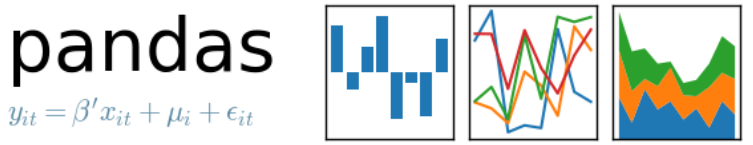
第三章 Pandas

目录

Pandas简介



Pandas(Python Data Analysis Library)

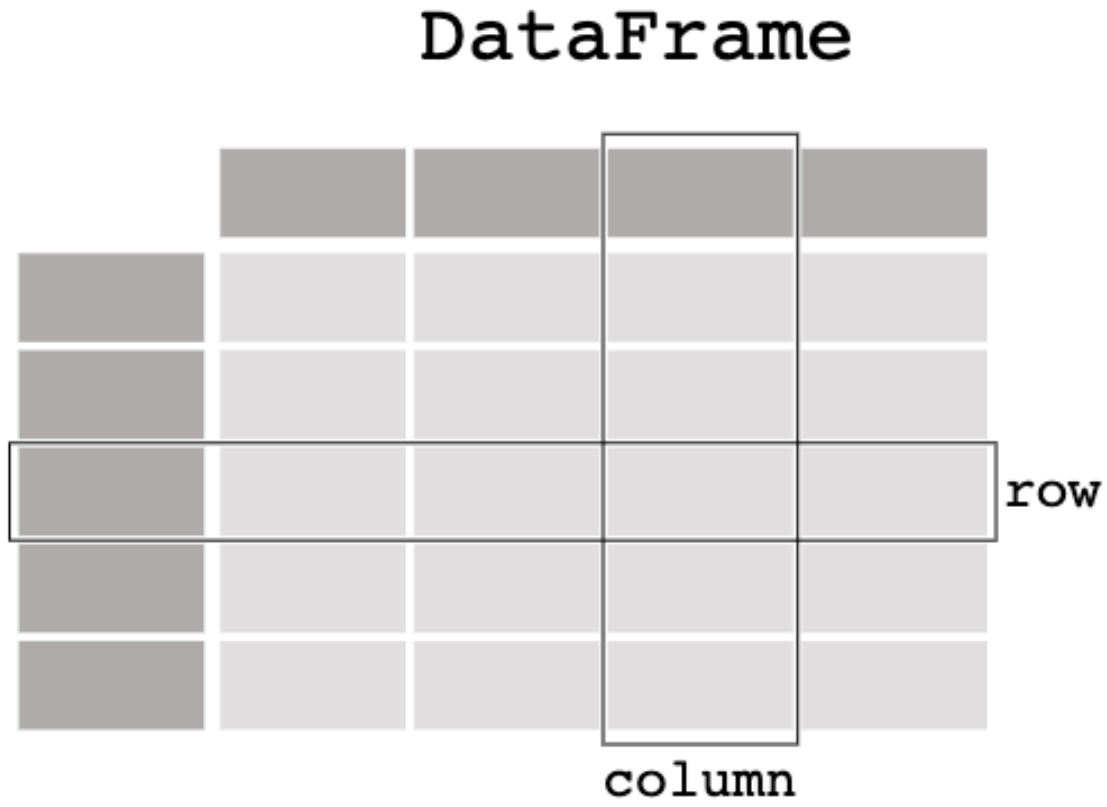
- 强大的Python数据分析工具包。
- 是一个开源的，BSD许可的库，为Python编程语言提供高性能，易于使用的数据结构和数据分析工具。

Pandas用于广泛的领域，包括金融，经济，统计，分析等学术和商业领域。

- Pandas的官方网站是： http://pandas.pydata.org/

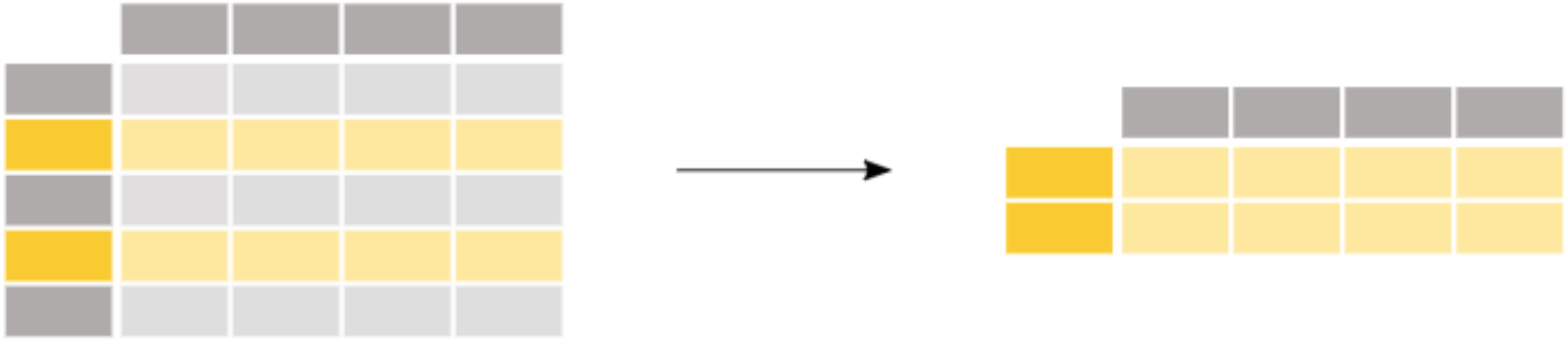
Pandas简介

What kind of data does pandas handle?

