

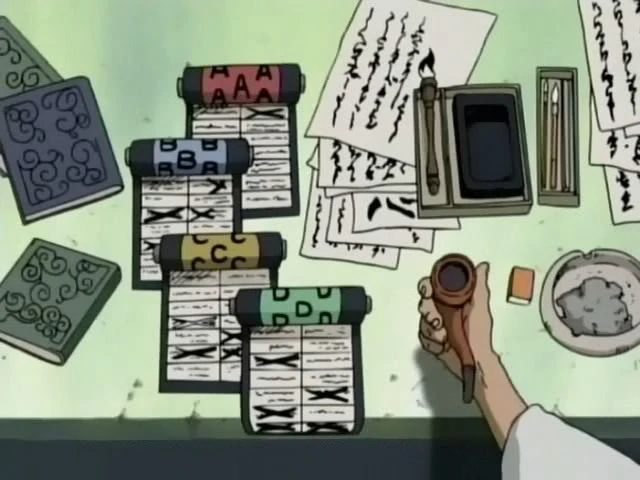


Welcome to CA’s Python Code Clinic 1’s Hands-on Exercise; Naruto Edition. In this exercise, you have 6 missions to complete within 1 hour and 30 minutes.

Each mission comes with several parts, although some parts can be done without completing the other. If you find yourself stuck at a particular mission, just skip and move on to the next one, it might be easier.

You may use any methods to solve these questions, although I would encourage you to stick to conventional solutions taught in class. Do NOT import any modules unless specified by the question. Refrain from using Google, you can’t do that in Lab Test anyway.

As you code, try to time yourself to solve it as fast as possible. The following timings are suggested for each difficulty.



| **D-Rank (\*)** | 5 minutes or less |
| --- | --- |
| **C-Rank (\*\*)** | 5 to 20 minutes (depending on difficulty) |
| **B-Rank (\*\*\*)** | 20 to 30 minutes, may be longer if there is more dialogue to read. |

You’re not ready for A and S ranked missions yet, saving it for Lab Test 2 preparation.

Good luck and stay safe!

***Stephen***

Click [here](https://drive.google.com/drive/folders/17oUnlVmRBqUV1YR2O2l4KLthXiTVywSI?usp=sharing) to download the source code files.

---------------------------------------------------------------------------------------------------------------------------

**Q1: Shuriken Toss**

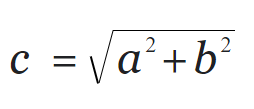
---------------------------------------------------------------------------------------------------------------------------



*“Shurikens are a ninja’s standard long-ranged throwing weapons. You need to use your coding knowledge to help improve the throwing capabilities of these ninjas”*

[**D-Ranked**] [PART A] In q1a.py, complete the function *get\_shuriken\_distance()* that takes in 2 parameters, *horizontal* and *vertical*, which represents the horizontal and vertical distance of the shuriken from the point of being thrown.

The function should return the shortest distance of the shuriken from its throwing point. You may test your code with the test cases provided in q1a.py

Hint: Use Pythagoras Theorem formula to help you solve the question. 

[**C-Ranked**] [PART B] In q1b.py, complete the function, *shoot\_shurikens()*. This function takes in a positive integer, which represents the number of shuriken being thrown.

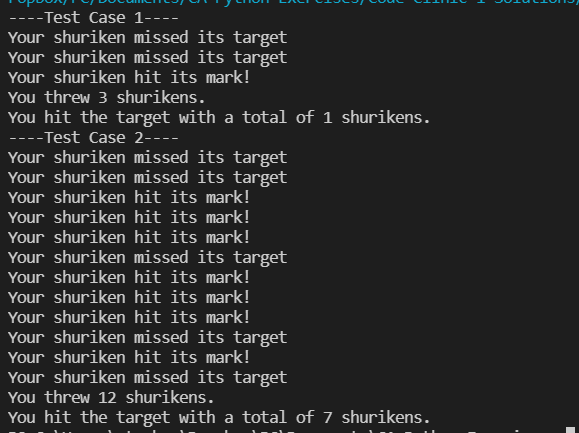
The function should process how many shurikens have hit the target. There should be a 50% chance that the shuriken hits the target, and a 50% chance that the shuriken misses the target. For each shuriken, the function should print whether the shuriken has hit or missed its target.

Once all shuriken has been thrown, the function should print the total number of shurikens thrown, as well as the number of shurikens that hit the target.

You may assume that the functions always takes in a positive integer.

Hint: Import the random module and make use of the method random.random() or random.randint().

If done correctly, running the test cases q1b.py should produce a similar output:



NOTE: The number of shurikens that hit the target may differ from your program, as it should be random. However, the total number of shurikens thrown should be the same.

