数据结构及类型

Python 实践练习 (shíjiàn liànxí)

第二章 课堂实操练习手册

2026.03

授课教师：李阳源

目录 (mùlù)

实践一：环境准备与 Pandas 入门

实践二：结构化数据——创建 Pandas DataFrame

实践三：结构化数据——DataFrame 基本操作

实践四：非结构化数据——文本数据处理

实践五：半结构化数据——JSON 数据处理

实践六：半结构化数据——XML 数据处理

实践七：三种数据类型综合对比

实践八：DataFrame 进阶操作——筛选、排序、聚合

课堂练习题

实践一：环境准备与 Pandas 入门

同学们，今天我们要把课堂上学的理论(lǐlùn)知识用 Python 代码来实现。

首先，我们要确认(quèrèn)环境是否准备好了。

Step 1：打开 PyCharm，新建一个 Python 文件

请在 PyCharm 中新建一个文件，命名(mìngmíng)为：data_structure_practice.py

Step 2：安装 Pandas 库(kù)

如果你还没有安装(ānzhuāng)Pandas，在终端(zhōngduān)中输入(shūrù)以下命令(mìnglìng)：

```
pip install pandas
```

Step 3：导入(dǎorù)必要的库

在文件最上方写下这些代码，导入我们今天需要用到的库：

```
# -*- coding: utf-8 -*-

# 第二章 数据结构及类型 - Python 实践练习

import pandas as pd      # 数据分析核心库

import json              # JSON 数据处理

import xml.etree.ElementTree as ET  # XML 数据处理

print("环境准备完成！开始实践练习！")

print("Pandas 版本:", pd.__version__)
```

运行(yùnxíng)代码，如果看到以下输出(shūchū)，说明环境准备成功(chénggōng)：

```
环境准备完成！开始实践练习！

Pandas 版本: 2.2.x
```

> ⚠ 老师提示：如果出现 ModuleNotFoundError，说明没有安装好，请重新(chóngxīn)运行 pip install pandas。

实践二：结构化数据——创建 Pandas DataFrame

还记得课堂上的大学生成绩表吗？现在我们用 Python 来创建它！

这就是“结构化数据(jiégòuhuà shùjù)”的典型例子。

Step 1：用字典(zìdiǎn)创建 DataFrame

字典是 Python 中的键值对(jiàn zhí duì)数据结构，跟我一起输入以下代码：

```
# ============================================

# 实践二：创建结构化数据 - 学生成绩表

# ============================================

print("=" * 50)

print("实践二：创建结构化数据")

print("=" * 50)


# 用字典创建数据

data = {

    "学号": [1001, 1002, 1003, 1004, 1005],

    "姓名": ["张伟", "李娜", "王浩", "赵雪", "陈明"],

    "专业": ["计算机工程", "统计学", "数据科学",

             "计算机工程", "数据科学"],

    "数学": [85, 95, 78, 90, 88],

    "统计学": [92, 90, 85, 88, 76],

    "编程": [88, 82, 91, 95, 84]

}
```

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


# 查看整个表格
print("\n★ 学生成绩表：")
print(df)
```

运行后，你会看到一个整齐(zhěngqí)的表格：

```
★ 学生成绩表：
     学号  姓名     专业  数学  统计学  编程
0  1001  张伟  计算机工程    85    92    88
1  1002  李娜    统计学    95    90    82
2  1003  王浩  数据科学    78    85    91
3  1004  赵雪  计算机工程    90    88    95
4  1005  陈明  数据科学    88    76    84
```

Step 2：查看 DataFrame 的基本信息(xìnxī)

让我们看看这个 DataFrame 的结构信息：

```
# 查看索引(Index)
print("\n★ 索引(suǒyǐn):", df.index.tolist())


# 查看列名(Columns)
print("★ 列名(liè míng):", df.columns.tolist())
```

```
# 查看数据类型
print("\n★ 每列的数据类型(shùjù lèixíng):")
print(df.dtypes)
# 查看形状(shape) - 行数和列数
print("\n★ 表格形状:", df.shape)
print(f"  共有 {df.shape[0]} 行, {df.shape[1]} 列")
```

运行结果：

```
★ 索引(suǒyǐn): [0, 1, 2, 3, 4]
★ 列名(liè míng): ['学号', '姓名', '专业', '数学', '统计学', '编程']

