網路爬蟲與資料分析
Python資料程式設計

Instructor: 馬豪尚

Pandas

- Pandas是一個用進行資料處理或分析強大的工具
- Pandas提供了Series和DataFrame兩個資料結構物件，基於這兩個資料結構物件又提供了非常多的函數和工具方便做資料的處理和分析
- 使用Pandas模組
  - Import pandas as pd

Pandas Series

Pandas Series

- Series是一維的資料陣列，帶有index屬性
  - pd.Series(data [, index=索引])
  - data可以是list、tuple 、dictionary 、Numpy陣列
  - index屬性是選填，預設是整數的串列

- 用列表創建
  - se=pd.Series(['a','b','c','d','e'])
  - se=pd.Series(['a','b','c','d','e'], index=[1,2,3,4,5])

Pandas Series

- 用字典創建
  - dict={'西瓜':15, '香蕉':20, '水蜜桃':25}
  - se=pd.Series(dict)
  - 字典中的key會自動預設為series的index，而value就是series的值
- 結合Numpy創建Series
  - se=pd.Series(np.arrange(0, 7, 2))

Pandas Series

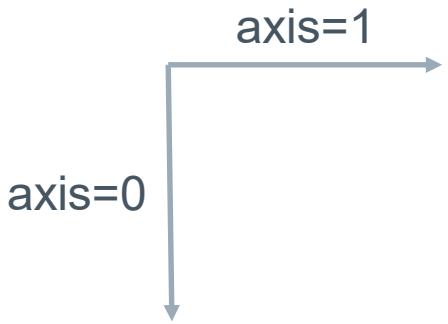
- Series取值
  - se[index]
  - se['index']
- Series取部分值
  - se[start:end]
- Series取得全部索引和全部值
  - se.index
  - se.values

Series 的運算

- 基本上可套用numpy的ndarray運算方法
- Series加減乘除(兩個series索引要相同)
  - se1 + se2
  - se1 – se2
  - se1 * se2
  - se1 / se2
- Series邏輯運算(兩個series索引要相同)
  - se1 > se2

Series 的運算

- 搭配numpy的運算方法
  - np.sin(se)
  - np.sum(se)
  - np.mean(se)
  - …
- 多個series串接
  - pd.concat([Series1, Series2, …], axis=0)
  - axis→定義series擺放方向(0為列方向, 1為欄方向)



Pandas DataFrame

Pandas DataFrame

- DataFrame是一種二維的資料結構，直覺上可以理解為類似Excel工作表
- 創建DataFrame物件
  - df_name = pd.DataFrame(data)
  - data為傳入的二維資料，例如:字典、字典的串列、Series的串列

DataFrame創建

- 創建DataFrame傳入串接的Series
  - df_name= pd.DataFrame(pd.concate([Series1, Series2, Series3], axis=1))

    Series 1→([1,2,3,4])
    Series 2→([5,6,7,8])
    Series 3→([9,1,2,3])

- columns設定欄位名稱
  - df_name.columns([“欄位1”, “欄位2”, “欄位3”, “欄位4”])

DataFrame創建

- 創建DataFrame傳入字典
  - cities = {'country':['China','Japan','Singapore'],
    'town':['Beijing','Tokyo','Singapore'],
    'population':[2000, 1600, 600]}
  - city_df = pd.DataFrame(cities)
  - 字典的Key會變成DataFrame的欄位名稱
  - 字典裡Value的串列型態轉換為一欄資料

DataFrame創建

- 創建DataFrame傳入字典的串列
  - fruit = [{'apple':50,'Orange':30,'Grape':80},
    {'apple':50,'Grape':80},
    {'apple':45, 'Orange':40}]
  - fruits_df = pd.DataFrame(fruit)
  - 字典的Key會變成DataFrame欄位名稱
  - 一個字典轉換為一列資料

DataFrame Index

- Index: 索引屬性，建立DataFrame的索引標籤
  - pd.DataFrame(data, index=索引)
    - 索引可以是一個"定義好的串列"或者是"某一欄位的資料"
    - 設定索引的時候要選適合當索引的資料，索引盡量不要有重複的值
- 創建DataFrame時用定義好的串列當作索引
  - cities = {'country':['China','Japan','Singapore'], 'town':['Beijing','Tokyo','Singapore'], 'population':[2000, 1600, 600]}
  - Indexlist=['first', 'second', 'third']
  - city_df = pd.DataFrame(cities, index= Indexlist)

