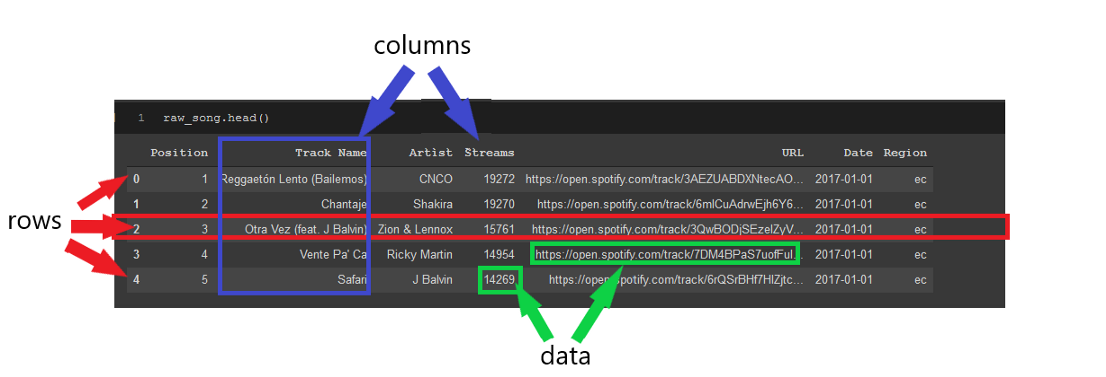
Pandas库基础知识

|  |  |  |
| --- | --- | --- |
|  | 文件标识： |  |
| 当前版本： | 1.0 |
| 作 者： | 马卫花 |
| 完成日期： | 2023-11 |

Python Data Analysis Library 或 pandas 是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas提供了大量能使我们快速便捷地处理数据的函数和方法。

import pandas as pd

Pandas API将Pandas数据帧定义为：二维、大小可变、潜在的异构表格数据。数据结构还包含轴（行和列）。算术运算对行标签和列标签进行对齐。可以认为是一个类似于dict的容器，用于存储序列对象。是Pandas主要的数据结构。



基本上，这意味着你有包含在格式中的数据，如下所示。在行和列中找到的数据：

数据帧非常有用，因为它们提供了一种高效的方法来可视化数据，然后按你希望的方式操作数据。

Pandas主要采用Series和DataFrame两种数据结构。Series是一种类似一维数据的数据结构，由数据(values)及索引(indexs)组成，而DataFrame是一个表格型的数据结构，它有一组序列，每列的数据可以为不同类型（NumPy数据组中数据要求为相同类型）,它既有行索引，也有列索引。

* Series

Series是一维带标签的数组，数组里可以放任意的数据(整数，浮点数，字符串， Python Object)。

s = pd.Series（data， index=index）

其中index是一个列表，用来作为数据的标签。

data的类型可以是Python字典、ndarray对象、一个标量值如3等。

|  |
| --- |
| a = np.array([1,2,3,4])  b = np.array([5,6,7,8])  c = np.array(['a','b','c','d'])  #转换为DataFrame结构  df = pd.DataFrame({'a':a,'b':b,'c':c})  print(df)  #创建Series  s = pd.Series([10,20,30,40,50])  print(s)  print('----------')  #Series的values  print(s.values)  #Series的索引值  print(s.index)  print('----------')  #定义Series的标签索引  s1 = pd.Series([10,20,30,40,50],index=['a','b','c','d','e'])  print(s1)  print('----数据访问------')  #数据访问  print(s1[2])  #通过索引值访问  print(s1['a'])  print('-------')  s2 = pd.Series([98.5,99.0,97.5,97,96,94],index=['admin','wang','li','a','b','c'])  print(s2)  print('---修改数据----')  s2['wang'] = 100.0  s2[0] = 99.5  print(s2)  #切片操作  print('----切片操作-----')  print(s2[1:3])  print('-----')  print(s2[:4])  #数据挖掘:数据缺失处理  #isnull:返回哪些值是缺失值  #dropna: 删掉缺失值  #fillna: 填充缺失值  print('---数据缺失处理---')  s3 =pd.Series([80,90,np.NaN,65],index=['a','b','c','d'])  #1.判断哪些值是缺失值  print(s3.isnull())  #2.删除缺失值  print(s3.dropna())  #print(s3)  #填充缺失值  print(s3.fillna(0)) |

创建Pandas数据帧的方法有几种：

* 从.csv文件导入数据（或其他文件类型，例如Excel、SQL数据库）
* 从列表中
* 从字典里
* 从numpy数组
* 其他

通常，你将主要将.csv文件或某种类型的数据源的数据放入Pandas数据框架中，而不是从头开始，因为这将需要非常长的时间来完成，这取决于你拥有的数据量。以下是python字典中的一个快速、简单的示例：

|  |
| --- |
| import pandas as pd  dict1 = {'Exercises': ['Running','Walking','Cycling'],  'Mileage': [250, 1000, 550]}  df = pd.DataFrame(dict1)  df |

输出：



字典键（“Exercises”和“Mileage”）成为相应的列标题。字典中的值是本示例中的列表，成为数据帧中的单个数据点。Running是“Exercises”列表中的第一个，250将被列在第二列的第一个。另外，你会注意到，由于没有为数据帧的索引指定标签，因此它会自动标记为0、1和2。

创建Pandas数据帧的最可能方法是从csv或其他类型的文件中

df = pd.read\_csv("file\_location.../file\_name.csv")

常用函数：

.head() & .tail() //它显示数据帧的前5行和每个列，以便你轻松地总结数据的外观。还可以在方法中指定一定数量的行，以显示更多行。

.info() //显示了所有列、它们的数据类型以及是否有null数据点

.describe() //显示关于这些列的许多描述性统计信息

计数（Count）： 数据的非缺失值数量。

均值（Mean）： 数据的平均值。

标准差（Standard Deviation）： 数据的标准偏差，用于衡量数据的离散程度。

最小值（Minimum）： 数据的最小值。

25th 百分位数（25th percentile）： 数据的第 25 百分位数，即四分之一位数。

中位数（Median）： 数据的中位数，即第 50 百分位数。

75th 百分位数（75th percentile）： 数据的第 75 百分位数，即四分之三位数。

最大值（Maximum）： 数据的最大值。

|  |
| --- |
| import numpy as np  import pandas as pd  #创建DataFrame对象  df = pd.DataFrame(np.arange(12).reshape(3,4),columns=['a1','a2','a3','a4'],  index=['a','b','c'])  print(df)  print('---获取结构属性---')  #获取DataFrame结构的属性  print(df.index)  #获取列名  print(df.columns)  #显示值  print(df.values)  print('-----将存储学生信息的字典转换为DataFrame-------')  data = {'name':['admin','system','root','user'],  'age':[18,15,17,14],  'addr':['anyang','luoyang','hebi','jiaozuo']  }  #df = pd.DataFrame(data)  #print(df)  df = pd.DataFrame(data,columns=['name','age','addr'],index =['a','b','c','d'])  print(df)   * 数据访问   #通过行索引和列索引访问  print('-----数据访问------')  #选取一列  print(df['name'])  print('~~~~~~~~')  print(df[['name']])  #选取多列  print(df[['name','addr']])  print('----行的选择----')  print(df['c':'d'])  print(df[1:3])  print('---过滤条件----')  #根据年龄进行过滤  print(df[df['age']>15])  #通过obj.loc[]或obj.iloc[]访问数据  #loc[]:通过标签获取行数据  #iloc[]:通过index的索引值获取数据  data=[[1,2,3],[4,5,6],[7,8,9]]  index =['a','b','c']  columns =['c1','c2','c3']  df = pd.DataFrame(data=data,columns=columns,index=index)  print(df)  print('---loc行标签获取数据--')  #根据行标签获取行数据  print(df.loc[['a','b']])  print('-----')  #根据行标签和列名获取数据  print(df.loc[['a','b'],['c1','c3']])  print('----iloc行索引值获取数据------')  print(df.iloc[1])  print('~~~~~~~~~~')  print(df.iloc[0:2])  #通过行号和列获取数据  print(df.iloc[1: ,1])  print(df.iloc[1:,[1,2]])  print('~~~~~~~~~~')  print(df.iloc[1:,1:3])  '''  '''  任务二：缺失值处理  isnull() :返回哪些值是缺失值  dropna(): 删掉缺失值  fillna(): 填充缺失值  '''  #数据样本  data={'one':[1,2,np.nan,3],  'two':[4,5,6,7],  'three':[8,9,10,11]}  df = pd.DataFrame(data,index=['a','b','c','d'])  print(df)  #print('---isnull----')  #print(df.isnull())  print('--dropna---')  #dropna():默认删除缺失值所在的行  # axis = 1 : 按列删除  # axis = 0 : 按行删除  #print(df.dropna())  print(df.dropna(axis=0))  print('--填充值---')  print(df.fillna(0)) |

