1. 文件和数据格式化

1、文件的使用

文件的类型：

文件是数据的抽象和集合

文件是存储在辅助存储器上的数据序列。

文件是数据存储的一种形式

文件展现形态：文本文件和二进制文件。

文本文件和二进制文件只是文件的展示方式；本质上，所有文件都是二进制形式存储

。形式上，所有文件采用两种方式展示。

文本文件：由单一特定编码组成的文件，如UTF-8编码。由于存在编码，也被看成是存储着长字符串

适用于例如: .txt文件 、 .py文件

二进制文件：直接由比特0和1组成，。没有统一字符编码。一般存在二进制0和1的组织结构，即文件格式。

适用于例如： .png文件 、 .avi文件等。

文件的打开和关闭

文件处理的步骤：打开——操作——关闭

文件的存储状态——文件的占用状态用a=open( , )

文件的占用状态——文件的存储状态用a.close()

读文件：

a.read(size)

a.readline(size)

a.readlines(hint)

写文件：

a.write(s)

a.writelines(lines)

a.seek(offset)

文件的打开：

<变量名>=open((文件名),(打开模式))

文件名：文件路径和名称（源文件同目录可省略）

打开模式：文本或二进制（读或写）

变量名：指文件句柄。

|  |  |
| --- | --- |
| 文件的打开模式 | 描述· |
| ‘r’ | 只读模式，默认值，如果文件不存在，返回FiledNotFoundError |
| ‘w’ | 覆盖写模式，文件不存在则创建，存在则完全覆盖 |
| ‘x’ | 创建写模式，文件不存在则创建，存在则返回FileExistsError |
| ‘a’ | 追加写模式，文件不存在则创建，存在则在文件最后追加内容 |
| ‘b’ | 二进制文件模式 |
| ‘t’ | 文本文件模式，默认值 |
| ‘+’ | 与r/w/x/a同一使用，在原功能基础上增加同时读写功能。 |

打开模式：

f=open(“f.txt”) 文本形式、只读模式、默认值

f=open(“f.txt”,”rt”) 文本形式、只读模式、同默认值

f=open(“f.open”,”w”) 文本形式、覆盖写模式

f=open(“f.txt”,”a+”) 文本形式、追加写模式+读文件

f=open(“f.txt”,”x”) 文本形式、创建写模式

f=open(“f.txt”,”b”) 二进制形式、只读模式

f=open(“f.txt”,”wb”) 二进制形式、覆盖写模式

文件的关闭：

<变量名>.close

文件内容的读取

|  |  |
| --- | --- |
| 操作方法 | 描述 |
| <f>.read(size=-1) | 读入全部内容，如果给出参数，读入前size长度 |
| <f>.readline(size=-1) | 读入一行内容，如果给出参数，读入该行的前size长度 |
| <f>.readlines(hint=-1) | 读入文件所有行，以每行为元素形成列表，如果给出参数，读入前hint行 |

遍历全文本：方法一

#bianliquanwenben1.py

fname=input("请输入要打开的文件名称：")

fo=open(fname,"r")

txt=fo.read()

#对全文txt进行处理,一次读入，统一处理

fo.close()

方法二：

#bianliquanwenben2.py

fname=input("请输入要打开的文件名称：")

fo=open(fname,"r")

txt=fo.read(2)

while txt !="":

#对txt进行处理,按数量读入，逐步处理

txt=fo.read(2)

fo.close()

逐行遍历文件：方法一

#一次读入，分行处理

fname=input("请输入要打开的文件名称：")

fo=open(fname,"r")

for line in fo.readlines():

print(line)

fo.close()

方法二：

#分行读入，逐行处理

fname=input("请输入要打开的文件名称：")

fo=open(fname,"r")

for line in fo:

print(line)

fo.close()

数据的文件写入

|  |  |
| --- | --- |
| 操作方法 | 描述 |
| <f>.write(s) | 向文件写入一个字符串或字节流 |
| <f>.writelines(lines) | 将一个元素全为字符串的列表写入文件 |
| <f>.seek(offset) | 改变当前文件操作指针的位置，offset含义如下：0-文件开头； 1-当前位置； 2-文件结尾 |

2、实例：自动轨迹绘制

需求：根据脚本来绘制图形

不是写代码而是写数据绘制轨迹

数据脚本是自动化最重要的第一步

基本思路：步骤一：定义数据文件格式（接口）

步骤二:编写程序，根据文件接口解析参数绘制图形

步骤三：编制数据文件



#autoTraceDraw.py

import turtle as t

t.title('自动轨迹绘制')

t.setup(800,600,0,0)

t.pencolor("red")

t.pensize(5)

#数据读取

datals=[]

f=open("data.txt")

for line in f:

line=line.replace("\n","")

datals.append(list(map(eval,line.split(","))))

f.close()

#自动绘制

for i in range(len(datals)):

t.pencolor(datals[i][3],datals[i][4],datals[i][5])

t.fd(datals[i][0])

if datals[i][1]:

t.right(datals[1][2])

else:

t.left(datals[i][2])