---------------------------------------------------------------------------------------------------------------------------

**Q2: The Weapons Store**

---------------------------------------------------------------------------------------------------------------------------



*“The weapons store in the Hidden Leaf wants to learn how to improve their cashier service to speed up transactions. ”*

[**D-Ranked**] [PART A] In q2a.py, complete the function *get\_total\_cost()* that takes in 2 parameters:

* purchase\_list (a list containing strings, which represent items)
* shop\_list (a list of tuples, with each tuple containing 2 elements; the item name and its price

The function should return the **total cost** of the items in purchase\_list. You may assume that all the items in purchase\_list is present in shop\_list.s

Run the test cases in q2a.py to test your code.

[**D-Ranked**] [PART B] In q2b.py, complete the function *make\_purchase()* that takes in 3 parameters:

* item\_list (a list containing strings, which represent items)
* shop\_list (a list of tuples, with each tuple containing 2 elements, the item name and its price).
* current\_money (a float that represents the amount of money the person has)

The function should **return a tuple**; the first element is a **boolean** which represents whether the person can afford the items in item\_list, while the second element represents the money remaining in the form of a **integer**.

If the boolean is True, the money should be subtracted and the remaining amount is returned.

If the boolean is False, the money returned should be the same as what was inputted.

You will need to import the function that you have used in q2a, DO NOT copy and paste the function from q2a.py. Run q2b.py to test your code using the test cases given.

---------------------------------------------------------------------------------------------------------------------------

**Q3: Grading Chunin Exams**

---------------------------------------------------------------------------------------------------------------------------



*“Ibiki has several chunin exam scripts to mark, and he needs a way to determine the best and the worse ninja in the lot, as well as the overall average scores of all the exam takers. ”*

[**C-Ranked**] In q3.py, complete the function *get\_min\_average\_max()*. The function has a parameter tup\_list, which is a list of tuples. Each tuple contains 3 integers, which represents the 3 test scores of a particular ninja.

The function should **return a tuple containing 3 elements in the following order**:

1. The **tuple** containing the scores of the ninja who has the LOWEST total score.
2. The average total score of all the ninjas’ scores in the list.
3. The **tuple** containing the scores of the ninja who has the HIGHEST total score.

You are NOT allowed to use the min(), max(), sum() or avg() method for this question

You may test your code using the test cases in q3.py

---------------------------------------------------------------------------------------------------------------------------

**Q4: Hokage’s Paperwork Mess**

---------------------------------------------------------------------------------------------------------------------------



*“Tsunade and Shizune have so much paperwork to do and it seems endless. As the Hokage, she orders you to create some functions to help speed up her decoding and sorting. ”*

[**C-Ranked**] [PART A] Complete the function *decode\_calculate()*, which takes in a string as its parameter.

This string can contain alphabets (can be upper or lowercase), integers and symbols.

The function should perform 2 actions.

1. It should add or subtract all integers in the string. If there is a minus symbol directly before an integer (e.g. ‘-8’), that integer should be treated as a negative integer. You may assume that all integers in the string are treated as SINGLE integers (e.g. ‘35’ is considered as 3 and 5 when calculating)
2. It should form up a new string that consists of only alphabets and symbols that are not involved in the calculation. In other words, minus symbols that come before an integer (e.g. ‘-5’) should NOT be added to this string.

The function should then return a tuple with 2 elements. The first element should be the new string formed, while the second element should be the total addition and/or subtraction of all integers in the string.

For example, decode\_calculate('aS-3o5AkE2b05#6') would return ('aSoAkEb#', 15), as -3 + 5 + 2 + 5 +6 = 15

You may test your code by running q4a.py

[**C-Ranked**] [PART B] Tsunade realizes that there are more complex calculations that need to be done, and they are all in a mess!

Complete function *decode\_formula()* that takes in a string as its parameter, similar to part A.

The function should form an equation by retrieving the necessary numbers and operator symbols, and then **return the answer** of that equation.

Your code must account for the following operators:

* Addition ( + )
* Subtraction ( - )
* Multiplication ( \* )
* Division ( / )
* Floor Division ( // )

You must make the following assumptions

* There will always be at least one number before the operator symbol in the string.
* You may assume that both numbers are positive.
* There will one and only one operator in the string (e.g. if there is a ‘-’ in the string, there will not be any other operators like ‘+’ or ‘/’)
* For the floor division operator ( // ), there may be characters in between (e.g. ‘/gr@/’)

For example, decode\_formula('@ka55Nsa/Sh/3iRo') returns 18 as 55 // 3 = 18

You may test your code by running q4b.py

[**C-Ranked**] [PART C] Great! Now one thing Tsunade needs to do now is to sort her paychecks. She has one last task for you.