★ 每列的数据类型(shùjù lèixíng):
学号      int64
姓名     object
专业     object
数学      int64
统计学     int64
编程      int64

★ 表格形状: (5, 6)
  共有 5 行, 6 列
```

> ⚠ 老师说明：看，这就是我们课堂上讲的 DataFrame 的三个组成部分——索引(Index)、列名(Columns)、数据(Values)！int64 表示整数，object 表示文字。

实践三：结构化数据——DataFrame 基本操作

现在我们来学习 DataFrame 的基本操作(jīběn cāozuò)。

Step 1：选择(xuǎnzé)特定的列

我们只想看姓名和数学成绩：

```
# ==========================================

# 实践三：DataFrame 基本操作

# ==========================================

print("\n" + "=" * 50)

print("实践三：DataFrame 基本操作")

print("=" * 50)



# 选择特定列

print("\n★ 只看姓名和数学成绩：")

print(df[["姓名", "数学"]])
```

Step 2：选择特定的行

用 loc 和 iloc 来选择行：

```
# 用索引选择第一行数据

print("\n★ 第一个学生的信息：")

print(df.iloc[0])    # iloc 用数字索引



# 选择前三行

print("\n★ 前三个学生：")
```

```
print(df.head(3))    # head(n)显示前 n 行
```

Step 3：筛选(shāixuǎn)数据

找出数学成绩大于等于(dàyú děng yú)85 分的学生：

```
# 筛选：数学 >= 85

print("\n★ 数学成绩 >= 85 分的学生：")

good_math = df[df["数学"] >= 85]

print(good_math)
```

运行结果：

```
★ 数学成绩 >= 85 分的学生：

    学号  姓名     专业  数学  统计学  编程

0  1001  张伟  计算机工程    85    92    88

1  1002  李娜     统计学    95    90    82

3  1004  赵雪  计算机工程    90    88    95

4  1005  陈明  数据科学    88    76    84
```

Step 4：添加新列——计算平均分(píngjūn fēn)

我们来计算每个学生的三科平均分：

```
# 计算平均分，添加新列

df["平均分"] = (df["数学"] + df["统计学"] + df["编程"]) / 3

df["平均分"] = df["平均分"].round(1)  # 保留一位小数


print("\n★ 添加平均分后的表格：")
```

```
print(df)
```

运行结果：

```
★ 添加平均分后的表格：
   学号  姓名     专业  数学  统计学  编程  平均分
0  1001  张伟  计算机工程    85    92    88  88.3
1  1002  李娜     统计学    95    90    82  89.0
2  1003  王浩  数据科学    78    85    91  84.7
3  1004  赵雪  计算机工程    90    88    95  91.0
4  1005  陈明  数据科学    88    76    84  82.7
```

> ⚠ 老师提示：DataFrame 可以像我们课堂上讲的那样，方便(fāngbiàn)地进行选择、筛选、转换等操作！

实践四：非结构化数据——文本数据处理

课堂上我们讲过，社交媒体(shèjiāo méitǐ)帖子是非结构化数据。

现在我们来看看如何用 Python 来处理这类数据。

Step 1：创建非结构化数据

```
# ============================================
# 实践四：非结构化数据处理
# ============================================
print("\n" + "=" * 50)
print("实践四：非结构化数据处理")
print("=" * 50)


# 社交媒体文本数据 - 没有固定格式！
posts = [
    "今天的数据结构课太有趣了！#大数据 #大学生活",
    "刚刚学完了 Pandas，感觉很强大！",
    "Python 编程真有意思，我爱数据分析！#编程",
    "下周考试啦，赶紧复习数据结构和统计学...",
    "用 Pandas 做了个小项目，很有成就感！#大数据 #Python"
]


print("\n★ 原始(非结构化)文本数据：")
for i, post in enumerate(posts):
    print(f"  [{i+1}] {post}")
```

Step 2：尝试(chángshì)从非结构化数据中提取信息

非结构化数据需要特殊(tèshū)处理技术，比如提取(tíqǔ)标签(hashtag)：

```
# 提取所有标签(hashtag)

print("\n★ 提取标签(biāoqiān)：")

all_tags = []

for post in posts:

    words = post.split()

    tags = [w for w in words if w.startswith("#")]

    if tags:

        print(f"  文本: {post[:20]}...

