**PART 1: Work with Python interactively**

* Use Python IDLE to complete this part and put your answer beside each question in **bold.**

**1a. Work with *lists*. Define the following lists:**

L1 = [3, 'word', 4.5, (3,4,5), [(1,'x'), 2, (3,'y')], 'Windsor', 'B', 75, 'Toronto']

L2 = ['sky', 'tree', 20, 5.00, 'grass', 'green']

i) Write Python commands to output the following and indicate the corresponding results:

1. 1st 2 elements of L1

**print(L1[0:2]);**

1. last 3 elements of L1

**print(L1[-3:]);**

1. elements 3-5 (inclusive) of L2

**print(L2[3:6]);**

1. 3rd character of last element of L2

**print(L2[-1][2]);**

ii) Indicate the results if you type the following at the Python prompt in IDLE interactive mode:

1. L1[3][1] **4**
2. L1[4][0] **(1 , ‘x’)**
3. L1[4][2][1] **‘y’**
4. len(L2) **6**
5. L1[14] **list index out of range**
6. L1[-4:-1] **[‘Windsor’, ‘B’, 75]**
7. L1[2:14] **[4.5, (3, 4, 5), [(1, 'x'), 2, (3, 'y')], 'Windsor', 'B', 75, 'Toronto']**
8. L1[-2:0] **[]**
9. L1+L2

**[3,**

**'word',**

**4.5,**

**(3, 4, 5),**

**[(1, 'x'), 2, (3, 'y')],**

**'Windsor',**

**'B',**

**75,**

**'Toronto',**

**'sky',**

**'tree',**

**20,**

**5.0,**

**'grass',**

**'green']**

1. L2\*2

**['sky',**

**'tree',**

**20,**

**5.0,**

**'grass',**

**'green',**

**'sky',**

**'tree',**

**20,**

**5.0,**

**'grass',**

**'green']**

1. L1.append(L2)

print(L1)

**[3,**

**'word',**

**4.5,**

**(3, 4, 5),**

**[(1, 'x'), 2, (3, 'y')],**

**'Windsor',**

**'B',**

**75,**

**'Toronto',**

**['sky', 'tree', 20, 5.0, 'grass', 'green']]**

1. L2.insert(2, 'xyz')

**print(L2)**

**['sky', 'tree', 'xyz', 20, 5.0, 'grass', 'green']**

1. L2.pop()

**green**

1. L1[4][1] = 10

**[3,**

**'word',**

**4.5,**

**(3, 4, 5),**

**[(1, 'x'), 10, (3, 'y')],**

**'Windsor',**

**'B',**

**75,**

**'Toronto',**

**['sky', 'tree', 'xyz', 20, 5.0, 'grass']]**

1. del L2[-3]

**['sky', 'tree', 'xyz', 5.0, 'grass']**

iii) Generate a list containing all numbers that are divisible by 7 but not multiple of 5 between 100 and 500 (both included)

**Code:**

**list=[] #declaring an empty list**

**for x in range(100, 500):**

**if (x%7==0) and (x%5!=0): #divisible by 7 but not multiple of 5**

**list.append(str(x)) #adds it to the list**

**print (','.join(list)) #prints the values in the list**

**O/P:**

**112,119,126,133,147,154,161,168,182,189,196,203,217,224,231,238,252,259,266,273,287,294,301,308,322,329,336,343,357,364,371,378,392,399,406,413,427,434,441,448,462,469,476,483,497**

**1b. Work with *strings*. Define the following strings:**

* s1= "Python is a very powerful language to learn!"
* s2 = 'Why we need to learn web development in Django?

i) Use **str** *methods* to do the following and indicate the corresponding results: (A list of **str** *methods* can be found in <https://docs.python.org/3/library/stdtypes.html#string-methods> )

