02



# 数据结构及类型
# Data Structure and Type

2026.03

授课教师：[李阳源]

# SL01:目录



1. 数据结构的概念
2. 结构化数据 (Structured Data)
3. 非结构化数据 (Unstructured Data)
4. 半结构化数据 (Semi-structured Data)
5. 数据类型比较
6. 熊猫数据框架 (kuàngjià Pandas DataFrame) 概述
7. 数据框架结构理解









# SL02: 课前导入...



今天将学习的内容:

- 数据分析基础的各种数据结构类型
- 结构化、非结构化、半结构化数据的特点和示例
- 熊猫数据框架的基本结构(Pandas DataFrame)
- 在实际工作中，我们需要区分不同形式的数据并适当(shìdàng)处理它们

# SL03:什么是数据结构?



数据结构 (Data Structure) 是指存储和组织数据的方式。

- 计算机高效管理数据的形式
- 影响数据分析时适当工具和技术的选择
- 决定数据的可访问性和处理便捷性

为什么重要?

- 选择适当的数据结构直接影响分析效率
- 数据存储方式影响处理速度和内存使用量






DYLON

# SL04:结构化数据 (Structured Data)



**定义**: 按照预先定义的格式或模式存储的数据

**特点**:

- 由行和列组成
- 具有明确的数据类型
- 主要用于关系型数据库(relational database)
- 可以轻松使用SQL查询

**示例**:

- Excel电子表格(excel spread sheet)
- 关系型数据库表(relational database table)

