

Report 3

Part 1: Course exercises

1.1:DataFrame

1.2:索引 数据

1.3:DataFrame

1.4:read_csv

1.5:multiIndex

2.1:T

2.2:T

2.3:T

2.4:F

2.5:F

3.1:C

3.2:D

3.3:A

3.4:A

3.5:A

4.1:

Series:

Series 是类似于一维数组的数据结构，主要由一组数据和与之相关的索引两部分组成，其数据类型为任意类型

- **一维数组:** Series 中的每个元素都有一个对应的索引值。
- **索引:** 每个数据元素都可以通过标签（索引）来访问，默认情况下索引是从 0 开始的整数，但你也可以自定义索引。
- **数据类型:** Series 可以容纳不同数据类型的元素，包括整数、浮点数、字符串、Python 对象等。
- **大小不变性:** Series 的大小在创建后是不变的，但可以通过某些操作（如 append 或 delete）来改变。
- **操作:** Series 支持各种操作，如数学运算、统计分析、字符串处理等。
- **缺失数据:** Series 可以包含缺失数据，Pandas 使用 NaN（Not a Number）来表示缺失或无值。
- **自动对齐:** 当对多个 Series 进行运算时，Pandas 会自动根据索引对齐数据，这使得数据处理更加高效。

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

data: Series 的数据部分，可以是列表、数组、字典、标量值等。如果不提供此参数，则创建一个空的 Series。

index: Series 的索引部分，用于对数据进行标记。可以是列表、数组、索引对象等。如果不提供此参数，则创建一个默认的整数索引。

dtype: 指定 Series 的数据类型。可以是 NumPy 的数据类型，例如 np.int64、np.float64 等。

如果不提供此参数，则根据数据自动推断数据类型。 如果不指定相应的 dtype 类型，则其类型默认为 object

name: Series 的名称，用于标识 Series 对象。如果提供了此参数，则创建的 Series 对象将具有指定的名称。

copy: 是否复制数据。默认为 False，表示不复制数据。如果设置为 True，则复制输入的数据。

fastpath: 是否启用快速路径。默认为 False。启用快速路径可能会在某些情况下提高性能。

如果一个变量的数据类型是 Series 的话，那么可以通过 index 和 value 分别获取其 索引 和 数据值

ser_obj.index # 获取其索引 其获取的值的类型是 Index 类的对象

ser_obj.value # 获取其数据值

index	获取 Series 的索引
values	获取 Series 的数据部分（返回 NumPy 数组）
head(n)	返回 Series 的前 n 行（默认为 5）
tail(n)	返回 Series 的后 n 行（默认为 5）
dtype	返回 Series 中数据的类型
shape	返回 Series 的形状（行数）
describe()	返回 Series 的统计描述（如均值、标准差、最小值等）
isnull()	返回一个布尔 Series，表示每个元素是否为 NaN
notnull()	返回一个布尔 Series，表示每个元素是否不是 NaN
unique()	返回 Series 中的唯一值（去重）
value_counts()	返回 Series 中每个唯一值的出现次数

map(func)	将指定函数应用于 Series 中的每个元素
apply(func)	将指定函数应用于 Series 中的每个元素，常用于自定义操作
astype(dtype)	将 Series 转换为指定的类型
sort_values()	对 Series 中的元素进行排序（按值排序）
sort_index()	对 Series 的索引进行排序
dropna()	删除 Series 中的缺失值（NaN）
fillna(value)	填充 Series 中的缺失值（NaN）
replace(to_replace, value)	替换 Series 中指定的值
cumsum()	返回 Series 的累计求和
cumprod()	返回 Series 的累计乘积
shift(periods)	将 Series 中的元素按指定的步数进行位移
rank()	返回 Series 中元素的排名
corr(other)	计算 Series 与另一个 Series 的相关性（皮尔逊相关系数）
cov(other)	计算 Series 与另一个 Series 的协方差
to_list()	将 Series 转换为 Python 列表
to_frame()	将 Series 转换为 DataFrame
iloc[]	通过位置索引来选择数据
loc[]	通过标签索引来选择数据

DataFrame:

- **二维结构:** DataFrame 是一个二维表格，可以被看作是一个 Excel 电子表格或 SQL 表，具有行和列。可以将其视为多个 Series 对象组成的字典。
- **列的数据类型:** 不同的列可以包含不同的数据类型，例如整数、浮点数、字符串或 Python 对象等。
- **索引:** DataFrame 可以拥有行索引和列索引，类似于 Excel 中的行号和列标。
- **大小可变:** 可以添加和删除列，类似于 Python 中的字典。
- **自动对齐:** 在进行算术运算或数据对齐操作时，DataFrame 会自动对齐索引。
- **处理缺失数据:** DataFrame 可以包含缺失数据，Pandas 使用 NaN (Not a Number) 来表示。
- **数据操作:** 支持数据切片、索引、子集分割等操作。

- **时间序列支持:** `DataFrame` 对时间序列数据有特别的支持, 可以轻松地进行时间数据的切片、索引和操作。
- **丰富的数据访问功能:** 通过 `.loc`、`.iloc` 和 `.query()` 方法, 可以灵活地访问和筛选数据。
- **灵活的数据处理功能:** 包括数据合并、重塑、透视、分组和聚合等。
- **数据可视化:** 虽然 `DataFrame` 本身不是可视化工具, 但它可以与 `Matplotlib` 或 `Seaborn` 等可视化库结合使用, 进行数据可视化。
- **高效的数据输入输出:** 可以方便地读取和写入数据, 支持多种格式, 如 CSV、Excel、SQL 数据库和 HDF5 格式。
- **描述性统计:** 提供了一系列方法来计算描述性统计数据, 如 `.describe()`、`.mean()`、`.sum()` 等。
- **灵活的数据对齐和集成:** 可以轻松地与其他 `DataFrame` 或 `Series` 对象进行合并、连接或更新操作。
- **转换功能:** 可以对数据集中的值进行转换, 例如使用 `.apply()` 方法应用自定义函数。
- **滚动窗口和时间序列分析:** 支持对数据集进行滚动窗口统计和时间序列分析。

```
pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

data: `DataFrame` 的数据部分, 可以是字典、二维数组、`Series`、`DataFrame` 或其他可转换为 `DataFrame` 的对象。如果不提供此参数, 则创建一个空的 `DataFrame`。

index: `DataFrame` 的行索引, 用于标识每行数据。可以是列表、数组、索引对象等。如果不提供此参数, 则创建一个默认的整数索引。

columns: `DataFrame` 的列索引, 用于标识每列数据。可以是列表、数组、索引对象等。如果不提供此参数, 则创建一个默认的整数索引。

dtype: 指定 `DataFrame` 的数据类型。可以是 `NumPy` 的数据类型, 例如 `np.int64`、`np.float64` 等。如果不提供此参数, 则根据数据自动推断数据类型。

copy: 是否复制数据。默认为 `False`, 表示不复制数据。如果设置为 `True`, 则复制输入的数据。

```
data = [['Google', 10], ['Runoob', 12], ['Wiki', 13]]
```

```
# 创建 DataFrame
```

```
df = pd.DataFrame(data, columns=['Site', 'Age'])
```

```
# 使用 astype 方法设置每列的数据类型
```

```
df['Site'] = df['Site'].astype(str)
```

```
df['Age'] = df['Age'].astype(float)
```

```
# 使用字典来创建
```

```
data = {'Site': ['Google', 'Runoob', 'Wiki'], 'Age': [10, 12, 13]}
```

```
df = pd.DataFrame(data)
```

```
# 创建一个包含网站和年龄的二维 ndarray
```

```
ndarray_data = np.array([
    ['Google', 10],
    ['Runoob', 12],
    ['Wiki', 13]
])
# 使用 DataFrame 构造函数创建数据帧
df = pd.DataFrame(ndarray_data, columns=['Site', 'Age'])