1. Check if the string s2 ends with the word ‘Django?’

**Code: if 'Django?' in s2:**

**print("Word is present in string")**

**else:**

**print("Word is not present in string")**

**O/P: Word is present in string**

1. Determine leftmost position of ‘pow’ in s1

**Code: s1.index(‘pow’)**

**O/P: 17**

1. Return a list of words from s2

**Code: print(s2.split())**

**O/P: ['Why’, ‘we’, ‘need’, ‘to’, ‘learn’, ‘web’, ‘development’, ‘in’, ‘Django?']**

1. Convert s1 and s2 to all uppercase letters

**Code: print(s1.upper())**

**print(s2.upper())**

**O/P: PYTHON IS A VERY POWERFUL LANGUAGE TO LEARN**

**WHY WE NEED TO LEARN WEB DEVELOPMENT IN DJANGO?**

1. Replace the string 'to learn' of s2 with empty string

**Code: print(s2.replace("to learn"," "))**

**O/P: Why we need web development in Django?**

1. Count the number of times ‘p’ occurs in s1

**Code: print(s1.count('p')) #capital ‘P’ is not included…. Only ‘p’ is counted**

**O/P: 1**

ii) Indicate the results if you type the following at the Python prompt in IDLE interactive mode:

|  |  |  |
| --- | --- | --- |
| a) | s1[:6] | **Python** |
| b) | s2[-1:-4] | ‘ ’ |
| c) | s2[-7:-1] | **Django** |
| d) | s2[-1:] | **?** |
| e) | s2[0:20:2] | **Wyw edt er** |
| f) | s1+" "+s2 | **Python is a very powerful language to learn! Why we need to learn web development in Django?** |

**1c. Work with *dicts*. Define the following *dicts*:**

* *#dictionary literals*

d1={"name": "John", "age": 25, (3, 'm'):['a', 'b', 'c'], 5: "Ontario", 20: 96, 12:27}

* *#dictionary using sequences*

d2 = dict([("name","Alice"), ('age', 24), ((1,2), ['u', 'v', 'w']), (0, 'blue'), (86, 20)])

* *#dictionary using keywords*

d3 = dict(id=123, name='Willis', siblings=['Alex', 'Bob', 'Cindy'])

* *#dictionary using zip( ) function*

d4 = dict(zip(("id", "name", "quantity"), (1234, "Disk Drive", 3)))

i) Indicate the results if you type the following at the Python prompt in IDLE interactive mode:

1. d1.keys()

**dict\_keys(['name', 'age', (3, 'm'), 5, 20, 12])**

1. d2.keys()

**dict\_keys(['name', 'age', (1, 2), 0, 86])**

1. d3.get('id')

**123**

1. d2.get('age’) **24**
2. d3.get('age') **NO OUTPUT**
3. d3.get('name', 'Tim') **Willis**
4. d2.items() **dict\_items([('name', 'Alice'), ('age', 24), ((1, 2), ['u', 'v', 'w']), (0, 'blue'), (86, 20)])**
5. d3['siblings'] **['Alex', 'Bob', 'Cindy']**
6. d2['siblings']

**---------------------------------------------------------------------------**

**KeyError Traceback (most recent call last)**

[**<ipython-input-37-b68ad526cb41>**](https://localhost:8080/) **in <module>()**

**----> 1 d2['siblings']**

**KeyError: 'siblings'**

1. d2.update(d3)

**{(1, 2): ['u', 'v', 'w'],**

**0: 'blue',**

**86: 20,**

**'age': 24,**

**'id': 123,**

**'name': 'Willis',**

**'siblings': ['Alex', 'Bob', 'Cindy']}**

1. d2[0] **blue**
2. d1.get((1,2)) **NO OUTPUT**
3. d2['siblings']**\* ['Alex', 'Bob', 'Cindy']**
4. d2['name']**\* Willis**
5. d1 == d2 **False**
6. len(d2) **7**
7. for key in d1.keys():

print(key)

**name**

**age**

**(3, 'm')**

**5**

**20**

**12**

1. for key in d2.keys():

print(d2[key])

**Willis**

**24**

**['u', 'v', 'w']**

**blue**

**20**

**123**

**['Alex', 'Bob', 'Cindy']**

**\****After* update.

**1d. Work with *datetime* module:**

i) Import the *datetime* module. Write Python commands to do following: (Check <https://docs.python.org/3/library/datetime.html> )

