数据预处理

字符串与字符处理

实践工作册

编程语言：Python

姓名：_____　　学号：_____

班级：_____　　日期：_____

目录

第一章　字符串基础操作 3

第二章　字符串索引与切片 6

第三章　字符串常用方法 9

第四章　字符串格式化 13

第五章　正则表达式基础 16

第六章　数据清洗实战 20

第七章　综合挑战题 24

参考答案 27

第一章 字符串基础操作

一、知识回顾

Python中的字符串（string）是不可变的序列类型，可以使用单引号、双引号或三引号创建。

操作	说明	示例
创建字符串	使用引号包裹文本	`s = "Hello"`
字符串拼接(pīnjiē)	使用 `+` 连接字符串	`"Hello" + " World"`
字符串重复	使用 `*` 重复字符串	`"ab" * 3  # "ababab"`
获取长度	`len()`函数	`len("Hello")  # 5`
类型转换	`str()`函数	`str(123)  # "123"`
成员检测	`in / not in`	`"ell" in "Hello"  # True`

二、基础练习

练习 1：字符串创建与基本操作

请预测以下代码的输出结果，并在空白处填写答案：

```
a = "Python"
b = "数据预处理"
c = a + " " + b
print(c)
```

答： _____

```
print(len(a))
```

答： _____

```
print(a * 3)
```

答： _____

```
print("数据" in b)
```

答： _____

练习 2：类型转换练习

请判断以下表达式的结果类型和值：

表达式	结果	类型
`str(100)`	_____	_____
`int("256")`	_____	_____
`float("3.14")`	_____	_____
`str(True)`	_____	_____
`bool("")`	_____	_____
`bool("0")`	_____	_____
`bool("False")`	_____	_____

> **提示：** 空字符串转换为布尔值为False，任何非空字符串（包括"0"和"False"）都为True。

练习 3：编程实践 - 字符串基本操作

请完成以下任务：

(1) 创建一个包含你名字的字符串，计算其长度

(2) 将名字与"你好，"拼接，生成问候语

(3) 检查名字中是否包含某个特定字符

请在下方编写代码：

```
# 你的代码：
```

第二章 字符串索引与切片

一、知识回顾

Python字符串支持正向索引（从0开始）和反向索引（从-1开始）。切片语法为 s[start:stop:step]。

```
#  索引示意图：
#  正向：  0  1  2  3  4  5
#          P  y  t  h  o  n
#  反向：-6 -5 -4 -3 -2 -1
```

操作	说明	示例
单个索引	获取单个字符	`s[0]  # "P"`
切片	获取子字符串	`s[1:4]  # "yth"`
步长切片	每隔n个字符取一个	`s[::2]  # "Pto"`
反向切片	从后往前取	`s[::-1]  # "nohtyP"`

二、基础练习

练习 1： 索引操作

已知 s = "Data Preprocessing"，请填写以下表达式的结果：

表达式	结果
`s[0]`	_____
`s[-1]`	_____
`s[5]`	_____

表达式	结果
`s[0:4]`	_____
`s[5:]`	_____
`s[:4]`	_____
`s[::2]`	_____
`s[::-1]`	_____
`s[5:17:3]`	_____

练习 2：切片应用

请使用切片操作完成以下任务（s = "Hello, World!"）：

(1) 提取 "Hello"

```
result = s[____:____]
```

答： _____

(2) 提取 "World"

```
result = s[____:____]
```

答： _____

(3) 反转字符串

```
result = s[____:____:____]
```

答： _____

(4) 每隔一个字符取一个(문자열에서 한 글자마다 하나씩 선택한다, 2칸 간격으로 획득)

```
result = s[____:____:____]
```

