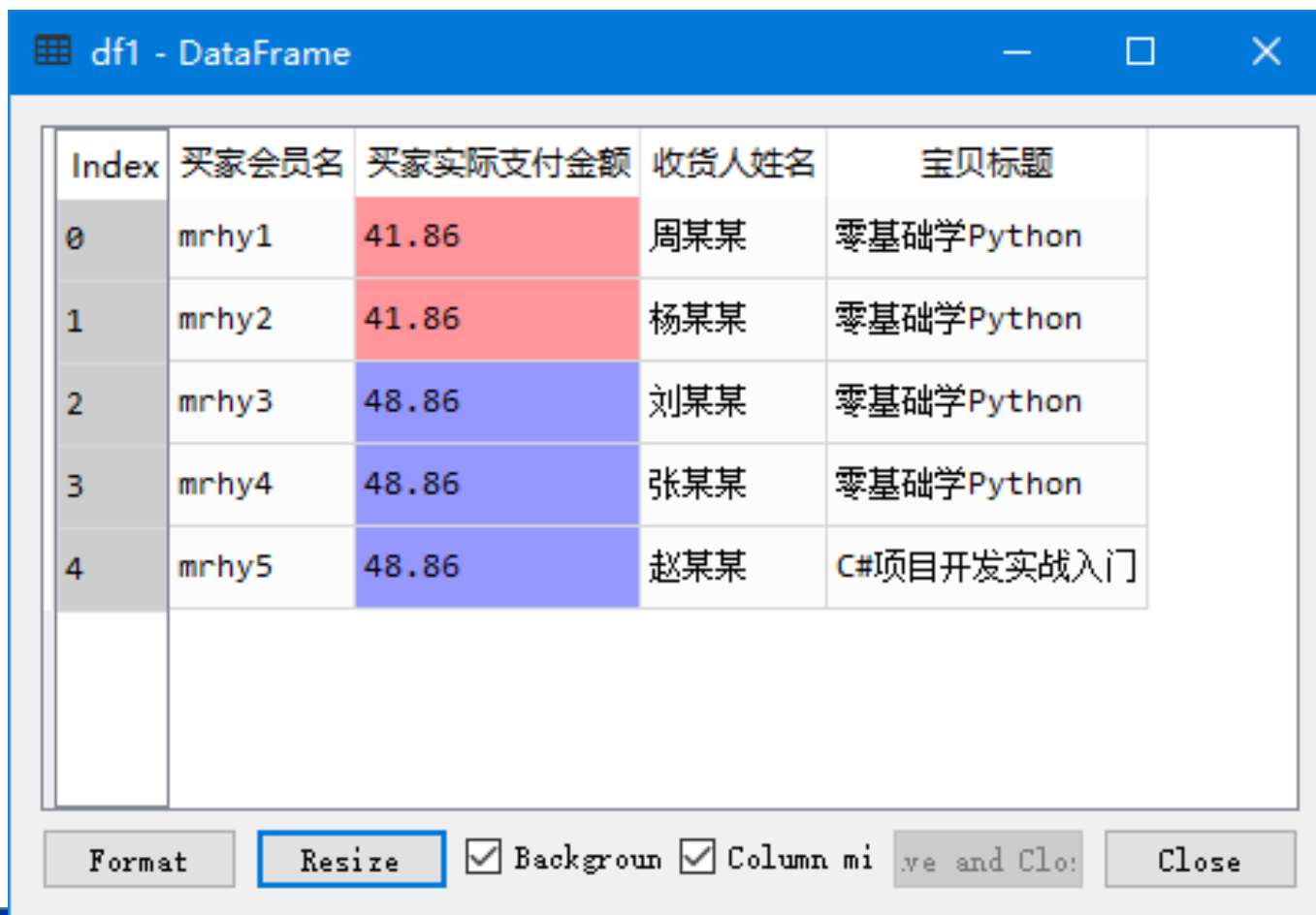
导入“1月.xlsx”Excel文件， 程序代码如下：

```
01  import pandas as pd
02  df=pd.read_excel('1月.xlsx')
03  df1=df.head()
```

运行程序， 输出前5条数据， 结果如图所示。



# 读取 1月淘宝销售数据（前5条数据）



df1 - DataFrame

Index	买家会员名	买家实际支付金额	收货人姓名	宝贝标题
0	mrhy1	41.86	周某某	零基础学Python
1	mrhy2	41.86	杨某某	零基础学Python
2	mrhy3	48.86	刘某某	零基础学Python
3	mrhy4	48.86	张某某	零基础学Python
4	mrhy5	48.86	赵某某	C#项目开发实战入门

Format Resize ☒ Background ☒ Column mi ve and Clo: Close