1. Display current date. Show the year, month and date separately.

**Code:**

**import datetime**

**x = datetime.date.today()**

**print(x)**

**Output:**

**2022-05-12**

1. Display current datetime

**Code:**

**x = datetime.datetime.now()**

**print(x)**

**Output:**

**2022-05-12 18:27:40.172125**

1. Calculate the number of days till the end of the year

**Code**

**# dates in string format**

**str\_d1 = '2022/5/12'**

**str\_d2 = '2022/12/31'**

**# convert string to date object**

**d1 = datetime.datetime.strptime(str\_d1, "%Y/%m/%d")**

**d2 = datetime.datetime.strptime(str\_d2, "%Y/%m/%d")**

**# difference between dates in timedelta**

**delta = d2 - d1**

**print(f'Difference is {delta.days} days')**

**Output**

**Difference is 233 days**

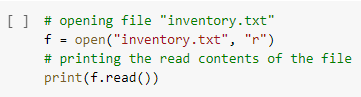
**PART 2: Programming in Python**

* Submit a separate .py file on BB for this part.

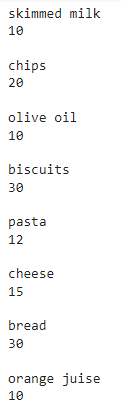
Write a **Python script, with comments**, to do the following:

1. Open a text file called “*inventory.txt*” for reading. The file contains products available in a store & their quantities. You can find a copy of the text file under **Resources**.

Code:



Output:

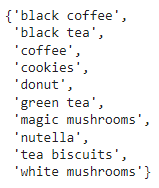


1. Define a list of strings called ***productList***. The list should contain at least 10 strings and each string represent a specific product.

Code:

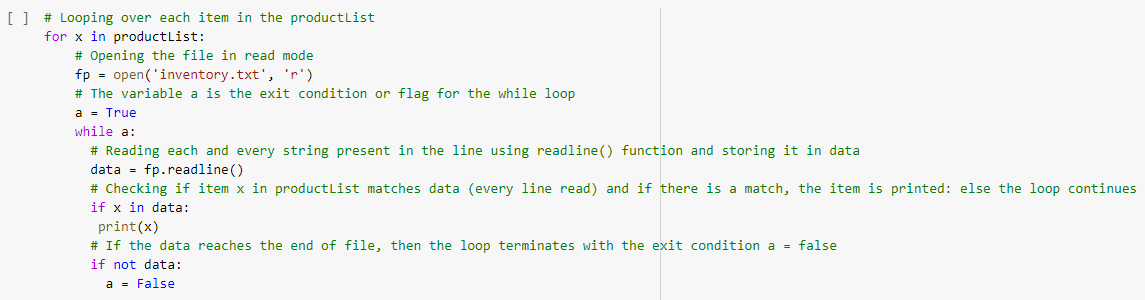


Output:

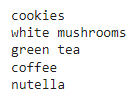


1. Loop over each element in ***productList*** and check if that element (product) matches any of the products in the file. You should use the function readline() to read a new line from the file and compare that line with the element from the list.

Code:

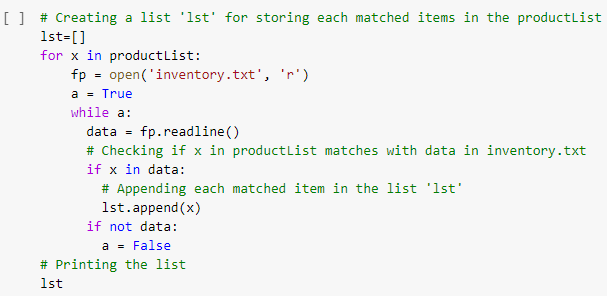


Output:



1. If there is a match, save the quantity of that product in a variable.

Code:

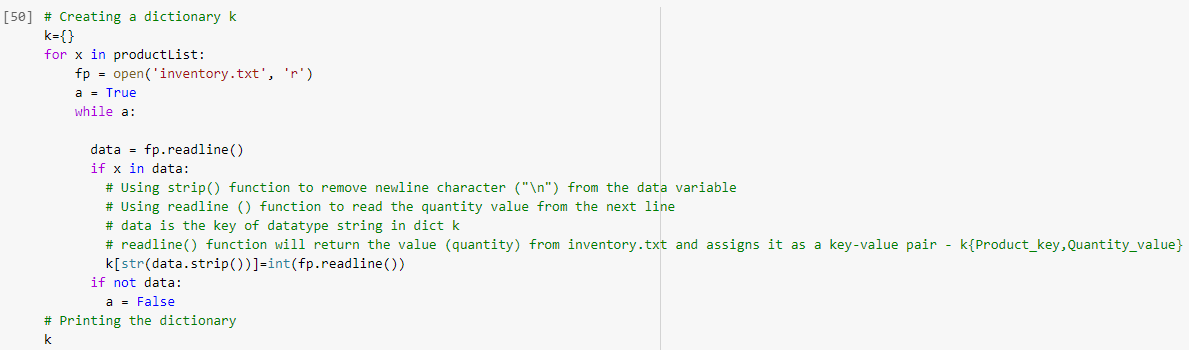


Output:

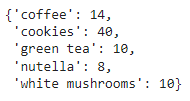


1. Create a **dict** ***d1*** with entries ***product: n***, ***product*** (**key**) is a product (string) from ***productList*** (Also the line read form the file) & ***n*** (**value**) is the quantity of ***product***. Use the function update() on ***d1*** to add a new item {***product***: ***n***} to the dictionary.

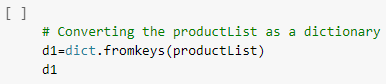
Code:



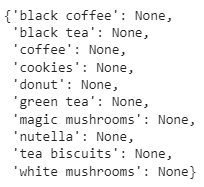
Output:



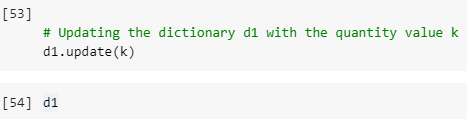
Code:



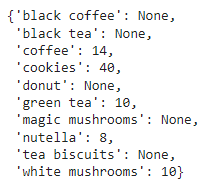
Output:



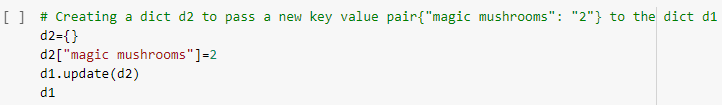
Code:



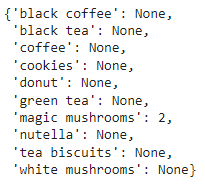
Output:



Code:

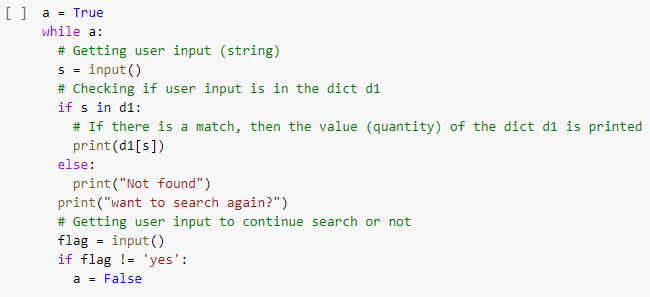


Output:

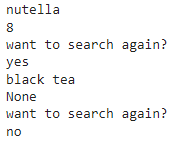


1. The program should ask the user to input a string ***s*** and retrieve the quantity of ***s*** from **dict** ***d1***. After display of the quantity corresponding to the product ***s***, the program asks if the user would like another search with (***yes***/***no***) options. If the user enters ***yes***, another quantity retrieval should be done. If the answer is ***no***, the program should exit.

Code:

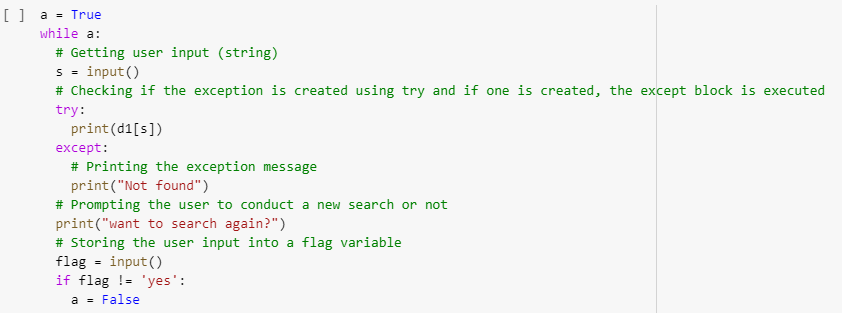


Output:



1. If the product name entered by the user does not correspond to a valid key, the program should catch an exception. When the exception occurs, display an appropriate error message then prompt the user to input another product name.

Code:



Output:

