Python 程序设计大作业报告

数据来源:

腾讯疫情数据 API
https://news.qq.com/zt2020/page/feiyan.htm#/global

基本功能:

1. **用独立图或表展示用户指定国家用户指定时间段的疫情有关数据。同时，展示用户指定国家疫情峰值信息，并且对曲线图做平滑处理。**

源代码:

```
# coding: utf-8
import requests
import json
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from scipy.interpolate import make_interp_spline
from datetime import datetime

print("本程序用独立图展示用户指定国家指定时间段指定相关数据的疫情变化图，并做了平滑处理")
query_url='https://api.inews.qq.com/newsqa/v1/automation/foreign/daily/list'
country_list=["美国","巴西","俄罗斯","西班牙","英国","意大利","法国","德国","土耳其","印度","伊朗","秘鲁","加拿大","智利","沙特阿拉伯","墨西哥","巴基斯坦","比利时","卡塔尔","荷兰","白俄罗斯","孟加拉","厄瓜多尔","瑞典","新加坡","阿联酋","葡萄牙","瑞士","南非","爱尔兰","哥伦比亚","印度尼西亚","科威特","波兰","乌克兰","埃及","罗马尼亚","以色列","日本本土","奥地利","多米尼加","菲律宾","阿根廷","阿富汗","巴拿马","丹麦","韩国","塞尔维亚","巴林","哈萨克斯坦","捷克","阿尔及利亚","尼日利亚","挪威","阿曼","亚美尼亚","玻利维亚","马来西亚","摩洛哥","摩尔多瓦","加纳","澳大利亚","芬兰","喀麦隆","伊拉克","洪都拉斯","阿塞拜疆","苏丹","危地马拉","卢森堡","匈牙利","塔吉克斯坦","乌兹别克斯坦","几内亚","塞内加尔","泰国","希腊","吉布提","科特迪瓦","刚果（金）","保加利亚","波黑","加蓬","克罗地亚","萨尔瓦多","北马其顿","古巴","爱沙尼亚","冰岛","索马里","立陶宛","吉尔吉斯斯坦","斯洛伐克","新西兰","肯尼亚","斯洛文尼亚","斯里兰卡","马尔代夫","海地","委内瑞拉","几内亚比绍","黎巴嫩","马里","赞比亚","拉脱维亚","突尼斯","阿尔巴尼亚","赤道几内亚","哥斯达黎加","尼日尔","塞浦路斯","尼泊尔","巴拉圭","布基纳法索","乌拉圭","塞拉利昂","安道尔","尼加拉瓜
```

```
","格鲁吉亚","埃塞俄比亚","约旦","乍得","钻石号邮轮","中非共和国","圣马力诺
","马达加斯加","马耳他","刚果（布）","牙买加","坦桑尼亚","巴勒斯坦","多哥","
佛得角","卢旺达","毛里求斯","越南","黑山","毛里塔尼亚","乌干达","斯威士兰","
利比里亚","也门","莫桑比克","贝宁","缅甸","蒙古","文莱","圭亚那","津巴布韦
","柬埔寨","叙利亚","特立尼达和多巴哥","马拉维","巴哈马","利比亚","摩纳哥","
巴巴多斯","科摩罗","列支敦士登公国","安哥拉","布隆迪","厄立特里亚","马提尼克
岛","博茨瓦纳","不丹","冈比亚","安提瓜和巴布达","东帝汶","格林纳达","纳米比亚
","老挝","斐济","伯利兹","圣文森特和格林纳丁斯","圣卢西亚","多米尼克","圣基茨
和尼维斯","梵蒂冈","苏里南","塞舌尔","巴布亚新几内亚","莱索托"]
print("请选择所想展示的国家（输入数字即可）：")
for i in range(len(country_list)):
    print(str(i)+"."+country_list[i])
n=int(input())

shuju=['每日新增', '确诊病例', '治愈病例', '死亡人数']
print("请输入你想展现的数据（输入数字即可）：")
for i in range(len(shuju)):
    print(str(i)+'.'+shuju[i])
n_shuju=int(input())

param = {"country": country_list[n]}
res = requests.post(query_url, param)
d=json.loads(res.content.decode())
data=pd.DataFrame(d['data'])
xs=pd.DataFrame(d['data']).date

for i in range(len(xs)):
    xs[i]="2020-"+xs[i][0]+xs[i][1]+'-'+xs[i][3]+xs[i][4]
xs = [datetime.strptime(d, '%Y-%m-%d').date() for d in xs]
print("请输入你想展现的时间（输入数字，间隔空格即可）：")
for i in range(len(xs)):
    print(str(i)+'.'+str(xs[i]))
start,end=map(int,input().split())
if n_shuju==0:
    ys=data.confirm_add
elif n_shuju==1:
    ys=data.confirm
elif n_shuju==2:
    ys=data.heal
else:
    ys=data.dead
y=list()
for i in range(start,end+1):
    y.append(ys[i])
```

```
startDate=xs[start]
endDate=xs[end]
index = pd.date_range(startDate, endDate)
cols = ['value']
df = pd.DataFrame(y, index=index, columns=cols)
fig, axs = plt.subplots(1,1, figsize=(18,5))
index_hourly = pd.date_range(startDate, endDate, freq='1H')
df_smooth = df.reindex(index=index_hourly).interpolate('cubic')
df_smooth = df_smooth.rename(columns={'value':'smooth'})
df_smooth.plot(ax=axs, alpha=100)
df.plot(ax=axs, alpha=100)
plt.xlabel('date')
plt.ylabel('number')
plt.title("Added trend graph after smoothing the infected person data")
plt.show()
max=ys.max()
time=list()
for i in range(len(xs)):
    if ys[i]==max:
        time.append(data.date[i])
print("所选数据峰值为"+str(max))
print("出现时间为：")
for i in range(len(time)):
    print(time[i])
```

