HW1_sol

May 1, 2020

[1]:
```
import requests

url = "https://covid-19-statistics.p.rapidapi.com/reports"

headers = {
    'x-rapidapi-host': "covid-19-statistics.p.rapidapi.com",
    'x-rapidapi-key': "5490de1e46mshe03fcf8c16e772fp1dfbd6jsn5b14620daf5f"
    }
```

0.0.1 P1 (2 pt). Extract the death count from "response.text" using json.

[2]:
```
querystring = {"region_province":"Beijing","iso":"CHN","date":"2020-04-14"}
response = requests.request("GET", url, headers=headers, params=querystring)


## Put your code here, and put the result in the following variable.
import json
death_count_beijing = json.loads(response.text)['data'][0]['deaths']
print("Q1:", death_count_beijing)
```

```
Q1: 8
```

0.0.2 P2 (2 pt). Based on the code above, write a function that will return the death count of a region on a specific date.

Hint: in the case that the country, region or date are unavailable, the function should return a death count as 0.

[3]:
```
def get_death_count(country, region, date):
  ## country is in the ISO format like "USA", region can be the name of the␣
 ↪state
  ## it should return one integer
  ## Put your code here to implement this function
  #return 0
  querystring = {"region_province":region,"iso":country,"date":date}
  response = requests.request("GET", url, headers=headers, params=querystring)
  if(len(json.loads(response.text)['data']) == 0):
    return 0
```

```
  else:
    return json.loads(response.text)['data'][0]['deaths']

## Here are some correct output examples. If you code is correct, the assertion␣
 ↪should not return error.
assert(get_death_count("USA", "Illinois", "2020-04-07") == 308)
assert(get_death_count("USA", "Illinois", "2019-04-07") == 0)
print("Q2-a:", get_death_count("USA", "Alabama", "2020-04-10"))
print("Q2-b:", get_death_count("USA", "New York", "2020-04-07"))
print("Q2-c:", get_death_count("USA", "California", "2020-04-04"))
```

```
Q2-a: 80
Q2-b: 5489
Q2-c: 289
```

0.0.3 P3 (1 pt). Obtain the list of death counts from 2020-01-01 to 2020-04-14 for New York.

[4]:
```
# The following function is help you solve this problem
from datetime import date, timedelta
def get_all_dates():
  sdate = date(2020, 3, 1)
  edate = date(2020, 4, 15)
  delta = edate - sdate
  return [(sdate + timedelta(days=i)).strftime("%Y-%m-%d") for i in range(delta.
 ↪days + 1)]

## Put your code here and your answer in the following varaible.
result_list = [get_death_count("USA", "New York", d) for d in get_all_dates()]
print("Q3:", result_list)
```

```
Q3: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 10, 13, 16, 34, 42, 60, 117,
158, 210, 285, 385, 527, 728, 965, 1218, 1550, 1941, 2373, 2935, 3565, 4159,
4698, 5489, 6268, 7067, 7867, 8627, 9385, 10058, 10842, 11617]
```

0.0.4 P4 (2 pt). Starting from the first day when the death count reached 10, count the number of days it takes for the death count to be doubled. Print a list of 7 integers for the next 7 doublings. What can you conclude from these 7 numbers?

[5]:
```
result_list2=[]
count = 10
index = 0
interval = 0;
for i in range(len(result_list)):
  index +=1
  if result_list[i] >= count:
```

```
    break

for i in range(index, len(result_list)):
  interval +=1
  if result_list[i] >= 2*count:
    result_list2.append(interval)
    interval = 0
    count = result_list[i]

  if(len(result_list2) == 7):
    break

assert(len(result_list2) == 7)
# The first three values are given
assert(result_list2[0] == 3) # 10 -> 13 -> 16 -> 34
assert(result_list2[1] == 3) # 34 -> 42 -> 60 -> 117
assert(result_list2[2] == 3) # 117 -> 158 -> 210 -> 285

## Put your code here and your answer in the following varaible. Also print␣
 ↪your findings below.
print("Q4-a:", result_list2)
print("Q4-b:", "The number of deaths increased exponentially at first and is␣
 ↪slowing down later.")
```

```
Q4-a: [3, 3, 3, 3, 3, 4, 6]
Q4-b: The number of deaths increased exponentially at first and is slowing down
later.
```

0.0.5 P5 (3 pt) Repeat the above process (P2 to P3), but this time for confirmed case across the US.

Hint1: You can a query string like querystring = {“iso”:“USA”, “date”:“2020-xx-xx”} to obtain the case report across US.

Hint2: You may need to sum up all confirmed cases.

[6]:
```
def get_confirmed_count(country, date):
  ## country is in the ISO format like "USA", region can be the name of the␣
 ↪state
  ## it should return one integer
  ## Your code here to implement this function
  #return 0
  querystring = {"iso":country,"date":date}
  response = requests.request("GET", url, headers=headers, params=querystring)
  if(len(json.loads(response.text)['data']) == 0):
    return 0
  else:
    j = json.loads(response.text)['data']
```

```
    return sum([state['confirmed'] for state in j])

result_list3 = [get_confirmed_count("USA", d) for d in get_all_dates()]
print("Q5:", result_list3)
```

```
Q5: [76, 101, 122, 153, 221, 278, 417, 537, 605, 959, 1281, 1663, 2179, 2726,
3499, 4632, 6421, 7786, 13680, 19101, 25493, 33848, 43663, 53736, 65778, 83836,
101657, 121465, 140909, 161831, 188172, 213372, 243599, 275586, 308853, 337072,
366667, 396223, 429052, 461437, 496535, 526396, 555313, 580619, 607670, 636350]
```