# 【Python代码说明文档】(以下图片详情附于Xmind)

[文档概述]  
本文档为Python代码说明文档，记录代码的设计思路、实现细节及注意事项。将涵盖以下主要内容：项目目标、流程逻辑介绍、函数方法介绍、技术介绍。

[项目目标]  
1、旨在实现从文件夹读取图片->实现创建数据库、tiles和metadata表->直接创建（地址）输出.db数据库功能。

2、旨在实现从左上角.db数据库变为左下角.db数据库功能。

3、旨在实现传入一个数据库（其包含了以左下角为原点的表b）,为其创建metadata并将b表变为标准的tiles表的功能。

4、前三者都会得到标准的数据库，最后一部修改文件后缀名为.mbtiles。

最后在geoserver上发布mbtiles文件进行验证，解决发布时候图片混乱的问题。

[环境]

Miniconda内置模块:

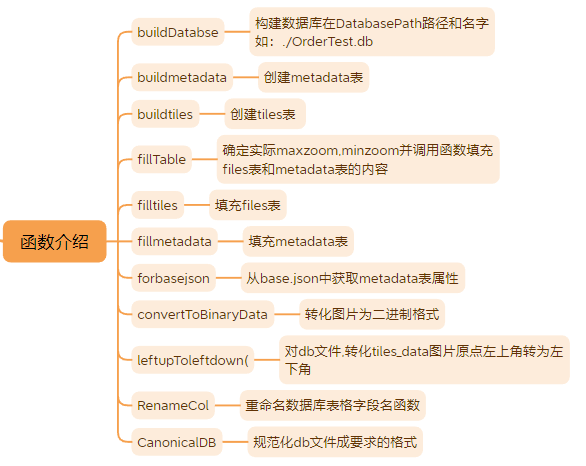
python :3.10.10

sqlite : 3.41.1

jupyter : 1.0.0

PyCharm 2023.1.2

[函数方法介绍]



该部分详细介绍函数方法。具体实现技术框架、依赖进行详细说明。同时，要给核心代码示例。

**1、**从文件夹读取图片->实现创建数据库、tiles和metadata表->直接创建（地址）输出.db数据库功能

1.1、依赖关系：该项目依赖于sqlite3、os、json、datatime库，这些是miniconda自带的库,无需额外安装。

依赖于函数方法:

|  |  |  |
| --- | --- | --- |
| name | meaning | 函数的**Parameter**和实现过程 |
| Buildmetadata | 创建metadata表 | **Parameter** :(sqliteConnection,cursor)  **实现过程:**调用执行创建语句：sql = '''CREATE TABLE metadata (name text, value text)''' |
| Buildtiles | 创建tiles表 | **Parameter** : (sqliteConnection,cursor)  **实现过程:**调用执行创建语句：sql = '''CREATE TABLE tiles (zoom\_level integer, tile\_column integer, tile\_row integer, tile\_data blob,uptime text)''' |
| convertToBinaryData | 转化图片为二进制格式 | **Parameter** : (filename)  **实现过程:**找到图片文件路径，用blobData = file.read()来获取该图片，则blobData为二进制格式 |
| forbasejson | 从base.json中获取metadata表属性 | **Parameter** : (jsonPath)  **实现过程:**先导入json文件：a = json.loads(content)为字典,然后提取里面的字段：  formatt = a["tileInfo"]['format'];description = a['description']  version = a["currentVersion"];  name = a["mapName"];tile\_width = a["tileInfo"]['rows'];  tile\_height = a["tileInfo"]['cols']; |
| fillmetadata | 填充metadata表 | **Parameter** : (sqliteConnection,cursor,maxzoom,minzoom)  **实现过程:** 先调用forbasejson获取到属性值，然后插入属性值。  sql = '''INSERT INTO metadata(name,value)  VALUES ('bounds',?),('format',?),('description',?), ('version',?),  ('type',?),('name',?),('tile\_width',?),('tile\_height',?), ('axis\_origin',?),('axis\_positive\_direction',?),('maxzoom',?),('minzoom',?)''' |
| filltiles | 填充tiles表 | **Parameter** : (sqliteConnection,cursor,maxzoom,minzoom)  **实现过程:** 共三次循环，  1、先以i在[minzoom,maxzoom]为索引，以filepath+i遍历文件夹下面的级数文件夹。  2、再分别进入各个级数文件夹，到里面后按照先大数字再小数字方式遍历行数文件夹。  3、最后按照观察原点由左上角变为左下角时，各元素的规律公式即其x坐标一致，即以左下角为原点的y坐标等于 (pow(2,level)-1 - 以左上角为原点的y坐标，原代码是：data\_tuple = (i,k,(orders-1)-j,empPhoto,datetime.datetime.now()) |
| fillTable | 确定实际maxzoom,minzoom并调用函数填充files表和metadata表的内容 | **Parameter** : (sqliteConnection,cursor)  **实现过程:** 先将文件夹下面各级数文件夹的名称填入整形列表temp(不能填入说明不是级数文件夹则try越过)，确定了实际的maxzoom=max(temp)和minzoom=min(temp),再调用fillmetadata, filltiles分别填充metadata表, tiles表 |
| buildDatabse | 构建数据库在DatabasePath路径和名字如：./OrderTest.db | **Parameter** : (DatabasePath)  **实现过程:** 先使用函数sqlite3.connect(DatabasePath)创建数据库，再调用buildmetadata, buildtiles创建初始的metadata,files表，形成初始的数据库,最后调用函数fillTable(sqliteConnection,cursor)进行填表内容。 |
| fileTodb | 从文件夹->实现创建->直接创建（地址）输出.db数据库(NEED), 默认以左下角为原点,将瓦片图片文件形成./OrderTest.db文件 | **Parameter** : (filepath,DatabasePath='./OrderTest.db',  bounds='-180.0,-90.0,180.0,90.0',  typee='baselayer',  axis\_origin='-180.0,-90.0',  axis\_positive\_direction='RightUp')  **实现过程:** 调用函数buildDatabse(DatabasePath)来从文件夹创建数据库。 |

**2、**从左上角.db数据库变为左下角.db数据库功能

2.1、依赖关系：该项目依赖于sqlite3库，这是miniconda自带的库,无需额外安装。

依赖于函数方法:

|  |  |  |
| --- | --- | --- |
| name | meaning | parameter |
| Buildmetadata | 创建metadata表 | **Parameter** : (sqliteConnection,cursor)  **实现过程:** 调用执行创建语句：sql = '''CREATE TABLE metadata (name text, value text)''' |
| Buildtiles | 创建tiles表 | **Parameter** : (sqliteConnection,cursor)  **实现过程:** 调用执行创建语句：sql = '''CREATE TABLE tiles (zoom\_level integer, tile\_column integer, tile\_row integer, tile\_data blob,uptime text)''' |
| leftupToleftdown | 对db文件,转化tiles\_data图片原点左上角转为左下角 | **Parameter** : (dbpath,orderTableName,  zoom\_level='zoom\_level',tile\_column='tile\_column'  ,tile\_row='tile\_row',tile\_data='tile\_data',  bounds='-180.0,-90.0,180.0,90.0',  description='description',version='1.3',  typee='baselayer',  name='Layers',tile\_width=256,tile\_height=256,  axis\_origin='-180.0,-90.0',  axis\_positive\_direction='RightUp')  **实现过程:**  1、先调用buildmetadata，buildtiles构建metadata和tiles表。  2、然后将orderTableName表的内容复制到tiles中便于后面修正。  3、再开始填充metadata表，先从tiles表中得到maxzoom=max(zoom\_level),minzoom=min(zoom\_level),format=tile\_data的数据切片，再结合参数的默认值得以填充。  4、最后orderTable表和tiles表联系，以公式进行tiles表tile\_data值的填入,公式即即以左下角原点的y坐标等于 (pow(2,level)-1 - 以左上角为原点的y坐标。代码:  sql = 'update tiles set tile\_data = (select '+tile\_data+' from '+orderTableName+' where '+\  zoom\_level+'=tiles.zoom\_level and '+tile\_column+'=tiles.tile\_column and '+\  tile\_row+'=(1<<tiles.zoom\_level)-1-tiles.tile\_row)' |