完成任务的重要部分：

指定国家通过 country_list[]函数输出来供用户选择，将网页上的时间通过 datetime.strptime 函数转换为 python3 内部可识别的时间，再逐次输出来让用户选择想展现的时间段。

平滑处理：通过 startdate 和 enddate 来储存用户指定的开始和结束时间，再利用 pd.date_range 和 df.reindex().interpolate()这两个函数，用“插值法”对数据进行插值处理，达到展现的折线图平滑化的效果

网页上爬取的新的本就是数组，那么通过 max()可以直接获得峰值信息，为了防止有多天同时出现峰值的情况，再通过遍历一次数组即可得出所有峰值数据的对应日期。

分工：

此项目由徐毅-18071132 完成

2. 变化趋势并预测未来可能性:

源代码:

```
# coding: utf-8
import requests
import json
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from datetime import datetime

query_url='https://api.inews.qq.com/newsqa/v1/automation/foreign/daily/
list'
country_list=["美国","巴西","俄罗斯","西班牙","英国","意大利","法国","德国
","土耳其","印度","伊朗","秘鲁","加拿大","智利","沙特阿拉伯","墨西哥","巴基
斯坦","比利时","卡塔尔","荷兰","白俄罗斯","孟加拉","厄瓜多尔","瑞典","新加坡
","阿联酋","葡萄牙","瑞士","南非","爱尔兰","哥伦比亚","印度尼西亚","科威特
","波兰","乌克兰","埃及","罗马尼亚","以色列","日本本土","奥地利","多米尼加
","菲律宾","阿根廷","阿富汗","巴拿马","丹麦","韩国","塞尔维亚","巴林","哈萨
克斯坦","捷克","阿尔及利亚","尼日利亚","挪威","阿曼","亚美尼亚","玻利维亚","
马来西亚","摩洛哥","摩尔多瓦","加纳","澳大利亚","芬兰","喀麦隆","伊拉克","洪
都拉斯","阿塞拜疆","苏丹","危地马拉","卢森堡","匈牙利","塔吉克斯坦","乌兹别克
斯坦","几内亚","塞内加尔","泰国","希腊","吉布提","科特迪瓦","刚果（金）","保
加利亚","波黑","加蓬","克罗地亚","萨尔瓦多","北马其顿","古巴","爱沙尼亚","冰
岛","索马里","立陶宛","吉尔吉斯斯坦","斯洛伐克","新西兰","肯尼亚","斯洛文尼亚
","斯里兰卡","马尔代夫","海地","委内瑞拉","几内亚比绍","黎巴嫩","马里","赞比
亚","拉脱维亚","突尼斯","阿尔巴尼亚","赤道几内亚","哥斯达黎加","尼日尔","塞浦
路斯","尼泊尔","巴拉圭","布基纳法索","乌拉圭","塞拉利昂","安道尔","尼加拉瓜
","格鲁吉亚","埃塞俄比亚","约旦","乍得","钻石号邮轮","中非共和国","圣马力诺
","马达加斯加","马耳他","刚果（布）","牙买加","坦桑尼亚","巴勒斯坦","多哥","
佛得角","卢旺达","毛里求斯","越南","黑山","毛里塔尼亚","乌干达","斯威士兰","
利比里亚","也门","莫桑比克","贝宁","缅甸","蒙古","文莱","圭亚那","津巴布韦
","柬埔寨","叙利亚","特立尼达和多巴哥","马拉维","巴哈马","利比亚","摩纳哥","
巴巴多斯","科摩罗","列支敦士登公国","安哥拉","布隆迪","厄立特里亚","马提尼克
岛","博茨瓦纳","不丹","冈比亚","安提瓜和巴布达","东帝汶","格林纳达","纳米比亚
","老挝","斐济","伯利兹","圣文森特和格林纳丁斯","圣卢西亚","多米尼克","圣基茨
和尼维斯","梵蒂冈","苏里南","塞舌尔","巴布亚新几内亚","莱索托"]
print("请选择所想展示的国家（输入数字即可）：")
for i in range(len(country_list)):
    print(str(i)+"."+country_list[i])
n=int(input())
param = {"country": country_list[n]}
res = requests.post(query_url, param)
d=json.loads(res.content.decode())
data=pd.DataFrame(d['data'])
xs=pd.DataFrame(d['data']).date
for i in range(len(xs)):
```

```
    xs[i]="2020-"+xs[i][0]+xs[i][1]+'-'+xs[i][3]+xs[i][4]
xs = [datetime.strptime(d, '%Y-%m-%d').date() for d in xs]
data.insert(0,'riqi',xs)
#data.to_csv("tzzs_data.csv")


x = np.arange(1, len(xs)+1, 1)
y = np.array(data.confirm)
z1 = np.polyfit(x, y,3)#用 n 次多项式拟合
p1 = np.poly1d(z1)
print(p1) #在屏幕上打印拟合多项式
yvals=p1(x)
plot1=plt.plot(x, y, '*',label='original values')
plot2=plt.plot(x, yvals, 'r',label='polyfit values')
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.legend(loc=4)
plt.title('polyfitting')
plt.show()
```

完成任务的重要部分：

首先，由已有的数学知识，可以证明，任意函数都可以表示为多项式形式。那么对于疫情数据的简单预测，可以转换为选择合适的函数进行拟合。经过查阅文献和多次尝试后发现，三次项多项式比较合适，所以，利用 np.polyfit 函数，采用三次项多项式对数据进行拟合处理，并通过拟合得出的函数，带入未来的数据，进行预测未来的值。