答： _____

练习 3：编程实践 - 身份证号解析

中国身份证号码为18位，其中：

- 第1-6位：地区编码
- 第7-14位：出生日期（YYYYMMDD）
- 第15-17位：顺序码（奇数为男，偶数为女）
- 第18位：校验码

请编写函数解析身份证号码：

```
def parse_id_card(id_number):
    """
    解析身份证号码，返回地区编码、出生日期、性别
    """
    region = id_number[______]      # 提取地区编码
    year = id_number[______]        # 提取出生年份
    month = id_number[______]       # 提取出生月份
    day = id_number[______]         # 提取出生日期
    gender_code = int(id_number[______])  # 提取顺序码末位
    gender = "男" if gender_code % 2 == 1 else "女"

    return {
        "地区编码": region,
        "出生日期": f"{year}年{month}月{day}日",
        "性别": gender
    }

# 测试
info = parse_id_card("110105199003077654")
print(info)
```

预期输出：

答： _____

第三章 字符串常用方法

一、知识回顾

大小写转换方法

方法	说明	示例
upper()	转为大写	"hello".upper() # "HELLO"
lower()	转为小写	"HELLO".lower() # "hello"
title()	每个单词首字母大写	"hello world".title()
capitalize()	首字母大写	"hello".capitalize()
swapcase()	大小写互换	"Hello".swapcase() # "hELLO"

查找与替换方法

方法	说明	示例
find(sub)	查找子串位置，未找到返回-1	"hello".find("ll") # 2
index(sub)	查找子串位置，未找到报错	"hello".index("ll") # 2
count(sub)	统计子串出现次数	"hello".count("l") # 2
replace(old,new)	替换子串	"hello".replace("l","L")
startswith(s)	判断是否以s开头	"hello".startswith("he")
endswith(s)	判断是否以s结尾	"hello".endswith("lo")

分割与连接方法

方法	说明	示例
split(sep)	按分隔符分割	"a,b,c".split(",")
rsplit(sep)	从右侧开始分割	"a,b,c".rsplit(",",1)
join(iterable)	连接可迭代对象	",".join(["a","b","c"])
strip()	去除两端空白字符	" hi ".strip() # "hi"

方法	说明	示例
lstrip()	去除左端空白	" hi ".lstrip() # "hi "
rstrip()	去除右端空白	" hi ".rstrip() # " hi"

二、基础练习

练习 1：方法预测

请预测以下表达式的结果：

表达式	结果
"Hello World".upper()	_____
"Hello World".lower()	_____
"hello world".title()	_____
"Hello World".replace("World", "Python")	_____
"Hello World".split()	_____
" ".join(["Data", "Pre", "Processing"])	_____
" Data Science ".strip()	_____
"hello".center(11, "*")	_____
"Python".find("th")	_____
"banana".count("a")	_____

练习 2：代码填空 - 文本清洗

请填写空白处的代码，完成文本清洗功能：

```
text = "  Hello, WORLD! Welcome to DATA preprocessing.  "


# 1. 去除两端空白
text = text._____()
```

```
# 2. 转换为小写
text = text._____()


# 3. 将 "data" 替换为 "数据"
text = text._____("data", "数据")


# 4. 按空格分割为单词列表
words = text._____()


# 5. 用下划线连接所有单词
result = "_____"._____(words)


print(result)
```

答： _____

练习 3：编程实践 - 电子邮件地址解析

编写一个函数，解析电子邮件地址，提取用户名和域名：

```
def parse_email(email):
    """
    解析邮件地址，返回用户名和域名
    例如: "zhangsan@example.com" -> ("用户名": "zhangsan", "域名":
"example.com")
    """
    # 请在下方编写代码


# 测试
emails = ["alice@gmail.com", "bob.smith@company.org", "admin@data-
science.edu"]
for email in emails:
    print(parse_email(email))
```

练习 4：编程实践 - CSV行解析器

编写一个函数，解析CSV格式的字符串：

```
def parse_csv_line(line):
    """
    解析CSV行，返回字段列表，并去除每个字段的前后空格
    """
    # 请在下方编写代码









# 测试
line = " 张三 , 25 , 北京 , 工程师 "
print(parse_csv_line(line))
# 预期输出: ["张三", "25", "北京", "工程师"]
```

