3.项目测试

（一）判断某一天是否为假期

S1、导入库和模块：

* 导入了holidays库以及用于处理日期的date类

import holidays

from datetime import date

S2、创建国家假期对象：

* 使用CountryHoliday类创建一个代表中国（国家代码为'CN'）的假期对象cn\_holidays。这个对象包含了中国的所有假期信息。

cn\_holidays = holidays.CountryHoliday('CN')

S3、检查特定日期是否是假期：使用in关键字检查特定日期（2024年1月1日）是否在中国的假期列表中。如果是假期，就打印相应的消息，包括该日期对应的假期名称，否则打印非假期的消息。

if date(2024, 1, 1) in cn\_holidays:

print("2024.1.1("+cn\_holidays.get('2024-01-01')+") is a holiday!")

else:

print("2024.1.1 is not a holiday :(")

（二）获得某个国家的假期

S1、导入库和模板：

import holidays

from datetime import date

S2、创建假期对象：

cn\_holidays = holidays.CountryHoliday('CN')

S3、遍历2024年的假期并打印：

* holidays.China(years=2024) 返回一个代表中国2024年的假期对象
* items() 方法返回一个包含日期和假期名称的元组列表
* sorted() 函数用于按日期排序输出
* 遍历排序后的元组列表，并使用 print 打印每个假期的日期和名称

for date, name in sorted(holidays.China(years=2024)。items()):

print(date, name)

（三）添加自定义假期：自定义一个新的假期类，继承自china.CN，通过覆盖\_populate方法来添加额外的自定义假期。

S1、导入库和模块：

from datetime import date

from holidays.countries import china

S2、自定义假期类 ‘MyHolidays’：

* 创建了一个名为MyHolidays的类，继承自china.CN，这表示它包含了中国的假期信息
* 覆盖了\_populate方法，首先调用了父类的方法（super().\_populate(year)），然后在父类的假期基础上，添加了一个自定义假期，即在每年的7月13日定义了一个名为"My Holiday"的假期

class MyHolidays(china.CN):

def \_populate(self, year):

super().\_populate(year)

self[date(year, 7, 13)] = "My Holiday"

S3、创建日历实例并查询假期：

* 创建了MyHolidays类的实例 holidays
* 使用get方法查询特定日期的假期信息。在例子中，分别查询了2023年7月13日和10月1日的假期信息

# 创建一个日历实例

holidays = MyHolidays()

# 查询特定日期的假期信息

print(holidays.get('2023-07-13')) # output: My Holiday

print(holidays.get('2023-10-01')) # output: 国庆节 (China National Day)