5.1 知识点定位

青少年编程能力“Python 三级”核心知识点 5：一维数据处理。

5.2 能 力 要 求

掌握并熟练编写使用一维数据的程序，具备解决一维数据处理问题的能力。

5.3 建议教学时长

本单元建议 2 课时。

5.4 教 学 目 标

1. 知识目标

本单元学习一维数据的处理，掌握在 Python 中进行一维数据表示、存储和数据处理及应用的能力。

2. 能力目标

通过一维数据处理的学习，初步建立数据分析和处理的能力。掌握对数据

表示、存储、读写处理的主要步骤，结合 Python 编程中文件的读写方法，能够以永久存储的形式进行一维数据在解决实际问题中的应用。

3. 素养目标

通过一维数据的表示、存储与处理的编程能力培养，初步培养学生获取数据的意识、分析数据的能力和整合数据的思维。

5.5 知识结构

本单元知识结构如图 5-1 所示。

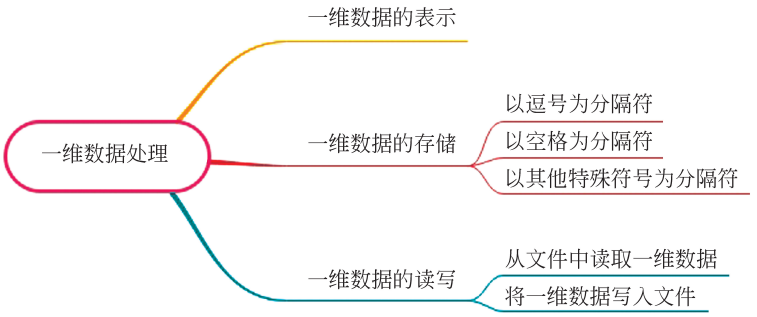

图 5-1　一维数据知识结构

5.6 补充知识

1. 一维数据的表示形式——序列

序列是一维数据的表示形式。一组对等关系的，有序或无序的数据，就构成了序列。序列采用线性方式组织，在 Python 中，有序的序列对应列表、元组、

字符串等类型，无序的序列对应集合、字典等类型。

生活中，具有一维数据特征的数据非常常见，例如，公交及地铁的线路图、各门功课的成绩、课外兴趣小组的成员名单等。一维数据还是我们日常归纳和统计的好帮手，例如，我国宋代著名文人苏轼，一生坎坷，先后在 15 个州县任职，留下了伟大的精神财富，供后人景仰。苏轼一生的行迹沉浮如图 5-2 所示。

图 5-2　苏轼一生的行迹沉浮

在图中，以时间为线索，即沿着时间维度，就非常容易厘清苏轼一生的任职地点以及职位的起伏。这就是一维数据在知识凝练、信息表达中的重要作用。

2. 序列的排序操作

序列，特别是列表，在一维数据表示中应用较为广泛。因此，在 Python 中运用有序序列的排序等操作所需的函数、方法，可以十分快捷地实现对一维数据的处理。下面简要列举两种序列的排序方法、函数。

1）内置函数 sorted()

该函数用于对所有可迭代的对象进行排序操作，其基本格式为：

```
sorted(iterable, cmp=None, key=None, reverse=False)
```

其中，key 是用来进行比较的元素，只有一个参数，具体的参数取自于可迭代对象，指定可迭代对象中的一个元素来进行排序。reverse 代表排序规则，reverse = True 为降序，reverse = False 为升序（默认）。统一举例如下。

```
>>>a = [5,3,4,1,2]
>>> b = sorted(a)
>>> a                    #a 列表保持不变
[5, 3, 4, 1, 2]
>>> b                    #b 列表升序排列
[1, 2, 3, 4, 5]
>>> stus = [('章', '女', 15),('李', '男', 10),('赵','男',12)]
>>> sorted(stus, key=lambda s: s[2], reverse=True)
# 按年龄降序排列
[('章', '女', 15), ('赵', '男', 12), ('李', '男', 10)]
```

2）列表的方法 sort( )

列表的 sort( ) 方法是把列表按照升序或者降序（也允许有排序的 key 值）重新排列，改变原有的列表。例如：

```
>>> a = [1,4,3,2]
>>> a.sort(reverse = True)
>>> a
[4, 3, 2, 1]
```

3. 列表解析表达式

列表解析式，也叫列表推导式，是将一个可迭代对象（如列表）转换成另一个列表的工具。在转换过程中，可以指定元素必须符合某一条件，并按照指定的表达式进行转换，快速生成新的列表。