3、一维数据的格式化和处理

数据组织的维度

一维数据：由对等关系的有序或无序数据构成，采用线性方式组织

一维数据对应列表、数组和集合等概念。

二维数据：由多个一维数据构成，是一维数据的组合形式。

多维数据：由一维或二维数据在新维度上扩展形成。

高维数据：仅利用最基本的二元关系展示数据间的复杂结构。

数据的操作周期：存储<->表示<->操作

存储指数据存储 ——关注存储格式

表示指数据表示——关注数据类型

操作指数据操作——关注操作方式

一维数据的表示

1、如果数据间有序：使用列表类型。列表类型可以表达一维有序数据。for循环可以遍历数据，进而对每个数据进行处理。

2、如果数据间无序：使用集合类型。集合类型可以表示一维无序数据。for循环可以遍历数据，进而对每个数据进行处理。

一维数据的存储

1、存储方式一：空格分隔。 使用一个或多个空格分隔进行存储，不换行。

缺点：数据中不能存在空格

1. 存储方式二：逗号分隔。 使用英文半角逗号分隔数据进行存储，不换行。

缺点：数据中不能有英文逗号

3、存储方式三：其它方式。 使用其他符号或符号组合分隔，建议采用特殊符号

缺点：需要根据数据特点定义，通用性较差

一维数据的处理

从空格分隔的文件中读入数据。txt.split()

从特殊符号分隔的文件中读入数据。 txt.split(“$”)

采用空格分隔方式将数据写入文件。 .write(‘ ‘.join(ls))

采用特殊分隔方式将数据写入文件。 .write(‘s’.join(ls))

4、二维数据的格式化和处理

二维数据的表示：

使用列表类型。列表类型可以表达二维数据。使用二维列表。

使用两层for循环遍历每个元素。外层列表中每个元素可以对应一行，也可以对应一列。

数据维度是数据的组织形式。

CSV格式与二维数据的存储

CSV：全称是Comma-separated Values

国际通用的一二维数据存储格式，一般.csv扩展名

每行一个一维数据，采用逗号分隔，无空行。

Excel和一般编辑软件都可以读入或另存为csv文件。

如果某个元素缺失，逗号仍要保留。

二维数据的表头可以作为数据存储，也可以另行存储。

逗号为英文半角逗号，逗号与数据之间无额外空格。

二维数据的存储：按行存或者按列存都可以，具体由程序决定。一般索引习惯：ls[row][column],先行后列。

根据一般习惯，外层列表每个元素是一行，按行存。

二维数据的处理

二维数据的读入处理

#从CSV格式的文件中读入数据

fo=open(fname)

ls =[]

for line in fo:

line=line.replace("\n","")

ls.append(line.split(","))

fo.close()

#将数据写入CSV格式的文件

ls=[[],[],[]]#二维列表

f=open(fname,'w')

for item in ls:

f.write(','.join(item)+'\n')

f.close()

二维数据的逐一处理

#采用二层循环

ls=[[1,2],[3,4],[5,6]]#二维列表

for row in ls:

for column in row:

print(column)

二维数据的格式化和处理

二维数据的表示：列表类型，其中每个元素也是一个列表

CSV格式：逗号分隔表示一维，按行分隔表示二维。

二维数据的处理：for循环+.split()和.join()

模块6：wordcloud库的使用

wordcloud库基本介绍：

wordcloud最优秀的词云展示第三方库

词云以词语为基本单位，更加直观和艺术的展示文本。

(cnd命令行) pip install wordcloud、

wordcloud库使用说明

wordcloud库把词云当作一个WordCloud对象

wordcloud.WordCloud()代表一个文本对应的词云： 可以根据文本中词语出现的频率等参数绘制词云； 绘制词云的形状、尺寸和颜色都可以设定。

w=wordcloud.WordCloud() 以WordCloud对象为基础； 配置参数、加载文本、输出文件；

|  |  |
| --- | --- |
| 方法 | 描述 |
| w.generate(txt) | 向WordCloud对象w中加载文本txt |
| w.to\_file(filename) | 将词云输出为图像文件，.png或.jpg格式 |

Wordcloud库常规方法：

步骤一：配置对象参数

步骤二：加载词云文本

步骤三：输出词云文件

#wordcloud.py

import wordcloud

c=wordcloud.WordCloud()

c.generate("wordcloud by Python")

c.to\_file("pywordcloud.png")

处理过程：

1、分隔：以空格分隔单词

2、统计：单词出现次数并过滤

3、字体：根据统计配置字号

4、布局：颜色环境尺寸

配置对象参数

w=wordcloud.WordCloud(<参数>)

|  |  |
| --- | --- |
| 参数 | 描述 |
| width | 指定词云对象生成图片的宽度，默认400像素 |
| height | 指定词云对象生成图片的高度，默认200像素 |

和字体有关的参数

|  |  |
| --- | --- |
| 参数 | 描述 |
| min\_font\_size | 指定词云中字体的最小字号，默认4号 |
| max\_font\_size | 指定词云中字体的最大字号，根据高度自动调节 |
| fonrt\_step | 指定词云中字体字号的步进间隔，默认为1 |
| font\_path | 指定字体文件的路径，默认None |
| max\_words | 指定词云显示的最大单词数量，默认200 |
| stop\_words | 指定词云的排除词列表，即不显示的单词列表 |
| mask | 指定词云形状，默认为长方形，需要引用imread()函数 |
| background\_color | 指定词云图片的背景颜色，默认为黑色 |

应用实例：

#wordcloud1.py

import wordcloud

txt="life is short,you need python"

w=wordcloud.WordCloud( \

background\_color="white")

w.generate(txt)

w.to\_file("pywcloud.png")

对汉字

import jieba

import wordcloud

txt="一溜串汉字"

w.wordCloud.WordCloud(width=1000,\

font\_path="msyh.ttc",height=700)

w.generate(" ".join(jieba.lcut(txt)))

w.to\_file("pywcloud.png")

#中文需要先分词并组成空格分隔字符串

实例：政府工作报告词云

需求;对于政府工作报告等政策文件，如何直观理解？

基本思路：

步骤1：读取文件、分词整理

步骤2设置并输出词云

步骤3：观察结果，优化迭代