DataFrame Index

- cities = {‘country’:[‘China’,‘Japan’,‘Singapore’], ‘town’:[‘Beijing’,‘Tokyo’,‘Singapore’], 'population':[2000, 1600, 600]}
- 創建DataFrame時用某一欄的資料做索引
  - city_df = pd.DataFrame(cities, index= country)
    - country這欄資料就會被當成索引使用
- 可以透過columns屬性選擇字典裡的key來創建
  - city_df = pd.DataFrame(cities, columns=[‘town’, ‘population’], index= country)
    - 這樣就只會選town和population來創建

DataFrame Index

- set_index: 設定 index
  - df_name.set_index(“索引欄位" , inplace=True)

- reset_index: 可以讓index重置成預設
  - df_name.reset_index(inplace=True)

讀取CSV到DataFrame

- pd.read_csv(path, sep, header, index_col, encoding, nrows, usecols)
  - path: csv檔案路徑
  - sep: 指定分隔字元，預設是’,’
  - header: 設定哪一個row為欄位標籤，預設是0
  - index_col: 設定哪一個欄位為索引
  - encoding:檔案編碼方式
  - nrows: 只讀取前幾row
  - usecols: 只讀取哪幾個欄位

寫入DataFrame到CSV檔案

- df_name.to_csv(path, sep, header, index, encoding)
  - path: csv檔案路徑
  - sep: 指定分隔字元，預設是','
  - header: True/Flase，是否保留columns名稱，預設是True
  - index: True/Flase，是否保留dataframe的索引，預設是True
  - encoding: 檔案編碼方式

DataFrame 函數取資料

- at: 使用index和columns內容取得或設定單一元素內容或陣列內容
- iat: 使用index和columns編號取得或設定單一元素內容
- loc: 使用index或columns內容取得或設定整個row或columns資料或陣列內容
- iloc: 使用index或columns編號取得或設定整個row或columns資料

DataFrame 函數取資料

- df_name.at['index', 'columns']
  - 返回在該欄位符合該索引的資料
  - 若不只一筆資料符合則返回一個串列(index有重複)

```
cities = {'country':['China','Japan','Singapore'],
'town':['Beijing','Tokyo','Singapore'],
'population':[2000, 1600, 600]}

city_df = pd.DataFrame(cities, index= country)

df_name.at['Japan', 'town']
```

DataFrame 函數取資料

- df_name.iat[row_number, columns_number]
  - 返回符合列編號與欄編號位置的資料

```
cities = {‘country’:[‘China’,‘Japan’,‘Singapore’],
‘town’:[‘Beijing’,‘Tokyo’,‘Singapore’],
'population':[2000, 1600, 600]}

city_df = pd.DataFrame(cities, index= country)

city_df.iat[2, 0]
```

DataFrame 函數取資料

- df_loc[‘index’]
  - 返回符合該索引的整列資料
- df_loc[[‘index1’, ‘index2’]]
  - 返回多筆符合索引的整列資料

```
cities = {‘country’:[‘China’,‘Japan’,‘Singapore’],
‘town’:[‘Beijing’,‘Tokyo’,‘Singapore’],
'population':[2000, 1600, 600]}
city_df = pd.DataFrame(cities, index= country)
df_loc[‘Japan’]
df_loc[[‘Japan’, ‘Singapore’]]
```

DataFrame 函數取資料

- df_iloc[row_number]
  - 返回符合該索引編號的整列資料
- df_iloc[[row_number1, row_number2]]
  - 返回多筆符合索引編號的整列資料
- df_iloc[start_row:end_row]
  - 返回從開始到結束編號的整列資料

DataFrame 函數取資料

- df_name.head(number)
  - 回傳一個dataframe，包含最前面幾筆的資料
- df_name.tail(number)
  - 回傳一個dataframe，包含最後面幾筆的資料

DataFrame 直接取資料方式

- 直接columns名稱方式取值，返回該columns的全部資料
  - df['columns']
- 直接用index和columns的名稱取值
  - df ['columns']['index']
- 直接用多個columns取資料
  - df[['columns1', 'columns2']]
- 直接用列編號取資料
  - df[:3]
- 邏輯判斷取資料
  - df[df['columns'] > number]

DataFrame Rename

- rename(): 可以改變DataFrame中的index或columns
- 修改index
  - df.rename(index={舊索引值 : 新索引值, ...}, inplace=True)
- 修改columns
  - df.rename(columns={舊欄位名稱: 新欄位名稱, ...}, inplace= True)