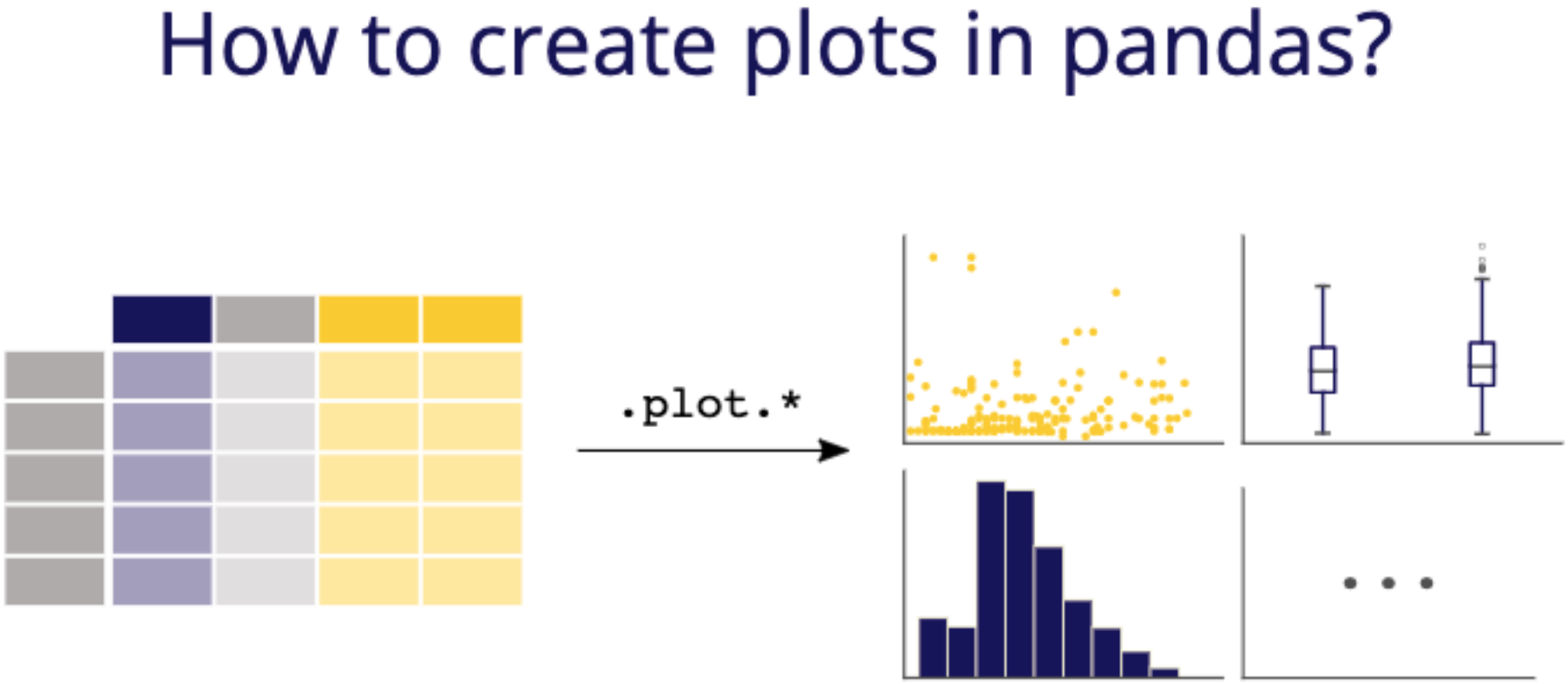


Pandas简介

How do I select a subset of a `DataFrame`?

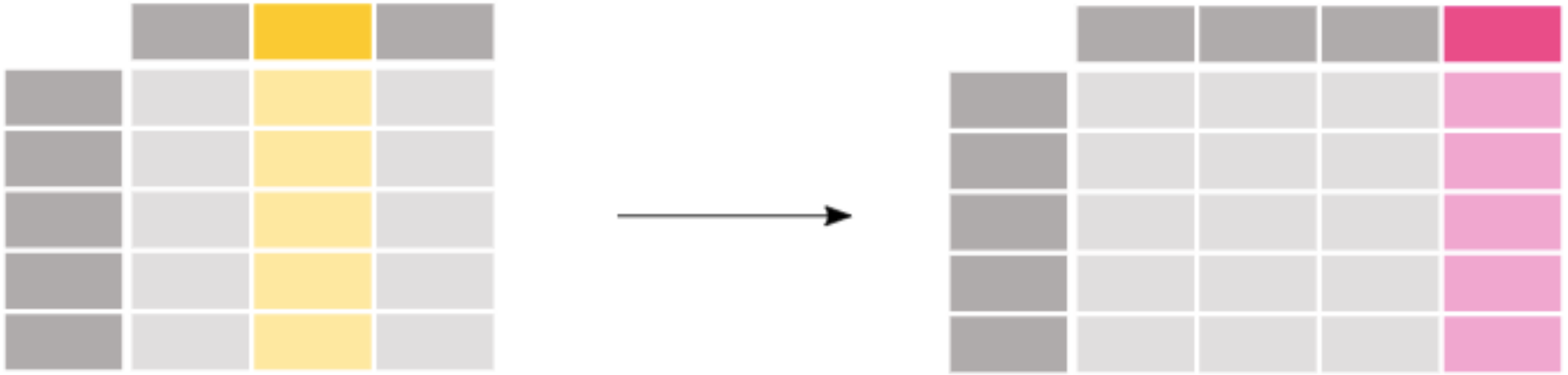


Pandas简介



Pandas简介

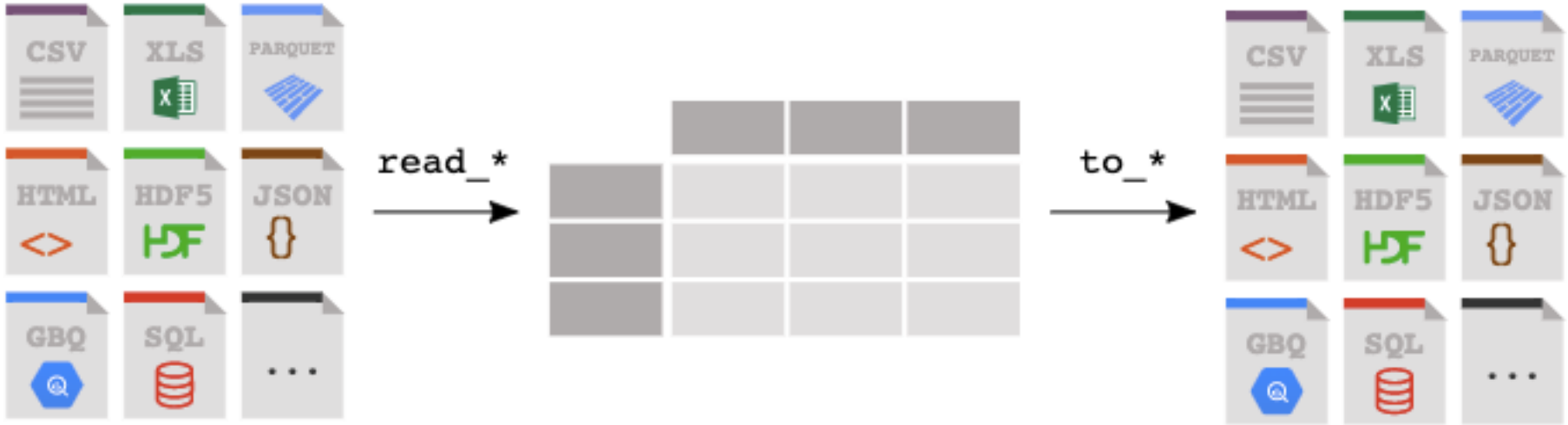
How to create new columns derived from existing columns?

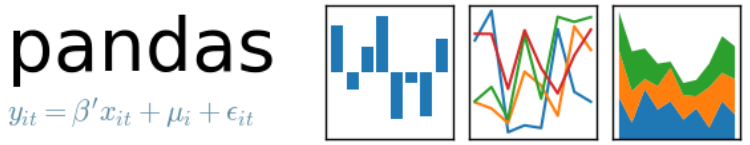


Pandas简介

How to calculate summary statistics?

How do I read and write tabular data?





Pandas/NumPy

多数据类型	数值型
数据分析	矩阵运算
有索引	无索引
Series、DataFrame	ndarray
数值型和非数值型统计	只能数值型进行统计

Pandas简介

- Pandas的主要特点：
  - 快速高效的DataFrame对象，具有默认和自定义的索引。
  - 将数据从不同文件格式加载到内存中的数据对象的工具。
  - 丢失数据的数据对齐和综合处理。
  - 基于标签的切片，索引和大数据集的子集。
  - 可以删除或插入来自数据结构的列。
  - 按数据分组进行聚合和转换。
  - 高性能合并和数据加入。
  - 时间序列功能。

Pandas安装

安装：

pip install pandas

```
import pandas as pd

print(pd.show_versions())
```

目录

Pandas基本数据类型

- Pandas主要提供了2种数据结构：
  - **Series**，带标签的一维数组；
  - **DataFrame**，带标签且大小可变的二维表格结构；

- DataFrame是Series的容器。

Pandas基本数据类型-系列(Series)

- 系列(Series)是一维标记的数组，能够保存任何数据类型（整数，字符串，浮点数，Python对象等）。
- 注意：
  - 关键点均匀数据
  - 数据的值可变

10	23	56	17	52	61	73	90	26	72

Pandas基本数据类型-系列(Series)

- Pandas系列可以使用以下构造函数创建
  - pandas.Series(data, index, dtype)
  - data：数据采取各种形式，如：ndarray，list，constants
  - index：必须是唯一的和散列的，与数据的长度相同。 默认np.arange(n)如果没有索引被传递。
  - dtype用于数据类型。如果没有，将推断数据类型

```
import pandas as pd

print(pd.Series())
```

Pandas基本数据类型-系列(Series)

