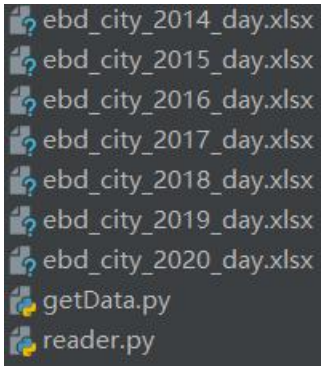
(归一化方法采用的是简单的取最大值最小值进行计算的)
使用说明：(该类只用于获取训练数据集以及结果的反归一化处理)

1. 第一次运行时将下列文件全部放在一个路径下
(reader.py 较之前有改动，2014 和 2020 的数据不需要)



2. import 文件

```
import getData
```

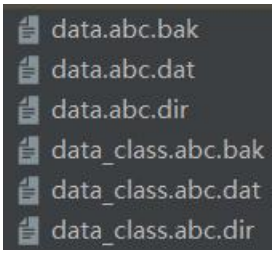
(reader.py 只需要放在相同路径下即可，不需要 import)

3. 使用(大致流程在 test.py 中有)

3.1 初次使用

```
# 第一次使用 运行后生成6个文件
test = getData.GetData()
print(test.get_nor_data())
test.save_class()
```

第一次运行时需要从 excel 中读入数据，速度可能较慢
第一次运行后会在 getData.py 所在目录下生成 6 个文件



三个为读出的数据所存文件，后三个为当前对象所存文件
后续再使用时可以直接从文件里读，速度可以加快不少

3.2 后续使用

```
# 后续使用 直接读取所生成的文件即可
test = getData.GetData.load_class()
print(test.get_nor_data())
```

后续可以用类的静态方法 load_class 来直接读取之前的对象
所存数据会在类中使用 无需接口调用

4. 内部方法介绍

不推荐自行调用内部除 test.py 中出现的其他方法(但是为了好扩展，还是没设为 private 的)调用逻辑不当可能导致文件中储存的数据错误

test.py 中的方法

```
def __init__(self):
    self.data = []  # 最后输出为(1812,367,6)
    self.data_isRead = False
    self.max = []
    self.min = []
    self.data_isNor = False
```

构造函数无需传入参数 所有读入写出地址均默认在当前所在路径下(懒得改了)

另注：data_isRead 和 data_isNor 两个 bool 变量标记了是否要从 excel 读取数据以及是否要对存储数据进行归一化计算。因此当新生成的对象想要读取之前类所创建的 data 文件时需要把 data_isRead 设为 True，而且因为归一化后的数据也是计算后直接存入 data 文件的，新对象直接调用 get_nor_data 方法会导致对已经归一化的数据再次归一化导致存储的数据错误，而且归一化过程中所存储的有关参数也会丢失。因此不推荐后续使用声明新的对象，只需要通过 load_class 方法读取之前声明的对象即可。

save_class()方法会在当前目录下生成 data_class.abc.bak/.dat/.dir 三个文件 多次调用会覆盖之前的文件

```
def save_class(self):
    with shelve.open("data_class.abc", 'n') as s:
        s['class'] = self
```

load_class()方法为静态方法，可类似单例模式直接读取对象

```
@staticmethod
def load_class():
    with shelve.open("data_class.abc", 'r') as s:
        return s['class']
```

get_nor_data()方法为推荐使用的封装好的方法会返回归一化好的(1812, 367, 6)的 tensor 变量，同时会将该变量存入文件中

```
def get_nor_data(self):

    self.normalization()
    self.save_data()

    return torch.tensor(self.data)
```

renormalization()方法用于将结果反归一化(必须调用进行归一化的那个对象的方法)参数为结果，以及其的三个维度，默认第三个维度为 6 且不可以大于 6 (因为不知道结果是几维的)

```python
def renormalization(self, data_to_re, x, y, z):

    for i in range(0, x):
        for j in range(0, y):
            for k in range(0, z):
                data_to_re[i][j][k] = data_to_re[i][j][k] * (self.max[k] - self.min[k]) + self.min[k]

    return data_to_re
```