分工：

此项目由徐琦-19035109 完成

3. 创意功能区：

1）对两个国家的指定数据进行对比，用平滑的折线图展示：

源代码：

```
# coding: utf-8
import requests
import json
import numpy as np
import pandas as pd
```

```
from matplotlib import pyplot as plt
from scipy.interpolate import make_interp_spline
from datetime import datetime

print("本程序用独立图展示用户指定两个国家指定时间段指定相关数据的疫情变化对比图，并做了平滑处理")
query_url='https://api.inews.qq.com/newsqa/v1/automation/foreign/daily/list'
country_list=["美国","巴西","俄罗斯","西班牙","英国","意大利","法国","德国","土耳其","印度","伊朗","秘鲁","加拿大","智利","沙特阿拉伯","墨西哥","巴基斯坦","比利时","卡塔尔","荷兰","白俄罗斯","孟加拉","厄瓜多尔","瑞典","新加坡","阿联酋","葡萄牙","瑞士","南非","爱尔兰","哥伦比亚","印度尼西亚","科威特","波兰","乌克兰","埃及","罗马尼亚","以色列","日本本土","奥地利","多米尼加","菲律宾","阿根廷","阿富汗","巴拿马","丹麦","韩国","塞尔维亚","巴林","哈萨克斯坦","捷克","阿尔及利亚","尼日利亚","挪威","阿曼","亚美尼亚","玻利维亚","马来西亚","摩洛哥","摩尔多瓦","加纳","澳大利亚","芬兰","喀麦隆","伊拉克","洪都拉斯","阿塞拜疆","苏丹","危地马拉","卢森堡","匈牙利","塔吉克斯坦","乌兹别克斯坦","几内亚","塞内加尔","泰国","希腊","吉布提","科特迪瓦","刚果（金）","保加利亚","波黑","加蓬","克罗地亚","萨尔瓦多","北马其顿","古巴","爱沙尼亚","冰岛","索马里","立陶宛","吉尔吉斯斯坦","斯洛伐克","新西兰","肯尼亚","斯洛文尼亚","斯里兰卡","马尔代夫","海地","委内瑞拉","几内亚比绍","黎巴嫩","马里","赞比亚","拉脱维亚","突尼斯","阿尔巴尼亚","赤道几内亚","哥斯达黎加","尼日尔","塞浦路斯","尼泊尔","巴拉圭","布基纳法索","乌拉圭","塞拉利昂","安道尔","尼加拉瓜","格鲁吉亚","埃塞俄比亚","约旦","乍得","钻石号邮轮","中非共和国","圣马力诺","马达加斯加","马耳他","刚果（布）","牙买加","坦桑尼亚","巴勒斯坦","多哥","佛得角","卢旺达","毛里求斯","越南","黑山","毛里塔尼亚","乌干达","斯威士兰","利比里亚","也门","莫桑比克","贝宁","缅甸","蒙古","文莱","圭亚那","津巴布韦","柬埔寨","叙利亚","特立尼达和多巴哥","马拉维","巴哈马","利比亚","摩纳哥","巴巴多斯","科摩罗","列支敦士登公国","安哥拉","布隆迪","厄立特里亚","马提尼克岛","博茨瓦纳","不丹","冈比亚","安提瓜和巴布达","东帝汶","格林纳达","纳米比亚","老挝","斐济","伯利兹","圣文森特和格林纳丁斯","圣卢西亚","多米尼克","圣基茨和尼维斯","梵蒂冈","苏里南","塞舌尔","巴布亚新几内亚","莱索托"]
print("请选择所想展示的国家（输入数字,空格间隔即可）：")
for i in range(len(country_list)):
    print(str(i)+"."+country_list[i])
n,n2=map(int,input().split())

shuju=['每日新增', '确诊病例', '治愈病例', '死亡人数']
print("请输入你想展现的数据（输入数字即可）：")
for i in range(len(shuju)):
    print(str(i)+'.'+shuju[i])
n_shuju=int(input())

param = {"country": country_list[n]}