- 从ndarray数据创建Pandas系列，则传递的索引必须具有相同的长度。 如果没有传递索引值，那么默认的索引将是范围(n)，其中n是数组长度。

```
import pandas as pd
import numpy as np

data = np.array(['a', 'b', 'c', 'd'])
s = pd.Series(data)
print(s)
s = pd.Series(data, index=[100, 101, 102, 103])
print(s)
```

Pandas基本数据类型-系列(Series)

- 从字典(dict)数据创建Pandas系列，字典作为输入传递，如果没有指定索引，则按排序顺序取得字典键以构造索引。 如果传递了索引，索引中与标签对应的数据中的值将被拉出。

```
import pandas as pd
import numpy as np

data = {'a': 0., 'b': 1., 'c': 2.}
s = pd.Series(data)
print(s)
s = pd.Series(data, ['b', 'c', 'd', 'a'])
print(s)
```

Pandas基本数据类型-系列(Series)

- 从标量数据创建Pandas系列，数据是标量值，则必须提供索引。将重复该值以匹配索引的长度。

```
import pandas as pd
import numpy as np

s = pd.Series(5, index=[0, 1, 3])
print(s)
```

Pandas基本数据类型-系列(Series)

Series是类似ndarray的数据类型-切片

```
import pandas as pd
import numpy as np
s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'])
# retrieve the first element
print(s[0])
# retrieve the first three element
print(s[:3])
# retrieve the last three element
print(s[-3:])
# retrieve a single element
print(s['a'])
# retrieve multiple elements
print(s[['a', 'c', 'd']])
```

Pandas基本数据类型-系列(Series)

Series是类似ndarray的数据类型-dtype

```
import pandas as pd
import numpy as np
s.dtype
```

```
dtype('int64')
```

Pandas基本数据类型-系列(Series)

Series是类似ndarray的数据类型-dtype

也可以传递到大多数期待ndarray的NumPy方法。

```
import pandas as pd
import numpy as np
print(s + s)
print(s.mean())
```

Pandas基本数据类型-系列(Series)

Series是类似ndarray的数据类型-dtype

也可以传递到大多数期待ndarray的NumPy方法。

Series和ndarray主要区别:Series会根据标签自动对齐数据。

```
import pandas as pd
import numpy as np
print(s[1:] + s[:-1])
```

Pandas基本数据类型-系列(Series)

Series是类似dict的数据类型

```
import pandas as pd
import numpy as np
data = np.array([1, 2, 3])
s = pd.Series(data, index = [‘a’, ‘b’, ‘c’])
print(s[‘a’])
```

Pandas基本数据类型-数据帧(DataFrame)

- 数据帧(DataFrame)是一个具有异构数据的二维数组。 例如：
  - 姓名（字符串）
  - 年龄（整数）
  - 性别（字符串）
  - 等级（整数）
- 注意：
  - 异构数据
  - 大小可变
  - 数据可变

姓名	年龄	性别	等级
Maxsu	25	男	4
Katie	34	女	3
Vina	46	女	2
Lia	12	女	3

Pandas基本数据类型-数据帧(DataFrame)

- 数据帧(DataFrame)是二维数据结构，即数据以行和列的表格方式排列。
- 功能：
  - 潜在的列是不同的类型
  - 大小可变
  - 标记轴(行和列)
  - 可以对行和列执行算术运算

Pandas基本数据类型-数据帧(DataFrame)

- Pandas中的DataFrame可以使用以下构造函数创建
  - pandas.DataFrame(data, index, columns, dtype)
  - 参数如下：
    - data数据采取各种形式，如:ndarray，series，map，lists，dict，constant和另一个DataFrame。
    - index对于行标签，要用于结果帧的索引是可选缺省值np.arrange(n)，如果没有传递索引值。
    - columns对于列标签，可选的默认语法是 - np.arange(n)。 这只有在没有索引传递的情况下才是这样。
    - dtype每列的数据类型。

Pandas基本数据类型-数据帧(DataFrame)

- Pandas数据帧(DataFrame)可以使用各种数据进行创建：
  - 列表
  - 字典
  - 系列
  - NumPy ndarray
  - Pandas DataFrame

```
import pandas as pd
import numpy as np

df = pd.DataFrame()#empty DataFrame
print(df)
data = [1, 2, 3, 4, 5]
df = pd.DataFrame(data)  #from list
print(df)
data = [['Alex', 10], ['Bob', 12], ['Clarke', 13]]
df = pd.DataFrame(data, columns=['Name', 'Age']) # from list
print(df)
data = {'Name': ['Tom', 'Jack', 'Steve'], 'Age': [28, 34, 29]} #from dict 列
df = pd.DataFrame(data)
print(df)
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]  #from dict 行
df = pd.DataFrame(data)
print(df)
d = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two': pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)  # from Series
print(df)
```

Pandas基本数据类型-数据帧(DataFrame)

- Pandas数据帧(DataFrame)的常见访问方式：
  - 列选择
  - 列添加
  - 列删除
  - 行选择，添加和删除

```
import pandas as pd
import numpy as np

d = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two': pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)
print(df['one'])
df['three'] = pd.Series([10, 20, 30], index=['a', 'b', 'c'])
print(df)
df['four'] = df['one'] + df['three']
print(df)
del df['one']
print(df)
```

Pandas基本数据类型-数据帧(DataFrame)

