**Exercise: While Loop**

Problems for in-class and homework exercises for the course ["Programming Basics" @ SoftUni](https://softuni.org/).

**Test** your solutions in the **Judge** system: <https://judge.softuni.org/Contests/3496/While-Loop-Exercise-PS>

## Old Books

Annie goes to her hometown after a very long period out of the country. After coming home, she sees her grandmother's old library and remembers her favorite book. Help Annie by writing a function in which she enters the **book (string)** she is looking for. **Until Annie finds her favorite book or checks out all the books in the library, the program must read the name of each following book (string). The books in the library are finished once you get the text "No More Books".**

* **If she does not find the book, print in two lines:**
* **"The book you search is not here!"**
* **"You checked {count} books."**
* **If she finds the book, print on a single line:**
  + **"You checked {count} books and found it."**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve(["Troy",  "Stronger",  "Life Style",  "Troy"]) | You checked 2 books and found it. | The book Annie is looking for in this case is Troy, and the library contains 3 books. The first one is Stronger, the second one is Life Style, the third book is the searched one - Troy and the program ends. |
| solve(["The Spot",  "Hunger Games",  "Harry Potter",  "Torronto",  "Spotify",  "No More Books"]) | The book you search is not here!  You checked 4 books. | The book Annie is looking for is The Spot.  The library contains 4 books.  The first one is Hunger Games, the second one Harry Potter, the third one Torronto, and the fourth one Spotify. Since there are no more books in the library, the name reading finished. Annie didn't find the book she was looking for. |
| solve(["Bourne",  "True Story",  "Forever",  "More Space",  "The Girl",  "Spaceship",  "Strongest",  "Profit",  "Tripple",  "Stella",  "The Matrix",  "Bourne"]) | You checked 10 books and found it. |  |

### Hints and Guidelines

1. Take the favorite book out of the data array that the function receives.

Screenshot_6.png

1. Make **two more help variables at the beginning** to keep track of **whether the book is found or if all books are checked**. The one variable will be a **counter** and should be a **number** with **an initial value of one**. It will keep track of **how many books have been checked**. The other variable should have an **initial value of false**.



1. If the book you received from the argument **matches Annie's favorite book**, overwrite the value of the **boolean variable** and **break the loop**, otherwise **increment the counter by one**.



1. Depending on whether the book is found, **print the necessary messages**.



### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3496#0>

## Exam Preparation

Write a function with which Michael solves problems from his exams **until he receives** the **"Enough"** command from his lecturer. For each problem he solves, he receives a grade. **The function should finish when receiving the "Enough" command or if Michael receives the specified number of poor grades.**

**A poor grade is any grade that is less than or equal to 4.**

### Input Data

* **First argument – number of poor grades – an integer in the range [1…5]**
* **Then two arguments are read multiple times:**
  + **Problem name – text (string)**
* **Grade – an integer in the range [2…6]**

### Output Data

* If Michael receives the **"Enough"** command, **print on 3 lines:**
* **"Average score: {average score}"**
* **"Number of problems: {number of all problems}"**
  + **"Last problem: {last problem name}"**
* **If he receives the specified number of poor grades:**
* **"You need a break, {number of poor grades} poor grades."**

**The average score must be formatted to two digits after the decimal point.**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve([3,  "Money",  6,  "Story",  4,  "Spring Time",  5,  "Bus",  6,  "Enough"]) | Average score: 5.25  Number of problems: 4  Last problem: Bus | The number of poor grades allowed is 3.  The first problem is called Money, Michael's grade is 6.  The second problem is Story, Michael's grade is 4.  The third problem is Spring Time, Michael's grade is 5.  The fourth problem is Bus, Michael's grade is 6.  The next command is Enough, and the program ends.  Average score: 21 / 4 = 5.25  Number of solved problems: 4  Last problem: Bus |
| **Input** | **Output** | **Comments** |
| solve([2,  "Income",  3,  "Game Info",  6,  "Best Player",  4]) | You need a break, 2 poor grades. | The number of poor grades allowed is 2.  The first problem is called Income, Michael's grade is 3.  The second problem is Game Info, Michael's grade is 6.  The third problem is Best Player, Michael's grade is 4.  Michael has reached the allowed number of poor grades, time for a break. |

### Hints and Guidelines

1. Make **four help variables** at the beginning to keep track **of the number of good grades, the number of poor grades, the sum of all grades, and which is the last problem**. The first, second, and third variables have an initial **value of zero**. The fourth has an initial value of the **blank text**.
2. Create a **while** loop that continues until **the number of poor grades is less than the number you received from the argument**. On **each** iteration of the loop, take **the name of the problem and its grade**.
   1. In case you receive an Enough command, find **Michael's average score**, **print** the required messages, and **break the loop**.
3. At **each loop, add** Michael's grade to **the sum of all his grades** and increase the **grades** counter. If the grade is **less than or equal to 4** increase the counter for **poor grades**. Rewrite the name of the **last problem**.
4. After the loop, if the number of **poor grades** has reached the **maximum poor grades**, print the appropriate message.

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3496#1>

## Vacation

Jessie has decided to raise money for a vacation and wants you to help her find out **if she will be able to collect the required amount. She saves** or **spends some of her money every day**. If she wants **to spend more than her available money**, she will spend **everything she has and will be left with 0 USD**.

### Input Data

The function receives **2 arguments**:

* **Money needed for the vacation** – **a floating-point number in the range [1.00... .25000.00]**
* **Available money** – **a floating-point number in the range [0.00... 25000.00]**

**Then 2 arguments are read multiple times:**

* **Action type** – **text with "spend" and "save" options.**
  + **Amount to save/spend** – **a floating-point number in the range [0.01… 25000.00]**

### Output Data

The function must end in the following cases:

* If in **5 consecutive days** Jessie **only spends, the console displays:**
* **"You can't save the money."**
* **"{Total days passed}"**
* **If** **Jessie collects the money for the vacation, the console displays:**
* **"You saved the money for {Total days passed} days."**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve([2000**,**  1000**,**  **"**spend**",**  1200**,**  **"**save**",**  2000**])** | You saved the money for 2 days. | Money needed for the vacation: **2000**  Available money: 1000  spend - subtract the next number from the money  (1000 - 1200 = -200, which is less than 0  => available money = 0)  ~ consecutive days spend = 1  - total days: 1  save - add the next number to the money  (0 + 2000 = 2000)  ~ consecutive days spend = 0  - total days: 2  Available money (2000) >= Money needed for the vacation (**2000**) |
| solve([110,  60,  "spend",  10,  "spend",  10,  "spend",  10,  "spend",  10,  "spend",  10]) | You can't save the money.  5 | Money needed for the vacation: **110**  Available money: **60**  spend – subtract the next number from the money (60 - 10 = 50)  ~ consecutive days spend = 1  - total days: 1  spend – subtract the next number from the money (50 - 10 = 40)  ~ consecutive days spend = 2  - total days: 2  spend – subtract the next number from the money (40 - 10 = 30)  ~ consecutive days spend = 3  - total days: 3  spend – subtract the next number from the money (30 - 10 = 20)  ~ consecutive days spend = 4  - total days: 4  spend – subtract the next number from the money (20 - 10 = 10)  ~ consecutive days spend = 5  - total days: 5  5 consecutive days spend => available money: 10  Available money (10) < Money needed for the vacation (**110**) |
| solve([250,  150,  "spend",  50,  "spend",  50,  "save",  100,  "save",  100]) | You saved the money for 4 days. | Money needed for the vacation: **250**  Available money: 150  spend - subtract the next number from the money (150 - 50 = 100)  ~ consecutive days spend = 1  - total days: 1  spend - subtract the next number from the money (100 - 50 = 50)  ~ consecutive days spend = 2  - total days: 2  save - add the next number to the money (50 + 100 = 150)  ~ consecutive days spend = 0  - total days: 3  save - add the next number to the money (150 + 100 = 250)  ~ consecutive days spend = 0  - total days: 4  Available money (250) >= Money needed for the vacation (**250**) |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3496#2>

## 4. Walking

Emma wants to start a healthy lifestyle and has set a goal to walk **10,000 steps every day**. However, some days she is very tired from work and will want to get home before she reaches her goal. Write a function that **reads from an array how many steps she walks** each time she goes out during the day, and **when she reaches her goal, the function should print "Goal reached! Good job!"** and how many more steps she has walked **"{difference between steps} steps over the goal!"**

If she wants to **go home before then**, she will enter **the command "Going home"** and **enter the steps she has walked while going home**. Then, if she has failed to reach her goal, the console should print: **"{difference between steps} more steps to reach the goal."**

### Sample Input and Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| solve([1000,  1500,  2000,  6500]) | Goal reached! Good job!  1000 steps over the goal! | solve([1500,  300,  2500,  3000,  "Going home",  200"]) | 2500 more steps to reach goal. |
| **Input** | **Output** | **Input** | **Output** |
| solve([1500,  3000,  250",  1548,  2000,  "Going home",  2000]) | Goal reached! Good job!  298 steps over the goal! | solve([125,  250,  4000,  30,  2678,  4682]) | Goal reached! Good job!  1765 steps over the goal! |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3496#3>

## 5. Coins

Vending machine manufacturers wanted to make their machines return **as few change coins as possible**. Write a function solve(amount) that receives **an amount - the change** that needs to be returned and calculates **with how few coins this can be done**. The coins can be 2.00 EUR, 1.00 EUR, 0.50 EUR, 0.20 EUR, 0.10 EUR, 0.05 EUR, 0.02 EUR, 0.01 EUR.

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve(1.23) | 4 | Our change is 1,23 EUR. The machine returns it to us with 4 coins: a coin of 1 EUR, a coin of 0.20 EUR, a coin of 0.02 EUR, and a coin of 0.01 EUR. |
| solve(2) | 1 | Our change is 2.00 EUR. The machine returns it to us with 1 coin of 2 EUR. |
| solve(0.56) | 3 | Our change is 0.56 EUR. The machine returns it to us with 3 coins: a coin of 0.50 EUR, a coin of 0.05 EUR, and a coin of 0.01 EUR. |
| solve(2.73) | 5 | Our change is 2.73 EUR. The machine returns it to us with 5 coins: a coin of 2 EUR, a coin of 0.50 EUR, a coin of 0.20 EUR, a coin of 0.02 EUR, and a coin of 0.01 EUR. |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3496#4>

## 6. Cake

You're invited to a 30th birthday party, where the birthday boy serves a huge cake. However, he doesn't know **how many pieces the guests can take from it**. Your task is to write a function that calculates **the** **number of pieces** the guests have taken before it is eaten. Your function will receive **the** **dimensions of the cake** (width and length - **integers** in the range [1...1000]) and then until you receive the **STOP** command or **until the cake is finished**, the number of pieces the guests take from it.

**Note: One piece of cake is 1x1 cm.**

**Print one** of the following lines on the console:

* {number of pieces} pieces are left. - if you receive **STOP** and have not finished the cake pieces
* **No more cake left! You need {number of not enough pieces} pieces more.**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve[(10,  10,  20,  20,  20,  20,  21]) | No more cake left! You need 1 pieces more. | The cake is **10 cm** long and **10 cm** wide  => number of pieces = **10** \* **10** = **100**  1st take -> 100 - 20 = 80  2nd take -> 80 - 20 = 60  3rd take -> 60 - 20 = 40  4th take -> 40 - 20 = 20  5th take -> 20 - 21 = -1 < 0  => no more cake left, 1 piece is not enough |
| solve[(10,  2,  2,  4,  6,  "STOP"]) | 8 pieces are left. | The cake is **10 cm** long and **2 cm** wide  => number of pieces = **10** \* 2 = 20  1st take -> 20 - 2 = 18  2nd take -> 18 - 4 = 14  3rd take -> 14 - 6 = 8  4th take -> command STOP  => pieces left: 8 |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3496#5>

## Moving

On Jose's eighteenth birthday, he decided that he was going to move out to live in an apartment. He packed his stuff in **boxes** and found a suitable ad for an apartment to rent. He began moving his stuff **in pieces** because he couldn't carry it all at once. He has limited **free space** in his new apartment where he can place his stuff so that the place is suitable for living.

Write **a function that calculates the free volume of Jose's apartment that remains after he moves his stuff**.

**Note: One box is with exact size: 1m. x 1m. x 1m.**

### Input Data

Your function receives the following arguments:

1. **Free space width - an integer in the range [1...1000]**
2. **Free space length - an integer in the range [1...1000]**
3. **Free space height - an integer in the range [1...1000]**
4. Next **arguments (until the Done command is received) – the number of boxes to be moved to the apartment - integers in the range** **[1...10000];**

**The function should finish by receiving the Done command or if the free space runs out.**

### Output Data

**Print one** of the following lines on the console:

* If you receive the **Done** command and there is still space available:

**{number of free cubic meters} Cubic meters left.**

* If the free space runs out before the Done command is received:

**No more free space! You need {number of not enough cubic meters} Cubic meters more.**

**Sample Input and Output**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve([10,  10,  2,  20,  20,  20,  20,  122]) | No more free space! You need 2 Cubic meters more. | 10 \* 10 \* 2 = 200 cubic meters available 20 + 20 + 20 + 20 + 122 = 202 cubic meters 200 - 202 = 2 needed cubic meters |
| solve([10,  1,  2,  4,  6,  "Done"]) | 10 Cubic meters left. | 10 \* 1 \* 2 = 20 cubic meters available 4 + 6 = 10 cubic meters 20 - 10 = 10 cubic meters |

### Testing in the Judge System

Test the solution to this problem here: <https://judge.softuni.org/Contests/Compete/Index/3496#6>