# SL05:结构化数据示例

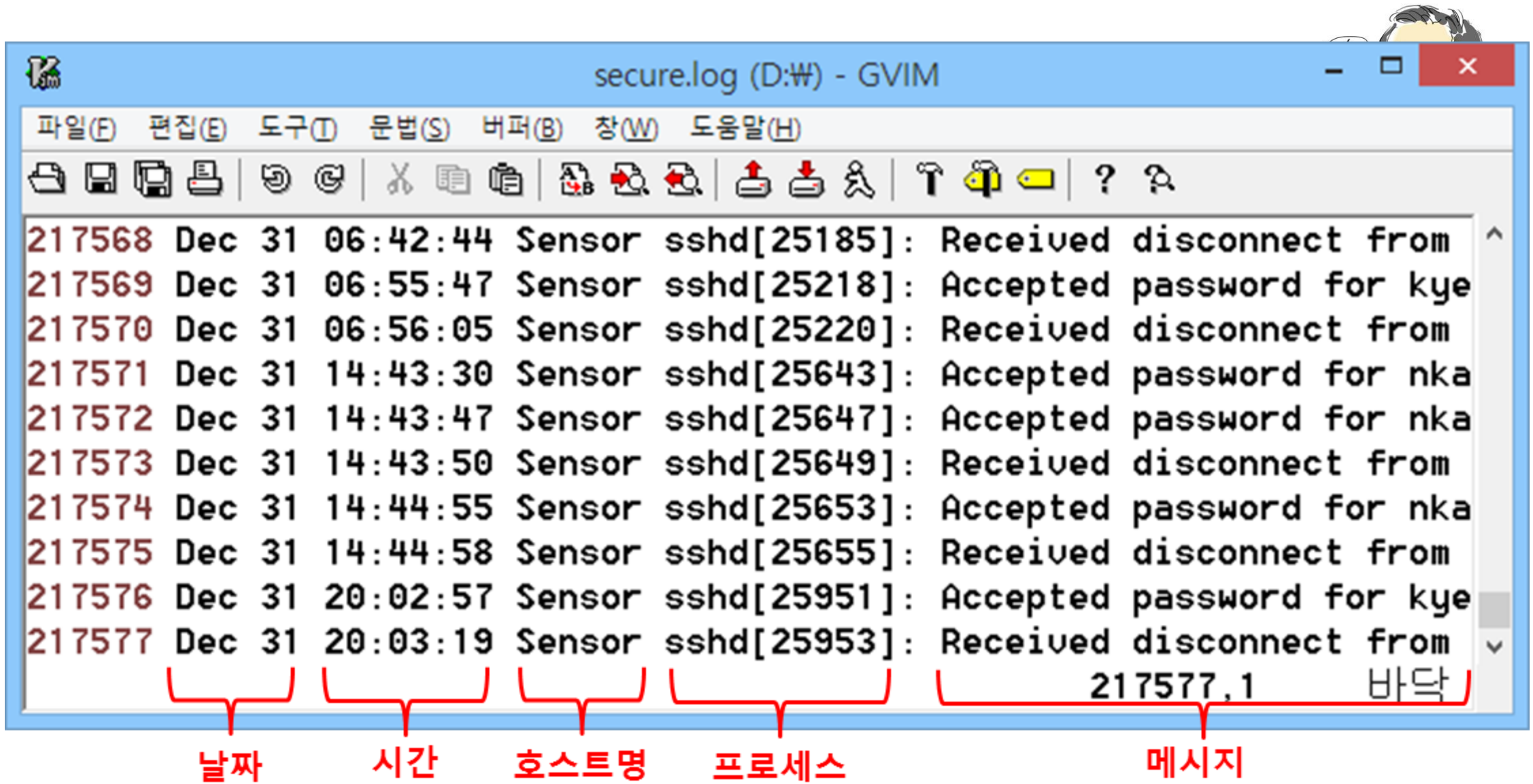


1. 大学生成绩数据:

| 学号 | 姓名 | 专业 | 数学 | 统计学 | 编程 |
|---|---|---|---|---|---|
| 1001 | 张伟 | 计算机工程 | 85 | 92 | 88 |
| 1002 | 李娜 | 统计学 | 95 | 90 | 82 |
| 1003 | 王浩 | 数据科学 | 78 | 85 | 91 |

2. 特点:

- 规则的形式
- 每列有固定的数据类型
- 行是单个记录(学生)，列是属性(科目)


secure.log (D:₩) - GVIM
파일(F) 편집(E) 도구(T) 문법(S) 버퍼(B) 창(W) 도움말(H)
217568 Dec 31 06:42:44 Sensor sshd[25185]: Received disconnect from
217569 Dec 31 06:55:47 Sensor sshd[25218]: Accepted password for kye
217570 Dec 31 06:56:05 Sensor sshd[25220]: Received disconnect from
217571 Dec 31 14:43:30 Sensor sshd[25643]: Accepted password for nka
217572 Dec 31 14:43:47 Sensor sshd[25647]: Accepted password for nka
217573 Dec 31 14:43:50 Sensor sshd[25649]: Received disconnect from
217574 Dec 31 14:44:55 Sensor sshd[25653]: Accepted password for nka
217575 Dec 31 14:44:58 Sensor sshd[25655]: Received disconnect from
217576 Dec 31 20:02:57 Sensor sshd[25951]: Accepted password for kye
217577 Dec 31 20:03:19 Sensor sshd[25953]: Received disconnect from
217577,1 바닥
날짜
시간
호스트명
프로세스
메시지

# SL06:非结构化数据 (Unstructured Data)



**定义**: 没有预先定义的模型或模式的数据

**特点**:

- 非标准化的形式
- 文本(text)、图像(image)、音频(audio)、视频(video)等
- 未经特殊处理难以分析
- 在大数据环境中占据很大比重

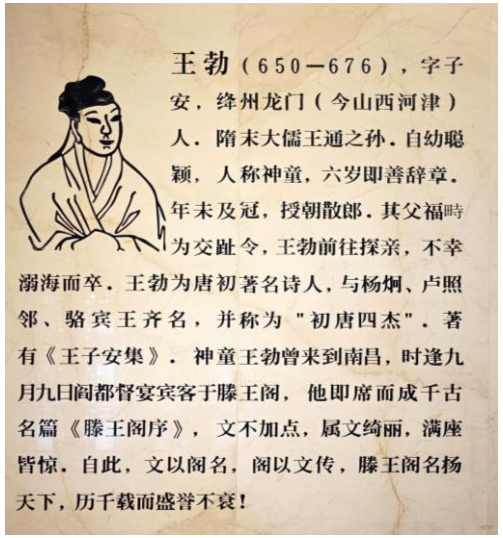
**示例**:

- 社交媒体帖子
- 电子邮件内容
- 语音录音
- 视频内容








王勃（650—676），字子安，绛州龙门（今山西河津）人。隋末大儒王通之孙。自幼聪颖，人称神童，六岁即善辞章。年未及冠，授朝散郎。其父福時为交趾令，王勃前往探亲，不幸溺海而卒。王勃为唐初著名诗人，与杨炯、卢照邻、骆宾王齐名，并称为"初唐四杰"。著有《王子安集》。神童王勃曾来到南昌，时逢九月九日阎都督宴宾客于滕王阁，他即席而成千古名篇《滕王阁序》，文不加点，属文绮丽，满座皆惊。自此，文以阁名，阁以文传，滕王阁名扬天下，历千载而盛誉不衰！

# SL07:非结构化数据示例



**社交媒体文本**☹social media text)
"今天的数据结构课太有趣了! 😊 #大数据 #大学生活"

**图像数据**☹image data)
- 照片、图表、图形等视觉资料

**文档数据**☹document data)
- 报告、论文、文章等以文本为中心的文档

**特点**:
- 没有固定的模式
- 数据长度和内容各不相同
- 需要特殊处理技术(文本挖掘wājué 、计算机视觉等)

# SL08:半结构化数据 (Semi-structured Data)



**定义**: 结构化和非结构化数据之间的中间形态

**特点**:

- 有一定结构但保持灵活性
- 用标签或标记区分数据元素
- 具有自描述性(self-describing)特性
- 可以有层次结构

**示例**:

- JSON (JavaScript Object Notation)
- XML (eXtensible Markup Language)
- HTML文档（HTML Document)
- 电子邮件 (标题是结构化的，正文是非结构化的)

# SL09:半结构化数据示例: JSON



JSON是JavaScript Object Notation的缩写

**特点**:

- 由键值对组成(key-value)
- 可以表示层次结构
- 可以嵌套数组和对象
- 可读性好，在网络上广泛使用

```
{
  "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"]
  }
}
```

# SL10:半结构化数据示例: XML



XML是eXtensible Markup Language的缩写

**特点**:

- 用标签区分数据元素
- 适合表示层次结构
- 兼容各种应用程序
- 比HTML有更严格的语法规则

```xml
<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>
```

# SL11:数据类型比较



| 特性 | 结构化数据 | 半结构化数据 | 非结构化数据 |
| --- | --- | --- | --- |
| 模式 | 固定 | 灵活 | 无 |
| 存储格式 | 表格 | 标签/键值 | 多样 |
| 查询便捷性 | 高 (SQL) | 中 (XPath, JSONPath) | 低 |
| 可扩展性 | 有限 | 高 | 非常高 |
| 处理难度 | 简单 | 中等 | 困难 |
| 示例 | 数据库表 | JSON, XML | 图像, 文本 |

# SL12:现实生活中的数据类型案例



**结构化数据**:

- 银行交易记录
- 学生成绩单
- 产品列表和库存

**半结构化数据**:

- 网站配置文件
- API响应数据
- 电子商务产品目录

**非结构化数据**:

- 客户评论文本
- 监控视频
- 语音通话录音

# SL13:熊猫数据框架 (Pandas DataFrame) 概述



**定义**:

- Python中处理结构化数据的二维表格数据结构
- 类似于电子表格的行列结构(spread sheet)

**特点**:

- 列(column)可以有不同的数据类型
- 可以给行和列添加标签(label)
- 提供处理缺失数据的功能
- 提供多种数据操作和分析功能
- 是大数据预处理的重要工具