```

```
res = requests.post(query_url, param)
d=json.loads(res.content.decode())
data=pd.DataFrame(d['data'])
xs=pd.DataFrame(d['data']).date

param2 = {"country": country_list[n2]}
res2 = requests.post(query_url, param2)
d2=json.loads(res2.content.decode())
data2=pd.DataFrame(d2['data'])
xs2=pd.DataFrame(d2['data']).date

for i in range(len(xs)):
    xs[i]="2020-"+xs[i][0]+xs[i][1]+'-'+xs[i][3]+xs[i][4]
xs = [datetime.strptime(d, '%Y-%m-%d').date() for d in xs]
for i in range(len(xs2)):
    xs2[i]="2020-"+xs2[i][0]+xs2[i][1]+'-'+xs2[i][3]+xs2[i][4]
xs2 = [datetime.strptime(d, '%Y-%m-%d').date() for d in xs2]
start=0
if len(xs)<len(xs2):
    startDate=xs[start]
    end=len(xs)-1
    endDate=xs[end]
else:
    startDate=xs2[start]
    end=len(xs2)-1
    endDate=xs2[end]

if n_shuju==0:
    ys=data.confirm_add
    ys2=data2.confirm_add
elif n_shuju==1:
    ys=data.confirm
    ys2=data2.confirm
elif n_shuju==2:
    ys=data.heal
    ys2=data2.heal
else:
    ys=data.dead
    ys2=data2.dead
y=list()
for i in range(start,end+1):
    y.append(ys[i])


y2=list()
```

```
for i in range(start,end+1):
    y2.append(ys2[i])

index = pd.date_range(startDate, endDate)
cols = ['value1']
cols2 = ['value2']
df = pd.DataFrame(y, index=index, columns=cols)
df2 = pd.DataFrame(y2, index=index, columns=cols2)

fig, axs = plt.subplots(1,1, figsize=(18,5))
index_hourly = pd.date_range(startDate, endDate, freq='1H')

df_smooth = df.reindex(index=index_hourly).interpolate('cubic')
df_smooth = df_smooth.rename(columns={"value1":"country-1"})

df_smooth2 = df2.reindex(index=index_hourly).interpolate('cubic')
df_smooth2 = df_smooth2.rename(columns={'value2':"country-2"})

df_smooth2.plot(ax=axs, alpha=100)
df2.plot(ax=axs, alpha=100)

df_smooth.plot(ax=axs, alpha=100)
df.plot(ax=axs, alpha=100)

plt.xlabel('date')
plt.ylabel('number')
plt.title("Added trend graph after smoothing the infected person data")
plt.show()
```

