数据预处理

数据结构及类型 — 工作册参考答案

Python 编程语言

— 教师用 —

第三部分 填空练习 参考答案

1. Python中的基本数值类型包括整数型 _____、浮点型 _____、和复数型 _____。

> **int、float、complex**

2. 列表是 _____（可变/不可变）的序列，而元组是 _____（可变/不可变）的序列。

> **可变、不可变**

3. 使用 _____ 函数可以查看一个变量的数据类型。

> **type()**

4. 字典的键必须是 _____（可变/不可变）类型，而值可以是 _____ 类型。

> **不可变、任意**

5. 集合（set）的两个核心特点是 _____ 和 _____。

> **无序、不重复（元素唯一）**

6. 将字符串 "123" 转换为整数的函数是 _____，将整数转换为字符串的函数是 _____。

> **int()、str()**

7. 列表推导式的基本语法是 _____。

> **[表达式 for 变量 in 可迭代对象] 或 [expr for var in iterable]**

8. 字典推导式的基本语法是 _____。

> **{key_expr: value_expr for 变量 in 可迭代对象} 或 {k: v for var in iterable}**

第四部分 代码阅读练习 参考答案

1. 预测以下代码的输出：

```
a = [1, 2, 3, 4, 5]
b = a
b.append(6)
print(a)
print(a is b)
```

输出：

[1, 2, 3, 4, 5, 6]

True

解析： b = a 是引用赋值，a和b指向同一个列表对象。列表是可变类型，通过b修改会直接影响a，因为它们是同一个对象。如果需要独立副本，应使用 b = a.copy() 或 b = a[:]

2. 预测以下代码的输出：

```
data = {'name': 'Alice', 'age': 25, 'scores': [88, 92, 76]}
print(type(data))
print(type(data['scores']))
print(len(data))
print(data.get('gender', 'Unknown'))
```

输出：

<class 'dict'>

<class 'list'>

3

Unknown

解析： data是字典类型，data['scores']是列表类型。len(data)返回字典的键值对数量（3个）。data.get('gender', 'Unknown')在键不存在时返回默认值'Unknown'，而不会报错。

3. 预测以下代码的输出：

```
s1 = {1, 2, 3, 4, 5}
s2 = {3, 4, 5, 6, 7}
print(s1 & s2)       # 交集
print(s1 | s2)       # 并集
print(s1 - s2)       # 差集
```

```
print(s1 ^ s2)       # 对称差集
```

输出：

{3, 4, 5}

{1, 2, 3, 4, 5, 6, 7}

{1, 2}

{1, 2, 6, 7}

解析： & 交集：两个集合共有的元素。| 并集：两个集合所有元素的合并。- 差集：在s1中但不在s2中的元素。^ 对称差集：只在其中一个集合中出现的元素。注意：集合输出顺序可能不同，因为集合是无序的。

4. 预测以下代码的输出：

```
x = (1, 2, [3, 4])
x[2].append(5)
print(x)


# 以下代码会发生什么？
# x[0] = 10
```

输出：

(1, 2, [3, 4, 5])

x[0] = 10 会引发 TypeError，因为元组本身不可变。

解析： 元组是不可变的，但元组内部的可变对象（如列表）可以被修改。x[2]是一个列表引用，可以对其调用append()。但x[0] = 10试图修改元组元素的引用，这是不允许的。

第五部分 编程实践练习 参考答案

练习 5.1 数据类型探索

```
# 1. 创建不同类型的变量
a = 42              # 整数
b = 3.14            # 浮点数
c = 'Hello'         # 字符串
d = True            # 布尔
e = None            # None值


# 2. 打印类型
print(type(a))      # <class 'int'>
print(type(b))      # <class 'float'>
print(type(c))      # <class 'str'>
print(type(d))      # <class 'bool'>
print(type(e))      # <class 'NoneType'>


# 3. 使用 isinstance() 检查数值类型
print(isinstance(a, (int, float)))   # True
print(isinstance(b, (int, float)))   # True
print(isinstance(c, (int, float)))   # False
print(isinstance(d, (int, float)))   # True (注意: bool是int的子类)
```