# SL14:熊猫数据框架结构(pandas data frame)



**组成部分**:

- **索引(Index)**: 行标签 (0, 1, 2, 3)
- **列名(Columns)**: 列标签 (A, B, C)
- **数据(Values)**: 实际存储的值

**特性**:

- 可以存储不同数据类型
- 支持类似SQL的操作
- 提供索引和切片功能

| | A | B | C |
|---|---|---|---|
| 0 | 1.0 | 5.0 | 9.0 |
| 1 | 2.0 | 6.0 | 10.0 |
| 2 | 3.0 | 7.0 | 11.0 |
| 3 | 4.0 | 8.0 | 12.0 |

# SL15:熊猫数据框架的优势



1. **支持多种数据源**
   - 读写CSV, Excel, SQL DB, JSON等多种格式
2. **灵活的数据处理**
   - 处理缺失值、数据筛选、分组等功能
3. **高效内存使用**
   - 针对大数据处理优化的内部结构
4. **数据可视化集成**
   - 与matplotlib, seaborn等轻松连接
5. **快速运算速度**
   - 内部使用C语言实现，提供高性能



瑞士军刀

# SL16:数据框架 vs SQL表格



**共同点**:

- 结构化的行列数据形式
- 数据筛选、连接、聚合功能(filtering, join, aggregation)
- 基于模式的数据结构 (schema)

**差异**:

- 数据框架: 内存中数据结构(internal memory)，临时性
- SQL表格: 持久存储，支持事务 (transaction)

SQL用于数据库，熊猫用于Python中的数据分析!

# SL17:数据框架基本操作



1. 创建: 从各种来源创建数据框架
2. 选择: 选择特定行/列
3. 筛选: 提取符合条件的数据
4. 转换: 改变数据形态
5. 聚合: 计算统计值
6. 合并: 合并多个数据框架
7. 分组: 按特定标准分组数据

数据预处理和分析过程中的核心操作!

# SL18:熊猫数据框架为什么重要



在大数据分析流程中:

1. **数据收集** → 2. **数据预处理** → 3. **分析和建模** → 4. **可视化**

- 熊猫在第2步(预处理)中扮演核心角色
- 大多数数据科学项目中60-80%的时间用于预处理
- 是创建干净数据的必备工具
- 提供数据结构变更的灵活性

# SL19:理解度测试



1. 以下哪项不是结构化数据的例子？ a) Excel电子表格 b) 社交媒体帖子 c) CSV文件 d) 关系型数据库表
2. 以下哪项不是熊猫数据框架的基本组成部分？ a) 索引 b) 列名 c) 事务 d) 数据值
3. 关于半结构化数据的特点，以下哪项是正确的？ a) 仅由行和列组成 b) 用标签或键值对区分数据 c) 完全没有模式 d) 可以轻松使用SQL查询

# S20:总结和下次课预告



**今天学习的内容**:

- 数据结构的三种类型: 结构化、半结构化、非结构化
- 各类型的特点和实际案例
- 熊猫数据框架的基本概念和重要性

**下次课预告**:

- 使用熊猫进行实际数据操作练习
- 数据加载、基本操作、简单分析技术
- 在PyCharm中搭建实践环境和编写代码

**有问题吗?**

# PL01:目录



1. 实践环境确认
2. Python基础语法复习
3. Numpy核心功能
4. Pandas基本结构
5. 数据加载实践
6. 数据探索基础
7. 实践作业指南
8. 问答与下节课介绍

# SL02: 实践环境确认



**PyCharm使用设置**:

- 项目目录: caowei
- Python解释器设置: Python 3.13+版本
- 必要包安装确认:

```
pip list
pip install --no-index --find-links=my_packages numpy
pandas matplotlib seaborn scikit-learn
```

**文件创建方法**:

1.新建Python文件: File → New → Python File

2.命名: dbp0416.py

# SL03:Python基础语法复习



## 1. 变量和数据类型:

```python
# week2_practice.py
# 基本数据类型
num_int = 10              # 整数型(Integer)
num_float = 3.14          # 浮点型(Float)
text = "Hello World"      # 字符串(String)
is_true = True            # 布尔值(Boolean)

# 输出确认
print("整数:", num_int, type(num_int))
print("浮点数:", num_float, type(num_float))
print("字符串:", text, type(text))
print("布尔值:", is_true, type(is_true))
```

# SL04:数据结构复习 (1)



1. 列表(List)和字典(Dictionary):

```python
# week2_practice.py
# 列表（List） - 有序集合
numbers = [1, 2, 3, 4, 5]
mixed_list = [1, "Python", 3.14, True]

# 列表操作
print("列表项访问:", numbers[0])  # 第一项
print("列表切片:", numbers[1:3])  # 第二项到第三项
print("列表长度:", len(numbers))

# 字典（Dictionary） - 键值对
student = {
    "id": 1001,
    "name": "张伟",
    "major": "计算机工程",
    "grades": {"math": 85, "statistics": 92, "programming": 88}
}

print("字典值访问:", student["name"])
print("嵌套字典访问:", student["grades"]["math"])
```

# SL05:数据结构复习 (1)



1. **元组(Tuple)和集合(Set)**:

```python
# week2_practice.py
# 元组（Tuple）- 不可修改的列表
coordinates = (10, 20)
person = ("张伟", 25, "北京")

# 元组操作
print("元组项访问:", coordinates[0])
print("元组解包:", *coordinates)  # 解包

# 集合（Set）- 无重复项的集合
fruits = {"苹果", "香蕉", "橙子", "苹果"}  # 重复的"苹果"只存储一次
print("集合内容:", fruits)
print("集合长度:", len(fruits))  # 3（去重）
```

# SL06:控制语句复习

1. 控制语句复习

```python
# week2_practice.py
# 条件语句（if-elif-else）
score = 85
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"
print(f"分数 {score}的等级: {grade}")

# 循环（for, while）
# 遍历列表中的所有项
for num in numbers:
    print(num ** 2, end=" ")  # 输出平方值
print()  # 换行

# 循环0到4
for i in range(5):
    print(i, end=" ")
print()
```

# SL07:函数定义与使用

1. **函数基本结构:**

```python
# week2_practice.py
# 函数定义
def calculate_average(numbers):
    """
    计算数字列表的平均值的函数

    Args:
        numbers (list): 数字列表

    Returns:
        float: 平均值
    """
    if not numbers:
        return 0
    return sum(numbers) / len(numbers)

# 函数调用
scores = [85, 92, 78, 90, 88]
avg = calculate_average(scores)
print(f"平均分数: {avg:.2f}")  # 保留两位小数
```

# SL08:Numpy核心功能 (1)

## 1. 数组创建与基本操作:

```python
# week2_numpy.py
import numpy as np

# 创建1D数组
arr1d = np.array([1, 2, 3, 4, 5])
print("1D数组:", arr1d)
print("数组形状:", arr1d.shape)  # (5,)
print("数组维度:", arr1d.ndim)   # 1

# 创建2D数组
arr2d = np.array([[1, 2, 3], [4, 5, 6]])
print("2D数组:\n", arr2d)
print("数组形状:", arr2d.shape)  # (2, 3)
print("数组维度:", arr2d.ndim)   # 2

# 特殊数组创建
zeros = np.zeros((2, 3))  # 2x3形状的0数组
ones = np.ones((3, 2))    # 3x2形状的1数组
identity = np.eye(3)      # 3x3单位矩阵
random_arr = np.random.rand(2, 2)  # 2x2随机数组(0-1之间)

print("zeros:\n", zeros)
print("ones:\n", ones)
print("identity:\n", identity)
print("random_arr:\n", random_arr)
```

