**Grid.ipynb为根据广州、重庆、上海等城市的充电站信息及POI信息进行选址。具体说明如下：**

**1. read\_data(filename: str, select\_city: str) -> pd.DataFrame**

**作用**:  
从指定的 Excel 文件中读取数据，并根据所选城市名称过滤出该城市的相关数据。

**输入**:

* filename: 包含数据的 Excel 文件路径。
* select\_city: 需要选择的城市名称。

**输出**:

* 返回过滤后的 pd.DataFrame，包含选定城市的数据。

**2. calculate\_scores(df: pd.DataFrame, col: pd.Index) -> pd.DataFrame**

**作用**:  
对指定列进行归一化处理（使用 MinMaxScaler），将结果转换为百分制评分，并计算所有评分列的总分。

**输入**:

* df: 原始数据集 pd.DataFrame。
* col: 需要进行归一化和评分的列名列表。

**输出**:

* 包含新的评分列以及总评分列的 pd.DataFrame。

**3. extract\_coordinates(df: pd.DataFrame) -> pd.DataFrame**

**作用**:  
从 "栅格编号" 列中提取经纬度信息，将其分为 "lon" 和 "lat" 列。

**输入**:

* df: 包含 "栅格编号" 列的 pd.DataFrame。

**输出**:

* 包含经纬度信息（lon 和 lat 列）的更新 pd.DataFrame。

**4. create\_geodataframe(df: pd.DataFrame) -> gpd.GeoDataFrame**

**作用**:  
将普通的 pd.DataFrame 转换为 GeoDataFrame，其中包含几何点信息（经纬度）。

**输入**:

* df: 包含经纬度信息的 pd.DataFrame。

**输出**:

* gpd.GeoDataFrame，包含几何信息（点）并与原始数据集相关联。

**5. score\_grid(df: pd.DataFrame, bounds: tuple) -> pd.DataFrame**

**作用**:  
基于输入的区域边界 (bounds) 生成格网，并根据栅格编号对每个格网计算总得分。

**输入**:

* df: 包含经纬度和总得分的 pd.DataFrame。
* bounds: 用于定义区域边界的坐标范围，格式为元组（tuple）。

**输出**:

* 返回包含总得分的格网 pd.DataFrame，只保留有得分的格网。

**6. classify\_grid(grid: pd.DataFrame) -> pd.DataFrame**

**作用**:  
根据格网的总得分对其进行降序排序，并将格网划分为 7 个评分等级。

**输入**:

* grid: 包含总得分的格网 pd.DataFrame。

**输出**:

* 按总得分排序并分类后的 pd.DataFrame，增加了评分等级列 num。

**7. select\_best\_sites(df: pd.DataFrame, grid: pd.DataFrame) -> pd.DataFrame**

**作用**:  
根据分类后的格网数据，从每个栅格中选择评分最高的站点。

**输入**:

* df: 原始数据集，包含坐标和总得分。
* grid: 已分类的格网数据集，包含评分等级。

**输出**:

* 返回按评分等级和总得分筛选出的最佳站点 pd.DataFrame。

**8. main(filename: str, select\_city: str, bounds: tuple) -> pd.DataFrame**

**作用**:  
执行完整的站点评分流程，从读取数据到筛选出最佳站点，并返回最终结果。

**输入**:

* filename: 数据文件路径。
* select\_city: 选择的城市名称。
* bounds: 定义的区域边界坐标范围，格式为元组。

**输出**:

* 最终筛选出的最佳站点 pd.DataFrame，包含每个栅格中的最佳站点信息。

代码如下：

import pandas as pd

import geopandas as gpd

from sklearn.preprocessing import MinMaxScaler

import transbigdata as tbd

# 读取数据文件，选择城市

def read\_data(filename, select\_city):

    """

    输入:

        filename (str): 文件路径

        select\_city (str): 选择的城市名称

    输出:

        pd.DataFrame: 过滤后的数据集，仅包含选定城市的数据

    """

    df = pd.read\_excel(filename)

    return df[df["市"] == select\_city]

# 对每个列进行MinMax标准化，并计算得分

def calculate\_scores(df, col):

    """