# loc 返回指定行的数据
data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

# 数据载入到 DataFrame 对象
df = pd.DataFrame(data)

# 返回第一行
print(df.loc[0])
# 返回第二行
print(df.loc[1])

df_obj.info()      # 查看 df_obj 对象的摘要信息
```

方法名称	功能描述
head(n)	返回 DataFrame 的前 n 行数据（默认前 5 行）
tail(n)	返回 DataFrame 的后 n 行数据（默认后 5 行）
info()	显示 DataFrame 的简要信息，包括列名、数据类型、非空值数量等
describe()	返回 DataFrame 数值列的统计信息，如均值、标准差、最小值等
shape	返回 DataFrame 的行数和列数（行数，列数）
columns	返回 DataFrame 的所有列名
index	返回 DataFrame 的行索引
dtypes	返回每一列的数值数据类型
sort_values(by)	按照指定列排序

sort_index()	按行索引排序
dropna()	删除含有缺失值 (NaN) 的行或列
fillna(value)	用指定的值填充缺失值
isnull()	判断缺失值, 返回一个布尔值 DataFrame
notnull()	判断非缺失值, 返回一个布尔值 DataFrame
loc[]	按标签索引选择数据
iloc[]	按位置索引选择数据
at[]	访问 DataFrame 中单个元素 (比 loc[] 更高效)
iat[]	访问 DataFrame 中单个元素 (比 iloc[] 更高效)
apply(func)	对 DataFrame 或 Series 应用一个函数
applymap(func)	对 DataFrame 的每个元素应用函数 (仅对 DataFrame)
groupby(by)	分组操作, 用于按某一列分组进行汇总统计
pivot_table()	创建透视表
merge()	合并多个 DataFrame (类似 SQL 的 JOIN 操作)
concat()	按行或按列连接多个 DataFrame
to_csv()	将 DataFrame 导出为 CSV 文件
to_excel()	将 DataFrame 导出为 Excel 文件
to_json()	将 DataFrame 导出为 JSON 格式
to_sql()	将 DataFrame 导出为 SQL 数据库
query()	使用 SQL 风格的语法查询 DataFrame
duplicated()	返回布尔值 DataFrame, 指示每行是否是重复的
drop_duplicates()	删除重复的行
set_index()	设置 DataFrame 的索引
reset_index()	重置 DataFrame 的索引
transpose()	转置 DataFrame (行列交换)

4.2:简述分层索引:

分层索引也就是在 Series 或 DataFrame 类对象还可以拥有更多层次的索引，可以在一个轴方向上具有两层甚至两层以上的索引，可以通过三种方式创建分层索引。

- 1、 from_tuples()
- 2、 from_arrays()
- 3、 from_product()

5.1：根据图示数据的结构创建一个 DataFrame 类对象

	A	B	C	D
0	1	5	8	8
1	2	2	4	9
2	7	4	2	3
3	3	0	5	2

图 3-14 数据示例

```
obj = pd.DataFrame(data=[  
    [1,5,8,8]  
    , [2,2,4,9]  
    , [7,4,2,3]  
    , [3,0,5,2]], columns=['A', 'B', 'C', 'D'])  
print(obj)
```

```
   A  B  C  D  
0  1  5  8  8  
1  2  2  4  9  
2  7  4  2  3  
3  3  0  5  2
```

5.2: 以 B 列为准，降序排列 DataFrame 类对象的数据

```
obj = pd.DataFrame(data=[  
    [1,5,8,8]  
    , [2,2,4,9]
```

```
, [7, 4, 2, 3]
, [3, 0, 5, 2]], columns=['A', 'B', 'C', 'D'])
# inplace 参数代表的是 是否将其应用到原本的变量值
obj.sort_values(by='B', ascending=False, inplace=True)
print(obj)
```

