Midterm 1 Questions

Submission conditions:

1- At the end of the documents, I have shared **FAQ** coming from previous semesters. Before you ask for more clarification, check the FAQ first, and if the answer is not there, ask me by email.

2- Please consider that the answers are time-bounded. Please upload your answers to the designated link in the blackboard before the due date.

3- Make sure your code runs smoothly. If your code does not run for simple syntax errors, you will lose the entire mark of that question

4- Make sure you check the inputs and validate them. I check your code with different inputs. In other words, I try to break your code. Your code must be resistant to different kinds of user inputs. Unless I have mentioned something is out of scope, you assume all the possible inputs.

5- If you do not follow the above instructions, you lose marks.

****

You already have seen the Binary search algorithm on the 10 practical questions. You see how we can bisect/ eliminate half of the items in the problem domain (in a list or any iterable object) in each iteration and come up with a shorter list to explore. This is the main idea of binary search:

Eliminating part of the problem space and narrowing it down to a smaller space to find the answer faster.

In question 1, below, I want you to have this matter in your mind. You must not forget that the binary search works on sorted lists.

Question 1)

Write a code that finds the elements of a sorted list that its **index** is equal to the **item** in that index. For example if the list is **lst= [-2,0,2,3,6,7,9]** then code has to return 2 and 3 since lst[**2**]=**2** and lst[3]=3

Hint1: do you think the above question has any relation to the binary search? Think about it. It is a sorted list, so it meets the first condition. Do you want to search for all the items on the list? What if the list has 1 million elements? Do you want to test 1 million items to see if **lst[i]=i**? (imagine the last item is the answer and you have to go through the entire list to get to the last item)

Can you use this fact that if **lst[j]>j** then no entry after j can satisfy the given criterion? This is because each element in the list is at least 1 greater than the previous element. For the same reason if **lst[j]<j** no entry before j can satisfy the given criterion.

The above observations can be used directly to create a binary search to find **lst[i]=i**

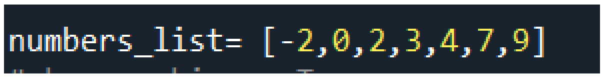
You can eliminate your search space with the two above conditions.

Hint2: you can also consider that if **lst[i]=i** then **lst[i]-i=o**. You can use this hint in your binary search too.

Now solve the above question by employing hint 1 or by hint 2.

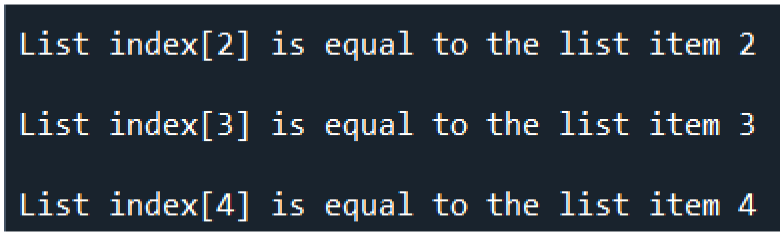
Sample output:

For a list like:



```
numbers_list= [-2,0,2,3,4,7,9]
```

We get an output like the below:



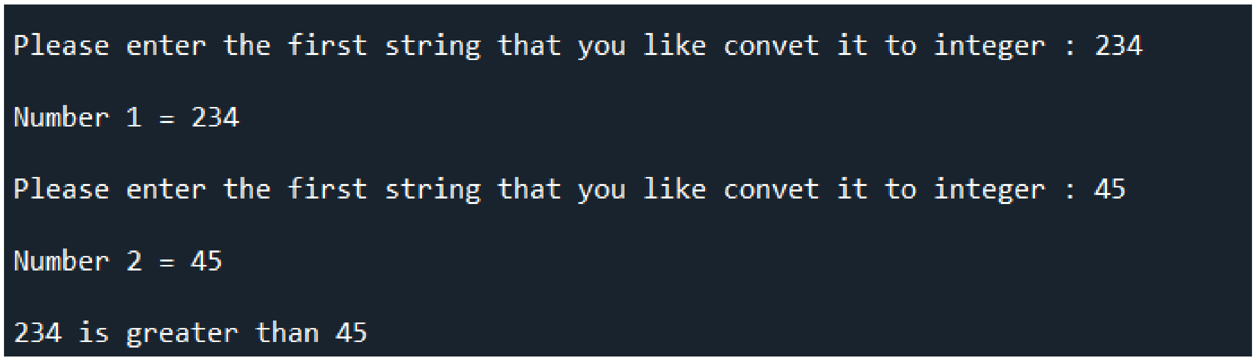
```
List index[2] is equal to the list item 2

List index[3] is equal to the list item 3

List index[4] is equal to the list item 4
```

