Lab: Dictionaries

This document defines the exercises for the "Python Fundamentals" course at @Software University.

Please submit your solutions (source code) to all the below-described problems in Judge.

1. Bakery

Your first task at your new job is to create a table of the stock in a bakery, and you really don't want to fail on your first day at work.

You will receive a single line containing some **food** (keys) and **quantities** (values). They will be separated by a **single space** (the **first** element is the **key**, the **second** – is the **value,** and so on). Create a **dictionary** with all the keys and values and **print** it on the console.

Example

Input	Output
`bread 10 butter 4 sugar 9 jam 12`	`{'bread': 10, 'butter': 4, 'sugar': 9, 'jam': 12}`
`eggs 3 sugar 7 salt 1 butter 3`	`{'eggs': 3, 'sugar': 7, 'salt': 1, 'butter': 3}`

Hint

Let us start with **reading the input** and **creating an empty dictionary**:

01-bakery.py

```
elements = input().split(" ")
bakery = {}   # bakery = dict()
```

- Note that there are **2 ways** of creating a dictionary (with the **curly braces** or using the **`dict()`** method).

Since we know that we will get **key-value pairs**, we can use a **for-loop** with **step 2.** We take the **key** and the **value** and **add** them to the dictionary, and after the **loop is over,** we **print** it:

```
for i in range(0, len(elements), 2):
    key = elements[i]
    value = elements[i + 1]
    bakery[key] = int(value)
print(bakery)
```

- Note that the **value** must be an **integer** (since it is a **quantity**).

2. Stock

After you have completed your first task, your boss decides to give you another one right away. Now, not only do you have to keep track of the stock, but you also need to answer customers about product availability.

You will be given **key-value** pairs of **products** and **quantities** (on a single line **separated by space**). On the following line, you will be given products to **search** for. Check for each product. You have **2 possibilities**:

- If you **have the product**, print "**`We have {quantity} of {product} left`**".
- **Otherwise**, print "**`Sorry, we don't have {product}`**".

Example

Input	Output
cheese 10 bread 5 ham 10 chocolate 3 jam cheese ham tomatoes	Sorry, we don't have jam We have 10 of cheese left We have 10 of ham left Sorry, we don't have tomatoes
eggs 5 bread 10 bread eggs	We have 10 of bread left We have 5 of eggs left

Hint

We repeat the steps from the previous task for reading the products and adding them to the dictionary.

```
02-stock.py
1  elements = input().split(" ")
2  bakery = {}
3  for i in range(0, len(elements), 2):
4      key = elements[i]
5      value = elements[i + 1]
6      bakery[key] = int(value)
```

Next, we **read** the products we have to **search** for and **check** for each of them.

```
8  searched_products = input().split(" ")
9  for product in searched_products:
10     if product in bakery:
11         print(f"We have {bakery[product]} of {product} left")
12     else:
13         print(f"Sorry, we don't have {product}")
```

3. Statistics

You seem to be doing great at your first job, so your boss decides to give you as your next task something more challenging. You have to accept all the new products coming into the bakery and finally gather some statistics.

You will be receiving **key-value** pairs on separate lines separated by **": "** until you receive the command **"statistics"**. Sometimes you may receive a product **more than once**. In that case, you have to **add** the **new quantity** to the existing one. When you receive the **"statistics"** command, print the following:

"Products in stock:

- {product1}: {quantity1}

- {product2}: {quantity2}

…

- {productN}: {quantityN}

Total Products: {count_all_products}

Total Quantity: {sum_all_quantities}"

Example

Input	Output
bread: 4 cheese: 2 ham: 1 bread: 1 statistics	Products in stock: - bread: 5 - cheese: 2 - ham: 1 Total Products: 3 Total Quantity: 8
eggs: 10 bread: 6 cheese: 8 milk: 7 statistics	Products in stock: - eggs: 10 - bread: 6 - cheese: 8 - milk: 7 Total Products: 4 Total Quantity: 31

Hint

Let us start by **creating the dictionary** and creating a **while loop**

03-statistics.py

```
products = {}

command = input()
while command != "statistics":
    command = input()
```

Now, let us get the **product** and the **quantity**

```
while command != "statistics":
    tokens = command.split(": ")
    product = tokens[0]
    quantity = int(tokens[1])
```

Then, we want to create a **check if** the product is **not already** in the dictionary and set its **value** to **0**

```
while command != "statistics":
    tokens = command.split(": ")
    product = tokens[0]
    quantity = int(tokens[1])
    if product not in products:
        products[product] = 0
```