- Pandas数据帧(DataFrame)的常见访问方式：
  - 列选择
  - 列添加
  - 列删除
  - 行选择，添加和删除

```
import pandas as pd

d = {'one': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two': pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)
print(df.loc['b'])
print(df.iloc[0])
print(df[2:4])
df = df.append(pd.DataFrame([[5, 6], [7, 8]], columns=['one',
'two']))
print(df)
df = df.drop(0)
print(df)
```

- 交换dataframe的第二第三两行

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

	0	1	2	3	4
0	0	1	2	3	4
1	10	11	12	13	14
2	5	6	7	8	9
3	15	16	17	18	19
4	20	21	22	23	24

```
df = pd.DataFrame(np.arange(25).reshape(5, -1))

a, b = df.iloc[1].copy(), df.iloc[2].copy()

df.iloc[1], df.iloc[2] = b, a
```

目录

Series基本操作

- Series的基本功能示例

```
import pandas as pd
import numpy as np

data = pd.Series(np.random.randint(0,4,5)
print(data)
print(data.axes)
print(data.empty)
print(data.ndim)
print(data.size)
print(data.values)
print(data.head(3))
print(data.tail(2))
```

Series基本操作

- Series的基本功能如下：

编号	属性或方法	描述
1	axes	返回行轴标签列表。
2	dtype	返回对象的数据类型(dtype)。
3	empty	如果系列为空，则返回True。
4	ndim	返回底层数据的维数，默认定义：1。
5	size	返回基础数据中的元素数。
6	values	将系列作为ndarray返回。
7	head()	返回前n行。
8	tail()	返回最后n行。

```
df=pd.DataFrame(
    {
        'a':range(100),
        'b':np.random.rand(100),
        'c':[1,2,3,4]*25,
        'd':['apple', 'banana', 'carrot']*33 + ['apple']
    }
)
```

```
df.rename(columns={'d':'fruit'})
```

DataFrame基本操作

- DataFrame的基本功能示例：

```
import pandas as pd
# Create a Dictionary of series
d = {'Name': pd.Series(['Tom', 'James', 'Ricky', 'Vin', 'Steve', 'Minsu', 'Jack']),
     'Age': pd.Series([25, 26, 25, 23, 30, 29, 23]),
     'Rating': pd.Series([4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8])}
# Create a DataFrame
data = pd.DataFrame(d)
print(data)
print(data.T)
print(data.axes)
print(data.dtypes)
print(data.empty)
print(data.ndim)
print(data.shape)
print(data.size)
print(data.values)
print(data.head(3))
print(data.tail(2))
```

DataFrame基本操作

- DataFrame的基本功能如下：

编号	属性或方法	描述
1	T	转置行和列。
2	axes	返回一个列，行轴标签和列轴标签作为唯一的成员。
3	dtypes	返回此对象中的数据类型(dtypes)。
4	empty	如果NDFrame完全为空[无项目]，则返回为True; 如果任何轴的长度为0。
5	ndim	轴/数组维度大小。
6	shape	返回表示DataFrame的维度的元组。
7	size	NDFrame中的元素数。
8	values	NDFrame的Numpy表示。
9	head()	返回开头前n行。
10	tail()	返回最后n行。

- 用0填充dataframe的对角线上的数

	0	1	2	3	4	5	6	7	8	9
0	0	30	53	36	24	55	59	41	91	0
1	23	0	72	97	39	44	3	46	0	93
2	97	3	0	39	92	85	74	0	31	39
3	93	4	23	0	29	51	0	4	24	79
4	40	57	22	32	0	0	79	94	95	56
5	24	8	68	71	0	0	53	57	33	13
6	23	73	26	0	18	22	0	34	4	15
7	37	69	0	12	51	33	22	0	99	83
8	65	0	53	67	41	21	26	34	0	94
9	0	28	39	99	37	81	45	78	48	0

```
df = pd.DataFrame(np.random.randint(1,100, 100).reshape(10, -1))

for i in range(df.shape[0]):
    df.iat[i, i] = 0
    df.iat[df.shape[0]-i-1, i] = 0
```

Pandas描述性统计操作

- Pandas常用的描述性统计信息的函数：

编号	函数	描述
1	count()	非空观测数量
2	sum()	所有值之和
3	mean()	所有值的平均值
4	median()	所有值的中位数
5	mode()	值的模值
6	std()	值的标准偏差
7	min()	所有值中的最小值
8	max()	所有值中的最大值
9	abs()	绝对值
10	prod()	数组元素的乘积
11	cumsum()	累计总和
12	cumprod()	累计乘积

Pandas描述性统计操作

- Pandas提供了描述性方法来进行数据的统计操作：
  - sum()方法返回所请求轴的值的总和。 默认情况下，轴为索引(axis=0)。
  - mean() 返回平均值。
  - std() 返回数字列的Bressel标准偏差。

```
import pandas as pd
# Create a Dictionary of series
d = {'Name': pd.Series(['Tom', 'James', 'Ricky', 'Vin', 'Steve', 'Minsu', 'Jack',
                        'Lee', 'David', 'Gasper', 'Betina', 'Andres']),
     'Age': pd.Series([25, 26, 25, 23, 30, 29, 23, 34, 40, 30, 51, 46]),
     'Rating': pd.Series([4.23, 3.24, 3.98, 2.56, 3.20, 4.6, 3.8, 3.78, 2.98,
4.80, 4.10, 3.65])}
# Create a DataFrame
data = pd.DataFrame(d)
print(data.sum())
print(data.mean())
print(data.std())
```

