目录

必做题

1. 摩斯码生成器（模块：textEx，所在文件名 text_hw.py，Level：★）

利用 Python 实现摩斯码符号生成，完成函数：

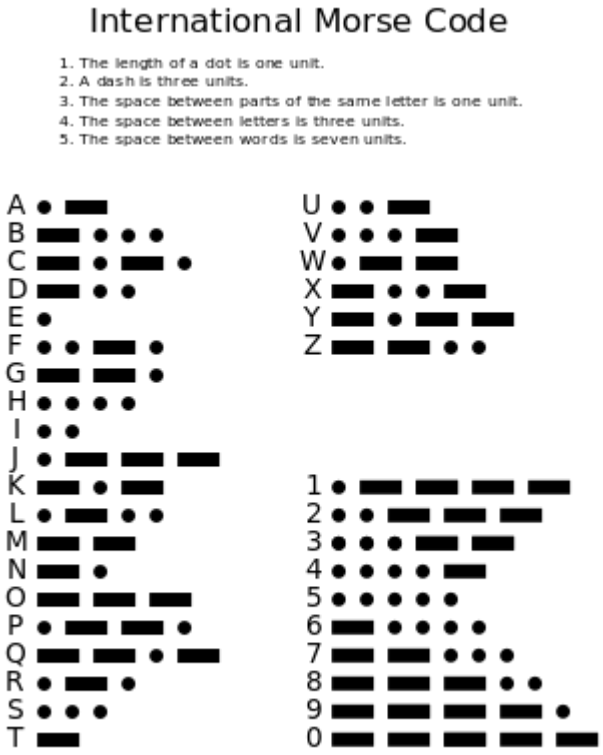
（1）摩斯码生成函数：

函数原型：def morse_code(usr_str)

参数 usr_str：字符串，需要转换为摩斯码的字符串。

返回值：输出 usr_str 对应的摩斯码字符串，用 . 代表点， - 代表破折号，点与点、破折号与破折号之间、点与破折号之间为一个空格，字符间为三个空格，单词之间为七个空格。注意输出的摩斯码首尾不含空格。

参考网站：https://en.wikipedia.org/wiki/Morse_code



2. 词频统计（模块：textEx，所在文件名 text_hw.py，Level：★★）

利用 Python 从文本文件中提取出现频次前十的单词，完成函数：

（1）词频提取函数：

函数原型：def word_freq(path)

参数 path：字符串，需要提取的文本文件路径。

返回值：列表，列表元素为二元组（单词，次数）；按从多到少的顺序列举出现最多的前十个单词与次数。如果单词出现的次数相同，则按单词的降序排序。统计时去除高频词（见sight word.txt）。

可逐行读取文本内容，并按空格进行切分，逐个统计该行单词的数目信息，存储于字典中，最终对字典中的数据进行排序，可转化为列表之后排序，输出前 10 个出现频率最高的单词及其出现的次数。单词不区分大小写，处理时需去除一些非必要的符号（!`~@#$%^&*()_-+=[]{}/?,.:\"<\/>），只保留单词，连写词如 it's，don't 等算一个词汇。

3. C 程序文件处理（模块：textEx，所在文件名 filter_hw.py，Level：★★★）

利用 Python 实现将 C 源代码文件（后缀 .c，.cpp）读入，去除代码中的空格、块注释、行注释、include 语句、空行、回车换行符号，形成一个长字符串，并写入到新的文件。

实现函数：

（1）C/C++文件过滤函数：

函数原型：def filter_c_file(path)

从 path 中找到后缀为 .c，.cpp 的文件，逐个按要求删除不必要的字符，形成一个新字符串，该字符串被写入到同级目录下的新文件“XXX.txt”，其中“XXX”为原 C/C++文件文件名称。

参数 path：需要过滤的 C/C++文件路径，包含有多个文件。

返回值：无。

4. 敏感词过滤（模块：textEx，所在文件名 text_hw.py，Level：★★）

提供一个敏感词库文件 sensitive.txt，敏感词按类型进行分组，如“第一类 形容词”，每组中一行代表一个敏感词，如“可怕”。要求读取敏感词库，获得敏感词列表，根据敏感词列表对输入的敏感词进行过滤。敏感词是几个字，就将对应的文本替换成几个*符号。比如敏感词为脉动，则将“脉动真好喝”替换为“**真好喝”。

注意敏感词类型文本不归入敏感词，比如“第一类 形容词”不在过滤范围。敏感词库文件中的行可能只有空格，读入时请去除，不包含在过滤词库中。

函数原型：def filter_words(user_input)

参数 user_input：字符串，为输入待处理的字符串。

返回值：字符串，过滤后的字符串。

5. Base64 编解码算法（模块：textEx，所在文件名 base64_hw.py，Level：★★★）

理解 Base64 编码的原理，设计两个函数实现二进制数据的编解码。

（1） Base64 编码：

函数原型：def b64en(path_in, path_out)

参数 path_in：需要进行 base64 编码的图片文件路径。

参数 path_out：以 UTF8 编码生成的文本文件路径。

返回值：无。可使用 base64 内置库。

（2） Base64 解码：

函数原型：def b64de(path_in, path_out)

参数 path_in：需要进行 base64 解码的 UTF8 文本文件路径。

参数 path_out：解码生成的图片文件路径。

返回值：无。**不允许**使用 base64 内置库。

6. XML 文件的生成与解析（模块：dataEx，所在文件名 xml_hw.py，Level：★★）

利用 Python 实现 XML 文件的读写，完成两个内容：

（1）创建 XML 文件，可使用 xml.dom.minidom，以生成 XML 文件。

函数原型：def create_xml(path)

参数 path：xml 文件的保存路径（包含文件名），要求支持相对路径。

返回值：无。

要求生成的 XML 文件结构与参考内容如下表所示。

```
<?xml version="1.0" ?>
<tilemap tilemapservice="http://tms.osgeo.org/1.0.0" version="1.0.0">
  <title>default</title>
  <abstract/>
  <srs>EPSG:4326</srs>
```

```
  <vsrs/>
  <boundingbox maxx="180.0" maxy="90.0" minx="-180.0" miny="-90.0" />
  <origin x="-180.0" y="-90.0" />
  <tileformat extension="tif" height="17" mime-type="image/tiff" width="17" />
  <tilesets profile="global-geodetic">
    <tileset href="" order="0" units-per-pixel="10.588" />
    <tileset href="" order="1" units-per-pixel="5.294" />
    <tileset href="" order="2" units-per-pixel="2.647" />
    <tileset href="" order="3" units-per-pixel="1.323" />
    <tileset href="" order="4" units-per-pixel="0.661" />
    <tileset href="" order="5" units-per-pixel="0.331" />
  </tilesets>
</tilemap>
```

（2）对指定的 XML 文件进行读取，可使用 xml.etree.ElementTree 解析 XML 文件。

函数原型：def parse_xml(path)

参数 path：要解析的 xml 文件路径，要求支持相对路径。

返回值：返回值类型为字典，如果解析成功，返回 dict 格式为：

{“tilemap service” : tilemap 节点 tilemapservice 属性的值，”title” : title 节点的值，”tileset count” : tileset 节点的个数，”tileset max” : tileset 节点中最大的 order 值（注意是整数）}

对应到上表的 XML 文件，返回值为：

{“tilemap service” : "http://tms.osgeo.org/1.0.0"，”title” : “default”，”tileset count” : 6，”tileset max” : 5}

解析过程中，如果缺少对应的值，则该项不在字典中出现；如果所有的值均不存在，就返回空的字典。

注意提供测试的 XML 中 tileset 节点的个数和属性值不是固定的。

7. 二进制数据报文构建与解析（模块：dataEx，所在文件名 data_hw.py，Level：★）

利用 Python 标准库中的 struct 模块实现二进制数据报文的构造与解析。完成两个内容：

（1）构建报文：

函数原型：def pack_message(data_dict)

参数 data_dict：报文字段值，为字典类型，例如：

{'type': 50, 'csum': 1, 'id': 'abcdefghigklmnop', 'dis1': 300, 'dis2': 100, 'count': 20}

返回值：二进制报文的字节序列。如果参数异常或者缺项，返回错误“Parameter Error.”

报文格式如下：共 27 字节

消息类型（type，1 字节，0-100 的整数） || 数据校验字节（csum，1 字节，后续的数据部分字节加法和）|| 禁飞区 ID（16 个字符，可按 UTF8 编码）| 禁飞区预警距离（dis1，整数，4 字节，大端序）| 禁飞区告警距离（dis2，整数，4 字节，大端序） | 禁飞区 1 点数（count，1 字节，0-255 整数）

（2）解析报文：

函数原型：def pack_message(message)

参数 message：经 pack_message 生成的二进制序列。

返回值：字典类型，包含有解析出来的数据。如果参数异常，返回错误“Parameter Error.”

8. 实现数据库的操作（模块：dataEx，所在文件名 db_hw.py，Level：★★）

利用 Python 实现针对 Sqlite3 数据库的操作，实现以下函数：

（1）初始化数据库：创建数据库文件、数据表

函数原型：def create_db(path)

参数 path：字符串，指明了数据库文件生成的位置。后续函数中的增删改查都是针对该数据库文件操作。在 dataEx 模块中可增加全局变量记录该路径。

在指定路径新建 Sqlite3 数据库，如果已经存在，则应首先删除原文件再创建。然后，建立两张数据表，即 Person 表与 Position 表。

返回值：创建成功，返回 0；失败返回-1。

人员信息表 Person：

序号	字段名称	字段类型	取值范围

1	NAME	字符串	姓名，32 字符
2	GENDER	字符串	性别，2 字符
3	BIRTH	日期	生日，2000 年 10 月 20 日
4	ID	字符串	身份证号，18 位身份证号，全局唯一，作为主键
5	POSITIONID	字符串	岗位名称，与岗位表关联

岗位表 Position：

序号	字段名称	字段类型	取值范围
1	POSITIONID	字符串	岗位名称，A、B、C、D；全局唯一，作为主键
2	SALARY	数字	薪水，10000，6000，3000，1000；每月的薪水

（2）新进人员：

函数原型：def new_employee(person，level)

参数 person：四元组，(姓名，性别，生日，身份证号)。

参数 level：字符串，岗位。

返回值：人员插入成功，返回 0；失败返回-1。

（3）删除人员：

函数原型：def delete_employee(person)

参数 person：字符串，被删除人员的身份证号。

返回值：删除成功，返回 0；失败返回-1。

（4）设置岗位薪水：

函数原型：def set_level_salary(level，salary)

参数 level：字符串，岗位级别，即 A、B、C、D 四个等级之一。

参数 level：整数，薪水。

返回值：设置成功，返回 0；失败返回-1。

（5）统计薪水开支：

函数原型：def get_total_salary()

返回值：整数，返回当前所有人员每月开支的薪水总和；失败返回-1。