解析：bool是int的子类，所以isinstance(True, int)返回True。这是一个常见的坑，学生应理解Python的类型继承关系。

练习 5.2 列表操作练习

```
students = ['Alice', 'Bob', 'Charlie', 'David', 'Eve']
scores = [88, 92, 76, 95, 83]


# 任务 1：将两个列表合并为字典
grade_dict = dict(zip(students, scores))
print(grade_dict)
# {'Alice': 88, 'Bob': 92, 'Charlie': 76, 'David': 95, 'Eve': 83}


# 任务 2：找出分数大于90的学生
top_students = [name for name, score in zip(students, scores) if score > 90]
print(top_students)  # ['Bob', 'David']


# 任务 3：计算平均分并替换低于平均分的成绩
avg = sum(scores) / len(scores)  # 86.8
adjusted = [s if s >= avg else avg for s in scores]
print(f'平均分: {avg}')
```

```
print(f'调整后: {adjusted}')


# 任务 4：按分数从高到低排序并输出
sorted_pairs = sorted(zip(scores, students), reverse=True)
for score, name in sorted_pairs:
    print(f'{name}: {score}')
# David: 95
# Bob: 92
# Alice: 88
# Eve: 83
# Charlie: 76
```

练习 5.3 字典的应用

```
students_data = [
    {'name': 'Alice', 'age': 20, 'major': 'CS', 'gpa': 3.8},
    {'name': 'Bob', 'age': 21, 'major': 'Math', 'gpa': 3.5},
    {'name': 'Charlie', 'age': 20, 'major': 'CS', 'gpa': 3.9},
    {'name': 'David', 'age': 22, 'major': 'Physics', 'gpa': 3.2},
    {'name': 'Eve', 'age': 21, 'major': 'CS', 'gpa': 3.7},
]


# 任务 1：统计每个专业的学生人数
major_count = {}
for s in students_data:
    major = s['major']
    major_count[major] = major_count.get(major, 0) + 1
print(major_count)  # {'CS': 3, 'Math': 1, 'Physics': 1}


# 任务 2：找出GPA最高的学生
best = max(students_data, key=lambda s: s['gpa'])
print(best)  # {'name': 'Charlie', 'age': 20, 'major': 'CS', 'gpa': 3.9}


# 任务 3：字典推导式创建 {name: gpa}
name_gpa = {s['name']: s['gpa'] for s in students_data}
print(name_gpa)
# {'Alice': 3.8, 'Bob': 3.5, 'Charlie': 3.9, 'David': 3.2, 'Eve': 3.7}


# 任务 4：按专业分组并计算各专业的平均GPA
major_gpas = {}
for s in students_data:
    major_gpas.setdefault(s['major'], []).append(s['gpa'])


avg_gpa = {m: sum(g)/len(g) for m, g in major_gpas.items()}
print(avg_gpa)
# {'CS': 3.8, 'Math': 3.5, 'Physics': 3.2}
```

练习 5.4 集合运算实践

```
exam1 = {'Alice', 'Bob', 'Charlie', 'David', 'Eve'}
exam2 = {'Bob', 'David', 'Frank', 'Grace', 'Eve'}


# 任务 1：两次都参加了的学生（交集）
both = exam1 & exam2
print(both)   # {'Bob', 'David', 'Eve'}


# 任务 2：只参加了第一次考试的学生（差集）
only_exam1 = exam1 - exam2
print(only_exam1)  # {'Alice', 'Charlie'}


# 任务 3：总共参加考试的学生（并集）
all_students = exam1 | exam2
print(all_students)
# {'Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace'}


# 任务 4：只参加了其中一次的学生（对称差集）
only_one = exam1 ^ exam2
print(only_one)  # {'Alice', 'Charlie', 'Frank', 'Grace'}
```

