输入：如何接收用户通过键盘输入的数据?

目录

输入设备与输入方式

- 输入，泛指计算机与人的交互方式
- 从计算机体系来看，输入的分层包括：输入硬件设备、硬件设备驱动、虚拟设备
- Python 编程主要关注虚拟设备（终端、图形界面输入）

* 本课程主要关注终端的交互和非交互两种输入

输入设备与输入方式

- 交互方式输入：

  input() 函数

- 非交互方式输入：

  命令行参数、文件

input 函数

```
text = input("用户输入前的提示信息")
```

input 函数

```
while True:
    print("按s键保存数据")
    print("按l键读取数据")
    print("按q键退出程序")
    keyboard = input("请输入控制指令")
    do something...
```

* 输入也经常用于循环中，实现暂停循环的功能

命令行参数

python3 –V

python3 ––help

python3 a.py b.py c.py

python3 ––noargs

参数支持不同的风格、数量和错误提示

命令行参数

实现参数处理的两个函数库：

- argparse——用户友好的命令行选项解析器

https://docs.python.org/zh-cn/3.10/library/argparse.html

- getopt——C 风格命令行选项解析器

https://docs.python.org/zh-cn/3.10/library/getopt.html

* 本课程中采用 argparse
* argparse 库需要 Python3.2 以上版本支持

命令行参数

一、编写命令行参数的一般逻辑

```
import argparse
parser = argparse.ArgumentParser(description="这个程序的用途")
parser.add_argument(要添加的参数和参数描述)
args = parser.parse_args()
```

* args 用于接收参数并进行处理
* 执行方式 python3 文件名.py 参数

命令行参数

二、增加一个可选参数

parser.add_argument("–number", help="输入一个数字")

执行方法:　python3 xx.py –number 100

三、增加一个必选参数

parser.add_argument("number", help="输入一个数字")

执行方法:　python3 xx.py 100

命令行参数

四、强制转换参数类型和设置默认值

```
parser.add_argument( "–number1", type=int,  default=10 )
```

总结

1. input 和命令行都能实现输入功能
2. 可以根据交互性和用户习惯选择输入方式
3. 用户不总会按照程序设计输入，应做全面判断

课后作业

编写一个程序，当用户输入 q 或 quit 时，程序退出，否则一直等待用户输入。

格式化输出：如何将执行结果通过屏幕输出?

目录

三种常用的格式化输出方式

1. 百分号 %

```
"%s is %s than %s"    %("Beautiful", "better", "ugly")
```

2. format() 函数

```
"{1} is {2} than {0}".format("ugly", "Beautiful", "better")
"{B} is {b} than {u}".format(u = "ugly", B = "Beautiful", b = "better")
```

3. f–strings

```
u = "ugly"; B = "Beautiful"; b = "better"
f"{B} is {b} than {u}"
```

百分号方式

百分号“%”用于定义格式和精度

%常用格式：

%s 格式化字符串

%d 格式化整数

%f 浮点数

```
>>> "%-5.3d" %(123.456)
'123  '     # 3后面有两个占位符
```

format() 函数方式

str.format() 的字符串部分使用 {} 替换内容

{} 既可以引用数字索引，也可以引用关键字参数

参考 https://docs.python.org/zh-cn/3.10/library/stdtypes.html#str.format

总结

1. 格式化输出是为了让提示信息和输出的结果更人性化
2. 可以根据输出的复杂度和特点，来选择不同的输出方法

课后作业

在程序中有字符串变量如下：

date="2022-05-20 13:14:00"

请使用格式化输出功能，将上述字符串进行格式转换，输出格式为：

智能助手为你报时，当前时间是 2022 年 5 月 20 日 13 点 14 分 0 秒

F–strings：如何通过定义好的格式进行输出?

目录

F-strings 介绍

- F-strings 是 Python3.6 新增的字符串格式化功能
- 比百分号和 str.format() 更友好
- 采用了早期为字符串添加关键字方式，如"u"、"r"、"b"

官方文档：
https://peps.python.org/pep-0498/

F-strings 的计算功能

- F-strings 的{}中可以实现数字计算、字符串连接、函数执行等计算任务
- 嵌入的内容可解释执行

```
f"{ 1 + 2 }"

f"{ 'a' + 'b' }"

f"{ print('c') }"
```

F-strings 宽度和精度调整

宽度和精度调整格式：

f"{对象:宽度.精度类型}"

* 类型同百分号格式化输出

F-string 宽度和精度调整

宽度

```
number = 123.456
f"{number:10}"
'    123.456' # 1 前面三个占位符

f"{number:010}"
'000123.456'

f"{number:4f}"
'123.456000' # 指定类型后，默认保存小数点后 6 位
```