        print(f"  → 标签: {tags}")

        all_tags.extend(tags)


print(f"\n  所有标签汇总(huìzǒng): {all_tags}")
```

Step 3：统计文本特征(tèzhēng)

我们还可以统计每条文本的长度(chángdù)：

```
# 统计每条文本的长度

print("\n★ 文本长度统计：")

for i, post in enumerate(posts):

    print(f"  [{i+1}] 长度: {len(post)} 字符")


lengths = [len(p) for p in posts]

print(f"\n  平均(píngjūn)长度: {sum(lengths)/len(lengths):.1f} 字符")
```

```
print(f"  最短: {min(lengths)} 字符")

print(f"  最长: {max(lengths)} 字符")
```

> ⚠ 老师说明：看，非结构化数据每条的长度都不一样，内容也各不相同。处理起来比结构化数据困难(kùnnan)多了！

实践五：半结构化数据——JSON 数据处理

还记得课堂上的 JSON 学生数据吗？现在我们用 Python 来处理它！

JSON 由键值对(jiàn zhí duì)组成，可以表示层次结构(céngcì jiégòu)。

Step 1：创建 JSON 数据

```
# ============================================
# 实践五：JSON 数据处理
# ============================================
print("\n" + "=" * 50)
print("实践五：JSON 数据处理")
print("=" * 50)


# 创建 JSON 数据（和课堂上的例子一样）
student_json = {
    "学生": {
        "学号": 1001,
        "姓名": "张伟",
        "专业": "计算机工程",
        "成绩": [
            {"科目": "数学", "分数": 85},
            {"科目": "统计学", "分数": 92},
            {"科目": "编程", "分数": 88}
        ],
        "社团": ["编程俱乐部", "AI 研究会"]
```

```
    }
}


# 转换为 JSON 字符串
json_str = json.dumps(student_json, ensure_ascii=False, indent=2)
print("\n★ JSON 格式数据：")
print(json_str)
```

Step 2：读取(dúqǔ)JSON 数据

我们来提取 JSON 里面的各种信息：

```
# 提取基本信息
student = student_json["学生"]
print("\n★ 提取 JSON 数据：")
print(f"  姓名: {student['姓名']}")
print(f"  学号: {student['学号']}")
print(f"  专业: {student['专业']}")


# 提取嵌套的成绩数据
print("\n★ 成绩明细(míngxì)：")
for score in student["成绩"]:
    print(f"  {score['科目']}: {score['分数']}分")


# 提取社团信息
print(f"\n★ 参加的社团(shètuán): {student['社团']}")
```

Step 3：JSON 转换(zhuǎnhuàn)为 DataFrame

半结构化数据可以转换成结构化数据！

```
# 把 JSON 中的成绩转换为 DataFrame

scores_df = pd.DataFrame(student["成绩"])

print("\n★ 成绩转换为 DataFrame：")

print(scores_df)
```

运行结果：

```
★ 成绩转换为 DataFrame：

    科目  分数

0   数学    85

1  统计学    92

2   编程    88
```

> ⚠ 老师说明：看，JSON（半结构化）可以转换成 DataFrame（结构化）！这在实际工作中非常常见，特别是处理 API 返回的数据时。

实践六：半结构化数据——XML 数据处理

接下来我们来处理 XML 数据。

XML 用标签(biāoqiān)来区分数据，就像课堂上讲的一样。

Step 1：创建 XML 数据

```
# ============================================
# 实践六：XML 数据处理
# ============================================
print("\n" + "=" * 50)
print("实践六：XML 数据处理")
print("=" * 50)


# 创建 XML 数据字符串（和课堂例子一样）
xml_data = """
<学生>
    <学号>1001</学号>
    <姓名>张伟</姓名>
    <专业>计算机工程</专业>
    <成绩>
        <科目 名称="数学" 分数="85" />
        <科目 名称="统计学" 分数="92" />
        <科目 名称="编程" 分数="88" />
    </成绩>
    <社团>
```

```
        <活动>编程俱乐部</活动>
        <活动>AI 研究会</活动>
    </社团>
</学生>
"""

print("\n★ XML 原始数据：")
print(xml_data)
```

Step 2：解析(jiěxī)XML 数据

用 Python 的 ElementTree 来解析 XML：

```
# 解析 XML
root = ET.fromstring(xml_data)

# 提取基本信息
print("★ XML 数据解析结果：")
print(f"  学号: {root.find('学号').text}")
print(f"  姓名: {root.find('姓名').text}")
print(f"  专业: {root.find('专业').text}")