**3、**传入一个数据库（其包含了以左下角为原点的表b）,为其创建metadata并将b表变为标准的tiles表的功能

3.1、依赖关系：该项目依赖于sqlite3、os、numpy、re、datatime库，这些是miniconda自带的库,无需额外安装。

依赖于函数方法:

|  |  |  |
| --- | --- | --- |
| name | meaning | parameter |
| Buildmetadata | 创建metadata表 | **Parameter** : (sqliteConnection,cursor)  **实现过程:** 调用执行创建语句：sql = '''CREATE TABLE metadata (name text, value text)''' |
| RenameCol | 重命名数据库表格字段名函数 | **Parameter** :(sqliteConnection,cursor,tablename,colname,newcolname)  **实现过程:** 使用sql语句: sql = 'alter table '+tablename+' rename column '+colname+' to '+newcolname修改表的字段名 |
| CanonicalDB | 传入数据库地址,变为mbtiles文件规范化db文件成要求的格式 | **Parameter** : (dbpath,orderTableName='tiles',  zoom\_level='zoom\_level',tile\_column='tile\_column'  ,tile\_row='tile\_row',tile\_data='tile\_data',  bounds='-180.0,-90.0,180.0,90.0',  description='description',version='1.3',  typee='baselayer',  name='Layers',tile\_width=256,tile\_height=256,  axis\_origin='-180.0,-90.0',  axis\_positive\_direction='RightUp')  **实现过程:** 1、先修改要规范(Canonical)的数据库的目标表名orderTableName为’tiles’。2、规范该表的字段名.3、调用buildmetadata函数创建metadata表。4、从tiles中获取到maxzoom=max(zoom\_level),minzoom=min(zoom\_level)。从tiles表中tile\_data中切片切出format。5、再结合一些默认如bounds等参数，插入metadata的各个值。 |

**4、**前三者都会得到标准的数据库，最后一部修改文件后缀名为.mbtiles。

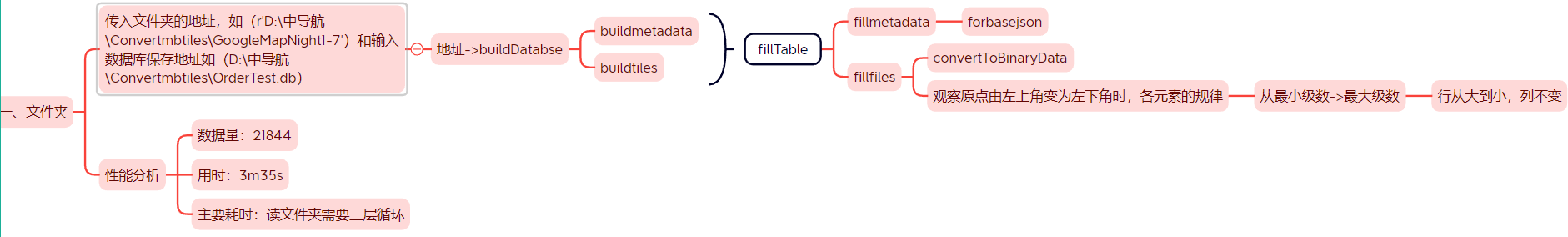
4.1、依赖关系：该项目依赖于os库，这是miniconda自带的库,无需额外安装。

依赖于函数方法:

|  |  |  |
| --- | --- | --- |
| name | meaning | parameter |
| batch\_rename | 修改文件的后缀名old\_ext到新的后缀名new\_ext | **Parameter** : (file\_dir,fname ,old\_ext, new\_ext)  **实现过程:** 先从文件夹地址读入各个文件名，并对比是否为目标文件，如果是则改名，并结束循环 |