F-string宽度和精度调整

精度

```
number = 123.4567

f"{number:.3f}"

'123.457'   # 保留小数点后 3 位
```

总结

1. F-strings 比百分号和 str.format() 函数更灵活

2. F-strings 的计算功能增强了格式化输出的友好性

课后作业

有一组由程序处理的浮点数，为了输出时保持工整，希望输出的字面值为 10 个数字，且小数点后最多包含 3 位，请你用 F-strings 对它们的格式进行调整并输出。

例：

数字为：123.4567

输出为：'000123.457'

常见常新：文件的打开

目录

文件的概念

类 Unix 系统的概念：一切皆文件

文件基本操作包括：打开、关闭、读取、写入

使用 open() 函数打开文件

- 打开文件是文件的第一步操作
- Python 使用 open() 函数实现了文件打开
- open() 函数的第一个参数就是要打开的文件名称

```
open(file, mode='r', buffering=-1, encoding=None, errors=None,
        newline=None, closefd=True, opener=None)
```

```
help(open)
Help on built-in function open in module io:
...
```

使用 open() 函数打开文件

```
file_handler = open("afile")
```

* name='a.file' mode='r' encoding='UTF-8'

* 如果没有报错，文件打开成功

* 为了避免误操作，默认为只读方式打开文件

文件路径处理

- 如果打开文件的路径不正确，会提示如下错误：

FileNotFoundError: [Errno 2] No such file or directory: 'xxx.file'

- 解决办法：

打开文件时增加路径

进入指定路径后，再打开文件

文件路径处理

```
file_handler = open("/tmp/afile")


import os

os.chdir("/tmp")

file_handler = open("afile")
```

文件打开模式

```
file_handler = open("/tmp/afile", mode="r")
```

常见的 mode 参数值

字符	含义
r	读取（默认）
w	写入，并覆盖文件
a	写入，如果文件存在则在末尾追加
b	二进制模式
+	支持读取与写入

总结

1. 文件操作包括：打开、关闭、读取、写入
2. 类 Unix 系统中设备、虚拟设备、网络套接字等都被当作文件
3. 你可以学习类 Unix 系统的方式，将对象定义为文件，保持编程接口的一致性

课后作业

如果你使用的是 Linux 或 Mac 系统，请使用只读模式打开 /etc/passwd 文件，

如果你使用的是 Windows 系统，请打开 C 盘下的任意一个以 .txt 结尾的文本文件。

要求：文件能够正常打开，在打开时不会报错。

文件编码：如何解决不同操作系统的文件乱码问题?

目录

为什么会产生乱码

- 计算机只能处理数字，如果处理文本，就需要把文本按特定规则转换为计算机能够处理的数字，这个过程就被叫做**编码**
- 比如你听过的 ASCII 编码，就是把英文字母、数字和常用符号编码成计算机能够认识的特定数字

为什么会产生乱码

- 计算机普及之后，各个国家有不同的标准，比如中国制定的 GBK 编码，一直被 Windows 沿用
- 为了避免编码混乱，后续产生了统一的 Unicode 编码
- 其中最广泛使用的 UTF-8 就是 Unicode 的一种实现方式

常见操作系统的文件编码

- Windows 默认使用 GBK 编码
- Linux 和 macOS 使用 UTF-8 编码

常见操作系统的文件编码

乱码举例：

```
string = "人生苦短，我用 Python"  # UTF8

string = "浜虹敓鑻﹀剰锛屾垜鐢 Python" # GBK
```

常见操作系统的文件编码

乱码举例：

```
string = "人生苦短，我用 Python"  # GBK
```

```
string = "锟斤拷锟斤拷锟斤拷锟斤拷锟斤拷 Python"
```

以不同的编码打开文件

```
open(filename, encoding=UTF-8)

open(filename, encoding=GBK)
```

总结

1. Windows 默认编码为 GBK，Mac 和 Linux 默认编码为 UTF-8
2. 以不正确的编码打开文件会产生乱码
3. 通过 encoding 参数能够以制定的编码打开文件

作业

使用 open() 函数打开任意一个文件，并观察执行结果。

打开文件时，请注意根据自己的操作系统使用正确的字符集，

如果你使用的是 Windows 系统则使用 GBK 字符编码，

如果你使用的是 macOS 系统则使用 UTF-8 字符编码。

常见常新：文件的读写

目录

文件读取

读取文件内容，可以采用多个函数或语句实现：

- read([size]) 返回指定 size 大小的内容，如果不指定 size，将返回整个文件
- readline() 返回一行数据
- readlines([size]) 返回指定 size 行数，如果不指定 size，将返回全部文件
- for line in file 迭代文件的每一行

