# Programming Fundamentals with Python: Exam Preparation

## 01. Password Reset

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2303#0**](https://judge.softuni.org/Contests/Practice/Index/2303#0)**.**

*Yet again, you have forgotten your password. Naturally, it's not the first time this has happened. Actually, you got so tired of it that you decided to help yourself with an intelligent solution.*

Write a password reset program that performs a series of commands upon a predefined string. First, you will receive a string, and afterward, until the command "**Done**" is given, you will be receiving strings with commands split by a single space. The commands will be the following:

* "TakeOdd"
  + Takes only the characters at **odd** **indices** and **concatenates** them to obtain the **new raw password** and then **prints** it.
* "Cut {index} {length}"
  + Gets the substring with the **given length** starting from the **given index** from the password and removes its **first occurrence**, then **prints** the password on the console.
  + The given index and the length will **always** be **valid**.
* "Substitute {substring} {substitute}"
  + If the raw password contains the given substring, replaces all of its occurrences with the substitute text given and prints the result.
  + If it doesn't, prints "Nothing to replace!".

### Input

* You will be receiving strings until the "**Done**" command is given.

### Output

* After the "Done" command is received, print:
  + "Your password is: {password}"

### Constraints

* The indexes from the "**Cut {index} {length}**" command will always be valid.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Siiceercaroetavm!:?:ahsott.:i:nstupmomceqr  TakeOdd  Cut 15 3  Substitute :: -  Substitute | ^  Done | icecream::hot::summer  icecream::hot::mer  icecream-hot-mer  Nothing to replace!  Your password is: icecream-hot-mer |
| **Comments** | |
| TakeOdd -> We only take the chars at odd indices 1, 3, 5 etc.  Siiceercaroetavm!:?:ahsott.:i:nstupmomceqr -> icecream::hot::summer  **Cut 15 3 ->** We cut a substring starting at index 15 with length 3, then remove it from the raw password:  icecream::hot::summer -> sum  **Substitute :: - ->** We replace "::" with "-":  icecream::hot::summer -> icream-hot-summer  **Substitute** | ^ **->** "|" is not found anywhere in the raw password, so we print **"Nothing to replace!"**  Finally, after receiving the "**Done**" command, we print the resulting password in the proper format. | |
| **Input** | **Output** |
| up8rgoyg3r1atmlmpiunagt!-irs7!1fgulnnnqy  TakeOdd  Cut 18 2  Substitute ! \*\*\*  Substitute ? .!.  Done | programming!is!funny  programming!is!fun  programming\*\*\*is\*\*\*fun  Nothing to replace!  Your password is: programming\*\*\*is\*\*\*fun |

## 02. Emoji Detector

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2302#1**](https://judge.softuni.org/Contests/Practice/Index/2302#1)**.**

Your task is to write a program that extracts emojis from a text and find the threshold based on the input.

You have to get your **cool threshold**. It is obtained by **multiplying all** the digits found in the input. The cool threshold could be a **huge number**, so be mindful.

An emoji is valid when:

* It is surrounded by 2 characters, either "::" or "\*\*"
* It is **at least 3** characters long (**without** the surrounding symbols)
* **It starts** with a **capital letter**
* Continues with **lowercase** letters **only**

Examples of valid emojis: ::Joy::, \*\*Banana\*\*, ::Wink::

Examples of invalid emojis: ::Joy\*\*, **::fox:es:**, **\*\*Monk3ys\*\*, :Snak::Es::**

You need to count **all valid emojis** in the text and calculate their **coolness**. The coolness of the emoji is **determined** by summing all the **ASCII values of all letters** in the emoji.

Examples: ::Joy:: - 306, \*\*Banana\*\* - 577, ::Wink:: - 409

You need to print the result of the cool threshold and, after that to take all emojis out of the text, count them and print **only the cool ones** on the console.

### Input

* On the single input, you will receive a piece of string.

### Output

* On the first line of the output, print the obtained Cool threshold in the format:

**"Cool threshold: {coolThresholdSum}"**

* On the following line, **print the** **count of all emojis** found in the text in format:

"{countOfAllEmojis} emojis found in the text. The cool ones are:

{cool emoji 1}

{cool emoji 2}

…

{cool emoji N}"

### Constraints

There will always be at least one digit in the text!