* **Pandas库GroupBy常用操作**

|  |
| --- |
| '''  任务一：  pandas库groupby的使用  产品销售数据.csv文件  store,product\_group,product\_code,stock\_qty,cost,price,last\_week\_sales,last\_month\_sales  店铺,产品类型,产品编号,库存数量,成本价，销售价,上周销售数量，上个月的销售数量  #思考：将聚合后的数据用图表进行展示。  '''  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  sale\_df = pd.read\_csv(r'data/sales\_data.csv',encoding='utf-8')  print(sale\_df)  '''  #1.计算每个店铺的平均库存数量 (单列聚合)  # 特征列: store  # stock\_qty  print(sale\_df.groupby("store")["stock\_qty"].mean())  #2.计算每个店铺的平均库存数量以及产品的平均销售价格(多列聚合)  # 特征列: store  # stock\_qty,price  listofdata = ['stock\_qty','price']  print(sale\_df.groupby("store")[listofdata].mean())  #3.计算每个店铺的平均库存数量和最大的库存数量（聚合） aggregate  print(sale\_df.groupby("store")["stock\_qty"].agg(["mean","max"]) )  #4.对聚合结果进行重命名  print(sale\_df.groupby("store").agg(avg\_stock\_qy=("stock\_qty","mean"),max\_stock\_qy=("stock\_qty","max")) )  #5.计算每个店铺的平均库存数量、最大的库存数量、产品销售的均价以及产品销售最大值  listofdata =['stock\_qty','price']  print(sale\_df.groupby("store")[listofdata].agg(["mean","max"]) )  #6.计算每个店铺的平均库存数量、产品销售的均价(对不同列的聚合然后进行命名)  print(sale\_df.groupby("store").agg(avg\_stock\_qy=("stock\_qty","mean"),avg\_price=("price","mean")))  # as\_index:将分析的结果作为DataFrame的一列  # mean:计算均价 count:计数 max:最大值 min:最小值  resultDF = sale\_df.groupby("store",as\_index=False).agg(avg\_stock\_qy=("stock\_qty","mean"),avg\_price=("price","mean"))  print(type(resultDF))  #7.计算每个店铺不同产品类型的上周销量的均值  # 特征列: store(店铺),  # product\_group(产品类型)  # last\_week\_sales(上周的销售数量)  listofdata =["store","product\_group"]  print(sale\_df.groupby(listofdata,as\_index=False).agg(avg\_sales=("last\_week\_sales","mean")))  #8.排序输出 sort\_values  # 计算每个店铺不同产品类型的上周销量的均值,然后销量的均值进行降序排列  # sort\_values:默认升序排列 ascending=False 降序排列  listofdata =["store","product\_group"]  print(sale\_df.groupby(listofdata,as\_index=False).agg(avg\_sales=("last\_week\_sales","mean")))  print('--排序---')  print(sale\_df.groupby(listofdata,as\_index=False)  .agg(avg\_sales=("last\_week\_sales","mean"))  .sort\_values(by="avg\_sales",ascending=False))  '''  #9.TopN  #nlargest:最大值  #nsmallest  # 统计每个店铺的上周销量的最大值  print(sale\_df.groupby("store")["last\_week\_sales"].nlargest(2))  print('-----')  print(sale\_df.groupby("store")["last\_week\_sales"].nsmallest(1))  2.DataFrame合并示例  [课堂示例]  '''  任务二：DataFrame合并操作  '''  import pandas as pd  import numpy as np  leftDF = pd.DataFrame({"key1":['k0','k0','k1','k2'],  "key2":['k0','k1','k0','k1'],  'A':['A0','A1','A2','A3'],  "B":['B0','B1','B2','B3']})  print(leftDF)  print('---------')  rightDF = pd.DataFrame({'key1':['k0','k1','k1','k2'],  "key2":['k0','k0','k0','k0'],  'C':['C0','C1','C2','C3'],  'D':['D0','D1','D2','D3']})  print(rightDF)  print('------')  #合并left和right两个DataFrame  #pd.merge(left,right,how='inner',on=None)  # how:inner 内连接(等值连接)  # left:左连接  # right：右连接  #print(pd.merge(leftDF,rightDF,on=['key1','key2']))  #print(pd.merge(leftDF,rightDF,how='left',on=['key1','key2']))  print(pd.merge(leftDF,rightDF,how='right',on=['key1','key2'])) |