

程序设计(Python)实训题集

张慧翔编

Version 0.2

2018 年 10 月 10 日

目 录

介绍.....	3
1. 二进制与十进制转换器（模块：numEx，所在文件名 number_hw.py）	4
2. 类似找零钱的操作（模块：numEx，所在文件名 number_hw.py）	4
3. 两地之间距离计算（模块：numEx，所在文件名 number_hw.py）	5
4. 计算 Fibonacci 序列的值（模块：numEx，所在文件名 number_hw.py）	6
5. 摩斯码生成器（模块：textEx，所在文件名 text_hw.py）	6
6. 词频统计（模块：textEx，所在文件名 text_hw.py）	7
7. C 程序文件处理（模块：textEx，所在文件名 text_hw.py）	7
8. 计算图形面积及周长（模块：classEx，所在文件名 class_hw.py）	9
9. XML 文件的生成与解析（模块：dataEx，所在文件名 xml_hw.py）	9
10. 二进制数据报文构建与解析（模块：dataEx，所在文件名 data_hw.py）	11
11. 实现数据库的操作（模块：dataEx，所在文件名 db_hw.py）	11
12. 获取当前天气情况（模块：netEx，所在文件名 net_hw.py）	13

介绍

以个人为单位上交训练题文件，个人文件夹以“班号_学号”形式命名，利用码云平台，Fork “Python 设计实训”项目，将个人文件存放到“ClassCode”目录下，然后通过 PR 方式反馈到老师的“Python 设计实训”项目中，等待老师审核，注意查看审核意见和结果，确保个人代码提交正确。

实验报告电子版采用 pdf 格式，以“report.pdf”形式命名；训练题代码按模块组织，源代码文件命名采用小写字母。

参考目录结构如下：

09061601_2014201325

```
| report.pdf
| ExCode
|   numEx
|       number_hw.py
|   textEx
|       text_hw.py
|   dataEx
|       xml_hw.py
|       data_hw.py
|       db_hw.py
|   classEx
|       class_hw.py
|   netEx
|       net_hw.py
```

.....

1. 二进制与十进制转换器（模块：numEx，所在文件名 num_hw.py）

利用 Python 实现二进制数与十进制数之间的相互转换。转换基于补码规则，32 位系统，完成两个函数：

（1）十进制转二进制

函数原型：def d2b(decimal_int)

参数 decimal_int：十进制整数，处于[INT32_MIN, INT32_MAX]之间。

返回值：如果输入的整数合规，则返回其对应的二进制序列字符串，该字符串的前导 0 全部被省略，如“11111111”；负数采用补码表示。如果输入的整数不合规，返回错误“Parameter Error.”

（2）二进制转十进制

函数原型：def b2d(binary_string, flag = False)

参数 binary_string：最多包含 32 个字符的字符串，代表了二进制序列，如“110011”。

参数 flag：指示是否按有符号数解释。如果为 True，如果 binary_string 最高位为 1，且长度为 32，则按有符号数规则解释。默认为 False。

返回值：如果输入的二进制字符串合规，则返回其对应的十进制整数；如果输入的二进制字符串不合规，返回错误“Parameter Error.”

2. 类似找零钱的操作（模块：numEx，所在文件名 num_hw.py）

实现找给用户找零的操作，最大面值为 100 元。找寻的零钱有以下几种：50 元，20 元，10 元，5 元，1 元，5 角，1 角。比如，物品：12.3 元，支付 100 元，程序应找寻：1 个 50 元，1 个 20 元，1 个 10 元，一个 5 元，2 个 1 元，1 个 0.5 元，2 个 0.1 元。可首先计算出差额，然后用整除的方式计算。

利用 Python 实现收银找零操作，完成以下函数：

（1）商品枚举

函数原型：def list_goods()

返回值：按商品名称顺序显示所有商品的名称和价格，按如下格式所示：

"item01": 2.3,

"item02": 35.8,

"item03": 16.3,

"item04": 12,
"item05": 13.6,
"item06": 29,
"item07": 17.4,
"item08": 63.9,
"item09": 56.7,
"item10": 23.8,

(2) 付款找零

函数原型: `def get_changes(items, pay)`

参数 `items`: list 类型, 表示需要购买的商品;