Pandas描述性统计操作

- Pandas 描述性统计函数，注意事项：
  - 由于DataFrame是异构数据结构。通用操作不适用于所有函数。
  - 类似于：sum()，cumsum()函数能与数字和字符(或)字符串数据元素一起工作，不会产生任何错误。
  - 由于这样的操作无法执行，因此，当DataFrame包含字符或字符串数据时，像abs()，cumprod()这样的函数会抛出异常。

Pandas函数应用

- 要将自定义或其他库的函数应用于Pandas对象，有三种方式：
  - pipe()：表格函数应用，通过将函数和适当数量的参数作为管道参数来执行自定义操作，对整个DataFrame执行操作。
  - **apply( )**：可以沿DataFrame的轴应用任意函数，它与描述性统计方法一样，采用可选的axis参数。
  - applymap() ：给DataFrame的所有元素应用任何Python函数，并且返回单个值。

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), columns=['col1',
'col2', 'col3'])
print(df)
df = df.apply(np.mean)
print(df)
```

- dataframe获取行之和大于100的数据, 并返回最后的两行

```
df = pd.DataFrame(np.random.randint(10, 40, 60).reshape(-1, 4))
rowsums = df.apply(np.sum, axis=1)
last_two_rows = df.iloc[np.where(rowsums > 100)[0][-2:]]
last_two_rows
```

Pandas数据的迭代访问

- Pandas对象之间的基本迭代的行为取决于类型。当迭代一个系列时，它被视为数组式，基本迭代产生这些值。
  - Series - 值
  - DataFrame - 列标签

```
import pandas as pd
import numpy as np
N = 5
df = pd.DataFrame({
    'X': np.linspace(0, stop=N - 1, num=N),
    'Y': np.random.rand(N),
    'C': np.random.choice(['Low', 'Medium', 'High'], N)
       .tolist(),
})
for key, value in df.iteritems():  # 按列访问值
    print(key, value)
for row_index, row in df.iterrows():  # 按行访问值
    print(row_index, row)
for row in df.itertuples():  # 按行访问值
    print(row)
```

Pandas中的数据排序

- Pandas中有两种排序方式：
  - 按标签排序：sort_index()方法通过传递axis参数和排序顺序，可以对DataFrame进行排序。ascending=true为升序，false为降序。axis=0排序行，1为排序列。
  - 按实际值：sort_values()是按值排序的方法。它接受一个by参数，指定排序列名。

```
import pandas as pd
import numpy as np

unsorted_df = pd.DataFrame(np.random.randn(10, 2),
                           index=[1, 4, 6, 2, 3, 5, 9, 8, 0, 7],
                           columns=['col2', 'col1'])
print(unsorted_df)
sorted_df = unsorted_df.sort_index()
print(sorted_df)  # 按索引排序
sorted_df = unsorted_df.sort_values(by='col1')
print(sorted_df)  # 按col1排序
```

目录

Pandas函数应用

- Pandas中提供Series和DataFrame两种最常用的数据类型，并且集成了大量的函数可以对这些数据进行操作。函数主要分为：
  - 字符串文本函数
  - 统计函数
  - 聚合函数
  - 分组函数
  - …

Pandas的字符串文本函数

- 常用字符串文本函数列表如下。

编号	函数	描述
1	lower()	将Series/Index中的字符串转换为小写。
2	upper()	将Series/Index中的字符串转换为大写。
3	len()	计算字符串长度。
4	strip()	帮助从两侧的系列/索引中的每个字符串中删除空格(包括换行符)。
5	split(' ')	用给定的模式拆分每个字符串。
6	cat(sep=' ')	使用给定的分隔符连接系列/索引元素。
7	get_dummies()	返回具有单热编码值的数据帧(DataFrame)。
8	contains(pattern)	如果元素中包含子字符串，则返回每个元素的布尔值True，否则为False。
9	replace(a,b)	将值a替换为值b。
10	repeat(value)	重复每个元素指定的次数。
11	count(pattern)	返回模式中每个元素的出现总数。
12	startswith(pattern)	如果系列/索引中的元素以模式开始，则返回true。
13	endswith(pattern)	如果系列/索引中的元素以模式结束，则返回true。
14	find(pattern)	返回模式第一次出现的位置。
15	findall(pattern)	返回模式的所有出现的列表。
16	swapcase	变换字母大小写。
17	islower()	检查系列/索引中每个字符串中的所有字符是否小写，返回布尔值
18	isupper()	检查系列/索引中每个字符串中的所有字符是否大写，返回布尔值
19	isnumeric()	检查系列/索引中每个字符串中的所有字符是否为数字，返回布尔值。

Pandas的字符串文本函数

- 示例

