1. **Best Time to Buy and Sell Stock**

You are given an array prices where prices[i] is the price of a given stock on the ith day.
You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.
Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.
Example 1:
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
Example 2:
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.
Constraints:
1 <= prices.length <= 105
0 <= prices[i] <= 104

2. There are N cars in line at the gas station. There are three tankers at the station, labeled X, Y, and Z. Each fuel dispenser contains a certain amount of fuel. At any time, the amount of available fuel is clearly displayed on each fuel dispenser.
When a car reaches the front of the queue, the driver can choose to drive to any distributor that is not occupied by another car. Suppose the fuel demand of this car is D litres. The driver must select a fuel dispenser with a fuel capacity of at least D liters. If the capacity of all unused dispensers is less than D liters, the driver must wait for other cars to finish refueling. If all the dispensers are not occupied and none are at least D litres, then the driver will not be able to refuel the car and will block the queue indefinitely. If there is at least D litre in more than one dispenser, the driver will choose the one with the smallest letter.
Every driver must wait for a while before starting to refuel the car. Calculate the longest waiting time between all drivers. Suppose it takes one second to refuel one liter of fuel, and the car in motion is instantaneous.

Write a function:
public static Integer solution(int[] A, int X, int Y, int Z)
X, Y, Z represent 3 fuel dispensers.
The position inside the array A represents the vehicle, and the element represents the amount of fuel required by the vehicle. Return the maximum waiting time of the car. If there is a car that cannot be refueled, the function returns -1.
If 1: X=7, Y=11, Z=3.
A[0]=2
A[1]=8

A[2]=4
A[3]=3
A[4]=2
The time that the car waits in the queue is 0, 0, 2, 2, and 8 seconds. The function returns 8.
If 2: X=4, Y=0, Z=3.
A[0]=5
The function returns -1.

Solution:
https://www.programmersought.com/article/82496639663/

3. Write a function:

def solution(A)

that, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

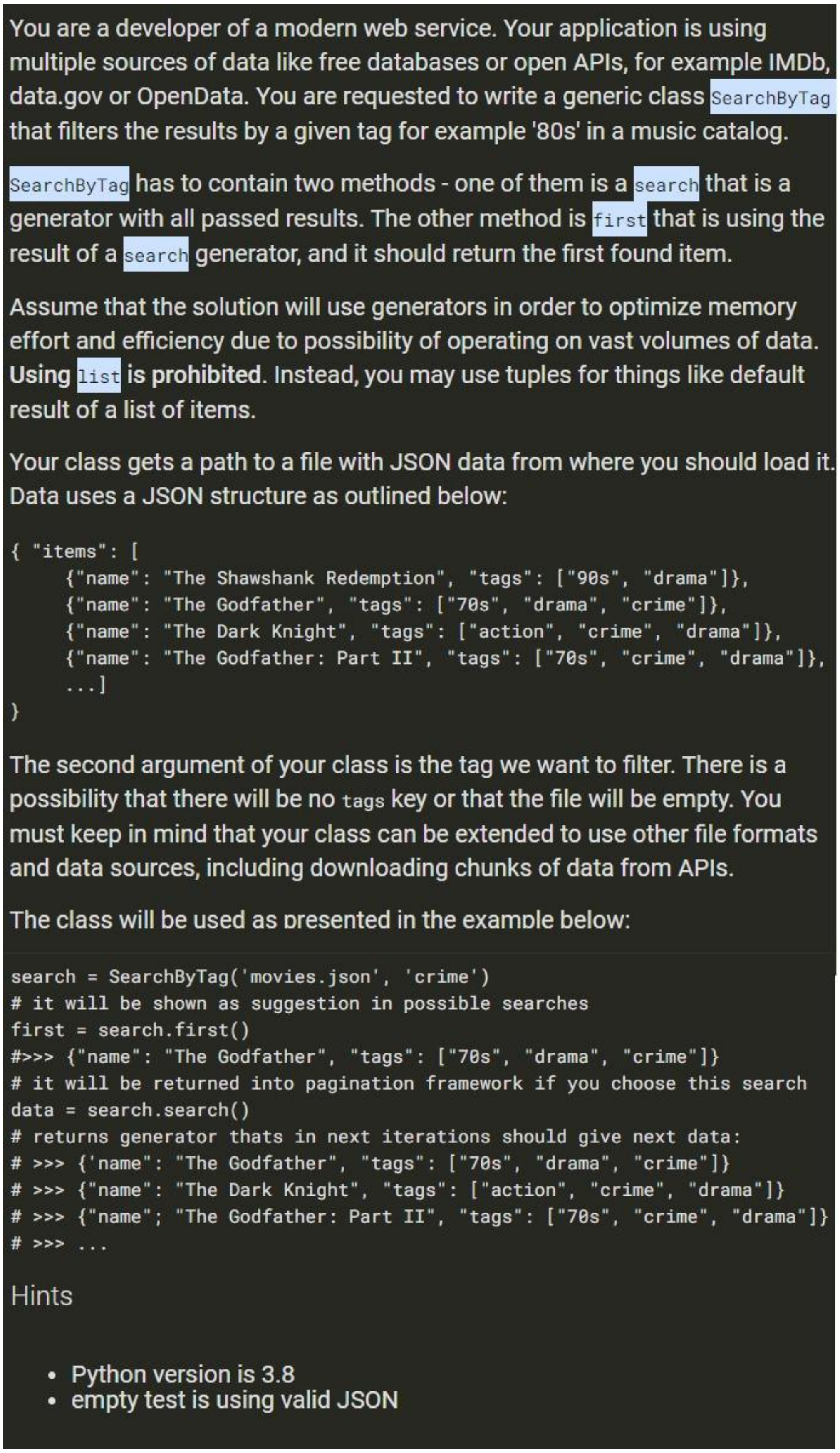
For example, given A = [1, 3, 6, 4, 1, 2], the function should return 5.

Given A = [1, 2, 3], the function should return 4. Given A = [−1, −3], the function should return 1.

Assume that:

N is an integer within the range [1..100,000]; each element of array A is an integer within the range [−1,000,000..1,000,000]. Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N) (not counting the storage required for input arguments).



You are a developer of a modern web service. Your application is using multiple sources of data like free databases or open APIs, for example IMDb, data.gov or OpenData. You are requested to write a generic class `SearchByTag` that filters the results by a given tag for example '80s' in a music catalog.

`SearchByTag` has to contain two methods - one of them is a `search` that is a generator with all passed results. The other method is `first` that is using the result of a `search` generator, and it should return the first found item.

Assume that the solution will use generators in order to optimize memory effort and efficiency due to possibility of operating on vast volumes of data. **Using `list` is prohibited**. Instead, you may use tuples for things like default result of a list of items.

Your class gets a path to a file with JSON data from where you should load it. Data uses a JSON structure as outlined below:

```
{ "items": [
     {"name": "The Shawshank Redemption", "tags": ["90s", "drama"]},
     {"name": "The Godfather", "tags": ["70s", "drama", "crime"]},
     {"name": "The Dark Knight", "tags": ["action", "crime", "drama"]},
     {"name": "The Godfather: Part II", "tags": ["70s", "crime", "drama"]},
     ...]
}
```

The second argument of your class is the tag we want to filter. There is a possibility that there will be no `tags` key or that the file will be empty. You must keep in mind that your class can be extended to use other file formats and data sources, including downloading chunks of data from APIs.

The class will be used as presented in the example below:

```
search = SearchByTag('movies.json', 'crime')
# it will be shown as suggestion in possible searches
first = search.first()
#>>> {"name": "The Godfather", "tags": ["70s", "drama", "crime"]}
# it will be returned into pagination framework if you choose this search
data = search.search()
# returns generator thats in next iterations should give next data:
# >>> {'name": "The Godfather", "tags": ["70s", "drama", "crime"]}
# >>> {"name": "The Dark Knight", "tags": ["action", "crime", "drama"]}
# >>> {"name"; "The Godfather: Part II", "tags": ["70s", "crime", "drama"]}
# >>> ...
```

Hints

- Python version is 3.8
- empty test is using valid JSON