其语法的基本格式为：

[表达式 for 元素 in 可迭代对象 if 条件]

举例如下：

```
# 生成一个由 1~9 的平方数构成的列表
>>> print([i**2 for i in range(1,10)])
[1, 4, 9, 16, 25, 36, 49, 64, 81]
# 在 1~9 范围内生成一个由偶数构成的列表
>>> print([i for i in range(1,10) if i%2 == 0])
[2, 4, 6, 8]
```

4. 程序资源

本单元阐述一维数据的表示、存储和操作，与后续单元的二维数据、多维数据、高维数据关系密切。因此，在本单元中配套了一维数据读出 .py、一维数据写入 .py 等程序，供授课教师选择演示，以帮助学生弄清楚如图 5-3 所示的数据处理周期。

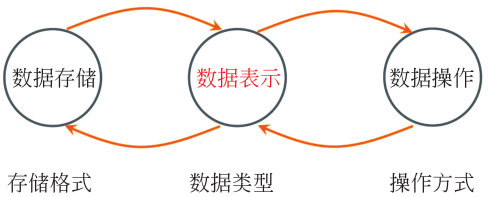

图 5-3　数据的处理周期

5.7　教学组织安排

教 学 环 节	教 学 过 程	建议课时
知识导入	在对数据维度的知识进行简单回顾以后，承上启下地引入关于数据处理的主要环节的介绍，使学生了解数据处理，主要考虑数据如何在 Python 程序中表示，以什么形式进行在计算内的存储，以及如何基于存储的数据进行读、写、统计、分析及计算	1 课时
一维数据的表示	分析一维数据的有序性和无序性，分别考虑用列表、集合进行数据的表示，当然如果数据是固定不变的，也可以使用元组表示。 通过课堂交互、代码演示等形式，对序列的统计、切片、遍历等常规运算进行复习巩固，对集合为代表的无序一维数据的运算也进行实践练习以复习巩固	
知识回顾	通过“练一练”环节的编程，巩固列表和集合的运算，熟悉一维数据的常见表示方法	

续表

教 学 环 节	教 学 过 程	建议课时
一维数据的存储	在实际编程处理一维数据之前，首先领会以下三点。 （1）列表和集合等只是适当的一维数据的表示方式，数据并没有被永久存储下来。 （2）要通过文件等永久存储的方式将数据存储下来，才有利于编程中多次读取和写入数据，以支持实际的应用。 （3）为方便在文件中读写数据，对于数据之间处于平等线性关系的一维数据，要用合适的符号进行数据之间的分隔，这样的符号一般使用英文的逗号、空格或者其他特殊的符号，要避免用于分隔符的符号在数据中出现	
一维数据的读取	从文件中读取一维数据，在读取中常运用以下编程知识。 （1）以固定编码格式打开文件。 （2）以特定的符号作为分隔符（依照文件内容的特点），将读取到的数据处理成为列表或者集合等	1 课时
一维数据的写入	将用列表或者集合等数据类型表示的一维数据写入指定的文件，写入中注意以下编程知识的运用。 （1）运用 `str.join(iterable)` 方法，用特定符号连接迭代的数据，形成字符串以便写入文件。 （2）`file.write( )` 方法仅支持字符串的写入，若一维数据存在其他类型数据，在写入时要注意转换成字符串。 （3）字符串的其他方法，可以帮助“清洗”和规格化处理要写入的字符串	
编程实践	通过“想一想”“练一练”环节，对文件的读写、数据的处理等编程知识进行复习巩固和实践练习	
本单元知识总结	回顾一维数据的表示、存储和读写处理等知识	

5.8　教学实施参考

1. 通过复习，渐进引入一维数据处理问题

通过提问、讨论等形式，对前面学习过的数据维度的知识进行复习。引导

学生思考怎样将抽象的数据信息通过编程在计算机中进行表示、存储和处理。即通过 Python 编程来实现一维数据处理。

2. 根据需求分析数据的特点，为之选取合适的表示方式

例如，表示某位同学“语文、数学、体育”三门课程的成绩，这样的数据便具有先后严格的顺序，否者分数就有可能出现“张冠李戴”的对位错误。这样的数据适合用列表表示。而表达诸如公园中鸟类的种类、文具店卖出商品的类别等信息，就无须考虑先后顺序，可以用集合表示。