```
2. \study \python \work
```

	A	B	C	D
0	1	5	8	8
2	7	4	2	3
1	2	2	4	9
3	3	0	5	2

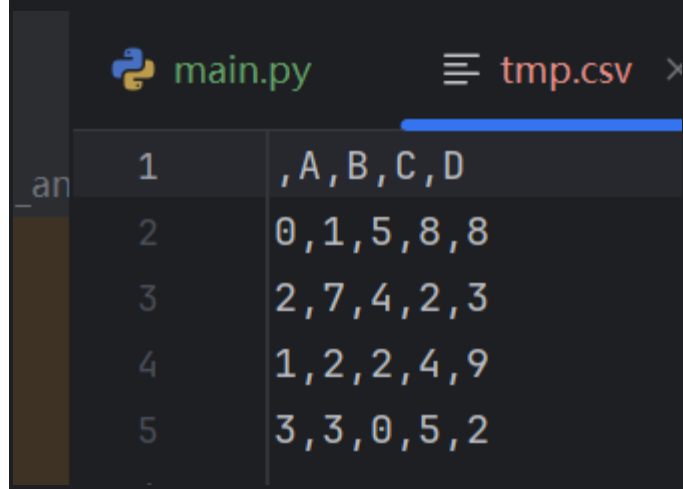
5.3: 将排序后的数据写入到 write_data.csv 文件。

```
import pandas as pd
import numpy as np

obj = pd.DataFrame(data=
    [[1, 5, 8, 8]
    , [2, 2, 4, 9]
    , [7, 4, 2, 3]
    , [3, 0, 5, 2]], columns=['A', 'B', 'C', 'D'])

obj.sort_values(by='B', ascending=False, inplace=True)

obj.to_csv('./tmp.csv')
```



	A	B	C	D
0	1	5	8	8
2	7	4	2	3
1	2	2	4	9
3	3	0	5	2

	A	B	C	D
1	5	8	8	1
2	4	2	3	7
3	2	2	4	9
4	0	5	2	3

Part 2:

[Download the Excel file from the Moodle and complete the following Pandas practices and screenshot your solutions and results.](#)

1. 读取文件并查看文件中数据的基本信息.

```
import pandas as pd
import numpy as np

df = pd.read_excel(r'./Report.xlsx')
print(df)

print(df.describe())
```

	购药时间	社保卡号	商品编码	商品名称	销售数量	应收金额	
0	2018-01-01 星期五	1.616528e+06	236701.0	强力VC银翘片	6.0	82.8	69.00
1	2018-01-02 星期六	1.616528e+06	236701.0	清热解毒口服液	1.0	28.0	24.64
2	2018-01-06 星期三	1.260283e+07	236701.0	感康	2.0	16.8	15.00
3	2018-01-11 星期一	1.007034e+10	236701.0	三九感冒灵	1.0	28.0	28.00
4	2018-01-15 星期五	1.015543e+08	236701.0	三九感冒灵	8.0	224.0	208.00
...
6573	2018-04-27 星期三	1.078861e+08	2367011.0	高特灵	10.0	56.0	54.80
6574	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6575	2018-04-27 星期三	1.008787e+10	2367011.0	高特灵	2.0	11.2	9.86
6576	2018-04-27 星期三	1.340663e+07	2367011.0	高特灵	1.0	5.6	5.00
6577	2018-04-28 星期四	1.192693e+07	2367011.0	高特灵	2.0	11.2	10.00

2. 把获取数据中“购药时间”改为售药时间.

```
df = pd.read_excel(r'./Report.xlsx')
new_df = df.reindex(columns=['售药时间','社保卡号','商品编码','商品名称','销售数量','应收金额','实收金额'])
new_df.loc[:, '售药时间'] = df.loc[:, '购药时间']
print(new_df)
```

	售药时间	社保卡号	商品编码	商品名称	销售数量	应收金额	实收金额
0	2018-01-01 星期五	1.616528e+06	236701.0	强力VC银翘片	6.0	82.8	69.00
1	2018-01-02 星期六	1.616528e+06	236701.0	清热解毒口服液	1.0	28.0	24.64

3. 删除数据中的缺失值.

```
df = pd.read_excel(r'./Report.xlsx')
new_df = df.reindex(columns=['售药时间','社保卡号','商品编码','商品名称','销售数量','应收金额','实收金额'])
new_df.loc[:, '售药时间'] = df.loc[:, '购药时间']

df_cleaned = new_df.dropna(axis=0)
print(df_cleaned)
```

6573	2018-04-27	星期三	1.078861e+08	2367011.0	高特灵	10.0	56.0	54.80
6574		NaN	NaN	NaN	NaN	NaN	NaN	NaN
6572	2018-04-27	星期三	1.006048e+10	2367011.0	高特灵	1.0	5.6	5.00
6573	2018-04-27	星期三	1.078861e+08	2367011.0	高特灵	10.0	56.0	54.80
6575	2018-04-27	星期三	1.008787e+10	2367011.0	高特灵	2.0	11.2	9.86
6576	2018-04-27	星期三	1.340663e+07	2367011.0	高特灵	1.0	5.6	5.00

4. 将字符串转换为浮点型数据.

```
new_df['售药时间']=new_df['售药时间'].astype(float)
```

5. 将“售药时间”中的星期去除，获取日期，并将“售药时间”这一列设置成所获取的日期.

```
df = pd.read_excel(r'./Report.xlsx')
new_df = df.reindex(columns=['售药时间','社保卡号','商品编码','商品名称','销售数量','应收金额','实收金额'])
new_df.loc[:, '售药时间'] = df.loc[:, '购药时间']

df_cleaned = new_df.dropna(axis=0)
# print(df_cleaned)
```

```
df_cleaned.loc[:, '售药时间'] = df_cleaned.loc[:, '售药时间'].str.split(' ').str[0]
```

	售药时间	
0	2018-01-01	1.616528e
1	2018-01-02	1.616528e
2	2018-01-06	1.260283e

```
print(df_cleaned)
```

6. 将获取的日期转换为时间格式.

```
df = pd.read_excel(r'./Report.xlsx')
new_df = df.reindex(columns=['售药时间', '社保卡号', '商品编码', '商品名称', '销售数量', '应收金额', '实收金额'])
new_df['售药时间'] = new_df['售药时间'].astype(str)
new_df.loc[:, '售药时间'] = df.loc[:, '购药时间']
df_cleaned = new_df.dropna(axis=0)
df_cleaned.loc[:, '售药时间'] = df_cleaned.loc[:, '售药时间'].str.split(' ').str[0]

df_cleaned['售药时间'] = pd.to_datetime(df_cleaned['售药时间'], format='%Y-%m-%d', errors='coerce')
print(df_cleaned)
```

7. 按照“售药时间”进行降序排列.

```
df = pd.read_excel(r'./Report.xlsx')
new_df = df.reindex(columns=['售药时间', '社保卡号', '商品编码', '商品名称', '销售数量', '应收金额', '实收金额'])
new_df['售药时间'] = new_df['售药时间'].astype(str)
new_df.loc[:, '售药时间'] = df.loc[:, '购药时间']
df_cleaned = new_df.dropna(axis=0)
df_cleaned.loc[:, '售药时间'] = df_cleaned.loc[:, '售药时间'].str.split(' ').str[0]

df_cleaned['售药时间'] = pd.to_datetime(df_cleaned['售药时间'], format='%Y-%m-%d', errors='coerce')
df_cleaned.sort_values(by='售药时间', ascending=False, inplace=True)

print(df_cleaned)
```


```
2404 2018-07-19
5990 2018-07-19
862 2018-07-19
---
```

8. 将“销售数量”、“应收金额”、“实收金额”这三列中的异常值排除掉.

```
df_cleaned = df_cleaned[df_cleaned['销售数量'] > 0]
df_cleaned = df_cleaned[df_cleaned['应收金额'] > 0]
df_cleaned = df_cleaned[df_cleaned['实收金额'] > 0]
```

9. 将最终的数据写入到新的.xlsx 文件，以学号命名.

```
df_cleaned.to_excel(r'2022015232.xlsx')
print('success!!')
```

 2022015232.xlsx