参数 `pay`: 数字类型, 表示付款纸币面额, 只付一张纸币。

返回值: 如果参数均合规, 返回字典, 输出各个面值的数目及对应个数; 如果购买商品列表清单中存在不合规的商品时, 输出“XX 商品不存在, 请重新选择。”; 如果支付金额不足时, 输出: “支付金额不足, 请重新支付。”

3. 两地之间距离计算 (模块: `numEx`, 所在文件名 `num_hw.py`)

利用 Python 实现地球上两点之间的距离计算, 地球上点的位置以经纬度坐标形式提供。
距离计算采用 Haversine 公式:

$$d = 2r \arcsin\left(\sqrt{\text{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1) \cos(\varphi_2) \text{hav}(\lambda_2 - \lambda_1)}\right) \\ = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

这里 r 是地球半径 6371Km, (φ, λ) 代表点的 (纬度, 经度) 坐标。

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

完成距离计算函数:

函数原型: `def sphere_distance(p1, p2)`

参数 `p1`: tuple 元组类型, 二元组, (纬度, 经度), 坐标精确到小数点后 7 位

参数 `p2`: tuple 元组类型, 二元组, (纬度, 经度), 坐标精确到小数点后 7 位

纬度取值范围: [0-90], 经度取值范围: [0-180], 单位均为角度; 而 Haversine 公式计算采用的是弧度, 注意转换。

返回值：如果输入的坐标数据合规，则返回两点之间的距离，单位为 Km，保留两位小数；如果输入的坐标不合规，返回错误“Parameter Error.”

4. 计算 Fibonacci 序列的值（模块：numEx，所在文件名 num_hw.py）

利用 Python 实现 Fibonacci 序列值的计算。实现两个函数：

（1）递归版本的 Fibonacci 序列值计算

函数原型：def fibonacci_recursion(number)

参数 number：Fibonacci 序列的第 number 项，number 为大于 0 的整数。

返回值：如果参数合规，则返回 Fibonacci 序列的第 number 项的值；如果参数不合规，返回错误“Parameter Error.”。

（2）循环版本的 Fibonacci 序列值计算

函数原型：def fibonacci_loop(number)

参数 number：Fibonacci 序列的第 number 项，number 为大于 0 的整数。

返回值：如果参数合规，则返回 Fibonacci 序列的第 number 项的值；如果参数不合规，返回错误“Parameter Error.”。

测试：

（1）查看 fibonacci_loop(36)与 fibonacci_recursion(36)的运行时间，哪个运行快？

（2）fibonacci_recursion 版本支持的最大输入是多少？最大值如何更改？

5. 摩斯码生成器（模块：textEx，所在文件名 text_hw.py）

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

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	• —	U	• • —
B	• — • •	V	• • • —
C	• — • —	W	• — —
D	• — • •	X	• • • —
E	•	Y	• • — —
F	• • — •	Z	• — — • •
G	• — — •		
H	• • • •		
I	• •		
J	• — — —		
K	• — • —	1	• — — — —
L	• — • •	2	• • — — —
M	• — —	3	• • • — —
N	• — •	4	• • • • —
O	• — — —	5	• • • • •
P	• — — •	6	• — • • •
Q	• — • —	7	• — — • •
R	• — • •	8	• — — — •
S	• • •	9	• — — — •
T	• —	0	• — — — —

6. 词频统计（模块：textEx，所在文件名 text_hw.py）

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，需要提取的文本字符串。

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。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。

7. （未完成）C 程序文件处理（模块：textEx，所在文件名 text_hw.py）

利用 Python 实现将 C 源代码文件读入，去除代码中的空格、空行、块注释、行注释，

形成一个长字符串，并写入到新的文件。实现函数：

(1) C 程序文件过滤函数：

函数原型：def c_code_filter(path, flag)

参数 path：需要过滤的 C 文件路径。

参数 flag：一个字节的整数，字节中的每个 bit 代表需要过滤的符号。该整数对应的二进制序列为：“****XXXX”，第 1 位（最低位）的“X”为 1 表示过滤空格；第 2 位的“X”为 1 表示过滤空行；第 3 位的“X”为 1 表示过滤块注释；第 4 位的“X”为 1 表示过滤行注释。高四位的“*”代表任意值（0 或 1），没有指示意义。

返回值：字符串，“ok”表示执行成功，如果执行错误，返回提示字符串，指明出错的行数与列数。过滤后的字符串被写入到同级目录下的新文件“XXX_string.txt”其中“XXX”为原文件名称。