这部分就是通过 country_list 函数，让用户选择两个国家，同第一题类似，设置两条曲线进行对比。

2）保存爬取的数据:

源代码:

```
# coding: utf-8
import requests
import json
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from datetime import datetime
```

```
query_url='https://api.inews.qq.com/newsqa/v1/automation/foreign/daily/list'
country_list=["美国","巴西","俄罗斯","西班牙","英国","意大利","法国","德国","土耳其","印度","伊朗","秘鲁","加拿大","智利","沙特阿拉伯","墨西哥","巴基斯坦","比利时","卡塔尔","荷兰","白俄罗斯","孟加拉","厄瓜多尔","瑞典","新加坡","阿联酋","葡萄牙","瑞士","南非","爱尔兰","哥伦比亚","印度尼西亚","科威特","波兰","乌克兰","埃及","罗马尼亚","以色列","日本本土","奥地利","多米尼加","菲律宾","阿根廷","阿富汗","巴拿马","丹麦","韩国","塞尔维亚","巴林","哈萨克斯坦","捷克","阿尔及利亚","尼日利亚","挪威","阿曼","亚美尼亚","玻利维亚","马来西亚","摩洛哥","摩尔多瓦","加纳","澳大利亚","芬兰","喀麦隆","伊拉克","洪都拉斯","阿塞拜疆","苏丹","危地马拉","卢森堡","匈牙利","塔吉克斯坦","乌兹别克斯坦","几内亚","塞内加尔","泰国","希腊","吉布提","科特迪瓦","刚果（金）","保加利亚","波黑","加蓬","克罗地亚","萨尔瓦多","北马其顿","古巴","爱沙尼亚","冰岛","索马里","立陶宛","吉尔吉斯斯坦","斯洛伐克","新西兰","肯尼亚","斯洛文尼亚","斯里兰卡","马尔代夫","海地","委内瑞拉","几内亚比绍","黎巴嫩","马里","赞比亚","拉脱维亚","突尼斯","阿尔巴尼亚","赤道几内亚","哥斯达黎加","尼日尔","塞浦路斯","尼泊尔","巴拉圭","布基纳法索","乌拉圭","塞拉利昂","安道尔","尼加拉瓜","格鲁吉亚","埃塞俄比亚","约旦","乍得","钻石号邮轮","中非共和国","圣马力诺","马达加斯加","马耳他","刚果（布）","牙买加","坦桑尼亚","巴勒斯坦","多哥","佛得角","卢旺达","毛里求斯","越南","黑山","毛里塔尼亚","乌干达","斯威士兰","利比里亚","也门","莫桑比克","贝宁","缅甸","蒙古","文莱","圭亚那","津巴布韦","柬埔寨","叙利亚","特立尼达和多巴哥","马拉维","巴哈马","利比亚","摩纳哥","巴巴多斯","科摩罗","列支敦士登公国","安哥拉","布隆迪","厄立特里亚","马提尼克岛","博茨瓦纳","不丹","冈比亚","安提瓜和巴布达","东帝汶","格林纳达","纳米比亚","老挝","斐济","伯利兹","圣文森特和格林纳丁斯","圣卢西亚","多米尼克","圣基茨和尼维斯","梵蒂冈","苏里南","塞舌尔","巴布亚新几内亚","莱索托"]
n=int(input())
param = {"country": country_list[n]}
res = requests.post(query_url, param)
d=json.loads(res.content.decode())
data=pd.DataFrame(d['data'])
xs=pd.DataFrame(d['data']).date
for i in range(len(xs)):
    xs[i]="2020-"+xs[i][0]+xs[i][1]+'-'+xs[i][3]+xs[i][4]
xs = [datetime.strptime(d, '%Y-%m-%d').date() for d in xs]
data.insert(0,'city',xs)
data.to_csv("eguo.csv")
```

思路：

通过 country_list 函数让用户选择国家，再爬取数据之后先把日期处理一下，不然 excel 会当成 double 类型，最后利用 `to_csv` 函数输出到`.csv` 文件之中

3）在世界地图上描绘出大部分国家的疫情严重程度:

源代码:

```
# -*- coding: utf-8 -*-
import json
import requests
import jsonpath
from pyecharts.charts import Map,Geo
from pyecharts import options as opts
from pyecharts.globals import GeoType,RenderType

url = 'https://api.inews.qq.com/newsqa/v1/automation/foreign/country/ranklist'
resp = requests.post(url).text
data = json.loads(resp)
name = jsonpath.jsonpath(data, "$..name")
confirm = jsonpath.jsonpath(data, "$..confirm")
a = zip(name, confirm)
nameMap = {
    'Singapore Rep.': '新加坡',    'Dominican Rep.': '多米尼加
',    'Palestine': '巴勒斯坦',    'Bahamas': '巴哈马',
    'Timor-Leste': '东帝汶',    'Afghanistan': '阿富汗',    'Guinea-
Bissau': '几内亚比绍',    "Côte d'Ivoire": '科特迪瓦',
    'Siachen Glacier': '锡亚琴冰川',    "Br. Indian Ocean Ter.": '英属印
度洋领土',    'Angola': '安哥拉',    'Albania': '阿尔巴尼亚
',    'United Arab Emirates': '阿联酋',    'Argentina': '阿根廷',
    'Armenia': '亚美尼亚',    'French Southern and Antarctic Lands': '法
属南半球和南极领地',    'Australia': '澳大利亚',    'Austria': '奥地利
',    'Azerbaijan': '阿塞拜疆',    'Burundi': '布隆迪',
    'Belgium': '比利时',    'Benin': '贝宁',    'Burkina Faso': '布基纳法
索',    'Bangladesh': '孟加拉国',
    'Bulgaria': '保加利亚',    'The Bahamas': '巴哈马
',    'Bosnia and Herz.': '波斯尼亚和黑塞哥维那',
    'Belarus': '白俄罗斯',    'Belize': '伯利兹',    'Bermuda': '百慕大
',    'Bolivia': '玻利维亚',    'Brazil': '巴西',    'Brunei': '文莱
',    'Bhutan': '不丹',    'Botswana': '博茨瓦纳',
    'Central African Rep.': '中非共和国',    'Canada': '加拿大
',    'Switzerland': '瑞士',    'Chile': '智利',
    'China': '中国',    'Ivory Coast': '象牙海岸',    'Cameroon': '喀麦隆
',    'Dem. Rep. Congo': '刚果（金）',
    'Congo': '刚果（布）',    'Colombia': '哥伦比亚',    'Costa Rica': '
哥斯达黎加',    'Cuba': '古巴',    'N. Cyprus': '北塞浦路斯
',    'Cyprus': '塞浦路斯',
```

```
    'Czech Rep.': '捷克',    'Germany': '德国',    'Djibouti': '吉布提
',    'Denmark': '丹麦',    'Algeria': '阿尔及利亚',    'Ecuador': '厄瓜
多尔',    'Egypt': '埃及',
    'Eritrea': '厄立特里亚',    'Spain': '西班牙',    'Estonia': '爱沙尼亚