DataFrame 新增資料

- 新增欄位資料
  - insert(欄位位置, column='欄位名稱', value=[欄位值])
    - 欄位位置代表在第幾個欄位插入
    - 欄為值可以是一個串列，包含值的數量必須和原本的資料表內的資料筆數相同
    - 會直接插入資料在原本的dataframe內

DataFrame 新增資料

- 新增一筆資料在資料表內
  - df_name.append(data, ignore_index)
    - data傳入時要指定傳入資料的欄位名稱和值，例如字典
    - ignore_index=True/False，指定是否忽略要新增的資料的index
    - 會回傳一個dataframe，裡面的資料是新增資料後新的dataframe，不會直接修改原本的dataframe
    - 未來的版本中可能不再支援append方法，可以用pd.concat代替

DataFrame 合併

- 以下方法會回傳一個dataframe，裡面的資料是合併後新的dataframe，不會直接修改原本的dataframe
- 垂直串接兩個dataframe
  - pd.concat([dataframe1, dataframe2], ignore_index, join)
    - Ignore_index=True/False，指定是否忽略要新增的資料的index
    - Join='outer': 預設模式，會直接把沒有的資料用 NaN 代替
    - Join='inner':會直接把沒有完整資料的刪除掉
- 水平合併兩個dataframe
  - pd.merge(dataframe1, dataframe2, on='columns')
    - on='columns': 設定用哪一個欄位來合併

DataFrame 刪除資料

- drop可以刪除DataFrame中的資料，會回傳一個dataframe，裡面的資料是刪除過後剩下的資料，不會直接修改原本的dataframe
- 刪除欄位
  - df.drop([“欄位名稱”,“欄位名稱"], axis="columns")
  - df.drop([“欄位名稱”,“欄位名稱"], axis=1)
- 刪除某列
  - df.drop(“index”, axis=“rows")
  - df.drop(“index”, axis=0)

Pandas DataFrame運算

DataFrame 四則運算

- 做四則運算時，是逐列逐欄位資料做運算，即兩個DataFrame相同欄位名稱且相同索引的值才會做運算，若兩邊有其中一邊的值缺失，該資料會無法進行運算並回傳NaN
- add(): 加法運算
  - df1.add(df2)
- sub(): 減法運算
- mul(): 乘法運算
- div(): 除法運算

DataFrame 邏輯運算

- 邏輯運算包含以下
  - gt()、lt() → 大於、小於
  - ge()、le() → 大於或等於、小於或等於
  - eq()、ne() → 等於、不等於
- df1.gt(df2)
  - 運算規則和四則運算一樣

DataFrame NaN

- dropna(): 將資料中的NaN刪除，傳回新的DataFrame
- fillna(value): 將NaN值由特定的傳入值取代，並傳回新的DataFrame
- isna(): 判斷是否為NaN，傳回True/False
- notna(): 判斷是否為NaN，傳回值和isna()相反

DataFrame 常用的數字類型統計函數

在DataFrame中，做統計時可以以columns為單位進行操作，例如使用loc函數取出想統計的columns

- describe(): 描述指定資料的基本統計
- sum(): 加總
- prod(): 乘積
- mean(): 平均值
- min(): 最小值
- max(): 最大值
- std(): 標準差
- var(): 變異數
- median(): 中位數
- argmin(): 最小元素值索引
- argmax(): 最大元素值索引
- cumsum(): 陣列元素累加
- cumprod(): 陣列元素累積
- corr(): 兩個資料分布的關聯性

DataFrame排序

- sort_values(by="columns", ascending=True/False)
  - 將某一個欄位內資料做排序， ascending=True為大到小排序

練習

- 參考csvReport.csv，該檔案為一間茶葉公司的業務成績，
  - Name: 業務員名,
  - Year: 年分
  - Product: 銷售產品
  - Price: 產品單價
  - Quantity: 銷售數量
  - Revenue: 銷售總額
  - Location: 地區

- 請分析以下的資料
  - 不同地區的銷售表現比較比較各地區（如 New York, Los Angeles, Tokyo 等）在不同年份的總銷售額，並找出銷售成績最佳的地區
    - 哪個地區在 2015-2024 年間的銷售額最高？
    - 哪個產品在該地區最受歡迎？
  - 業務員銷售效率評估計算每位業務員的年平均銷售額，並進行排序。
    - 哪位業務員的銷售績效最佳？
    - 每位業務員最擅長銷售哪種茶葉？
  - 產品銷售趨勢分析研究不同產品在不同年份的銷售額變化趨勢。
    - 哪一種產品的銷售額在過去幾年中增長最快？哪種產品的銷售額有下降的趨勢？