For this section, you may find it helpful to note that " " is represented by "%20"

1a.

Sign up for an API key for a news site  
 e.g. The Guardian <https://open-platform.theguardian.com/access/>

(Docs: <https://open-platform.theguardian.com/documentation/search>)

You can test that your api key works using this snippet:   
 query\_params = {'api-key': 'your-api-key-here'}   
 response = requests.get("<https://content.guardianapis.com/search>", params=query\_params)  
 print(response.status\_code)

1b. Find a url to any article about clouds.

1c. Find a url to any article 5 years or older about bitcoin.

1d. Count the number of articles written this year about 3D printing (hint: use the tag "technology/3d-printing" for The Guardian).

1e. Find the first article written in the science section about the landing of the Mars Rover Perseverance. The landing was on 18 Feb 2021. Don't try too hard, the search function isn't the best.

For this section, you may find it helpful to also use this page to help you find the name of the metric you are looking for: <https://coronavirus.data.gov.uk/details/download>

2a. Use the UK government's COVID API to create a list of daily cases in the UK, with each element in the list representing one day's cases.

2b. Use list slicing on the above list to create a new list of the number of cases for the last 7 days, and print the 7 day average to the nearest whole number. You may find the inbuilt functions sum() and round() helpful.

2c. Plot the number of cases for the whole UK on a graph.You can use this code snippet as a reference:

import matplotlib.pyplot as plt  
 import matplotlib.dates cases = ["2", "3", "2", "4"]  
 dates = ["2020-02-17", "2020-02-18", "2020-02-19", "2020-02-20"]  
 converted\_dates = matplotlib.dates.datestr2num(dates)  
 plt.plot\_date(converted\_dates, cases, '-', color='red')  
 plt.ylabel('Daily Cases')  
 plt.xlabel('Date')  
 plt.show()

2d. Plot the number of cases for your local area.

2e. Plot the number of cases for each nation (England, Scotland, Wales, Northern Ireland) on a graph. hint: areaName for Northern Ireland is "northern%20ireland", " " is represented by "%20"

2f. Write a csv with the number of daily cases for your lcoal area with columns for dates and cases.

3. Create small shopping list program, reading and writing to a file path hardcoded into your script to store the contents of the shopping list between running the program.

Usage:The program will wait for user input  
Type in items on your shopping list, separated by commas between items.  
After you hit Enter, the contents of your current shopping list will be shown.  
e.g butter, onions, beetroot

My Names's Shopping List:beetrootonionsbuttercookies

If an item is already on the list, do not add the item to the list again.

If no input is provided, display the current shopping list

Update the list(Extension) if the keyword "reset" is used as an input to your shopping list, remove all the current items in the shopping list(Extension)

if the keyword "remove" is used before an item, remove that item from the shopping list

After you've got working code, try to go back and refactor it to make sure it's easy to read and look for areas of improvement.

Suggested Order:

Get input from the user using input(), splitting the input string by ",".

Print this list.  
Read from a file and create a list from the content, splitting by either "/n" or ",".

Print the new list.  
Combine both lists and print this after getting the input.  
Create a set from both lists, to remove duplicates. Print the set.  
Either append the new list to the existing file or overwrite the file with the combined list. Think about why you might choose to do one over the other, but either will work here.  
Format the output from your script so that list is presented in a format that is easy to read.  
(Extension 1)Add a check if 'reset' was in the input string. If so, write a new file to the same file path.

(Extension 2)  
For each item in the new list that contains the string "remove", put them a new list with that item, but remove the string "remove".

#Ans Q1  
#Part A  
query\_params = {'api-key': 'c5fc13e3-93be-4025-88f7-1492ee1dae50', 'q':'clouds'}  
response = requests.get("<https://content.guardianapis.com/search>", params=query\_params)  
print(response.json()['response']['results'][0]['webUrl'])#Part B  
query\_params = {'api-key': 'c5fc13e3-93be-4025-88f7-1492ee1dae50', 'q':'bitcoin', 'to-date':'2016-04-01'}  
response = requests.get("<https://content.guardianapis.com/search>", params=query\_params)  
print(response.json()['response']['results'][0]['webUrl'])#Part C  
query\_params = {'api-key': 'c5fc13e3-93be-4025-88f7-1492ee1dae50', 'tag':'technology/3d-printing', 'from-date':'2021-01-01'}  
response = requests.get("<https://content.guardianapis.com/search>", params=query\_params)  
print(response.json()["response"]["total"])#Part D  
query\_params = {'api-key': 'c5fc13e3-93be-4025-88f7-1492ee1dae50', 'q':'perseverance', 'order-by':'oldest', 'from-date':'2021-02-18', 'to-date':'2021-02-20', 'section':'science'}  
response = requests.get("<https://content.guardianapis.com/search>", params=query\_params)  
print(response.json()["response"]["results"][1]["webUrl"])