',
    'Ethiopia': '埃塞俄比亚',    'Finland': '芬兰',
    'Fiji': '斐',
    'Falkland Islands': '福克兰群岛',
    'France': '法国',
    'Gabon': '加蓬',
    'United Kingdom': '英国',
    'Georgia': '格鲁吉亚',
    'Ghana': '加纳',
    'Guinea': '几内亚',
    'Gambia': '冈比亚',
    'Guinea Bissau': '几内亚比绍',
    'Eq. Guinea': '赤道几内亚',
    'Greece': '希腊',
    'Greenland': '格陵兰',
    'Guatemala': '危地马拉',
    'French Guiana': '法属圭亚那',
    'Guyana': '圭亚那',
    'Honduras': '洪都拉斯',
    'Croatia': '克罗地亚',
    'Haiti': '海地',
    'Hungary': '匈牙利',
    'Indonesia': '印度尼西亚',
    'India': '印度',
    'Ireland': '爱尔兰',
    'Iran': '伊朗',
    'Iraq': '伊拉克',
    'Iceland': '冰岛',
    'Israel': '以色列',
    'Italy': '意大利',
    'Jamaica': '牙买加',
    'Jordan': '约旦',
    'Japan': '日本',
    'Japan': '日本本土',
    'Kazakhstan': '哈萨克斯坦',
    'Kenya': '肯尼亚',
    'Kyrgyzstan': '吉尔吉斯斯坦',
    'Cambodia': '柬埔寨',
    'Korea': '韩国',
    'Kosovo': '科索沃',
```

```
    'Kuwait': '科威特',
    'Lao PDR': '老挝',
    'Lebanon': '黎巴嫩',
    'Liberia': '利比里亚',
    'Libya': '利比亚',
    'Sri Lanka': '斯里兰卡',
    'Lesotho': '莱索托',
    'Lithuania': '立陶宛',
    'Luxembourg': '卢森堡',
    'Latvia': '拉脱维亚',
    'Morocco': '摩洛哥',
    'Moldova': '摩尔多瓦',
    'Madagascar': '马达加斯加',
    'Mexico': '墨西哥',
    'Macedonia': '马其顿',
    'Mali': '马里',
    'Myanmar': '缅甸',
    'Montenegro': '黑山',
    'Mongolia': '蒙古',
    'Mozambique': '莫桑比克',
    'Mauritania': '毛里塔尼亚',
    'Malawi': '马拉维',
    'Malaysia': '马来西亚',
    'Namibia': '纳米比亚',
    'New Caledonia': '新喀里多尼亚',
    'Niger': '尼日尔',
    'Nigeria': '尼日利亚',
    'Nicaragua': '尼加拉瓜',
    'Netherlands': '荷兰',
    'Norway': '挪威',
    'Nepal': '尼泊尔',
    'New Zealand': '新西兰',
    'Oman': '阿曼',
    'Pakistan': '巴基斯坦',
    'Panama': '巴拿马',
    'Peru': '秘鲁',
    'Philippines': '菲律宾',
    'Papua New Guinea': '巴布亚新几内亚',
    'Poland': '波兰',
    'Puerto Rico': '波多黎各',
    'Dem. Rep. Korea': '朝鲜',
    'Portugal': '葡萄牙',
    'Paraguay': '巴拉圭',
    'Qatar': '卡塔尔',
```

```
    'Romania': '罗马尼亚',
    'Russia': '俄罗斯',
    'Rwanda': '卢旺达',
    'W. Sahara': '西撒哈拉',
    'Saudi Arabia': '沙特阿拉伯',