# SL09:Numpy核心功能 (2)

## 1. 数组创建与基本操作:

```python
# week2_numpy.py
# 数组运算(向量化操作)
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print("a + b =", a + b)  # 元素级加法
print("a * b =", a * b)  # 元素级乘法
print("a * 2 =", a * 2)  # 标量乘法

# 统计函数
print("平均值:", np.mean(a))
print("总和:", np.sum(a))
print("最大值:", np.max(a))
print("最小值:", np.min(a))

# 索引与切片
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print("arr:\n", arr)
print("arr[0, 0] =", arr[0, 0])  # 第一行, 第一列
print("arr[1, :] =", arr[1, :])  # 第二行全部
print("arr[:, 1] =", arr[:, 1])  # 第二列全部
print("arr[1:3, 2:4] =\n", arr[1:3, 2:4])  # 部分矩阵
```

# SL10:Pandas基本结构 (1)



## 1. Series与DataFrame概述:

```python
# week2_pandas.py
import pandas as pd
import numpy as np

# 创建Series(一维数组)
s1 = pd.Series([1, 2, 3, 4, 5])
print("基本Series:\n", s1)

# 指定索引的Series
s2 = pd.Series([10, 20, 30, 40, 50], index=['a', 'b', 'c', 'd', 'e'])
print("指定索引的Series:\n", s2)

# 从字典创建Series
s3 = pd.Series({'a': 100, 'b': 200, 'c': 300})
print("从字典创建的Series:\n", s3)
```

# SL11:Pandas基本结构 (2)

## 1. DataFrame

**创建与基本操作:**

```python
# week2_pandas.py
# 从字典创建DataFrame
data = {
    'name': ['张伟', '李娜', '王浩', '赵雪'],
    'age': [22, 20, 21, 23],
    'major': ['计算机工程', '统计学', '数据科学', '人工智能'],
    'grade': [3.5, 4.2, 3.8, 4.0]
}

df = pd.DataFrame(data)
print("基本DataFrame:\n", df)

# 查看DataFrame信息
print("\nDataFrame信息:")
print("大小:", df.shape)  # (行, 列)
print("列名:", df.columns)
print("索引:", df.index)
print("数据类型:\n", df.dtypes)

# DataFrame基础统计
print("\n基础统计量:\n", df.describe())

# 特定列访问
print("\n姓名列:\n", df['name'])
print("\n年龄和成绩:\n", df[['age', 'grade']])
```

# SL12:Pandas 基本结构 (3)

1. **DataFrame 行和列访问**:

```python
# week2_pandas.py
# 行访问(基于索引)
print("\n前两行:\n", df.iloc[0:2])

# 行访问(基于标签)
print("\n索引1, 3的行:\n", df.loc[[1, 3]])

# 基于条件的筛选
print("\n21岁以上的学生:\n", df[df['age'] >= 21])
print("\n计算机工程专业的学生:\n", df[df['major'] == '计算机工程'])

# 添加新列
df['year'] = [2, 1, 2, 4]  # 添加年级
print("\n添加年级后:\n", df)

# 删除列
df_drop = df.drop('year', axis=1)  # axis=1表示列方向
print("\n删除年级后:\n", df_drop)
```

# SL13:数据加载实践

1. 读取各种格式的文件:

```python
# week2_data_loading.py
import pandas as pd
import os

# 确认当前工作目录
print("当前工作目录:", os.getcwd())

# 读取CSV文件(例: student_data.csv)
# 文件应该在同一目录下
try:
    students_csv = pd.read_csv('student_data.csv')
    print("\nCSV文件加载成功:\n", students_csv.head())
except FileNotFoundError:
    print("找不到CSV文件。")

    # 创建示例数据(如果文件不存在)
    students_data = {
        'id': range(1001, 1011),
        'name': ['学生'+str(i) for i in range(1, 11)],
        'math': [85, 76, 92, 88, 65, 78, 94, 82, 71, 89],
        'english': [92, 88, 95, 77, 84, 91, 79, 85, 83, 90],
        'science': [79, 85, 88, 91, 76, 82, 95, 79, 85, 92]
    }
    students_csv = pd.DataFrame(students_data)
    print("\n创建示例数据:\n", students_csv.head())

    # 保存为CSV文件
    students_csv.to_csv('student_data.csv', index=False)
    print("已将示例数据保存为'student_data.csv'。")
```