通过上述举例思考，让同学们树立需求分析的意识，用适合的数据类型表达不同特征的一维数据，并进一步思考不同数据类型所支持的方法、运算的不同之处。

3. 数据存储问题的考虑

一维数据只是一个简单的线性数列，关于它的存储可以用文本文件来实现。通过探讨，让学生领会临时存储和永久存储的区别。

在回顾文件读写方法的基础上，侧重注意文件读写过程中可能遇到的问题。

（1）文件的编码。

（2）读、写文件的路径表示方法。

（3）文件读、写过程中分隔符的设定。

4. 知识点一：一维数据的表示

（1）列表适合有序一维数据的表示。

（2）集合适于无序一维数据的表示。

（3）一维数据可以用更多数据类型来表示，例如，用字典、元组同样可以表示一维数据。

5. 知识点二：一维数据的读取

（1）可以从文件中读取一维数据，并根据需要将其转换为列表、集合等合适的数据类型。

（2）读取一维数据时须考虑分隔符，以利于正确地将读出的字符串转换为合适的数据类型。

6. 知识点三：一维数据的写入

（1）写入过程中同样要考虑数据间的分隔符设置。

（2）写入数据前注意数据类型的转换。

（3）列表等其他类型在写入之前转换成字符，但可能带有边界符号等，需考虑通过运算将其清洗、规格化处理。

（4）通过学生用书中的“想一想”“练一练”环节，锻炼学生对一维数据中处理相关环节操作方法的综合应用能力。

7. 单元总结

小结本单元的内容，布置课后作业。

5.9 拓展练习

已知苏轼出生在 1037 年，列表 `tsu=[('惠州',1094),('黄州',1079),('儋州',1097)]` 代表他去三地任职的地点及年份。请编程，得到如图 5-4 所示的输出结果。

```
42 岁：黄州
57 岁：惠州
60 岁：儋州
```

图 5-4　程序输出效果

5.10 问题解答

【问题 5-1】 古诗中生字表的生成属于无序的一维数据的表示及求解问题。解题思路为：

（1）生成古诗的集合，在集合生成中，可以利用集合数据类型的特性排除重复的汉字。

（2）为学生学习过的汉字建立一个生字排除集合。

（3）通过两个集合的差集计算，生成生字集合并输出。

当然如果需要写入文件之中，还可以通过读写文件的方式，将生字集合写入文件之中。“练一练”的参考代码如下。

```
t_words = set('咏柳碧玉妆成一树高万条垂下绿丝绦不知\
细叶谁裁出二月春风似剪刀')
t_exc =   set('柳玉成一树万下绿不叶出二月春风刀')
t_nwords = t_words - t_exc
print("古诗《咏柳》中的生字有：")
for i in t_nwords:
    print("{:3}".format(i),end="")
```

运行结果为：

```
古诗《咏柳》中的生字有：
碧 高 垂 妆  咏 绦 条 细 裁 知 丝 谁 剪 似
```

【问题 5-2】 如果文本中的数据有多行，将每一行信息都读取并处理成一个列表或者集合，应该如何编程实现?

该问题可以通过循环，逐行读取文件，然后将每一行文本数据均按照特定的分隔符进行列表化或者集合转换处理即可。以处理成列表为例，文件信息如图 5-5 所示。

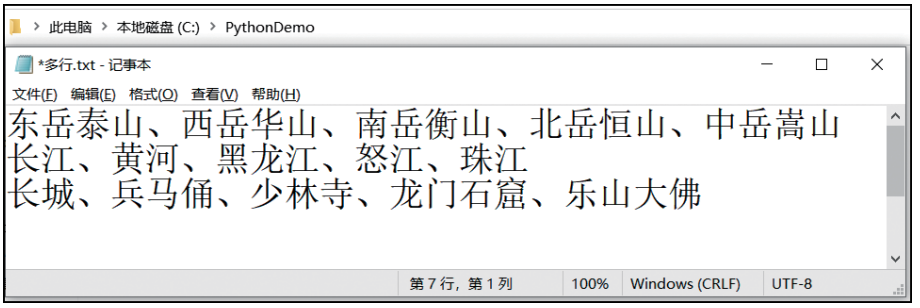

图 5-5　多行文本的文件信息

参考代码如下。

```
f = open('C:/PythonDemo/多行.txt','r',encoding='UTF-8')
i=1
line = f.readline( )
while line:
    tstr = str(line).strip("\n")   # 去除行字串中的 "\n" 换行符
```