文件写入

文件写入由 write() 函数实现，写入方式由 open() 函数控制

文件写入

不同的 open() 函数打开模式，执行写入时，写入的位置和结果也不同：

- mode = "r"　报错：　io.UnsupportedOperation: not writable
- mode = "w" 或 mode = "w+" 清空文件，写入内容，返回写入字符数量
- mode = "a"　或 mode = "a+" 在文件结尾追加

文件写入

配合不同函数可以实现指定位置的文件写入：

- tell() 函数——返回文件指针位置
- seek(偏移量, [起始位置])——移动指针
- close()——关闭文件

总结

1. 使用不同函数可以实现不同的读取方式
2. 使用 open() 函数控制写入方式，使用 seek() 函数控制写入位置
3. 文件写入完成应使用 close() 函数关闭文件

课后作业

1. 请将以下字符串内容保存到文本文件 a.txt 中。

2. 请使用文件读写功能将 a.txt 文件复制为 b.txt 并保存。

```
string = """Python 3.11 is up to 10–60% faster than Python 3.10.
On average, we measured a 1.25x speedup on the standard
benchmark suite. See Faster CPython for details. """
```

常见常新：文件的关闭

目录

文件关闭的内部工作过程

为什么要及时关闭文件?

- 文件写入函数执行完成后，写入部分仍然在内存中
- 由于内存是易失存储，突然断电会导致数据丢失
- close() 函数可以告知内核文件在内存中有更新

close() 函数

- write() 函数写入后，必须显式 close()
- read 族函数读取后，不必 close() 但建议关闭，避免文件描述符耗尽
- open() 失败，不必 close()

with 语句

with 语句可以使打开和关闭文件保持一致性，即：

使用 with 语句打开的文件，离开 with 语句块作用域会自动关闭

with 语句

- with open(文件) as 文件描述符:

  文件读写操作

- with 语句块结束，自动关闭文件

总结

1. 文件关闭有助于保存文件内容
2. 文件关闭也利于多文件操作时，及时释放文件描述符
3. 使用 with 语句可以使文件打开和关闭保持一致性
4. 文件打开失败可以不用关闭文件

课后作业

请打开一个不存在的文件，观察报错，并使用 close() 函数关闭文件，继续观察报错信息。

小试牛刀：如何使用 Python 合并多个文件?

目录

合并文件需求分析

合并文件 = 读取原始文件 + 追加写入新的文件

合并两个文件

关键任务：

1. 以可读取的模式打开原始文件
2. 以写入的模式创建并打开要合并的文件
3. 正确关闭文件

遍历多个文件

关键任务：

1. 将多个文件的文件名写入列表

2. 遍历列表

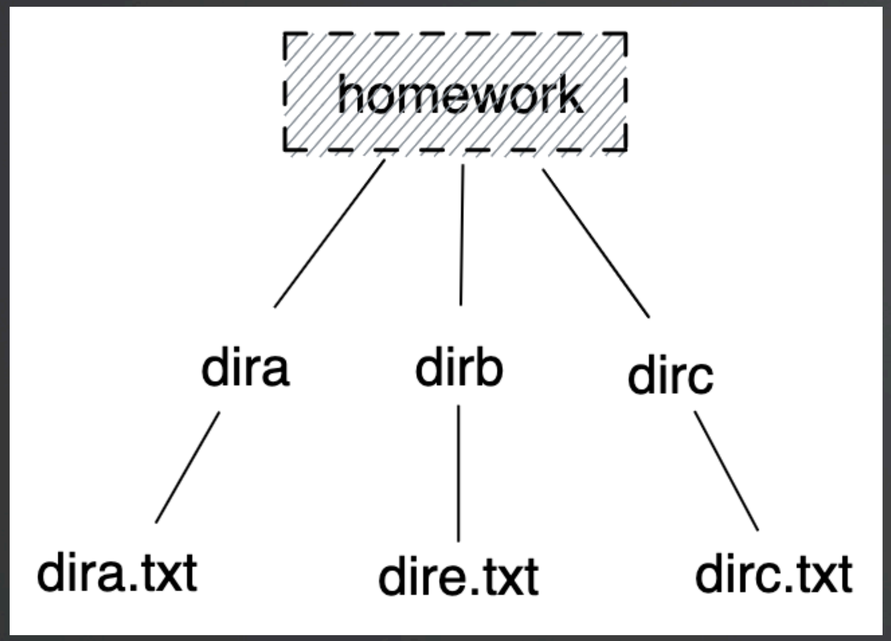
3. 重构文件读、写代码

总结

1. 文件的读、写是实现文件合并的基础
2. 通过遍历可以实现多个文件自动合并

课后作业

在 homework 文件夹下，有三个文件夹分别为 dira、dirb、dirc，三个文件夹下各自存放了和文件夹同名的文本文件，组成了如下格式：



现需要将三个 txt 文件，合并为 homework.txt 并放在 homework 下，请你用 Python 实现该需求。

THANKS