- That way, we make sure that the **product will exist** before we add the **quantity**

Then we **add up the quantity**

```
    if product not in products:
        products[product] = 0
    products[product] += quantity
```

And finally, we **print** the result

```
13  print("Products in stock:")
14  for (product, quantity) in products.items():
15      print(f"- {product}: {quantity}")
16  print(f"Total Products: {len(products.keys())}")
17  print(f"Total Quantity: {sum(products.values())}")
```

- For the total products, we get the **length of the keys**
- For the total quantity, we **sum the values**

*Another way of implementing lines 14 and 15 would be by using **dictionary comprehension**

```
16  [print(f"- {product}: {quantity}") for (product, quantity) in products]
```

4. Students

You will be receiving names of students, their ID, and a course of programming they have taken in the format **"{name}:{ID}:{course}"**. On the last line, you will receive the name of a course in snake case lowercase letters. You should print only the information of the students who have taken the corresponding course in the format: **"{name} - {ID}"** on separate lines.

Note: each student's ID will always be unique

Input	Output
Peter:123:programming basics John:5622:fundamentals Maya:89:fundamentals Lilly:633:fundamentals fundamentals	John - 5622 Maya - 89 Lilly - 633
Alex:6:programming basics Maria:7:programming basics Kaloyan:9:advanced Todor:10:fundamentals programming_basics	Alex - 6 Maria - 7

5. ASCII Values

Write a program that receives a **list of characters** separated by **", "**. It should create a dictionary with each **character** as a **key** and its **ASCII** value as a **value**. Try solving that problem using **comprehension**.

Examples

Input	Output
a, b, c, a	{'a': 97, 'b': 98, 'c': 99}
d, c, m, h	{'d': 100, 'c': 99, 'm': 109, 'h': 104}

6. Odd Occurrences

Write a program that **prints** all elements from a given sequence of words that occur an **odd number of times** (case-insensitive) in it.

- Words are given on a **single line**, space-separated.

- Print the result elements in **lowercase**, in **their order of appearance**.

Examples

Input	Output
Java C# PHP PHP JAVA C java	java c# c
3 5 5 hi pi HO Hi 5 ho 3 hi pi	5 hi
a a A SQL xx a xx a A a XX c	a sql xx c

Hints

Read a line from the console **split it by a space**, and create a **dictionary**:

```
04-odd-occurrences.py
1  words = input().split(" ")
2  dictionary = {}
```

Create a **loop** and check for **each word** (**lower case**) if it is **in the dictionary**, and if it is not, add it:

```
3  for word in words:
4      word_lower = word.lower()
5      if word_lower not in dictionary:
6          dictionary[word_lower] = 0
7      dictionary[word_lower] += 1
```

Then create another loop using the **`items()`** method and check if the number of **occurrences** of the current word is **odd. If it is, print it:**

```
8  for (key, value) in dictionary.items():
9      if value % 2 != 0:
10         print(key, end=" ")
```

7. Word Synonyms

Write a program, which keeps a dictionary with synonyms. The **key** of the dictionary will be the **word**. The **value** will be a **list of all the synonyms of that word**. You will be given a number **n – the count of the words**. After each term, you will be given a synonym, so the count of lines you should read from the console is **2 * n. You will be receiving** a **word** and a **synonym** each on a separate line like this:

- **`{word}`**
- **`{synonym}`**

If you get the same word **twice**, just **add the new synonym** to the list.

Print the words in the following format:

`{word} - {synonym1, synonym2 … synonymN}`

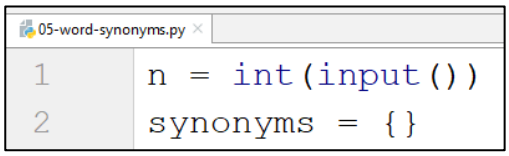
Examples

Input	Output
3 cute adorable	cute - adorable, charming smart - clever

cute charming smart clever	
2 task problem task assignment	task - problem, assignment

Hint

We start by reading the number **n** and creating the **dictionary with synonyms**:



Then we create a **for loop** to read the **word-synonym pairs**:

```
3  for i in range(n):
4      word = input()
5      synonym = input()
```

We check if the **word is not** in the dictionary, and in that case, we set its **value to an empty list** (since one word can have **multiple synonyms**), and we **append the new synonym** to that list:

```
6      if word not in synonyms:
7          synonyms[word] = []
8      synonyms[word].append(synonym)
```

Finally, we **print** the result:

```
9  for word in synonyms:
10     print(f"{word} - {', '.join(synonyms[word])}")
```