    输入:

        df (pd.DataFrame): 输入的数据集

        col (pd.Index): 需要进行标准化的列

    输出:

        pd.DataFrame: 返回包含标准化得分的新数据集

    """

    for c in col:

        scaler = MinMaxScaler()

        new\_col = c + "\_score"

        df[new\_col] = scaler.fit\_transform(df[[c]]) \* 100

    df["total\_score"] = df[[c for c in df.columns if "score" in c]].sum(axis=1)

    return df

# 提取经纬度坐标信息

def extract\_coordinates(df):

    """

    输入:

        df (pd.DataFrame): 输入的数据集，包含栅格编号

    输出:

        pd.DataFrame: 返回提取了经纬度坐标的更新数据集

    """

    df["lon"] = df["栅格编号"].str.split("-", expand=True)[1].astype("float")

    df["lat"] = df["栅格编号"].str.split("-", expand=True)[2].astype("float")

    return df

# 将数据转换为GeoDataFrame

def create\_geodataframe(df):

    """

    输入:

        df (pd.DataFrame): 包含经纬度的普通DataFrame

    输出:

        gpd.GeoDataFrame: 包含几何信息的GeoDataFrame

    """

    return gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df["lon"], df["lat"]))

# 对栅格评分，并生成格网

def score\_grid(df, bounds):

    """

    输入:

        df (pd.DataFrame): 包含经纬度及评分的DataFrame

        bounds (tuple): 定义区域边界的坐标

    输出:

        pd.DataFrame: 包含总评分的格网数据

    """

    p = tbd.area\_to\_params(bounds, accuracy=5000)

    grid, p = tbd.area\_to\_grid(bounds, params=p)

    df["LONCOL"], df["LATCOL"] = tbd.GPS\_to\_grid(df["lon"], df["lat"], params=p)

    score = df.groupby(["LONCOL", "LATCOL"])["total\_score"].sum().reset\_index()

    score.columns = ["LONCOL", "LATCOL", "total\_score"]

    grid = pd.merge(grid, score, on=["LONCOL", "LATCOL"], how="left")

    return grid[grid["total\_score"].notna()]

# 根据评分将格网进行分类

def classify\_grid(grid):

    """

    输入:

        grid (pd.DataFrame): 包含格网和评分的数据

    输出:

        pd.DataFrame: 排序后的格网数据，并按评分分类

    """

    grid.sort\_values(by="total\_score", ascending=False, inplace=True)

    grid["num"] = pd.cut(grid["total\_score"], bins=7, labels=["0", "1", "2", "3", "4", "5", "6"])

    return grid

# 选择最佳站点

def select\_best\_sites(df, grid):

    """

    输入:

        df (pd.DataFrame): 原始数据集，包含坐标和评分

        grid (pd.DataFrame): 已经分类的格网数据

    输出:

        pd.DataFrame: 每个栅格中的最佳站点选择

    """

    select = pd.merge(df[["geometry", "LONCOL", "LATCOL", "total\_score"]],

                      grid[["LONCOL", "LATCOL", "num"]])

    select["num"] = select["num"].astype(int)

    select = select.sort\_values(by=["LONCOL", "LATCOL", "total\_score"], ascending=[True, True, False])

    f = select.groupby(["LONCOL", "LATCOL"]).apply(lambda x: x.head(x["num"].iloc[0])).reset\_index(drop=True)

    return f

# 主函数，用于调用所有步骤

def main(filename, select\_city, bounds):

    """

    输入:

        filename (str): 文件路径

        select\_city (str): 选择的城市

        bounds (tuple): 定义区域边界的坐标

    输出:

        pd.DataFrame: 最终的站点选择结果

    """

    df\_gz = read\_data(filename, select\_city)

    cols = df\_gz.columns[4:]

    df\_gz = calculate\_scores(df\_gz, cols)

    df\_gz = extract\_coordinates(df\_gz)

    df\_gz = create\_geodataframe(df\_gz)

    grid = score\_grid(df\_gz, bounds)

    grid = classify\_grid(grid)

    best\_sites = select\_best\_sites(df\_gz, grid)

    return best\_sites

# 调用主函数

# best\_sites = main(filename, select\_city, bounds)