下面是一个 Python Pandas 入门学习文档，内容涵盖 Pandas 的基本概念、数据结构、常用操作以及一些示例。

Python Pandas 入门学习文档

1. 简介

Pandas 是一个强大的 Python 数据分析库，它提供了高性能、易于使用的数据结构和数据分析工具。Pandas 特别擅长处理表格型数据（类似于 Excel 表格或 SQL 数据表），并且提供了丰富的功能来清洗、转换、分析和可视化数据。

2. 安装 Pandas

在开始之前，你需要先安装 Pandas。可以使用 pip 包管理器来安装：

```
pip install pandas
```

3. Pandas 的两种主要数据结构

Pandas 提供了两种主要的数据结构：

- **Series（序列）：** 一维带标签的数组，可以存储任何数据类型（整数、浮点数、字符串等）。
- **DataFrame（数据框）：** 二维表格型数据结构，可以看作是 Series 的容器。每一列都是一个 Series，并且所有列共享相同的索引。

4. Series

4.1 创建 Series

可以使用多种方式创建 Series：

- **从列表创建：**

```
import pandas as pd

data = [10, 20, 30, 40, 50]
s = pd.Series(data)
print(s)
```

输出：

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

- **从 NumPy 数组创建：**

```
import pandas as pd
import numpy as np

data = np.array([10, 20, 30, 40, 50])
s = pd.Series(data)
print(s)
```

- **从字典创建：**

```
import pandas as pd

data = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
s = pd.Series(data)
print(s)
```

输出：

```
a    10
b    20
c    30
d    40
dtype: int64
```

4.2 Series 的属性

- `index`：获取 Series 的索引。
- `values`：获取 Series 的值。

- `dtype`：获取 Series 的数据类型。

```
import pandas as pd

data = [10, 20, 30, 40, 50]
s = pd.Series(data, index=['a', 'b', 'c', 'd', 'e'])

print(s.index)
print(s.values)
print(s.dtype)
```

4.3 Series 的基本操作

- **索引：** 可以使用标签或位置来访问 Series 中的元素。

```
import pandas as pd

data = [10, 20, 30, 40, 50]
s = pd.Series(data, index=['a', 'b', 'c', 'd', 'e'])

print(s['a'])  # 使用标签索引
print(s[0])    # 使用位置索引
```

- **切片：**

```
import pandas as pd

data = [10, 20, 30, 40, 50]
s = pd.Series(data, index=['a', 'b', 'c', 'd', 'e'])

print(s['a':'c']) # 标签切片
print(s[0:3])   # 位置切片
```

- **筛选：** 可以使用条件表达式来筛选 Series 中的元素。

```
import pandas as pd

data = [10, 20, 30, 40, 50]
s = pd.Series(data, index=['a', 'b', 'c', 'd', 'e'])
```

```
print(s[s > 25])
```

- **数学运算：** 可以对 Series 进行各种数学运算。

```
import pandas as pd

data = [10, 20, 30, 40, 50]
s = pd.Series(data, index=['a', 'b', 'c', 'd', 'e'])

print(s + 5)
print(s * 2)
print(s.mean())
```

5. DataFrame

5.1 创建 DataFrame

可以使用多种方式创建 DataFrame：

- **从字典创建：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)
print(df)
```

输出：

```
      name  age      city
0    Alice   25  New York
1      Bob   30    London
2  Charlie   28     Paris
3    David   22     Tokyo
```

- **从列表创建：**

```
import pandas as pd

data = [
    ['Alice', 25, 'New York'],
    ['Bob', 30, 'London'],
    ['Charlie', 28, 'Paris'],
    ['David', 22, 'Tokyo']
]
df = pd.DataFrame(data, columns=['name', 'age', 'city'])
print(df)
```

- **从 CSV 文件创建：**

```
import pandas as pd

df = pd.read_csv('data.csv')  # 假设有一个名为 data.csv 的文件
print(df)
```

5.2 DataFrame 的属性

- `index`：获取 DataFrame 的索引。
- `columns`：获取 DataFrame 的列名。
- `values`：获取 DataFrame 的值（NumPy 数组）。
- `dtypes`：获取 DataFrame 每列的数据类型。
- `shape`：获取 DataFrame 的形状（行数，列数）。

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

print(df.index)
print(df.columns)
print(df.values)
```

```
print(df.dtypes)
print(df.shape)
```

5.3 DataFrame 的基本操作

- **选择列：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

print(df['name'])  # 选择 'name' 列
print(df[['name', 'age']])  # 选择 'name' 和 'age' 列
```

- **选择行：**
  - `loc`：基于标签选择行。
  - `iloc`：基于位置选择行。

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data, index=['a', 'b', 'c', 'd'])

print(df.loc['a'])   # 选择索引为 'a' 的行
print(df.iloc[0])    # 选择第一行
print(df.loc['a':'c']) # 选择索引 'a' 到 'c' 的行
print(df.iloc[0:3])  # 选择前三行
```

- **筛选：** 可以使用条件表达式来筛选 DataFrame 中的行。

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

print(df[df['age'] > 25])  # 选择年龄大于 25 的行
print(df[df['city'] == 'London']) # 选择城市为 'London' 的行
```

- **添加列：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

df['salary'] = [50000, 60000, 55000, 45000]  # 添加 'salary' 列
print(df)
```

- **删除列：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

df = df.drop('city', axis=1)  # 删除 'city' 列
print(df)
```

- **排序：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

df = df.sort_values(by='age')  # 按 'age' 列排序
print(df)
```

- **分组：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'age': [25, 30, 25, 22, 30],
    'city': ['New York', 'London', 'New York', 'Tokyo', 'London']
}
df = pd.DataFrame(data)

grouped = df.groupby('city')
print(grouped.mean())  # 按城市分组，计算每组的平均年龄
```

6. 数据清洗

Pandas 提供了许多用于数据清洗的功能：

- **处理缺失值：**
  - `fillna()`：填充缺失值。
  - `dropna()`：删除包含缺失值的行或列。
  - `isna()`：检测缺失值。

```
import pandas as pd
import numpy as np

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, np.nan, 28, 22],
```

```
    'city': ['New York', 'London', 'Paris', np.nan]
}
df = pd.DataFrame(data)

print(df.isna())
df = df.fillna({'age': df['age'].mean(), 'city': 'Unknown'})  # 填充缺失值
print(df)
df = df.dropna() # 删除包含缺失值的行
print(df)
```

- **处理重复值：**
  - `duplicated()`：检测重复值。
  - `drop_duplicates()`：删除重复值。

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Alice', 'David'],
    'age': [25, 30, 25, 22],
    'city': ['New York', 'London', 'New York', 'Tokyo']
}
df = pd.DataFrame(data)

print(df.duplicated())
df = df.drop_duplicates()
print(df)
```

7. 数据合并和连接

Pandas 提供了多种方式来合并和连接 DataFrame：

- **`concat()`：沿着轴方向连接 DataFrame。**
- **`merge()`：基于共同列连接 DataFrame（类似于 SQL 的 JOIN 操作）。**

```
import pandas as pd

df1 = pd.DataFrame({'key': ['A', 'B', 'C', 'D'], 'value': [1, 2, 3, 4]})
df2 = pd.DataFrame({'key': ['B', 'D', 'E', 'F'], 'value': [5, 6, 7, 8]})

# Concatenate
df_concat = pd.concat([df1, df2])
```

```
print("Concatenated DataFrame:\n", df_concat)

# Merge
df_merged = pd.merge(df1, df2, on='key', how='inner')
print("\nMerged DataFrame:\n", df_merged)
```

8. 数据导出

可以将 DataFrame 导出到多种格式的文件：

- **导出到 CSV 文件：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

df.to_csv('output.csv', index=False)  # 导出到 output.csv 文件
```

- **导出到 Excel 文件：**

```
import pandas as pd

data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'age': [25, 30, 28, 22],
    'city': ['New York', 'London', 'Paris', 'Tokyo']
}
df = pd.DataFrame(data)

df.to_excel('output.xlsx', index=False)  # 导出到 output.xlsx 文件
```

9. 总结

这只是 Pandas 的一个简单入门。Pandas 提供了非常丰富的功能，可以满足各种数据分析的需求。 要深入学习 Pandas，建议参考 Pandas 官方文档和其他在线教程。

10. 学习资源

- **Pandas 官方文档：** https://pandas.pydata.org/docs/
- **Pandas 教程：** https://pandas.pydata.org/docs/getting_started/index.html
- **各种在线教程和博客文章。**

希望这个入门文档能帮助你开始学习 Pandas! 祝你学习顺利！