Question 2) Let’s say I give you two strings that contain digits like “234” and “45”. How do you compare the equivalent numeric values of those two strings **without converting them to an integer by the int() method**? For example, 234>45 in the above case.

If your idea is to compare the very left positions, then you must be careful since 10>2 even though **1<2**

Make sure your code does not break if the string is empty. *Assume we do not give a negative number or non-numeric characters to the program, so you do not need to check them*. However, they can be equal to each other or equal to zero. Print the appropriate message to use if they are equal.

Sample outputs:



```
Please enter the first string that you like convet it to integer : 234

Number 1 = 234

Please enter the first string that you like convet it to integer : 45

Number 2 = 45

234 is greater than 45
```

Question 3) Let’s say we want to check if the sum of two numbers in a list is equal to 10. The list can be very big. <u>We do not want to iterate through the list more than once</u>. Do not use the brute force method.

Hint: let's say our list is [3,4,1,2,9]. In a brute force approach, you can check the first item (i.e. 3) with all the other items in the list and if that is 10, print those two items. Otherwise, try the second item and compare it with the third, fourth, and so on. You have to do this comparison for all the items as it is possible no two items in the list add up to 10. The problem is if the list has **n** list, you check each item with the other **n-1** items in each iteration. Since you have n items, in total you perform n(n-1) comparisons (since you pick one of the n numbers and compare that with the remaining (n-1) numbers). But the question asks you **not** to iterate through the list more than once (n). How can we do that?

You should use a more innovative solution. For example, you can use a dictionary where the keys are the list items and the value is 1 if (value=10-item) does not exist. For example, for a list like this:

[3,4,1,2,9]

The 10-3(first value in the list)=7. We do not have 7 in the dictionary yet (we just started to populate the dictionary). So we insert the 3 in the dictionary and put 1 in its value

Key	value
3	1

Then we calculate 10-4(second value in the list)=6. We have just 3 in the dictionary, so we go ahead and add that to the dictionary too:

Key	value
3	1
4	1

Then 10-1=9. We have 3 and 4 in the dictionary. So, let's add that to the dictionary:

Key	value
3	1
4	1
1	1

The next one is 10-2=8

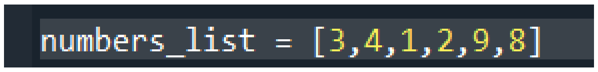
Key	value
3	1
4	1
1	1
2	1

Now we come to the last item: 10-9=1. This time we can see that we already have **1** in the dictionary (see the item in the last but one position. So, we found the pair (9 and 1)

The beauty of this solution is we did not iterate the list more than once.

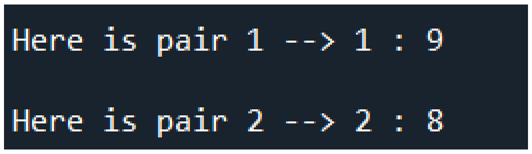
Now you implement this solution.

Sample output for a list like below:



```
numbers_list = [3,4,1,2,9,8]
```

Is:



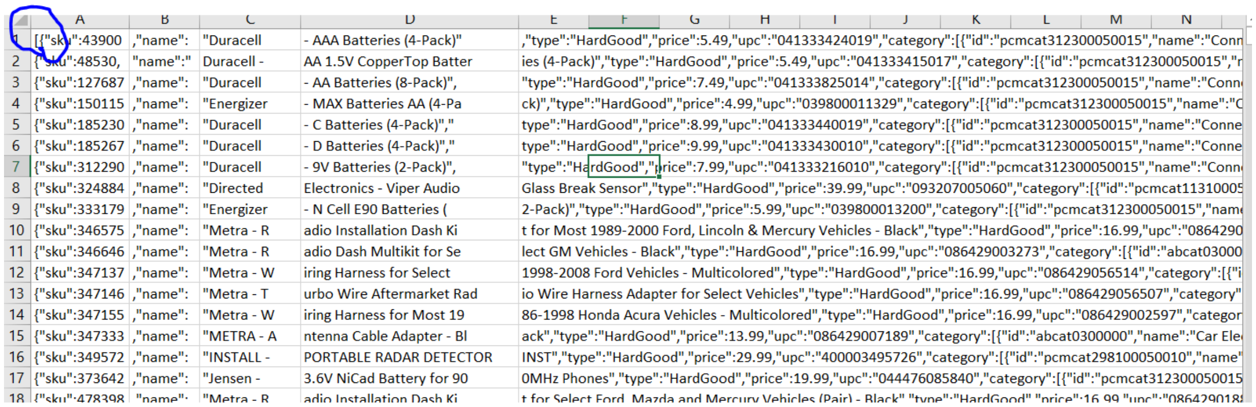
```
Here is pair 1 --> 1 : 9

Here is pair 2 --> 2 : 8
```

Question 4)

I have shown you how to make a dictionary suggestion in the Day 07 notes. Now I want you to use that knowledge in answering the following question:

We want to implement an autocomplete feature that provides suggestions as a user types a phrase in the search field. The autocomplete is like the Google search suggests a feature. When you start searching for a keyword in the Google search, it suggests a few items to you to make life easier for you. The list we have in mind currently has a product catalog of around 3 million objects and only wants to autocomplete the **product name**. Use the **products.json** in the following link: https://github.com/BestBuyAPIs/open-data-set

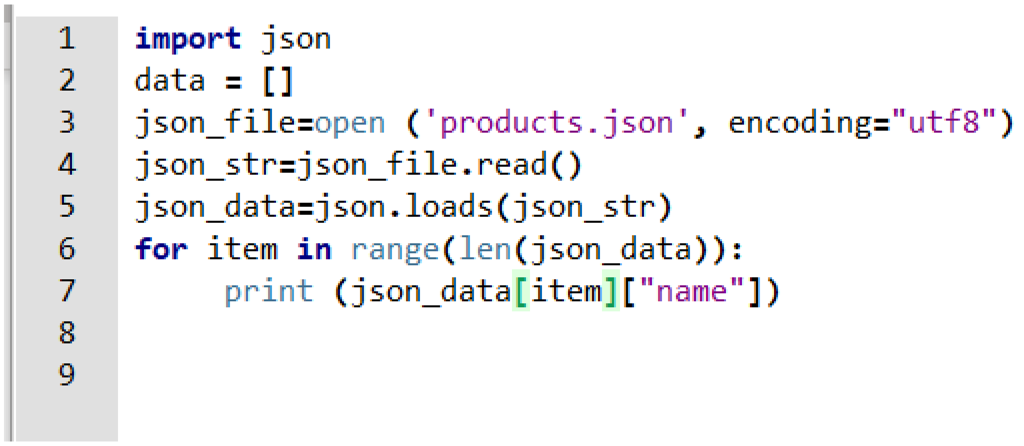
These are the BestBuy product names. If you save the products.json on the local desktop and open that in MS Excel, you get some idea about its format. This is what it looks like:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	[{"sku":43900	,"name":	"Duracell	- AAA Batteries (4-Pack)"	,"type":"HardGood","price":5.49,"upc":"041333424019","category":[{"id":"pcmcat312300050015","name":"Conn									
2	{"sku":48530,	"name":"	Duracell -	AA 1.5V CopperTop Batter	ies (4-Pack)","type":"HardGood","price":5.49,"upc":"041333415017","category":[{"id":"pcmcat312300050015","r									
3	{"sku":127687	,"name":	"Duracell	- AA Batteries (8-Pack)",	"type":"HardGood","price":7.49,"upc":"041333825014","category":[{"id":"pcmcat312300050015","name":"Conn									
4	{"sku":150115	,"name":	"Energizer	- MAX Batteries AA (4-Pa	ck)","type":"HardGood","price":4.99,"upc":"039800011329","category":[{"id":"pcmcat312300050015","name":"C									
5	{"sku":185230	,"name":	"Duracell	- C Batteries (4-Pack)","	type":"HardGood","price":8.99,"upc":"041333440019","category":[{"id":"pcmcat312300050015","name":"Conne									
6	{"sku":185267	,"name":	"Duracell	- D Batteries (4-Pack)","	type":"HardGood","price":9.99,"upc":"041333430010","category":[{"id":"pcmcat312300050015","name":"Conne									
7	{"sku":312290	,"name":	"Duracell	- 9V Batteries (2-Pack)",	"type":"HardGood","price":7.99,"upc":"041333216010","category":[{"id":"pcmcat312300050015","name":"Conn									
8	{"sku":324884	,"name":	"Directed	Electronics - Viper Audio	Glass Break Sensor","type":"HardGood","price":39.99,"upc":"093207005060","category":[{"id":"pcmcat11310005									
9	{"sku":333179	,"name":	"Energizer	- N Cell E90 Batteries (	2-Pack)","type":"HardGood","price":5.99,"upc":"039800013200","category":[{"id":"pcmcat312300050015","name									
10	{"sku":346575	,"name":	"Metra - R	adio Installation Dash Ki	t for Most 1989-2000 Ford, Lincoln & Mercury Vehicles - Black","type":"HardGood","price":16.99,"upc":"0864290									
11	{"sku":346646	,"name":	"Metra - R	adio Dash Multikit for Se	lect GM Vehicles - Black","type":"HardGood","price":16.99,"upc":"086429003273","category":[{"id":"abcat03000									
12	{"sku":347137	,"name":	"Metra - W	iring Harness for Select	1998-2008 Ford Vehicles - Multicolored","type":"HardGood","price":16.99,"upc":"086429056514","category":[{"i									
13	{"sku":347146	,"name":	"Metra - T	urbo Wire Aftermarket Rad	io Wire Harness Adapter for Select Vehicles","type":"HardGood","price":16.99,"upc":"086429056507","category"									
14	{"sku":347155	,"name":	"Metra - W	iring Harness for Most 19	86-1998 Honda Acura Vehicles - Multicolored","type":"HardGood","price":16.99,"upc":"086429002597","categor									
15	{"sku":347333	,"name":	"METRA - A	ntenna Cable Adapter - Bl	ack","type":"HardGood","price":13.99,"upc":"086429007189","category":[{"id":"abcat0300000","name":"Car Ele									
16	{"sku":349572	,"name":	"INSTALL -	PORTABLE RADAR DETECTOR	INST","type":"HardGood","price":29.99,"upc":"400003495726","category":[{"id":"pcmcat298100050010","name"									
17	{"sku":373642	,"name":	"Jensen -	3.6V NiCad Battery for 90	0MHz Phones","type":"HardGood","price":19.99,"upc":"044476085840","category":[{"id":"pcmcat312300050015									
18	{"sku":478398	,"name":	"Metra - R	adio Installation Dash Ki	t for Select Ford, Mazda and Mercury Vehicles (Pair) - Black","type":"HardGood","price":16.99,"upc":"08642901									

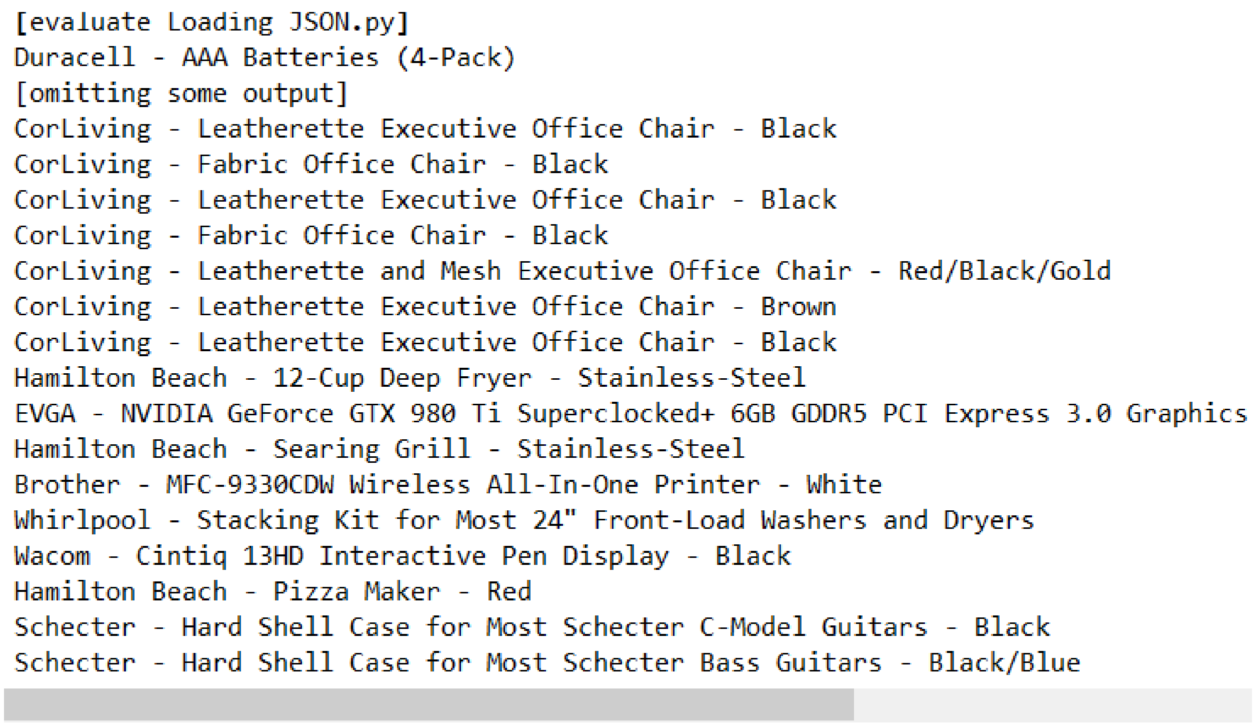
It has a **list** (i.e. the entire product is in []) that each of its elements is a python **dictionary** with keys and values. You have seen a similar data model when we tried to model the Spotify website in the class. You are just interested to pick the key =name. The columns C and D sound good candidates to build the dictionary structure. You do not care about the rest. To simplify the solution you can limit the search suggestions to **5** suggestions to users.

I have written a code for you not to worry about loading the keys and values. Run this code in the same folder where your JSON file is located:



```python
import json
data = []
json_file=open ('products.json', encoding="utf8")
json_str=json_file.read()
json_data=json.loads(json_str)
for item in range(len(json_data)):
     print (json_data[item]["name"])


```

If you run this code you get this result:



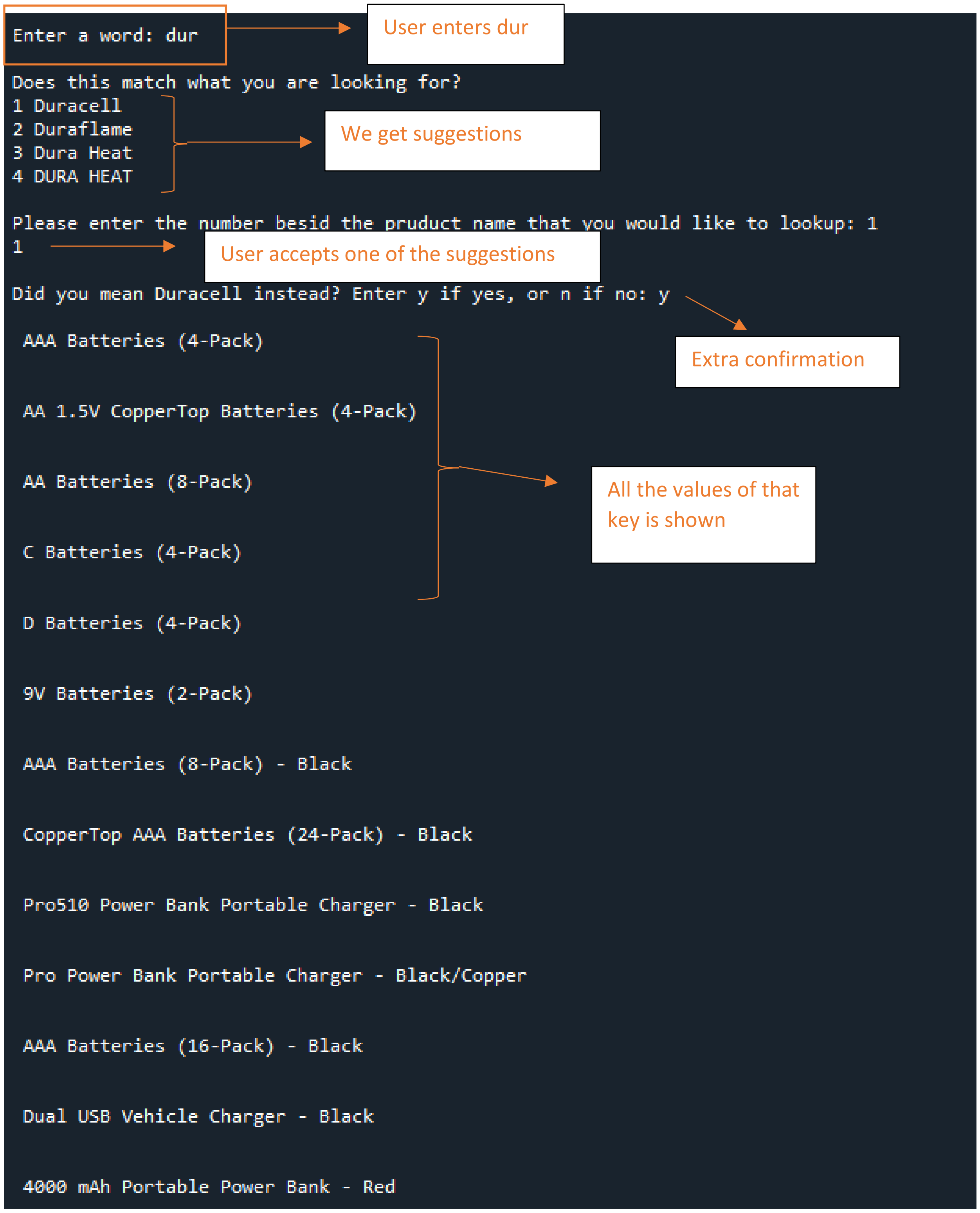
```
[evaluate Loading JSON.py]
Duracell - AAA Batteries (4-Pack)
[omitting some output]
CorLiving - Leatherette Executive Office Chair - Black
CorLiving - Fabric Office Chair - Black
CorLiving - Leatherette Executive Office Chair - Black
CorLiving - Fabric Office Chair - Black
CorLiving - Leatherette and Mesh Executive Office Chair - Red/Black/Gold
CorLiving - Leatherette Executive Office Chair - Brown
CorLiving - Leatherette Executive Office Chair - Black
Hamilton Beach - 12-Cup Deep Fryer - Stainless-Steel
EVGA - NVIDIA GeForce GTX 980 Ti Superclocked+ 6GB GDDR5 PCI Express 3.0 Graphics
Hamilton Beach - Searing Grill - Stainless-Steel
Brother - MFC-9330CDW Wireless All-In-One Printer - White
Whirlpool - Stacking Kit for Most 24" Front-Load Washers and Dryers
Wacom - Cintiq 13HD Interactive Pen Display - Black
Hamilton Beach - Pizza Maker - Red
Schecter - Hard Shell Case for Most Schecter C-Model Guitars - Black
Schecter - Hard Shell Case for Most Schecter Bass Guitars - Black/Blue
```

As you see in the above screenshot you have repeated names like **Corliving**. Please merge them into one dictionary record. That means one key **Corliving** will have more than one value. So you will have a dictionary with a key that may have many values and each value is an item in a list.