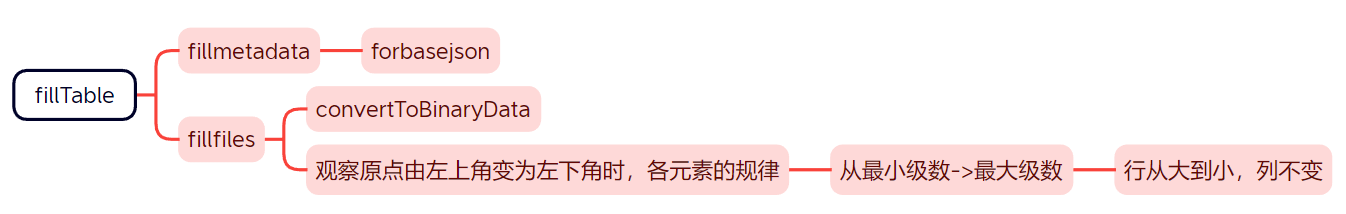
[代码逻辑介绍]  
这部分主要介绍项目的代码逻辑、核心代码。同时，针对可能出现的异常情况进行了try越过和except显示错误。

**功能一的逻辑，**先传入级数文件夹地址,用来作数据库的映射和输入数据库保存地址,若没有该数据库，则会自动创建在该目录下面。从文件夹读取图片->实现创建数据库、tiles和metadata表->直接创建（地址）输出.db数据库**：**



**核心代码（tiles表的映射）：**

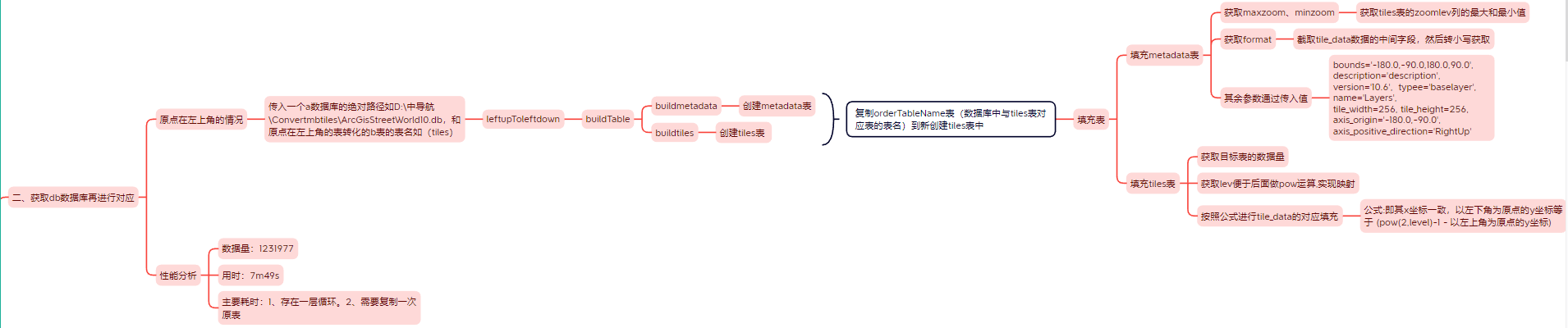
**思路图：**



**代码：**

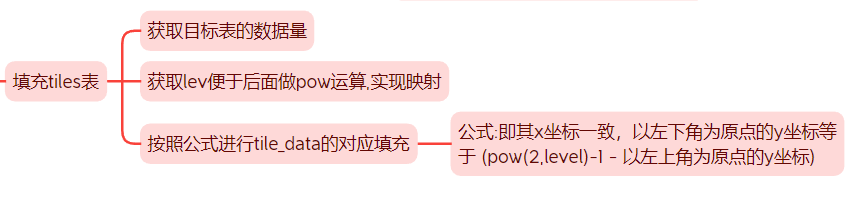
|  |
| --- |
| **#开始tiles表的映射**  **for i in range(minzoom,maxzoom+1):**  **path1 = filepath+'/'+str(i) #文件夹目录**  **files1= os.listdir(path1) #得到文件夹下的所有文件名称**  **#print(files1)**  **orders=pow(2,i)**  **try:**  **for j in range(orders-1,-1,-1):**  **path2 = path1+'/'+str(j)#文件夹目录**  **if formatt==' ':**  **files2= os.listdir(path2) #得到文件夹下的所有文件名称**  **pic=files2[0]**  **formatt=pic[pic.index('.')+1:len(pic)]**  **#print(formatt)**  **try:**  **for k in range(0,orders):**  **#tiles表开始填入数据**  **sqlite\_insert\_blob\_query = """ INSERT INTO tiles**  **(zoom\_level, tile\_column, tile\_row, tile\_data,uptime) VALUES (?, ?, ?, ?,?)"""**  **empPhoto = convertToBinaryData(path2+'/'+str(k)+'.'+formatt)**  **# Convert data into tuple format**  **data\_tuple = (i,k,(orders-1)-j,empPhoto,datetime.datetime.now())**  **cursor.execute(sqlite\_insert\_blob\_query, data\_tuple)**  **sqliteConnection.commit()**  **except sqlite3.Error as error:**  **print('error,映射tiles表出错,找不到文件,',k,'列下标图片是否连续？',error)**  **except sqlite3.Error as error:**  **print('error,映射tiles表出错,找不到文件,',j,'行文件数是否连续？',error)**  **print('tiles表映射结束。')** |