第四章 字符串格式化

一、知识回顾

Python提供了多种字符串格式化方式，常用的有三种：

1. f-string格式化（推荐）

```
name = "小明"
age = 20
print(f"我叫{name}，今年{age}岁")
print(f"明年{age + 1}岁")
print(f"PI = {3.14159:.2f}")  # 保留两位小数
```

2. format()方法

```
print("我叫{}，今年{}岁".format("小明", 20))
print("我叫{name}，今年{age}岁".format(name="小明", age=20))
print("{:>10}".format("right"))    # 右对齐
print("{:<10}".format("left"))     # 左对齐
print("{:^10}".format("center"))   # 居中
```

3. %操作符（旧式）

```
print("我叫%s，今年%d岁" % ("小明", 20))
print("价格：%.2f元" % 99.9)
```

常用格式化说明符

说明符	说明	示例
:d	整数	f"{42:d}" # "42"
:.nf	浮点数，保留n位小数	f"{3.14159:.2f}" # "3.14"
:,	千位分隔符	f"{1000000:,}" # "1,000,000"
:%	百分比	f"{0.85:%}" # "85.000000%"
:>n	右对齐，宽n	f"{"hi":>10}"
:<n	左对齐，宽n	f"{"hi":<10}"
:^n	居中，宽n	f"{"hi":^10}"
:0nd	零填充	f"{42:05d}" # "00042"

二、基础练习

练习 1： 格式化输出练习

请使用f-string完成以下格式化任务：

```
product = "笔记本电脑"
price = 5999.5
quantity = 3
discount = 0.85


# 1. 输出："商品：笔记本电脑，单价：5999.50元"
print(f"_____")


# 2. 输出："数量：003"
print(f"_____")


# 3. 输出："折扣：85.0%"
print(f"_____")


# 4. 输出："总价：15,298.78元"
```

```
total = price * quantity * discount
print(f"_____")
```

练习 2：编程实践 - 成绩单生成器

编写一个函数，生成格式化的成绩单：

```
def generate_report(students):
    """
    生成成绩单
    students: [("姓名", 语文, 数学, 英语), ...]
    """
    print("=" * 50)
    print(f"{"_____":^50}")
    print("=" * 50)
    # 表头：姓名(10字符宽) 语文(8) 数学(8) 英语(8) 平均分(8)

print(f"{"_____":<10}{"_____":>8}{"_____":>8}{"_____":>8}{"_____":>8}"
)
    print("-" * 50)
    for name, chinese, math, english in students:
        avg = (chinese + math + english) / 3
        # 请完成格式化输出
        print(f"_____")
    print("=" * 50)


# 测试数据
students = [
    ("张三", 92, 85, 88),
    ("李四", 78, 95, 82),
    ("王五", 88, 90, 95),
]
generate_report(students)
```

第五章 正则表达式基础

一、知识回顾

正则表达式（Regular Expression）是一种强大的字符串匹配工具，Python通过re模块提供支持。

常用元字符

元字符	说明	示例
.	匹配任意字符（除\n）	"a.c" 匹配 "abc"
\d	匹配数字 [0-9]	"\d+" 匹配 "123"
\w	匹配字母数字下划线	"\w+" 匹配 "hello_1"
\s	匹配空白字符	"\s+" 匹配 " "
^	匹配字符串开头	"^Hello"
$	匹配字符串结尾	"world$"
*	重复0次或多次	"ab*" 匹配 "a","ab","abb"
+	重复1次或多次	"ab+" 匹配 "ab","abb"
?	重复0次或1次	"ab?" 匹配 "a","ab"
{n,m}	重复n到m次	"a{2,4}" 匹配 "aa","aaa"
[]	字符集合	"[aeiou]" 匹配元音
()	分组捕获	"(\d+)" 捕获数字