You know how to search and suggest a Python dictionary data structure when you worked on the Notes on 10 practical question set **(question 1)**. Use the same technique to answer this question.

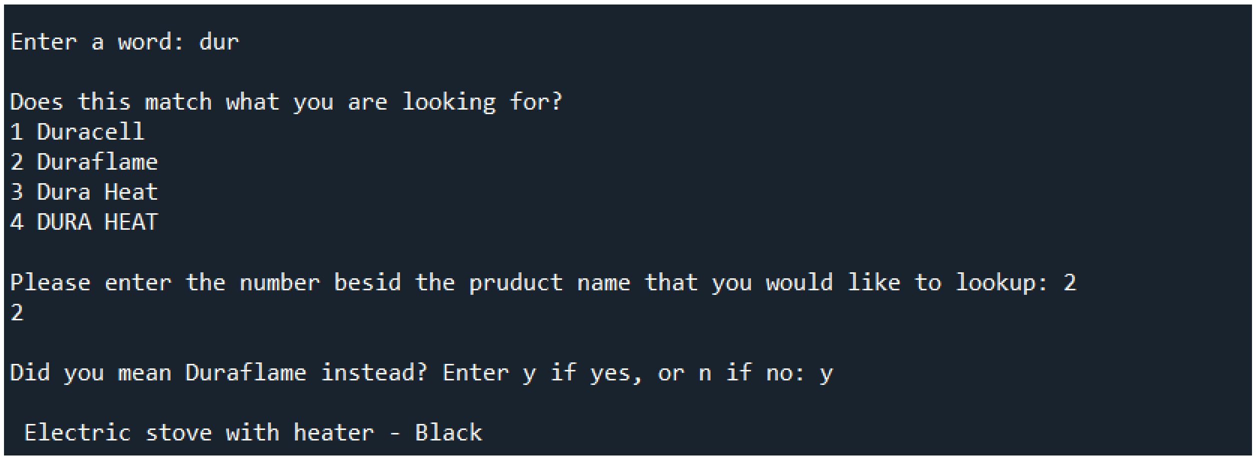
NOTE: if you need to remove the empty lines in the **producs.json** file, it is OK to do that manually.

Sample outputs:


User enters dur
We get suggestions
User accepts one of the suggestions
Extra confirmation
All the values of that key is shown