    'Sudan': '苏丹',
    'S. Sudan': '南苏丹',
    'Senegal': '塞内加尔',
    'Solomon Is.': '所罗门群岛',
    'Sierra Leone': '塞拉利昂',
    'El Salvador': '萨尔瓦多',
    'Somaliland': '索马里兰',
    'Somalia': '索马里',
    'Serbia': '塞尔维亚',
    'Suriname': '苏里南',
    'Slovakia': '斯洛伐克',
    'Slovenia': '斯洛文尼亚',
    'Sweden': '瑞典',
    'Swaziland': '斯威士兰',
    'Syria': '叙利亚',
    'Chad': '乍得',
    'Togo': '多哥',
    'Thailand': '泰国',
    'Tajikistan': '塔吉克斯坦',
    'Turkmenistan': '土库曼斯坦',
    'East Timor': '东帝汶',
    'Trinidad and Tobago': '特里尼达和多巴哥',
    'Tunisia': '突尼斯',
    'Turkey': '土耳其',
    'Tanzania': '坦桑尼亚',
    'Uganda': '乌干达',
    'Ukraine': '乌克兰',
    'Uruguay': '乌拉圭',
    'United States': '美国',
    'Uzbekistan': '乌兹别克斯坦',
    'Venezuela': '委内瑞拉',
    'Vietnam': '越南',
    'Vanuatu': '瓦努阿图',
    'West Bank': '西岸',
    'Yemen': '也门',
    'South Africa': '南非',
    'Zambia': '赞比亚',
    'Zimbabwe': '津巴布韦'
}
```

```
map_ = Map(opts.InitOpts(width='1200px', height='600px')).add(series_na
me="世界各国确诊人数",
                 data_pair=a,
                 maptype="world",
                 name_map=nameMap,
                 is_map_symbol_show=False
                 )
map_.set_series_opts(label_opts=opts.LabelOpts(is_show=False))  # 显示国
家名称
map_.set_global_opts(title_opts=opts.TitleOpts(title="国外疫情情况"),
                     visualmap_opts=opts.VisualMapOpts(max_=1000000, is_
piecewise=True))
map_.render("国外疫情情况.html")
print("ok")
```

思路：

这里利用了 map 包，先通过 a = zip(name, confirm)创建元组，将国家与数据一一对应，接着用 Map()函数，设置一些基本参数以及 data_pair=a 将刚才的元组传入其中，最后展现出来。

以上创意部分，以及文本，ppt 的制作，由吕彦溥-18071122 和杨静-19035110 共同完成

碰到的问题以及解决思路:

1.处理平滑:

怎么平滑处理是第一个碰到的问题，因为一般来说，插值法 x 轴都是连续的数字，而这次却是日期类型，后来发现 pandas 内有 **pd.date_range** 函数，可以产生连续的日期，那样，用插值连续数字的方法，也可以插值日期了。

2.预测未来数据

怎么预测未来数据也是碰到的难题，后来查阅官网发现，**polyfit**，Fit a polynomial of degree *deg* to points *(x, y)*. Returns a vector of coefficients *p* that minimises the squared error in the order *deg*, *deg-1*, … *0*.`p(x) = p[0] * x**deg + ... + p[deg]。`

这个函数可以拟合多项式，并且将现有疫情数据分 80%为实验集，20%为检测集，发现通过前 80%预测后 20%，准确率不低，故采用此方法。