# SL14:数据加载实践(继续)

## 1. 读取/写入各种格式的文件:

```
# week2_data_loading.py
# 读取/写入Excel文件
try:
    # 保存Excel文件
    students_csv.to_excel('student_data.xlsx', index=False, shee
    print("\n已保存'student_data.xlsx'文件。")

    # 读取Excel文件
    students_excel = pd.read_excel('student_data.xlsx', sheet_na
    print("Excel文件加载成功:\n", students_excel.head())
except Exception as e:
    print("处理Excel文件时出错:", e)

# 读取/写入JSON文件
try:
    # 保存JSON文件
    students_csv.to_json('student_data.json', orient='records')
    print("\n已保存'student_data.json'文件。")

    # 读取JSON文件
    students_json = pd.read_json('student_data.json')
    print("JSON文件加载成功:\n", students_json.head())
except Exception as e:
    print("处理JSON文件时出错:", e)
```

# SL15:数据探索基础 (1)



## 1. 数据摘要与探索:

```python
# week2_data_exploration.py
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 加载示例数据(之前步骤中创建的CSV文件)
students = pd.read_csv('student_data.csv')

# 检查基本信息
print("数据大小:", students.shape)
print("\n前5行:\n", students.head())
print("\n基本统计量:\n", students.describe())

# 检查缺失值
print("\n缺失值数量:\n", students.isnull().sum())

# 计算相关性
numeric_data = students.select_dtypes(include=['float64', 'int64'])
correlation = numeric_data.corr()
print("\n相关性:\n", correlation)
```

```python
# 简单可视化
plt.figure(figsize=(10, 6))

# 相关性热图
plt.subplot(1, 2, 1)
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title('科目间相关性')

# 数学成绩直方图
plt.subplot(1, 2, 2)
sns.histplot(students['math'], kde=True)
plt.title('数学成绩分布')

plt.tight_layout()
plt.savefig('data_exploration.png')  # 保存图像
plt.close()

print("\n已将可视化图像保存为'data_exploration.png'。")
```

# SL16:数据探索基础 (2)



## 1. 分组和数据透视表:

```python
# week2_data_exploration.py
# 成绩等级分配函数
def assign_grade(score):
    if score >= 90:
        return 'A'
    elif score >= 80:
        return 'B'
    elif score >= 70:
        return 'C'
    else:
        return 'D'

# 为每个科目计算等级
for subject in ['math', 'english', 'science']:
    students[subject + '_grade'] = students[subject].apply(assign_grade)

print("\n添加等级后的数据:\n", students.head())
```

```python
# 计算每个等级的学生数
grade_counts = {
    'math': students['math_grade'].value_counts(),
    'english': students['english_grade'].value_counts(),
    'science': students['science_grade'].value_counts()
}

print("\n数学各等级学生数:\n", grade_counts['math'])
print("\n英语各等级学生数:\n", grade_counts['english'])
print("\n科学各等级学生数:\n", grade_counts['science'])
```

# SL17:实践作业指南



**个人实践作业**:

1.使用课堂上创建的student_data.csv文件:

- 添加所有科目的总分和平均分列
- 找出平均成绩前5名的学生
- 计算每个科目的最高分和最低分之间的差异

2.文件名：week2_assignment.py

3.提交方法:

- 在下节课开始前通过电子邮件提交 (ywlee1212@outlook.kr)
- 包含代码和简短的结果说明

# SL18:问答与下节课介绍



**今天学习的内容**:

- Python基本语法和数据结构复习
- Numpy和Pandas核心功能
- 数据文件读取/写入
- 基本数据探索方法

**下节课预告**:

- 缺失值(Missing Value)处理方法
- Pandas .isnull(), .dropna(), .fillna() 等函数应用
- 比较各种缺失值替换方法

**有问题吗?**