第六部分 综合挑战 参考答案

挑战 6.1 数据清洗实战

```
raw_orders = [
    {'id': '001', 'product': 'Apple', 'price': '5.5', 'qty': '3'},
    {'id': '002', 'product': 'Banana', 'price': 'N/A', 'qty': '5'},
    {'id': '003', 'product': 'apple', 'price': '5.5', 'qty': '2'},
    {'id': '001', 'product': 'Apple', 'price': '5.5', 'qty': '3'},
    {'id': '004', 'product': 'Cherry', 'price': '15.0', 'qty': ''},
    {'id': '005', 'product': 'Durian', 'price': '45.0', 'qty': '1'},
]


# 1. 去除重复订单（根据 id 判断）
seen_ids = set()
unique_orders = []
for order in raw_orders:
    if order['id'] not in seen_ids:
        seen_ids.add(order['id'])
        unique_orders.append(order)


# 2. 将 price 和 qty 转换为数值类型
def safe_float(val):
    try:
        return float(val)
    except (ValueError, TypeError):
        return 0


def safe_int(val):
    try:
        return int(val)
    except (ValueError, TypeError):
        return 0


for order in unique_orders:
    order['price'] = safe_float(order['price'])
    order['qty'] = safe_int(order['qty'])


# 3. 统一产品名称为首字母大写
for order in unique_orders:
    order['product'] = order['product'].capitalize()


# 4. 计算每个订单的总价
for order in unique_orders:
    order['total'] = order['price'] * order['qty']


# 5. 找出总价最高的订单
max_order = max(unique_orders, key=lambda x: x['total'])
print(f'总价最高的订单: {max_order}')
```

```
# {'id': '005', 'product': 'Durian', 'price': 45.0, 'qty': 1, 'total': 45.0}


# 打印所有清洗后的订单
for order in unique_orders:
    print(order)
```

挑战 6.2 数据结构选择

1. 场景：存储学生的姓名和对应的成绩

数据结构：dict（字典）

理由：字典提供键值对映射，可以用姓名作为键、成绩作为值，查找效率为O(1)。

2. 场景：记录一周七天的气温（不需要修改）

数据结构：tuple（元组）

理由：数据固定且不需要修改，元组的不可变性可以保证数据安全，同时占用更少内存。

3. 场景：统计一篇文章中每个单词出现的次数

数据结构：dict（字典）或 collections.Counter

理由：字典可以将单词作为键、出现次数作为值，Counter是专门用于计数的字典子类，更加简洁。

4. 场景：存储一组不重复的用户ID

数据结构：set（集合）

理由：集合自动去重，且查找、添加的时间复杂度为O(1)，非常适合存储唯一标识符。

第七部分 选择题 参考答案

1. 以下哪个不是Python的不可变类型?

> **C. list**

解析： list（列表）是可变类型，可以原地修改元素。int、str、tuple都是不可变类型，修改时会创建新对象。

2. 执行 d = {}; d[[1,2]] = 'test' 会发生什么?

> **B. TypeError，因为列表不可哈希**

解析： 字典的键必须是可哈希的（即不可变类型）。列表是可变类型，无法哈希，因此不能作为字典的键。如果需要使用序列作为键，可以转换为元组：d[(1,2)] = 'test'

3. 以下哪个表达式的结果是 True?

> **B 和 D**

解析： A: list和tuple是不同类型，结果为False。B: bool是int的子类，所以isinstance(True, int)返回True。C: int和float是不同类型，结果为False。D: 集合比较只看元素内容，重复元素自动去除，{1,2}和{2,1,2}相等。

4. 关于字典的说法，哪个是错误的?

> **D. 字典可以使用列表作为键**

解析： A正确：字典键必须不可变。B正确：Python 3.7+字典保持插入顺序。C正确：值可以是任意类型。D错误：列表是可变类型，不能作为字典的键（不可哈希）。