Complete the function *sort\_numbers()* that takes in num\_list, which is a list that contains at least one or more numbers.

Your function should sort the numbers in ascending order, and then **return** that newly sorted list.

You are NOT allowed to use any sorting methods (e.g. sort(), ksort() etc.), max() and min() methods.

You may test your cases by running q4c.py

---------------------------------------------------------------------------------------------------------------------------

**Q5: Transcribe the Scrolls**

---------------------------------------------------------------------------------------------------------------------------



*“There has been a discovery of several scrolls containing unknown writings. If we can find a way to understand what they mean, they may reveal to us secrets that will become very vital. Well? That’s your job now, to help decode them.”*

[**C-Ranked**] [PART A] Complete the function *transcribe\_numlist()* that takes in a string. This string is a list of integers in string form.

The function should return the list in the string in its list form, with all the elements inside it (Take a look at test case expected results). You are NOT allowed to use the list() method to help you.  
  
You may assume that the input is a proper list in string form. In other words, you will not see invalid lists like '[1,,,,32]]'.

Run q5a.py to test your code.

[**B-Ranked**] [PART B] Unfortunately, it looks like there are some scrolls that have been tampered with, and the numbers aren’t so visible.

Complete the function *transcribe\_problematic\_numlist()* that takes in a list in string form, similar to the input in Part A.

The function should return the list of numbers, similar to Part A, but this time you have to account for improper list formatting, such as invalid comma positions, or alphabets or symbols that are improperly added into the string.

You may assume that there are always only one set of square brackets [ ] to represent the list.

For example, transcribe\_problematic\_numlist('aP,,#[&3,140B,23,,S2,20aa,34^]') should return [3, 140, 23, 2, 20, 34], after removing all the invalid commas, as well as alphabets and symbols.

---------------------------------------------------------------------------------------------------------------------------

**Q6: Mysterious Roman Notes**

---------------------------------------------------------------------------------------------------------------------------



*“Mysterious characters have been discovered on paper bombs all over the village. This looks pretty dangerous, it’s your job to convert these characters into readable numbers in order to deactivate them. (By the way, those aren’t actually roman numerals in the picture but let’s just pretend they are). ”*

Roman numerals originated in ancient Rome as early as 900 B.C. They were developed out of a need to count beyond ten fingers. The counting system was developed based on a person’s hands.



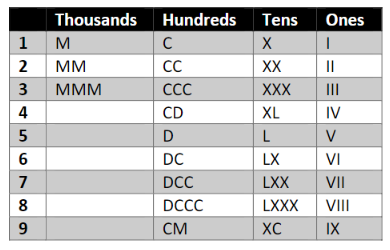
To read roman numerals, the numbers are formed by combining the letters and finding the sum of those values. The letters are placed left to right and the order determines whether their values are added or subtracted:

1. If a letter is placed after a letter of equal or greater value, add it. (e.g. VI = 5 + 1 = 6)
2. If a letter is placed before a letter of greater value, subtract it. (e.g. IV = 5 - 1 = 4)

There are other rules to observe:

1. Only C, X, I are used for subtraction, e.g. XCV = 95
2. Only one letter can be subtracted from another. E.g. VIII = 8
3. Cannot subtract a letter from another which is more than ten times greater e.g. XCIX = 99
4. No letter more than three times consecutively in a row (e.g. 4 should not be IIII, it should be IV).

The table below shows the well-formed numbers:



For this question, we will only focus on Roman numerals from 1 up to 3,999 inclusive. There are variants of the roman numeral system to represent 0, fractions and large numbers, but they are not to be considered for this question.

[**B-Ranked**] [PART A] Complete the function *convert\_to\_roman()* that takes in a positive integer, *num*.

The function should convert this integer into its roman numeral. It should then **return** the roman numeral in *string form*.

You may assume that the input is always a positive integer between 1 to 3999 inclusive.

You may take a look and run q6a.py to test your code.

HINT: You may want to break down your code into smaller functions to make it easier to code.

[**B-Ranked**] [PART B] Now that you know how to convert an integer to a roman numeral, how about the other way around?

Complete the function *convert\_to\_arabic()*. This function takes in a roman numeral in *string form*.

The function must then convert this roman numeral into its integer form, and **return** it.

You may assume that the roman numeral entered is always valid, and is of an integer between 1 to 3999 inclusive.

You may take a look and run q6b.py to test your code.

HINT: You may want to break down your code into smaller functions to make it easier to code.