常用re模块函数

函数	说明	返回值
re.search(p, s)	搜索第一个匹配	Match对象或None
re.match(p, s)	从开头匹配	Match对象或None
re.findall(p, s)	找所有匹配	列表

函数	说明	返回值
re.sub(p, r, s)	替换匹配内容	新字符串
re.split(p, s)	按模式分割	列表
re.compile(p)	编译正则表达式	Pattern对象

二、基础练习

练习 1：正则匹配练习

请写出以下任务对应的正则表达式：

任务	正则表达式
匹配手机号（11位数字，1开头）	_____
匹配电子邮箱地址	_____
匹配日期（YYYY-MM-DD格式）	_____
匹配中文字符	_____
匹配URL地址	_____
匹配IP地址	_____

练习 2：代码填空 - 正则表达式应用

请填写空白处的正则表达式：

```
import re


text = "联系电话：13800138000，邮箱：test@example.com，日期：2024-03-15"


# 1. 提取手机号
phone = re.findall(r"_____", text)
print("手机号:", phone)
```

```
# 2. 提取邮箱
email = re.findall(r"______", text)
print("邮箱:", email)


# 3. 提取日期
date = re.findall(r"______", text)
print("日期:", date)
```

练习 3：编程实践 - 数据提取器

编写一个函数，从文本中提取所有数字并计算总和：

```
import re


def extract_and_sum(text):
    """
    从文本中提取所有数字（包括小数）并求和
    """
    # 请在下方编写代码









# 测试
text = "苹果3.5元/斤，买了2斤，香蕉买2.8元/斤，买了3斤"
print(extract_and_sum(text))
```

练习 4：编程实践 - 敏感信息脱敏

编写一个函数，将文本中的手机号和邮箱进行脱敏处理：

```
import re
```

```
def desensitize(text):
    """
    手机号: 138****8000
    邮箱: t***@example.com
    """
    # 请在下方编写代码









# 测试
text = "用户张三，电话13800138000，邮箱zhangsan@gmail.com"
print(desensitize(text))
```

第六章 数据清洗实战

一、知识回顾

数据清洗是数据预处理的重要环节，常见的字符串清洗任务包括：

- 去除多余空白和特殊字符
- 统一数据格式（日期、电话、地址等）
- 处理缺失值和异常值
- 文本标准化和规范化

二、实战练习

练习 1：姓名清洗

编写函数清洗姓名数据：

```
def clean_name(name):
    """
    清洗规则：
    1. 去除前后空格
    2. 去除名字中的特殊字符（如 #、@、*）
    3. 将英文名字转换为首字母大写
    4. 将多个连续空格替换为单个空格
    """
    # 请在下方编写代码
```

```
# 测试
names = [
    "  张 三  ",
    "john  smith",
    "#李@四*",
    "  WANG   WEI  ",
]
for name in names:
    print(f"\"{name}\" -> \"{clean_name(name)}\"")
```

练习 2：日期格式统一

编写函数将不同格式的日期统一为 YYYY-MM-DD 格式：

```
import re


def normalize_date(date_str):
    """
    支持的输入格式：
    - "2024/03/15"
    - "2024.03.15"
    - "2024年3月15日"
    - "03-15-2024"
    - "15/03/2024"
    输出格式: "2024-03-15"
    """
    # 请在下方编写代码
```

```
# 测试
dates = ["2024/03/15", "2024.3.15", "2024年3月15日", "03-15-2024"]
for d in dates:
    print(f"\"{d}\" -> \"{normalize_date(d)}\"")
```