#Ans Q2  
#Part A  
uk\_endpoint = '[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=overview&structure={](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=overview&structure=%7b)"date":"date","newCases":"newCasesByPublishDate"}'  
r = requests.get(uk\_endpoint)  
response = r.json()uk\_cases = []  
for day in response["data"]:  
 uk\_cases.append((day["newCases"]))# You could also use a list comprehension to create a list   
uk\_cases = [day["newCases"] for day in response["data"]]print(len(uk\_cases))#Part B  
uk\_endpoint = '[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=overview&structure={](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=overview&structure=%7b)"date":"date","newCases":"newCasesByPublishDate"}'  
r = requests.get(uk\_endpoint)  
response = r.json()  
uk\_cases = [day["newCases"] for day in response["data"]]  
uk\_dates = [day["date"] for day in response["data"]]  
uk\_7\_days = uk\_cases[:6]  
print(round(sum(uk\_7\_days)/7))#Part C  
uk\_endpoint = '[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=overview&structure={](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=overview&structure=%7b)"date":"date","newCases":"newCasesByPublishDate"}'  
r = requests.get(uk\_endpoint)  
response = r.json()  
uk\_cases = [day["newCases"] for day in response["data"]]  
uk\_dates = [day["date"] for day in response["data"]]uk\_dates = matplotlib.dates.datestr2num(dates)  
plt.plot\_date(uk\_dates, uk\_cases, '-', color='blue')  
plt.ylabel('Daily Cases')  
plt.xlabel('Date')  
plt.show()#Part E  
england\_endpoint = '[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=england&structure={](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=england&structure=%7b)"date":"date","newCases":"newCasesByPublishDate"}'  
r = requests.get(england\_endpoint)  
response = r.json()  
england\_cases = []  
england\_dates = []  
for day in response["data"]:  
 england\_cases.append((day["newCases"]))  
 england\_dates.append(day["date"])england\_dates = matplotlib.dates.datestr2num(england\_dates)  
plt.plot\_date(england\_dates, england\_cases, '-', color='red')wales\_endpoint = '[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=wales&structure={](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=wales&structure=%7b)"date":"date","newCases":"newCasesByPublishDate"}'  
r = requests.get(wales\_endpoint)  
response = r.json()  
wales\_cases = [day["newCases"] for day in response["data"]]  
wales\_dates = [day["date"] for day in response["data"]]  
wales\_dates = matplotlib.dates.datestr2num(wales\_dates)  
plt.plot\_date(wales\_dates, wales\_cases, '-', color='orange')scotland\_endpoint = '[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=scotland&structure={](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=scotland&structure=%7b)"date":"date","newCases":"newCasesByPublishDate"}'  
r = requests.get(scotland\_endpoint)  
response = r.json()  
scotland\_cases = [day["newCases"] for day in response["data"]]  
scotland\_dates = [day["date"] for day in response["data"]]  
scotland\_dates = matplotlib.dates.datestr2num(scotland\_dates)  
plt.plot\_date(scotland\_dates, scotland\_cases, '-', color='blue')ni\_endpoint = '[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=Northern%20Ireland&structure={](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=nation;areaName=Northern%20Ireland&structure=%7b)"date":"date","newCases":"newCasesByPublishDate"}'  
r = requests.get(ni\_endpoint)  
response = r.json()  
ni\_cases = [day["newCases"] for day in response["data"]]  
ni\_dates = [day["date"] for day in response["data"]]  
ni\_dates = matplotlib.dates.datestr2num(ni\_dates)  
plt.plot\_date(ni\_dates, ni\_cases, '-', color='green')#You could also make Part D more efficient, like this   
areas = {"england":"red", "scotland":"blue", "wales":"orange", "northern%20ireland":"green"}import matplotlib.pyplot as plt  
import matplotlib.datesdef endpoint\_builder(areaType, areaName, metric):  
 structure = '{"date":"date", "newCases":"'+ metric + '"}'  
 return f"[https://api.coronavirus.data.gov.uk/v1/data?filters=areaType={areaType};areaName={areaName}&structure={structure}](https://api.coronavirus.data.gov.uk/v1/data?filters=areaType=%7bareaType%7d;areaName=%7bareaName%7d&structure=%7bstructure%7d)"for nation in areas.keys():  
 r = requests.get(endpoint\_builder("nation", nation, "newCasesByPublishDate"))  
 response = r.json()  
 cases = [day["newCases"] for day in response["data"]]  
 dates = [day["date"] for day in response["data"]]  
 dates = matplotlib.dates.datestr2num(dates)  
 plt.plot\_date(dates, cases, '-', color=areas[nation])plt.ylabel('Daily Cases')  
plt.xlabel('Date')  
plt.show()

3.

def read\_list\_from\_file(path):  
 shopping\_list = []  
 with open(path, "r") as file:  
 for item in file.readlines():  
 shopping\_list.append(item.strip())  
 return shopping\_listdef write\_to\_file\_from\_list(path, list):  
 with open(path, "w") as file:  
 for item in list:  
 file.write(f"{item}\n")def reset\_file(path):  
 with open(path, "w") as file:  
 pass  
 print("Resetting shopping list")def check\_for\_items\_to\_remove(existing\_items\_list, new\_items\_list):  
 for item in new\_items:  
 if 'remove' in item:  
 original\_item = item.replace("remove"," ").strip()  
 existing\_items\_list.remove(original\_item)  
 new\_items\_list.remove(item)def show\_shopping\_list(shopping\_items):  
 print("Janki's Shopping List:\n{0}".format('\n'.join((shopping\_items))))shopping\_list\_path = "C:\\Users\\Janki\\Desktop\\shopping\_list.txt"  
existing\_items = read\_list\_from\_file(shopping\_list\_path)  
show\_shopping\_list(existing\_items)new\_items\_input = input("Would you like to add something to the list?")  
if "reset" in new\_items\_input:  
 new\_items\_input = new\_items\_input.replace("reset", "")  
 reset\_file(shopping\_list\_path)  
 existing\_items = []new\_items = new\_items\_input.split(",")  
#Check for remove items command  
check\_for\_items\_to\_remove(existing\_items, new\_items)  
#Remove Duplicate Items  
existing\_items = set(existing\_items + new\_items)write\_to\_file\_from\_list(shopping\_list\_path, existing\_items)  
show\_shopping\_list(existing\_items)