```
import pandas as pd
import numpy as np

s = pd.Series(['Tom', 'William Rick', 'John',
'Alber@t', np.nan, '1234', 'SteveMinsu'])

print(s.str.lower())
print(s.str.upper())
print(s.str.len())
print(s.str.find('e'))
print(s.str.count('m'))
```

Pandas中索引和选择数据函数

- Pandas现在支持三种类型的多轴索引：
  - DataFrame.loc() 方法通过标签来完成DataFrame的索引。
    - 单个标量标签
    - 标签列表
    - 切片对象
    - 布尔数组

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
                  index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'],
                  columns=['A', 'B', 'C', 'D'])

# select all rows for a specific column
print(df.loc[:, 'A'])
print(df.loc[:, ['A', 'C']])
print(df.loc[['a', 'b', 'f', 'h'], ['A', 'C']])
print(df.loc['a':'h'])
print(df.loc['a'] > 0)
```

Pandas中索引和选择数据函数

- Pandas现在支持三种类型的多轴索引：
  - DataFrame.iloc() 方法通过基于始0的下标来完成DataFrame的索引。
    - 整数
    - 整数列表
    - 系列值

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
                  index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'],
                  columns=['A', 'B', 'C', 'D'])

# select all rows for a specific column
print(df.iloc[:4])
print(df.iloc[1:5, 2:4])
print(df.iloc[[1, 3, 5], [1, 3]])
print(df.iloc[1:3, :])
print(df.iloc[:, 1:3])
```

Pandas中索引和选择数据函数

- Pandas现在支持三种类型的多轴索引：
  - DataFrame.ix() 方法通过混合标签和下标的方式来完成DataFrame的索引。

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
                  index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'],
                  columns=['A', 'B', 'C', 'D'])

# select all rows for a specific column
print(df.ix[:4])
print(df.ix[:, 'A'])
```

Pandas中索引和选择数据函数

- Pandas还支持通过属性运算符.来选择列。

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
                  index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'],
                  columns=['A', 'B', 'C', 'D'])

print(df.A)
```

Pandas统计函数

- 统计方法有助于理解和分析数据的行为。Pandas也提供了统计函数。
  - 差分函数：pct_change( ) 函数将每个元素与其前一个元素进行比较，并计算变化百分比。
  - 协方差函数：协方差适用于系列数据。Series对象有cov( )方法用来计算序列对象之间的协方差。NA将被自动排除。
  - 相关性函数： corr( )用于计算某两列值的相关性。
  - 数据排名函数：rank( ) 用于为元素数组中的每个元素生成排名。

Pandas统计函数

- Pandas统计函数示例:

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(3, 2), columns=['a', 'b'])
print(df.pct_change())
s1 = pd.Series(np.random.randn(10))
s2 = pd.Series(np.random.randn(10))
print(s1.cov(s2))

print(df['a'].cov(df['b']))  # a列和b列的协方差
print(df.cov())  # df协方差矩阵

print(df['a'].corr(df['b']))# a列和b列的相关性
print(df.corr()) #df的相关矩阵

s = pd.Series(np.random.np.random.randn(6), index=list('abcdee'))
print(s)
print(s.rank())
```

Pandas分组（GroupBy）

- 任何分组(groupby)操作都涉及原始对象的以下操作
  - 根据指定条件分割对象集合：df.groupby('key')
  - 在每个分割后的集合上应用函数：聚合函数，转换函数，过滤函数
  - 整合结果并显示

Pandas分组（GroupBy）

- 分组(groupby)函数示例

```
import pandas as pd
import numpy as np
ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
                     'Kings', 'Kings', 'Kings', 'Riders', 'Royals',
                     'Royals', 'Riders'],
            'Rank': [1, 2, 2, 3, 3, 4, 1, 1, 2, 4, 1, 2],
            'Year': [2014, 2015, 2014, 2015,
                     2014, 2015, 2016, 2017, 2016, 2014, 2015, 2017],
            'Points': [876, 789, 863, 673, 741, 812, 756, 788, 694, 701, 804,
690]}
df = pd.DataFrame(ipl_data)
grouped = df.groupby('Team')
for name, group in grouped:
    print(name, group)
print(grouped['Rank'].agg(np.mean))
```

Pandas IO函数

- Pandas提供了方便的数据获取方法，可以直接从外部数据源得到DataFrame类型的数据。
- 读取文本文件(或平面文件)的两个主要功能是read_csv()和read_table()。它们都使用相同的解析代码来智能地将表格数据转换为DataFrame对象。
  - pandas.read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer',names=None, index_col=None, usecols=None)

```
import pandas as pd
import numpy as np
df = pd.read_csv("temp.csv")
df = pd.read_csv("temp.csv", index_col=['S.No'])
df = pd.read_csv("temp.csv", dtype={'Salary': np.float64})
df = pd.read_csv("temp.csv", names=['a', 'b', 'c', 'd', 'e'], header=0)
df = pd.read_csv("temp.csv", skiprows=2)
print(df)
```

课堂实验

实验一

- 请分析不同国家酒消费的数据集
  - 打印每列每个大洲的平均饮酒量

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
2	Afghanistan	0	0	0	0	AS
3	Albania	89	132	54	4.9	EU
4	Algeria	25	0	14	0.7	AF
5	Andorra	245	138	312	12.4	EU
6	Angola	217	57	45	5.9	AF
7	Antigua & Barbuda	102	128	45	4.9	NA
8	Argentina	193	25	221	8.3	SA
9	Armenia	21	179	11	3.8	EU
10	Australia	261	72	212	10.4	OC
11	Austria	279	75	191	9.7	EU
12	Azerbaijan	21	46	5	1.3	EU
13	Bahamas	122	176	51	6.3	NA
14	Bahrain	42	63	7	2	AS
15	Bangladesh	0	0	0	0	AS
16	Barbados	143	173	36	6.3	NA

Exercise 1.py

实验二

- 根据给出的美国儿童的名字数据集进行如下统计分析：
  - 分别统计男生(M)和女生(F)的名字个数
  - 统计一共有多少个不重复的名字

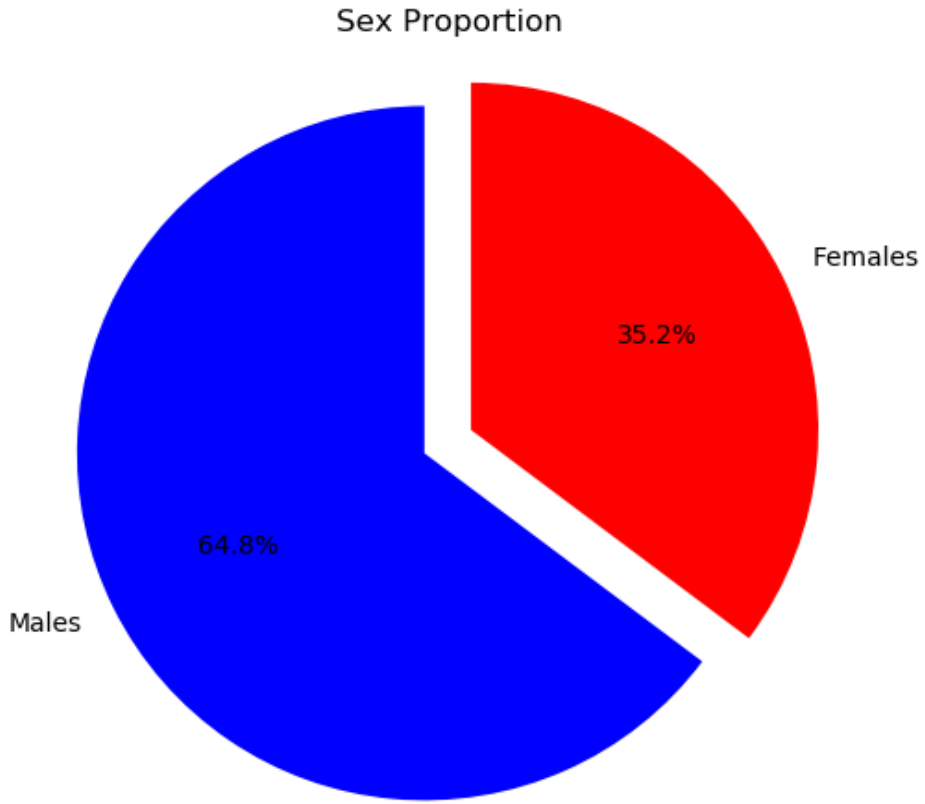
	A	B	C	D	E	F	G
1		Id	Name	Year	Gender	State	Count
2	11349	11350	Emma	2004	F	AK	62
3	11350	11351	Madison	2004	F	AK	48
4	11351	11352	Hannah	2004	F	AK	46
5	11352	11353	Grace	2004	F	AK	44
6	11353	11354	Emily	2004	F	AK	41
7	11354	11355	Abigail	2004	F	AK	37
8	11355	11356	Olivia	2004	F	AK	33
9	11356	11357	Isabella	2004	F	AK	30
10	11357	11358	Alyssa	2004	F	AK	29
11	11358	11359	Sophia	2004	F	AK	28
12	11359	11360	Alexis	2004	F	AK	27
13	11360	11361	Elizabeth	2004	F	AK	27
14	11361	11362	Hailey	2004	F	AK	27
15	11362	11363	Anna	2004	F	AK	26
16	11363	11364	Natalie	2004	F	AK	25
17	11364	11365	Sarah	2004	F	AK	25
18	11365	11366	Sydney	2004	F	AK	25
19	11366	11367	Ava	2004	F	AK	23

Exercise 2.py

实验三

- 对泰坦尼克幸存人员数据集进行下载并分析幸存人员比例，制作饼图。

	A	B	C	D	E	F	G	H	I	J	K	L
1	Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, M	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen,	female	26	0	0	STON/O2.	7.925		S
5	4	1	1	Futrelle, Mr	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. V	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, M	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, Mr	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstrom	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, Mi	female	58	0	0	113783	26.55	C103	S
14	13	0	3	Saunderco	male	20	0	0	A/5. 2151	8.05		S
15	14	0	3	Andersson	male	39	1	5	347082	31.275		S
16	15	0	3	Vestrom, M	female	14	0	0	350406	7.8542		S
17	16	1	2	Hewlett, M	female	55	0	0	248706	16		S
18	17	0	3	Rice, Maste	male	2	4	1	382652	29.125		Q
19	18	1	2	Williams, M	male		0	0	244373	13		S
20	19	0	3	Vander Pla	female	31	1	0	345763	18		S



Exercise 3.py

内容回顾

1. Pandas简介及安装
2. Pandas基本数据类型
3. Pandas基本操作
4. Pandas函数应用

Thank you !