**Call Base**

*It looks like our store accountant makes long distance calls too often and his phone bill has become much too expensive. We ought to control his use of the communications systems and calculate how much it costs.*

*Perhaps we should put him on a texting plan instead.*

The bill is represented as an array with information about the calls that our accountant has made. Write a function to calculate the cost of these calls.

Each call is represented as a string with the date, time and duration of the call in seconds in the follow format:

"YYYY-MM-DD hh:mm:ss duration"

The date and time in this information describes the start of the call.

Space-Time Communications Co. has several rules on how it calculates the cost of calls made on their network:

* First 100 (one hundred) minutes in one day are priced at 1 coin per minute;
* After the first 100 minutes in one day, each minute costs 2 coins per minute;
* All calls are rounded up to the nearest minute. For example 59 sec ≈ 1 min, 61 sec ≈ 2 min;
* Calls are billed based on the day when they began So if a call was started at 2014-01-01 23:59:59, then it counts as having started on 2014-01-01.

Here's an example communications bill:

2014-01-01 01:12:13 181

2014-01-02 20:11:10 600

2014-01-03 01:12:13 6009

2014-01-03 12:13:55 200

* First day -- 181s≈4m -- 4 coins;
* Second day -- 600s=10m -- 10 coins;
* Third day -- 6009s≈101m + 200s≈4m -- 100 + 5 \* 2 = 110 coins;
* Total -- 124 coins.

**Input:** Information about calls as a tuple of strings.

**Output:** The total cost as an integer.

**Example:**

total\_cost(("2014-01-01 01:12:13 181",

 "2014-01-02 20:11:10 600",

 "2014-01-03 01:12:13 6009",

 "2014-01-03 12:13:55 200")) == 124

**Precondition:**

0 < |calls| ≤ 30

0 < call\_duration ≤ 7200

The bill is sorted by datetime.

**How it is used:**

This mission will teach you how to parse and analyse various types data and encourages you to single out important fragments from a larger dataset.

**Most Numbers**

*To check an automated sieve for ore we use a variety of sample sets to find the smallest and the largest stones. The difference between these values is then used to decide if the sample is worth checking.*

You are given an array of numbers from which you must find the difference between the maximum and minimum elements. Your function should be able to handle an undefined amount of arguments. For an empty argument list, the function should return 0.

Floating-point numbers are represented in computer hardware as base 2 (binary) fractions, since this is the case, we should check that the result is within 0.001 precision.

**Input:** An arbitrary number of arguments as numbers (int, float).

**Output:** The difference between the maximum and minimum as a number (int, float).

**Example:**

most\_difference(1, 2, 3) == 2

most\_difference(5, -5) == 10

most\_difference(10.2, -2.2, 0, 1.1, 0.5) == 12.4

most\_difference() == 0

**Precondition:**

0 ≤ |arguments| ≤ 20

**How it is used:**

The important concept to learn from this mission is about passing an undefined amount of arguments to functions.

[Repository](https://github.com/Checkio-Game-Missions/checkio-empire-most-numbers.git)

#### Pangram

*A pangram (Greek:παν γράμμα, pan gramma, "every letter") or holoalphabetic sentence for a given alphabet is a sentence using every letter of the alphabet at least once. Perhaps you are familiar with the well known pangram "The quick brown fox jumps over the lazy dog".*

*But we've found one interesting usage of pangrams, we are using them to train our snipers focus.*

For this mission, we will use the latin alphabet (A-Z). You are given a text with latin letters and punctuation symbols. You need to check if the sentence is a pangram or not. Case does not matter.

**Input:** A text as a string.

**Output:** Whether the sentence is a pangram or not as a boolean.

**Example:**

check\_pangram("The quick brown fox jumps over the lazy dog.") == True

check\_pangram("ABCDEF.") == False

**Precondition:**

Input text can contain only ASCII letters, whitespaces and punctuation symbols.

0 < |text|

**How it is used:**

Pangrams have been used to display typefaces, test equipment, and develop skills in handwriting, calligraphy, and keyboarding for ages.