Sherine_Homework6

October 12, 2023

1 Homework 6

1.1 Question 1 (40 points)

You might find this a bit challenging at first, but trust me: with what you learned so far, you got this!

Re-implement the `ip_address2country_code` function of Homework 4 in Python. Show that your implementation works by executing your function using the following list of IP addresses as input:

```
[
    "171.182.200.160",
    "203.45.67.89",
    "10.0.0.23",
    "172.16.32.55",
    "67.195.44.68"
]
```

Hints: - You will need the `requests` library, and in particular you will need to use the `requests.post` function. - You can use a list comprehension to create the request body (which you will pass to the `json` argument of the `request.post` function). According to the ip-api API documentation, it is acceptable to format the request body as a list of dictionaries:

```
[
    {"query": "171.182.200.160"},
    {"query": "203.45.67.89"},
    ...
]
```

- You can parse the body of the response using the `json` method e.g.,

  ```
  response = requests.post(...)
  response.json() # <- this is a list of dictionaries, with a dictionary for each input ip a
  ```

- Finally, you can use another list comprehension on the output of `response.json()` to create the output list of country codes, but be careful handling cases where the `countryCode` key is not available (e.g., because the IP address is private). You might find it convenient to use the `get` method of Python dictionaries, which allows to extract the value associated with a key when the key exists and return `None` when the key does not exist.

[1]:
```
import requests

def ip_address2country_code(ip_addresses):
    ip_api_url = "http://ip-api.com/batch/"
    request_body = [{"query": ip} for ip in ip_addresses]
    response = requests.post(url=ip_api_url, json=request_body)
    if response.status_code == 200:
        response_data = response.json()
        country_codes = [entry.get("countryCode", None) for entry in␣
 ↪response_data]
        return country_codes
    else:
        return None
ip_addresses = [
    "171.182.200.160",
    "203.45.67.89",
    "10.0.0.23",
    "172.16.32.55",
    "67.195.44.68"
]
country_codes = ip_address2country_code(ip_addresses)
print(country_codes)
```

```
['GB', 'AU', None, None, 'US']
```

1.2 Question 2 (30 points)

Use the following code to read the `log.txt` file into a list, such that each each line of the log file is an element of the list:

```
with open("./data/log.txt") as log_file:
    log = log_file.readlines()
```

Using what you know about for loops and regular expressions, iterate over the log line by line and extract into a list all IP addresses that appear in the log.

Hints:

- For this exercise, you can assume that there is at most 1 IP address on each line.
- There are in total 20 IP addresses (some appearing more than once possibly) in the log. Your final list should have length 20.
- You might find the `append` or `extend` methods of Python `lists` useful in this exercise.

[3]:
```
import re
ip_addresses = []
with open(r"log.txt") as log_file:
    log = log_file.readlines()
ip_pattern = r"\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b"
for line in log:
```

```
    # Use regular expression search to find IP addresses in the line
    matches = re.findall(ip_pattern, line)

    # If any matches are found, add them to the ip_addresses list
    if matches:
        ip_addresses.extend(matches)

# Print the list of extracted IP addresses
print(ip_addresses)
```

```
['129.1.1.1', '9.37.65.139', '9.67.100.1', '9.67.101.1', '9.67.116.98',
'9.67.117.98', '127.0.0.1', '129.1.1.1', '9.37.65.139', '9.67.100.1',
'9.67.101.1', '9.67.116.98', '9.67.117.98', '127.0.0.1', '9.67.116.99',
'9.67.116.98', '9.67.116.98', '0.0.0.0', '9.67.116.98', '9.67.116.99']
```

1.3 Question 3 (30 points)

Solve again the Exercise 1 of Lab 6, this time using Python.

Hints:

- Read the dataset into a `pandas.DataFrame` and name the data frame `purchases`.
- Write a function that takes an email as input (e.g., `"john.doe@gmail.com"` and returns the masked email as output (e.g., `"jxxxxxxe@gmail.com"`). You may find the `split` and `join` methods of Python strings handy.
- Use the `map` method of `purchases["email"]` (which is an example of a `pandas.Series`) to apply your function to all elements of that column.

[4]:
```
import pandas as pd
data = {
    'email': [
        'john.doe@gmail.com',
        'emma.lee@yahoo.com',
        'james@gmail.com',
        'mary@fancyuni.edu',
        'roger.mcguire@outlook.com'
    ],
    'item': [
        'vacuum cleaner',
        'standing desk',
        '$25 amazon gift card',
        'chewing gum',
        'pencils'
    ],
    'n_purchases': [1, 1, 3, 5, 10]
}
purchases = pd.DataFrame(data)
def mask_email(email):
```

```
    parts = email.split('@')
    if len(parts) != 2:
        return email
    username, domain = parts
    masked_username = username[0] + 'x' * (len(username) - 2) + username[-1]
    return masked_username + '@' + domain
purchases["email"] = purchases["email"].map(mask_email)
print(purchases)
```

```
                      email                  item  n_purchases
0         jxxxxxxe@gmail.com        vacuum cleaner            1
1         exxxxxxe@yahoo.com         standing desk            1
2            jxxxs@gmail.com  $25 amazon gift card            3
3          mxxy@fancyuni.edu           chewing gum            5
4  rxxxxxxxxxxxe@outlook.com               pencils           10
```

[ ]: