# Programming Fundamentals with Python: Exam Preparation

# Black Flag

**Submit your solutions in the SoftUni judge system at [https://judge.softuni.org/Contests/Practice/Index/1773#0](https://judge.softuni.org/Contests/Practice/Index/1773" \l "0).**

*Pirates are invading the sea, and you're tasked to help them plunder*

Create a program that checks if **target plunder** is **reached**. First, you will receive how many **days** the pirating lasts. Then you will receive how much the pirates **plunder for a day**. Last you will receive the **expected plunder** at the end.

Calculate how much **plunder** the pirates manage to **gather**. Each **day** they gather the **plunder**. Keep in mind that they attack more ships every third day and add additional plunder to their total gain, which is **50% of the daily plunder**. Every **fifth day** the pirates encounter a warship, and after the battle, they **lose 30%** of their **total plunder**.

If the gained plunder is **more or equal** to the target, print the following:

**"Ahoy! {totalPlunder} plunder gained."**

If the gained plunder is **less** than the target. Calculate the **percentage left** and print the following:

**"Collected only {percentage}% of the plunder."**

Both numbers should be **formatted** to the **2nd decimal place**.

## Input

* On the **1st line,** you will receive the **days** of the plunder – an **integer number** in the range [0…100000]
* On the **2nd line,** you will receive the **daily plunder** – an **integer number** in the range [0…50]
* On the **3rd line,** you will receive the **expected plunder** – a **real number** in the range [0.0…10000.0]

## Output

* In the end, print whether the plunder **was successful** or **not,** following the format **described above**.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 40 100 | Ahoy! 154.00 plunder gained. |
| **Comments** | |
| The days are 5, and the daily plunder is 40. On the third day, the total plunder is 120, and since it is a third day, they gain an additional 50% from the daily plunder, which adds up to 140. On the fifth day, the plunder is 220, but they battle with a warship and lose 30% of the collected cargo, and the total becomes 154. That is more than expected. | |
|  | |
| 10  20  380 | Collected only 36.29% of the plunder. |

# Mu Online

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2028#0**](https://judge.softuni.org/Contests/Practice/Index/2028#0)**.**

You have **initial health 100 and initial bitcoins 0**. You will be given **a string representing the dungeon's rooms**. Each room is separated with **'|'** (vertical bar): **"room1|room2|room3…"**

Each room contains **a command** and a **number**, separated by space. The command can be:

* **"potion"**
  + You are healed with the number in the second part. But your health **cannot exceed** your **initial health (100)**.
  + First print: **"You healed for {amount} hp."**
  + After that, print your current health: **"Current health: {health} hp."**
* **"chest"**
  + You've found some bitcoins, the number in the second part.
  + Print: **"You found {amount} bitcoins."**
* In **any other case,** you are **facing a monster**, which you will **fight**. The **second part of the room** contains the **attack** of the monster. You should remove the monster's attack from your health.
  + If you are not dead (health <= 0), you've slain the monster, and you should print: **"You slayed {monster}."**
  + If you've died, print **"You died! Killed by {monster}."** and your quest is over. Print the best room you've manage to reach: **"Best room: {room}"**

If you managed to **go through all the rooms** in the dungeon, print on the **following three lines**:

**"You've made it!"**

**"Bitcoins: {bitcoins}"**

**"Health: {health}"**

## Input / Constraints

You receive a **string** representing the dungeon's rooms, separated with **'|'** (vertical bar): **"room1|room2|room3…"**.

## Output

Print the corresponding messages described above.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| rat 10|bat 20|potion 10|rat 10|chest 100|boss 70|chest 1000 | You slayed rat.  You slayed bat.  You healed for 10 hp.  Current health: 80 hp.  You slayed rat.  You found 100 bitcoins.  You died! Killed by boss.  Best room: 6 |
| **Input** | **Output** |
| cat 10|potion 30|orc 10|chest 10|snake 25|chest 110 | You slayed cat.  You healed for 10 hp.  Current health: 100 hp.  You slayed orc.  You found 10 bitcoins.  You slayed snake.  You found 110 bitcoins.  You've made it!  Bitcoins: 120  Health: 65 |

# Memory Game

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2517#2**](https://judge.softuni.org/Contests/Practice/Index/2517#2)**.**

Write a program that recreates the **Memory game**.

On the first line, you will **receive a sequence of elements**. Each element in the sequence **will have a** **twin**. Until the player receives **"end"** from the console, you will receive **strings with two integers** separated by a space, representing **the indexes** of elements in the sequence.

If the player **tries to cheat** and enters **two equal indexes** or indexes which are **out of bounds of the sequence**, you should **add** two matching elements at the middle of the sequence in the following format:

**"-{number of moves until now}a"**

Then print this message on the console:

**"Invalid input! Adding additional elements to the board"**

## Input

* On the **first** line**,** you will receive a **sequence of elements**
* On the **following** lines, you will receive **integers** until the command **"end"**

## Output

* Every time the player hit **two matching elements**, you should **remove** them from the sequence and **print** on the console the following message:

**"Congrats! You have found matching elements - ${element}!"**

* If the player hit **two different elements**, you should **print** on the console the following message:

**"Try again!"**

* If the player hit **all matching elements** before he receives **"end"** from the console, you should **print** on the console the following message:

**"You have won in {number of moves until now} turns!"**

* If the player receives **"end"** **before** **he hits all matching elements**, you should **print** on the console the following message:

**"Sorry you lose :(**

**{the current sequence's state}"**

## Constraints

* **All elements in the sequence will always have a matching element.**

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 1 2 2 3 3 4 4 5 5  1 0  -1 0  1 0  1 0  1 0  end | Congrats! You have found matching elements - 1!  Invalid input! Adding additional elements to the board  Congrats! You have found matching elements - 2!  Congrats! You have found matching elements - 3!  Congrats! You have found matching elements - -2a!  Sorry you lose :(  4 4 5 5 |
| **Comment** | |
| 1)  1 0  1 1 2 2 3 3 4 4 5 5 –> 1 = 1, equal elements, so remove them. Moves: 1  2)  -1 0  -1 is invalid index so we add additional elements  2 2 3 3 -2а -2а 4 4 5 5, Moves: 2  3)  1 0  2 2 3 3 -2а -2а 4 4 5 5 -> 2 = 2, equal elements, so remove them. Moves: 3  4)  1 0  3 3 -2а -2а 4 4 5 5 -> 3 = 3, equal elements, so remove them. Moves: 4  5)  1 0  -2а -2а 4 4 5 5 -> -2а = -2а, equal elements, so remove them. Moves: 5  6)  You receive the end command.  There are still elements in the sequence, so the player loses the game.  Final state - 4 4 5 5 | |
| a 2 4 a 2 4  0 3  0 2  0 1  0 1  end | Congrats! You have found matching elements - a!  Congrats! You have found matching elements - 2!  Congrats! You have found matching elements - 4!  You have won in 3 turns! |
| a 2 4 a 2 4  4 0  0 2  0 1  0 1  end | Try again!  Try again!  Try again!  Try again!  Sorry you lose :(  a 2 4 a 2 4 |