**功能二的逻辑，**传入一个a数据库的绝对路径如D:\中导航\Convertmbtiles\ArcGisStreetWorld10.db，和原点在左上角的表转化的b表的表名如（tiles）。将左上角.db数据库变为左下角.db数据库**：**



**核心代码：**

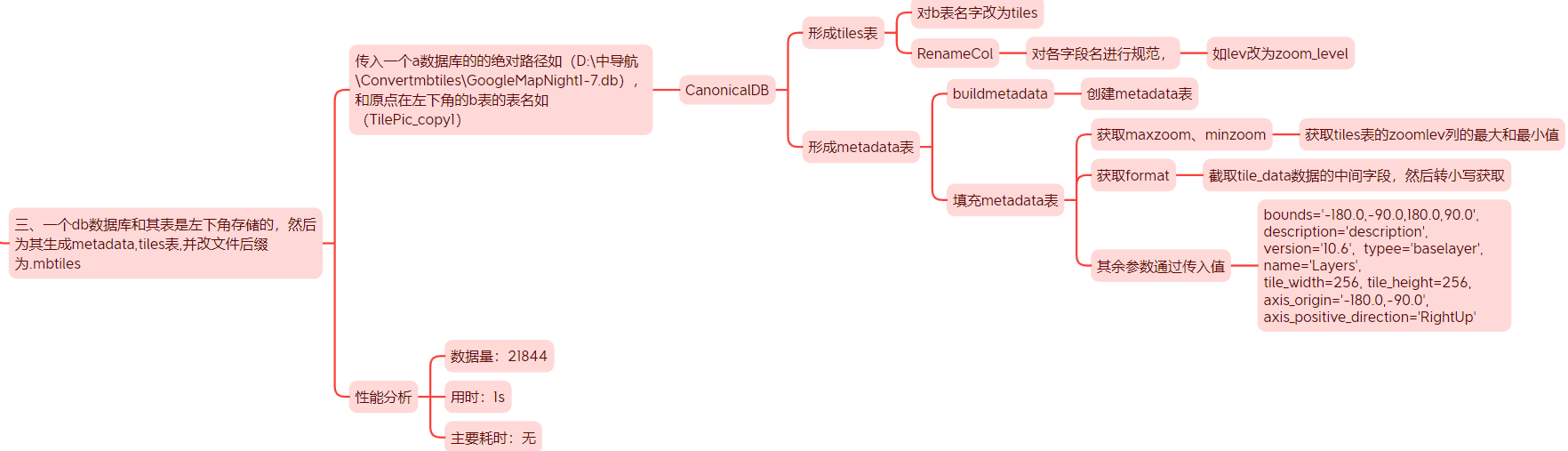
**思路图：**



**代码：**

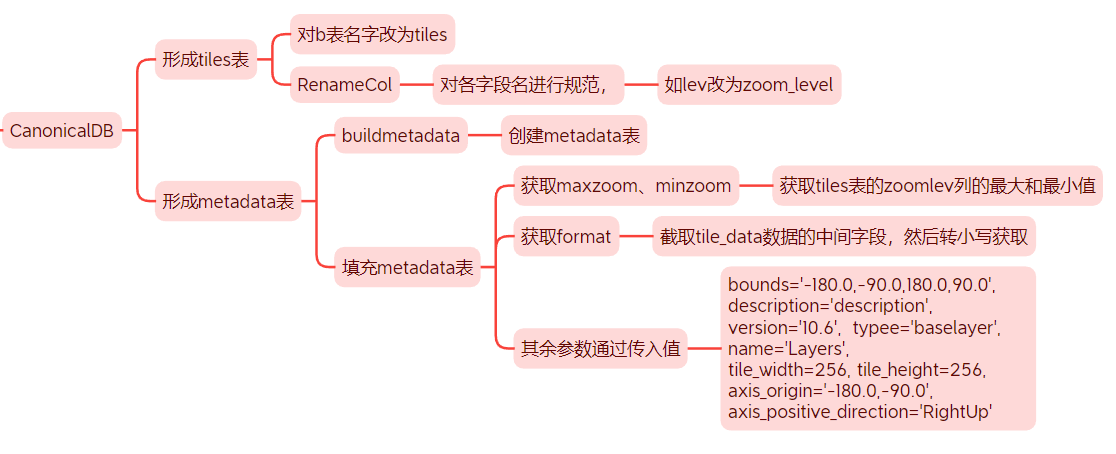
|  |
| --- |
| **print('开始tiles表的映射......')**  **sql = 'update tiles set tile\_data = (select '+tile\_data+' from '+orderTableName+' where '+\**  **zoom\_level+'=tiles.zoom\_level and '+tile\_column+'=tiles.tile\_column and '+\**  **tile\_row+'=(1<<tiles.zoom\_level)-1-tiles.tile\_row)'**  **#print(sql)**  **cursor.execute(sql)**  **sqliteConnection.commit()**  **print('tiles表映射结束。')** |

**功能三的逻辑，**传入一个a数据库的的绝对路径如（D:\中导航\Convertmbtiles\GoogleMapNight1-7.db），和原点在左下角的b表的表名如（TilePic\_copy1）,为其创建metadata并将b表变为标准的tiles表**：**



**核心代码（规范目标表为tiles表,获取metadata的maxzoom、minzoom、format属性值）**

**思路图：**



**代码：**

|  |
| --- |