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| In the Sofia Zoo there are 311 animals in total! ::Smiley:: This includes 3 \*\*Tigers\*\*, 1 ::Elephant:, 12 \*\*Monk3ys\*\*, a \*\*Gorilla::, 5 ::fox:es: and 21 different types of :Snak::Es::. ::Mooning:: \*\*Shy\*\* | Cool threshold: 540  4 emojis found in the text. The cool ones are:  ::Smiley::  \*\*Tigers\*\*  ::Mooning:: |
| **Comments** | |
| You can see all the valid emojis in green. There are various reasons why the rest are not valid, examine them carefully. The "cool threshold" is 3\*1\*1\*3\*1\*1\*2\*3\*5\*2\*1 = 540.  ::Smiley:: -> 83 + 109 + 105 + 108 + 101 + 121 = 627 > 540 -> cool  \*\*Tigers\*\* -> 84 + 105 + 103 + 101 + 114 + 115 = 622 > 540 -> cool  ::Mooning:: -> 77 + 111 + 111 + 110 + 105 + 110 + 103 = 727 > 540 -> cool  \*\*Shy\*\* -> 83 + 104 + 121 = 308 < 540 -> not cool  In the end, we print the count of all valid emojis found and each of the cool ones on a new line. | |
| **Input** | **Output** |
| 5, 4, 3, 2, 1, go! The 1-th consecutive banana-eating contest has begun! ::Joy:: \*\*Banana\*\* ::Wink:: \*\*Vali\*\* ::valid\_emoji:: | Cool threshold: 120  4 emojis found in the text. The cool ones are:  ::Joy::  \*\*Banana\*\*  ::Wink::  \*\*Vali\*\* |
| **Input** | **Output** |
| It is a long established fact that 1 a reader will be distracted by 9 the readable content of a page when looking at its layout. The point of using ::LoremIpsum:: is that it has a more-or-less normal 3 distribution of 8 letters, as opposed to using 'Content here, content 99 here', making it look like readable \*\*English\*\*. | Cool threshold: 17496  1 emojis found in the text. The cool ones are: |
| **Comments** | |
| You can see \*\*English\*\* is a valid emoji, but the sum of ASCII **is not** **bigger** than the cool threshold. That's why we **don't** print anything in the end. | |

## 03. Plant Discovery

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2518#2**](https://judge.softuni.org/Contests/Practice/Index/2518#2)**.**

*You have now returned from your world tour. On your way, you have discovered some new plants, and you want to gather some information about them and create an exhibition to see which plant is highest rated.*

On the **first line,** you will receive a number **n**. On the next **n lines**, you will be given some information about the plants that you have discovered in the format: **"{plant}<->{rarity}"**. **Store** that **information** because you will need it later. If you receive a plant **more than once**, **update** its rarity.

After that, until you receive the **command** **"Exhibition"**, you will be given some of these **commands**:

* **"Rate: {plant} - {rating}"** – **add** the given **rating** to the plant (**store all ratings**)
* **"Update: {plant} - {new\_rarity}"** – **update** the **rarity** of the plant with the **new one**
* **"Reset: {plant}"** – **remove all** the **ratings** of the given plant

**Note: If any given plant name is invalid, print "error"**

After the command **"Exhibition"**, print the information that you have about the plants in the following format:

**"Plants for the exhibition:  
- {plant\_name1}; Rarity: {rarity}; Rating: {average\_rating}**

**- {plant\_name2}; Rarity: {rarity}; Rating: {average\_rating}  
…**

**- {plant\_nameN}; Rarity: {rarity}; Rating: {average\_rating}"**

The **average rating** should be formatted to the **second decimal place.**

### Input / Constraints

* You will receive the input as **described above**
* **JavaScript**: you will receive a **list of strings**

### Output

* Print the **information** about all plants as **described above**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Arnoldii<->4  Woodii<->7  Welwitschia<->2  Rate: Woodii - 10  Rate: Welwitschia - 7  Rate: Arnoldii - 3  Rate: Woodii - 5  Update: Woodii - 5  Reset: Arnoldii  Exhibition | Plants for the exhibition:  - Arnoldii; Rarity: 4; Rating: 0.00  - Woodii; Rarity: 5; Rating: 7.50  - Welwitschia; Rarity: 2; Rating: 7.00 |
| 2  Candelabra<->10  Oahu<->10  Rate: Oahu - 7  Rate: Candelabra - 6  Exhibition | Plants for the exhibition:  - Candelabra; Rarity: 10; Rating: 6.00  - Oahu; Rarity: 10; Rating: 7.00 |