# 提取成绩（从属性中读取）
print("\n★ 成绩信息：")
scores_list = []
for subject in root.find('成绩'):
    name = subject.get('名称')
```

```
    score = subject.get('分数')
    print(f"  {name}: {score}分")
    scores_list.append({"\u79d1\u76ee": name, "\u5206\u6570": int(score)})


# 提取社团
print("\n★ 社团信息：")
for activity in root.find('社团'):
    print(f"  - {activity.text}")
```

Step 3：XML 数据转换为 DataFrame

```
# 把 XML 提取的成绩转换为 DataFrame
xml_scores_df = pd.DataFrame(scores_list)
print("\n★ XML 成绩转换为 DataFrame：")
print(xml_scores_df)
```

> ⚠ 老师说明：JSON 和 XML 都是半结构化数据，但它们的格式不同。JSON 用键值对，XML 用标签。两者都可以转换成 DataFrame！

实践七：三种数据类型综合对比 (zōnghé duìbǐ)

现在我们用 Python 来直观(zhíguān)地比较三种数据类型的差异(chāyì)。

同一份数据，三种不同的表现(biǎoxiàn)形式

```
# ============================================

# 实践七：三种数据类型综合对比

# ============================================

print("\n" + "=" * 50)

print("实践七：三种数据类型对比")

print("=" * 50)



# ① 结构化数据：整齐的表格

print("\n【结构化数据】表格形式：")

structured = pd.DataFrame({

    "姓名": ["张伟"],

    "数学": [85],

    "统计学": [92],

    "编程": [88]

})

print(structured)

print(f"→ 类型: 固定的行列结构, 每列有固定类型")



# ② 半结构化数据：JSON 格式

print("\n【半结构化数据】JSON 形式：")
```

```
semi_structured = {
    "姓名": "张伟",
    "成绩": [{"\u79d1\u76ee": "\u6570\u5b66", "\u5206\u6570": 85},
             {"\u79d1\u76ee": "\u7edf\u8ba1\u5b66", "\u5206\u6570": 92}],
    "社团": ["编程俱乐部"]  # 可以有不同数量的字段
}
print(json.dumps(semi_structured, ensure_ascii=False, indent=2))
print("→ 类型：有结构但灵活，可以嵌套")


# ③ 非结构化数据：纯文本
print("\n【非结构化数据】纯文本形式：")
unstructured = "张伟同学计算机工程专业数学考了 85 分统计学 92 分编程 88 分"
print(f"  {unstructured}")
print("→ 类型：没有固定格式，难以直接分析")
```

运行后对比三种输出，同学们可以清楚(qīngchǔ)地看到差异！

> ⚠ 课堂讨论：同学们觉得哪种格式最容易(róngyì)处理？为什么？

实践八：DataFrame 进阶(jìnjiē)操作——筛选、排序、聚合

课堂上我们讲过 DataFrame 的七个基本操作。

现在我们来练习更多的操作！

Step 1：排序(páixù) - Sort

按平均分从高到低排序：

```
# ============================================

# 实践八：DataFrame 进阶操作

# ============================================

print("\n" + "=" * 50)

print("实践八：DataFrame 进阶操作")

print("=" * 50)



# 按平均分排序（从高到低）

print("\n★ 按平均分排序（从高到低）：")

sorted_df = df.sort_values("平均分", ascending=False)

print(sorted_df[["姓名", "专业", "平均分"]])
```

Step 2：分组(fēnzǔ)聚合(jùhé) - GroupBy

按专业分组，计算每个专业的平均成绩：

```
# 按专业分组，计算平均分

print("\n★ 按专业(zhuānyè)分组计算平均分：")

grouped = df.groupby("专业")[["数学", "统计学", "编程"]].mean()

grouped = grouped.round(1)
```

```
print(grouped)
```

运行结果：

```
★ 按专业(zhuānyè)分组计算平均分：
          数学  统计学   编程
专业
计算机工程  87.5   90.0  91.5
数据科学  83.0   80.5  87.5
统计学   95.0   90.0  82.0
```

Step 3：统计描述(miáoshù) - Describe

一行代码就能看到所有统计信息：