| **#重命名数据库表格字段名函数,parameter:sqliteConnection,cursor,tablename,colname,newcolname**  **RenameCol(sqliteConnection,cursor,orderTableName,zoom\_level,'zoom\_level'); RenameCol(sqliteConnection,cursor,orderTableName,tile\_column,'tile\_column');**  **RenameCol(sqliteConnection,cursor,orderTableName,tile\_row,'tile\_row'); RenameCol(sqliteConnection,cursor,orderTableName,tile\_data,'tile\_data');**  **#1、获取maxzoom、minzoom**  **sql = '''SELECT max(zoom\_level),min(zoom\_level) FROM tiles'''#从规范的tiles中的zoom\_level列，获取到maxzoom、minzoom**  **cursor.execute(sql)**  **ll = cursor.fetchall()**  **maxzoom = ll[0][0];minzoom = ll[0][1]**  **#print(maxzoom,minzoom)**  **#2、获取format**  **formatt=''**  **sql = '''SELECT tile\_data FROM tiles where Rowid=1'''#得到tiles表中第一行tile\_data的数据，从中得到数据类型**  **cursor.execute(sql)**  **tile\_data1 = cursor.fetchall()**  **t = tile\_data1[0][0]#得到tiles表中第一行tile\_data的数据t**  **formatt=''.join(list(str(t[1:5]))[2:5]).lower()#从数据t中拆分出展示其数据类型的字段,如：png#print(formatt)** |

