PROBLEM SOLVING

CODEWARS

8 kyu difficulty

Invert values

Given a set of numbers, return the additive inverse of each. Each positive becomes negatives, and the negatives become positives.

invert([1,2,3,4,5]) == [-1,-2,-3,-4,-5]

invert([1,-2,3,-4,5]) == [-1,2,-3,4,-5]

invert([]) == []

You can assume that all values are integers. Do not mutate the input array/list.

```
List<int>? invert(List<int> arr) {
  List<int> newList = [];
  if (arr.length > 0) {
    arr.forEach((element) {
      newList.add(-element);
    });
    return newList;
  } else {
    return [];
  }
}
```

Reduce but grow

Given a non-empty array of integers, return the result of multiplying the values together in order. Example:

[1, 2, 3, 4] => 1 * 2 * 3 * 4 = 24

```dart
int grow(List<int> arr) {
  int mult = arr[0];
  for (int i = 1; i < arr.length; i++) {
    mult *= arr[i];
  }
  return mult;
}
```

Square(n) Sum

Complete the square sum function so that it squares each number passed into it and then sums the results together.

```dart
int squareSum(List<int> numbers) {
  int sum = 0;
  numbers.forEach((element) {
    sum += (element * element);
  });
  return sum;
}
```

Century from year

Introduction

The first century spans from the year 1 up to and including the year 100, the second century - from the year 101 up to and including the year 200, etc.

Task

Given a year, return the century it is in.

```dart
int century(year) {
  return ((year / 100).ceil());
}
```

Is n divisible by x AND y?

Create a function that checks if a number n is divisible by two numbers x **AND** y. All inputs are positive, non-zero numbers.

```
bool isDivisible(int n, int x, int y) {
  if (n % x == 0 && n % y == 0) return true;
  return false;
}
```

Reversed String

Complete the solution so that it reverses the string passed into it.

```
'world'  =>  'dlrow'

'word'   =>  'drow'
```

```
String solution(String str) {
  return str.split("").reversed.join("");
}
```

Subtract the Sum

NOTE! This kata can be more difficult than regular 8-kyu katas (lets say 7 or 6 kyu)

Complete the function which get an input number n such that n >= 10 and n < 10000, then:

Sum all the digits of n.

Subtract the sum from n, and it is your new n.

If the new n is in the list below return the associated fruit, otherwise return back to task 1.

Example

n = 325

sum = 3+2+5 = 10

n = 325-10 = 315 (not in the list)

sum = 3+1+5 = 9

n = 315-9 = 306 (not in the list)

sum = 3+0+6 = 9

n =306-9 = 297 (not in the list)

...until you find the first n in the list below.

There is no preloaded code to help you. This is not about coding skills; think before you code.

- **Note that this problem teaches you to just notice things very well. If you noticed in the solution area in Codewars there's a comment in the function saying that (return something like "apple"), and guess what you just have to uncomment it!**

- **If you enter any number to the function, let's say 356 for example, then sum each digit to find them = 14, then subtract it from n: 356 – 14 = 342.**
- **Now if you sum every digit of 342, you'll find the result is 9.**
- **If you try it with any number again, you'll find the result in the end 9 too.**
- **And if you look at the list given in the problem, you'll notice that "apple" is in the index 9 and its multiplies.**
- **SO THE ANSWER IS JUST "apple".**

```
String subtractSum(n){
  return "apple";
}
```

Grasshopper – Summation

Write a program that finds the summation of every number from 1 to num. The number will always be a positive integer greater than 0.

For example (Input -> Output):

2 -> 3 (1 + 2)

8 -> 36 (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8)

```
int summation(int n) {
  int sum = 0;
  for (int i = n; i > 0; i--) {
    sum += i;
  }
  return sum;
}
```

Is it Palindrome?

Write a function that checks if a given string (case insensitive) is a palindrome.

```
bool isPalindrome(String x) {
  return x.split('').reversed.join('').toLowerCase() ==
x.toLowerCase();
}
```

Difference of Volumes of Cuboids

In this simple exercise, you will create a program that will take two lists of integers, a and b. Each list will consist of 3 positive integers above 0, representing the dimensions of cuboids a and b. You must find the difference of the cuboids' volumes regardless of which is bigger.

For example, if the parameters passed are ([2, 2, 3], [5, 4, 1]), the volume of a is 12 and the volume of b is 20. Therefore, the function should return 8.

Your function will be tested with pre-made examples as well as random ones.

If you can, try writing it in one line of code.

```
int findDifference(List<int> a, List<int> b) {
  return (a.reduce((value, element) => value * element) -
          b.reduce((value, element) => value * element))
      .abs();
}
```

Get the mean of an Array!

It's the academic year's end, fateful moment of your school report. The averages must be calculated. All the students come to you and entreat you to calculate their average for them. Easy! You just need to write a script.

Return the average of the given array rounded down to its nearest integer.

The array will never be empty.

```
int getAverage(List<int> arr) {
  return ((arr.reduce((value, element) => value + element)) /
arr.length)
      .floor();
}
```

Return the first name from a full name

```
void main() {
  print(firstName("Ahmed Elsayed Ghaly"));
}

String firstName(String fullName) {
  // Split the full name into an array of words
  List<String> words = fullName.split(' ');

  // Retrieve the first word which represents the first name
  String fName = words[0];

  return fName;
}
```

If you can’t sleep, just count sheep!

Given a non-negative integer, 3 for example, return a string with a murmur: "1 sheep...2 sheep...3 sheep...". Input will always be valid, i.e. no negative integers.

```dart
String countSheep(int numb) {
  String murmur = "";
  for (int i = 1; i <= numb; i++) {
    murmur += "${i} sheep...";
  }
  return murmur;
}
```

My head is at the wrong end!

You're at the zoo... all the meerkats look weird. Something has gone terribly wrong - someone has gone and switched their heads and tails around!

Save the animals by switching them back. You will be given an array which will have three values (tail, body, head). It is your job to re-arrange the array so that the animal is the right way round (head, body, tail).

Same goes for all the other arrays/lists that you will get in the tests: you have to change the element positions with the same exact logics.

["tail", "body", "head"] => ["head", "body", "tail"]

["lower legs", "torso", "upper legs"] => ["upper legs", "torso", "lower legs"])

```dart
List<String> fixTheMeerkat(List<String> arr) {
  return arr.reversed.toList();
}
```

Thinkful – Logic Drills: Traffic light

You're writing code to control your town's traffic lights. You need a function to handle each change from green, to yellow, to red, and then to green again.

Complete the function that takes a string as an argument representing the current state of the light and returns a string representing the state the light should change to.

For example, when the input is green, output should be yellow.

```
String updateLight(String current) {
  if (current == "green")
    return "yellow";
  else if (current == "yellow")
    return "red";
  else
    return "green";
}
```

Alan Partridge || - Apple Turnover

Alan is known for referring to the temperature of the apple turnover as Hotter than the sun!. According to space.com the temperature of the sun's corona is 2,000,000 degrees Celsius, but we will ignore the science for now.

Task: Your job is simple, if x squared is more than 1000, return It's hotter than the sun!!, else, return Help yourself to a honeycomb Yorkie for the glovebox.

Note: Input will either be a positive integer (or a string for untyped languages).

```
String apple(dynamic a) {
  if (a is String) {
    a = int.parse(a);
    if (pow(a, 2) > 1000) {
      return "It's hotter than the sun!!";
    } else {
```

```
      return "Help yourself to a honeycomb Yorkie for the
glovebox.";
    }
  } else if (a is int) {
    if (pow(a, 2) > 1000) {
      return "It's hotter than the sun!!";
    } else {
      return "Help yourself to a honeycomb Yorkie for the
glovebox.";
    }
  }
  return "";
}
```

Count the monkeys.

You take your son to the forest to see the monkeys. You know that there are a certain number there (n), but your son is too young to just appreciate the full number, he has to start counting them from 1.

As a good parent, you will sit and count with him. Given the number (n), populate an array with all numbers up to and including that number, but excluding zero.

For example (Input --> Output):

10 --> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

1 --> [1]

```
List<int> monkeyCount(int n) {
  List<int> list = [];
  for (int i = 1; i <= n; i++) {
    list.add(i);
  }
  return list;
}
```

Is the String uppercase?

Task

Create a method to see whether the string is ALL CAPS.

Examples (input -> output)

"c" -> False

"C" -> True

"hello I AM DONALD" -> False

"HELLO I AM DONALD" -> True

"ACSKLDFJSgSKLDFJSKLDFJ" -> False

"ACSKLDFJSGSKLDFJSKLDFJ" -> True

```
bool isUpperCase(String str) {
  if (str == str.toUpperCase()) return true;
  return false;
}
```

Count odd numbers below n

Given a number n, return the number of positive odd numbers below n, EASY!

Examples (Input -> Output)

7 -> 3 (because odd numbers below 7 are [1, 3, 5])

15 -> 7 (because odd numbers below 15 are [1, 3, 5, 7, 9, 11, 13])

Expect large Inputs!

```
int oddCount(n) {
  int counter = 0;
  for (int i = 1; i < n; i++) {
    if (!(i % 2 == 0)) {
      counter++;
    }
  }
  return counter;
}
```

Return negative

In this simple assignment you are given a number and have to make it negative. But maybe the number is already negative?

Examples

makeNegative(1); // return -1

makeNegative(-5); // return -5

makeNegative(0); // return 0

makeNegative(0.12); // return -0.12

Notes

- The number can be negative already, in which case no change is required.
- Zero (0) is not checked for any specific sign. Negative zeros make no mathematical sense.

```
num makeNegative(n) {
  if (n == 0 || n < 0)
    return n;
  else
    return n * -1;
}
```

Lost without a map

Given an array of integers, return a new array with each value doubled.

For example:

[1, 2, 3] --> [2, 4, 6]

```
List<int> maps(List<int> arr) {
  List<int> doubledArr = [];
  for (int i = 0; i < arr.length; i++) {
    doubledArr.add(arr[i] * 2);
  }

  return doubledArr;
}
```