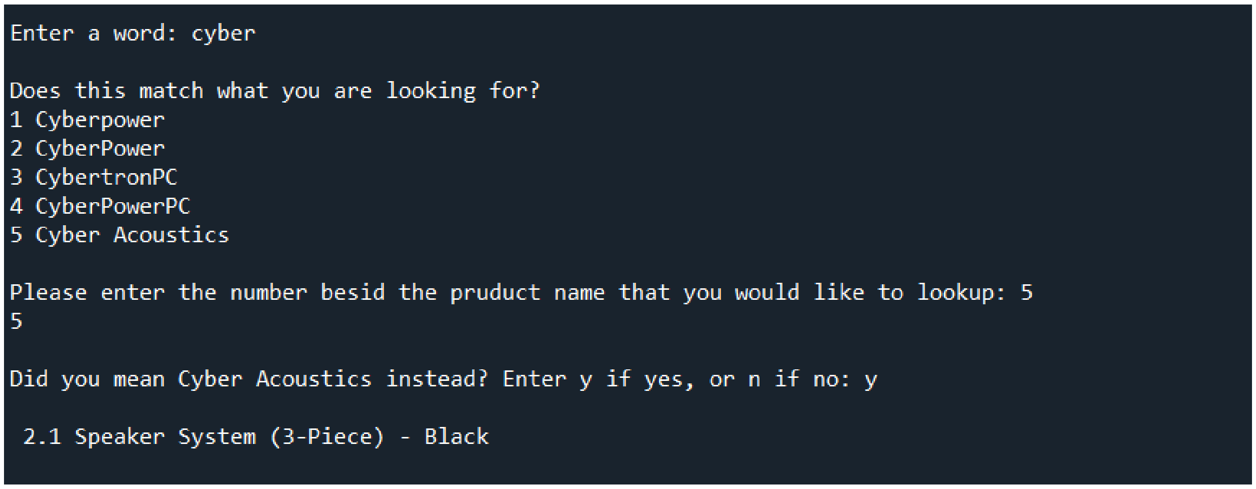
```
Enter a word: dur

Does this match what you are looking for?
1 Duracell
2 Duraflame
3 Dura Heat
4 DURA HEAT

Please enter the number besid the pruduct name that you would like to lookup: 1
1

Did you mean Duracell instead? Enter y if yes, or n if no: y

 AAA Batteries (4-Pack)


 AA 1.5V CopperTop Batteries (4-Pack)


 AA Batteries (8-Pack)


 C Batteries (4-Pack)


 D Batteries (4-Pack)


 9V Batteries (2-Pack)


 AAA Batteries (8-Pack) - Black


 CopperTop AAA Batteries (24-Pack) - Black


 Pro510 Power Bank Portable Charger - Black


 Pro Power Bank Portable Charger - Black/Copper


 AAA Batteries (16-Pack) - Black


 Dual USB Vehicle Charger - Black


 4000 mAh Portable Power Bank - Red
```

Or:



```
Enter a word: dur

Does this match what you are looking for?
1 Duracell
2 Duraflame
3 Dura Heat
4 DURA HEAT

Please enter the number besid the pruduct name that you would like to lookup: 2
2

Did you mean Duraflame instead? Enter y if yes, or n if no: y

 Electric stove with heater - Black
```

Or:



```
Enter a word: cyber

Does this match what you are looking for?
1 Cyberpower
2 CyberPower
3 CybertronPC
4 CyberPowerPC
5 Cyber Acoustics

Please enter the number besid the pruduct name that you would like to lookup: 5
5

Did you mean Cyber Acoustics instead? Enter y if yes, or n if no: y

 2.1 Speaker System (3-Piece) - Black
```

In terms of the messages to be given to the user, I have shown an example above. When the user enters a word close to one of the keys, then we ask the user if that's the word that is looking for (if we have more matches, we show a list of words that are close to the user input). When the user confirms, we show the value(s) of that key. If there is no such key, then we should say there is no such key (specifically when we cannot find anything close to suggest).

FAQs:

FAQ 1- In questions 1 and 3, should I take user input, or will you just pop the test list into my code?

A: You can hardcode the list in the code. That is easier for you and for me to test. I will change it while testing the application. See FAQ 4 below.

FAQ 2- In question 2, I was a bit unclear on the goal, is it to simply return if the 2 numbers are less, equal, or greater than each other? And I can do that anyway except cast to int?

A: Yes, without casting, please consider that I may give nothing (empty string as well). That is actually an easy question, there is no trick there.

FAQ 3- In Question 2: Does it mean only inputs are integers? Will there be decimal input?

A: I will test with decimal numbers too. In the question, there is no indication that we do not test with non-integer values, so decimals will be tested.

Also, the user should enter a valid number so if the user enters 10 31 (with a space between 10 and 31) that is not a valid number, so the code will show a message saying that is not a valid digit.

FAQ 4- Is there any validation needed for Questions 1 and 3?

A: If you hard code the list, then no validation is needed for Q1 and 3.

FAQ 5- In Question 2: Will inputs with blank space(s) to either or both sides be a valid number (it was previously mentioned in FAQ 3 above that blank space(s) between numbers make it invalid)? For example **" 123   "** is valid?

A: You have to either handle that by code to remove white spaces on the two sides or show an error message to the user to fix the issue.

FAQ 6- In Question 3: Is it only finding and printing the "first" two numbers that equal 10? No need to find a second, third ... pair?

A: Printing one combination is good enough.