**功能四的逻辑，**从dbpath中切出文件夹地址，和文件名用于修改后缀，修改文件.db的后缀名，变.mbtiles文件**：**

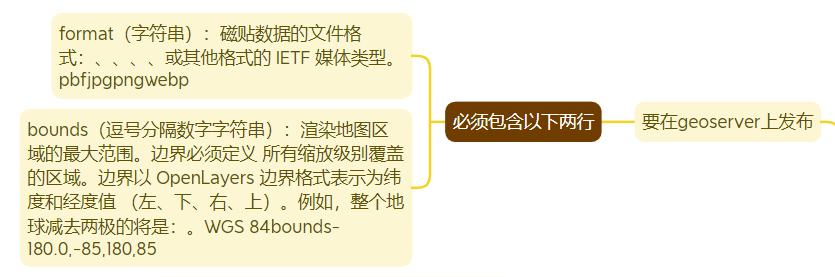
[技术介绍]  
该项目主要采用以下技术实现：

1. Python语言。
2. Miniconda内置模块。
3. SQLite数据库。
4. 异常处理：对异常进行捕获和处理，确保系统的稳定性和可用性。
5. 文档测试：定期进行文档测试，保证文档的质量和可读性。

[mbtiles文件介绍]

进行mbtiles测试流程:

用geoserver进行metadata表的属性删除实验，得出要在geoserver上发布至少包括bounds和format。



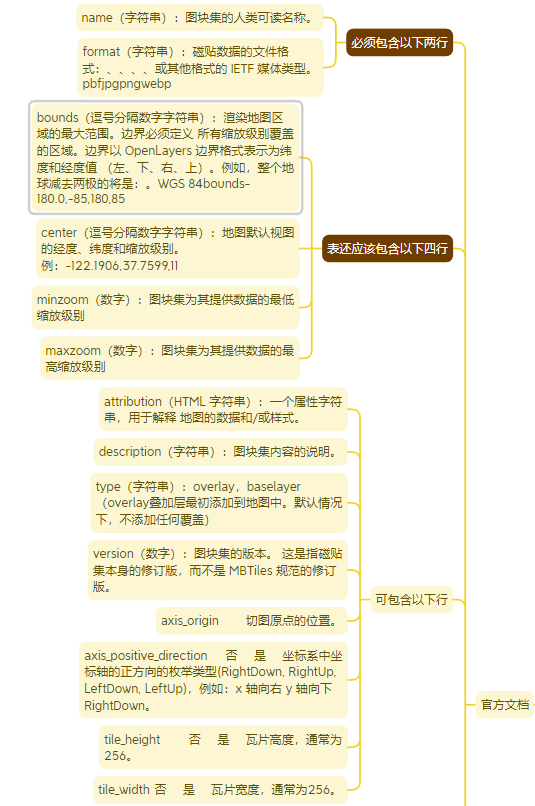
**.mbtiles文件特点:**

****

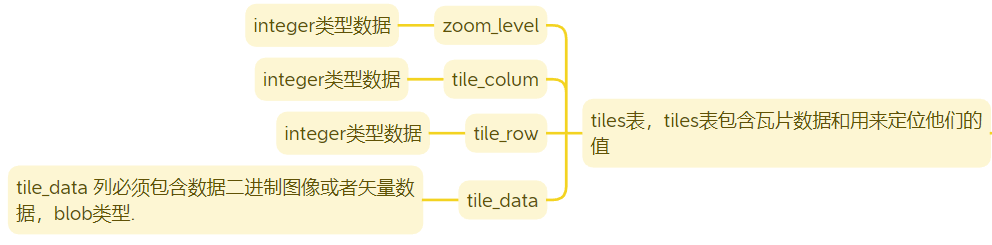
**.mbtiles文件构成:**

**数据库必须包含metadata元数据字段，tiles表，可以包含map 数据表，images 瓦片数据表。**

**metadata元数据字段:**



**tiles表:**

**map 数据表和images 瓦片数据表:**

****

[注意事项]



Problem::

其中power在python的sqlite中会找不到如以下语句:

UPDATE tiles

SET tile\_data = (

SELECT

pic

FROM

TilePic

WHERE

lev = tiles.zoom\_level

AND mapCol = tiles.tile\_column

AND mapRow = power ( 2, tiles.zoom\_level )-1 - tiles.tile\_row

)

在python中运行会报错:no such function: power

这里用位运算符代替如power(2,0) = 1 = 1<<0; power(2,1)=2=1<<1...最多select (1 << 63)，即如果(1<<64)则返回0，超限。