```
# 基本统计描述
print("\n★ 成绩统计描述(tǒngjì miáoshù)：")
print(df[["数学", "统计学", "编程", "平均分"]].describe().round(1))
```

运行结果会显示：count(计数)、mean(平均值)、std(标准差)、min(最小值)、25%/50%/75%(百分位数)、max(最大值)。

Step 4：导出(dǎochū)为 CSV 文件

把 DataFrame 保存(bǎocún)为 CSV 文件：

```
# 导出为 CSV 文件
df.to_csv("student_scores.csv", index=False, encoding="utf-8-sig")
print("\n★ 数据已保存为 student_scores.csv 文件！")
```

```
# 从 CSV 文件重新读取

df_loaded = pd.read_csv("student_scores.csv")

print("\n★ 今 CSV 文件重新读取的数据：")

print(df_loaded)
```

> ⚠ 老师提示：Pandas 可以读写 CSV、Excel、SQL、JSON 等多种格式，就像课堂上讲的“瑞士军刀”一样！

课堂练习题 (liànxí tí)

同学们，现在请自己动手(dòngshǒu)完成以下练习！

练习一：扩展(kuòzhǎn)DataFrame

请在现有的学生成绩表中添加两个新学生的数据：

- 学号 1006，刘芝，统计学专业，数学 92，统计学 88，编程 79
- 学号 1007，杨德，计算机工程专业，数学 81，统计学 94，编程 90

提示(tíshì)代码：

```
# 提示：用 pd.concat()添加新行
new_data = pd.DataFrame({
    "学号": [1006, 1007],
    "姓名": ["???", "???"],   # 请填写
    "专业": ["???", "???"],   # 请填写
    "数学": [??, ??],
    "统计学": [??, ??],
    "编程": [??, ??]
})
df_new = pd.concat([df, new_data], ignore_index=True)
print(df_new)
```

练习二：多条件筛选

找出同时满足(mǎnzú)以下条件的学生：

- 数学成绩 >= 85 分

- 编程成绩 >= 85 分

提示代码：

```
# 提示：用 & 连接多个条件

result = df[(df["数学"] >= 85) & (df["编程"] >= 85)]

print(result)
```

练习三：创建自己的 JSON 数据

请创建一个包含你自己信息的 JSON 数据，包括：

- 姓名、学号、专业
- 至少两门课的成绩
- 参加的社团或爱好(hào)

提示代码：

```
# 提示：参考实践五的 JSON 格式

my_data = {

    "学生": {

        "姓名": "???",       # 填写你的名字

        "学号": ???,

        "专业": "???",

        "成绩": [

            # 填写你的成绩

        ],

        "爱好": [

            # 填写你的爱好
```

```
        ]
    }
}
print(json.dumps(my_data, ensure_ascii=False, indent=2))
```

练习四：挑战题(tiǎozhàn tí)

尝试创建一个包含 5 个同学信息的 JSON 数据，

然后将它转换为 Pandas DataFrame，并执行以下操作：

- 计算每个同学的平均分
- 找出平均分最高的同学
- 按平均分排序

提示代码：

```
# 提示：
# 1. 创建包含多个学生的列表(list)
# 2. 用 pd.DataFrame() 转换
# 3. 用 df["新列"] = 计算公式 添加平均分
# 4. 用 df.sort_values() 排序
# 5. 用 df.iloc[0] 找出第一名
```

课堂总结 (zǒngjié)

同学们辛苦(xīnkǔ)了！今天的实践练习我们学习了很多内容。

今天的实践成果：

- ✅ 学会了用 Python 创建 Pandas DataFrame（结构化数据）
- ✅ 学会了 DataFrame 的基本操作：选择、筛选、添加列
- ✅ 体验了非结构化数据（文本）的处理
- ✅ 练习了 JSON 和 XML（半结构化数据）的解析
- ✅ 完成了三种数据类型的直观对比
- ✅ 学会了排序、分组聚合、统计描述、导出等进阶操作

理论与实践的对应(duìyìng)关系：

- 结构化数据 → Pandas DataFrame, CSV, Excel
- 非结构化数据 → 文本处理, 标签提取
- 半结构化数据 → JSON 解析, XML 解析
- 数据类型转换 → JSON/XML → DataFrame

下次课预告(yùgào)：

下次课我们会学习更多的 Pandas 操作和数据可视化(kěshìhuà)。

请同学们课后多练习今天的内容！

有问题随时来找老师！加油(jiāyóu)！