练习 3：电话号码标准化

编写函数将各种格式的电话号码统一为标准格式：

```
import re


def normalize_phone(phone):
    """
    将各种格式的电话号码统一为 "XXX-XXXX-XXXX"
    输入可能是：
    - "13800138000"
    - "138 0013 8000"
    - "138-0013-8000"
    - "+86 13800138000"
    - "+86-138-0013-8000"
    """
    # 请在下方编写代码









# 测试
```

```
phones = ["13800138000", "138 0013 8000", "+86 13800138000", "+86-138-0013-8000"]
for p in phones:
    print(f"\"{p}\" -> \"{normalize_phone(p)}\"")
```

练习 4：批量数据清洗

编写函数对一批用户数据进行全面清洗：

```
import re


def clean_user_data(users):
    """
    users: [{"name": ..., "phone": ..., "email": ..., "age": ...}, ...]
    清洗规则：
    1. 姓名：去除空格和特殊字符
    2. 电话：统一格式
    3. 邮箱：转小写
    4. 年龄：转为整数，无效值设为None
    """
    cleaned = []
    for user in users:
        # 请在下方编写代码









    return cleaned


# 测试数据
raw_data = [
```

```
    {"name": " 张#三 ", "phone": "+86 13800138000", "email":
"ZHANG@Gmail.COM", "age": "25"},
    {"name": "li si", "phone": "139-0013-9000", "email": "lisi@QQ.com",
"age": "二十"},
    {"name": "  WANG  WU  ", "phone": "15000150000", "email":
"Wang@163.COM", "age": "30"},
]
for user in clean_user_data(raw_data):
    print(user)
```

第七章 综合挑战题

以下题目综合运用本工作册所学的所有知识点，请独立完成。

练习 1：密码强度检测器

编写一个密码强度检测函数，要求：

- 长度至少8位：弱
- 包含大小写字母：中
- 包含数字：强
- 包含特殊字符：非常强

```
import re


def check_password_strength(password):
    """
    返回密码强度等级和建议
    """
    # 请在下方编写代码









# 测试
```

```
passwords = ["123", "abcdefgh", "Abcdefgh", "Abc12345", "Abc@1234!"]
for pwd in passwords:
    result = check_password_strength(pwd)
    print(f"\"{pwd}\" -> {result}")
```

练习 2：文本统计分析器

编写一个文本分析函数，统计：

- 字符总数、单词数、句子数
- 每个单词的出现频率
- 最常见的5个单词
- 平均句子长度

```
import re
from collections import Counter


def analyze_text(text):
    """
    返回文本统计信息的字典
    """
    # 请在下方编写代码
```

```
# 测试
sample_text = """
Python is a great programming language. It is widely used in data science.
Data preprocessing is an important step. Python makes it easy to process
strings.
Learning Python is fun and rewarding.
"""
result = analyze_text(sample_text)
for key, value in result.items():
    print(f"{key}: {value}")
```

练习 3：简易模板引擎

实现一个简易的字符串模板引擎，支持变量替换：

```
import re


def render_template(template, variables):
    """
    将模板中的 {{variable_name}} 替换为实际值
    示例：render_template("Hello {{name}}", {"name": "小明"})
          -> "Hello 小明"
    要求：如果变量未定义，保留原始占位符
    """
    # 请在下方编写代码









# 测试
```

```
template = "尊敬的{{name}}，您的订单{{order_id}}已确认，金额{{amount}}元。编号：{{code}}"
data = {"name": "张三", "order_id": "ORD-2024-001", "amount": "299.00"}
print(render_template(template, data))
# 预期：尊敬的张三，您的订单ORD-2024-001已确认，金额299.00元。编号：{{code}}
```

练习 4：日志文件解析器

编写一个日志解析器，从日志文本中提取结构化信息：

```
import re


def parse_log(log_text):
    """
    解析日志文本，提取每条日志的：
    - 时间戳
    - 日志级别（INFO/WARNING/ERROR）
    - 消息内容
    返回字典列表
    """
    # 请在下方编写代码









# 测试数据
logs = """
[2024-03-15 10:30:45] INFO: 用户登录成功，用户ID=1001
[2024-03-15 10:31:02] WARNING: 磁盘空间不足，剩余：5.2GB
[2024-03-15 10:31:15] ERROR: 数据库连接失败，错误码：1045
```

```
[2024-03-15 10:32:00] INFO: 定时任务执行完成
"""


for entry in parse_log(logs):
    print(entry)
```

参考答案

第一章参考答案

练习1

```
# 输出: Python 数据预处理
# 输出: 6
# 输出: PythonPythonPython
# 输出: True
```

练习2

```
# str(100) -> "100", str类型
# int("256") -> 256, int类型
# float("3.14") -> 3.14, float类型
# str(True) -> "True", str类型
# bool("") -> False, bool类型
# bool("0") -> True, bool类型
# bool("False") -> True, bool类型
```

第二章参考答案

练习1

```
# s[0] -> "D"
# s[-1] -> "g"
# s[5] -> "P"
# s[0:4] -> "Data"
# s[5:] -> "Preprocessing"
# s[:4] -> "Data"
# s[::2] -> "Dt rpoesg"
# s[::-1] -> "gnissecorperP ataD"
# s[5:17:3] -> "Pors"
```

练习2

```
# (1) s[0:5]
# (2) s[7:12]
# (3) s[::-1]
# (4) s[::2]
```

练习3 身份证解析

```
region = id_number[0:6]
year = id_number[6:10]
month = id_number[10:12]
day = id_number[12:14]
gender_code = int(id_number[16])
# 输出: {"地区编码":"110105", "出生日期":"1990年03月07日", "性别":"男"}
```

第三章参考答案

练习1

```
# "HELLO WORLD"
# "hello world"
# "Hello World"
# "Hello Python"
# ["Hello", "World"]
# "Data Pre Processing"
# "Data Science"
# "***hello***"
# 2
# 3
```

练习2

```
text = text.strip()
text = text.lower()
text = text.replace("data", "数据")
words = text.split()
```

```
result = "_".join(words)
```

练习3 邮箱解析

```
def parse_email(email):
    parts = email.split("@")
    return {"用户名": parts[0], "域名": parts[1]}
```

练习4 CSV解析

```
def parse_csv_line(line):
    fields = []                          # 결과를 저장할 빈 리스트 생성
    raw_fields = line.split(",")         # 쉼표를 기준으로 문자열 분할

    for field in raw_fields:             # 분할된 각 항목을 반복
        cleaned_field = field.strip()    # 앞뒤 공백 제거
        fields.append(cleaned_field)     # 정제된 항목을 리스트에 추가

    return fields                        # 최종 리스트 반환
```

第四章参考答案

练习1

```
print(f"商品：{product}，单价：{price:.2f}元")
print(f"数量：{quantity:03d}")
print(f"折扣：{discount:.1%}")
print(f"总价：{total:,.2f}元")
```

第五章参考答案

练习1

```
# 手机号: r"1\d{10}"
# 邮箱: r"[\w.+-]+@[\w-]+\.[\w.]+"
# 日期: r"\d{4}-\d{2}-\d{2}"
```

```
# 中文: r"[\u4e00-\u9fff]+"
```

练习2

```
phone = re.findall(r"1\d{10}", text)
email = re.findall(r"[\w.+-]+@[\w-]+\.[\w.]+", text)
date = re.findall(r"\d{4}-\d{2}-\d{2}", text)
```

练习3 数据提取

```
def extract_and_sum(text):
    numbers = re.findall(r"\d+\.?\d*", text)
    nums = [float(n) for n in numbers]
    return {"numbers": nums, "sum": sum(nums)}
```

练习4 敏感信息脱敏

```
def desensitize(text):
    text = re.sub(r"(1\d{2})\d{4}(\d{4})", r"\1****\2", text)
    text = re.sub(r"([\w])[\w.]*@", r"\1***@", text)
    return text
```

— 工作册完 —