参考流程：

按行读入

(1) 去掉空行

(2) 去掉空格

(3) 去掉块注释： 查找本行内的块注释符号“/”，如果该符号的前置符号为空、“(”、“)”、“}”、“{”、“;” 则认定为块注释开始，在本行内寻找匹配的块注释结束符号“/”；如果没有读入新行，直到找到结束符号“*/”； 两者之间的文本被去除，结束符号如果存在后续文本，则重新查找块注释开始符号

(4) 去掉行注释： 查找本行内的行注释符号“//”，如果该符号的前置符号为空、“(”、“)”、“}”、“{”、“;” 则认定为行注释，删除//及后续内容

(2) 测试用例：

No.	测试用例	返回值
1	c_code_filter(“math.c”, 15)	“ok”
2	c_code_filter(“math.c”, 15)	“第 10 行，第 8 列处理出错。”

8. 计算图形面积及周长（模块：classEx，所在文件名 class_hw.py）

利用 Python 尝试采用面向对象的设计方法。

（1）设计一个基类 Shape：

包含两个成员函数：

```
def cal_area(): 计算并返回该图形的面积，保留两位小数；  
def cal_perimeter(): 计算并返回该图形的周长，保留两位小数。  
def display(): 三行字符串，分别显示名称、面积、周长，如下：
```

```
id 是 rect  
面积是 6  
周长是 10
```

包含三个变量：

```
name: 表示 id，字符串类型；  
area: 表示面积；  
perimeter: 表示周长。
```

（2）设计三个派生类：Rectangle、Triangle、Circle；派生类分别实现基类中的两个成员函数。

Rectangle: 构造函数参数（name, length, width），name 为 id，其他均为浮点数，两位小数，代表长和宽。

Triangle: 构造函数参数（name, a, b, c），name 为 id，其他均为浮点数，两位小数，代表三边的长度。

Circle: 构造函数参数（name, radius），name 为 id，其他均为浮点数，两位小数，代表圆的半径。

9. （未完成）XML 文件的生成与解析（模块：dataEx，所在文件名 xml_hw.py）

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}

对应到上表的 XML 文件，返回值为： `["http://tms.osgeo.org/1.0.0", default, 6, 5]`。

{“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` 节点的个数和属性不是固定的。

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

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。

返回值：二进制报文的字节序列。

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。

11. （未完成）实现数据库的操作（模块：`dataEx`，所在文件名 `db_hw.py`）

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。

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。

返回值：创建成功，返回“ok”；失败返回“error”。

人员信息表 `Person`：

序号	字段名称	字段类型	取值范围
1	姓名 NAME	字符串	32 字符
2	性别 GENDER	字符串	2 字符
3	生日 BIRTH	日期	2000 年 10 月 20 日
4	身份证号 ID	字符串	18 位身份证号，全局唯一，作为主键
5	岗位 POSITIONID	字符串	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	岗位名称 POSITIONID	字符串	A、B、C、D；全局唯一，作为主键
2	薪水 SALARY	数字	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“ok”；失败返回错误原因，如“身份证号已存在”。

(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“ok”；失败返回错误原因，如“该身份证号不存在”。

(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 salary：整数，薪水。

返回值：设置成功，返回“ok”；失败返回“error”。

(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。

12. （未完成）获取当前天气情况（模块：netEx，所在文件名 net_hw.py）

利用 Python 从互联网公开服务中获取天气预报信息。天气信息来源网站：

http://www.webxml.com.cn/zh_cn/index.aspx

实现以下函数：

（1）获取天气：

函数原型：def get_weather(cityname)

参数 cityname：字符串，城市名称。

返回值：字典类型，如果可获取当前城市的天气信息，返回：时间、气温、风向风力、湿度、紫外线强度、空气质量等级等信息。如果无法获取天气信息或者出错，则返回空字典。

参考网站：

<https://blog.csdn.net/cw123458945/article/details/8146984>

http://www.webxml.com.cn/zh_cn/index.aspx

<http://ws.webxml.com.cn/WebServices/WeatherWS.asmx?op=getWeather>

（2）测试用例

No.	测试用例	返回值
1	get_weather(“西安”)	{“气温”:“19℃”, “风力风向”:“东风 2 级”, “湿度”:“38%”}