# Node HW: 9

Nick Sotula • February 21, 2020

## Task 1:

### **Tic-Tac-Toe Checker**

If we were to set up a Tic-Tac-Toe game, we would want to know whether the board's current state is solved, wouldn't we? Our goal is to create a function that will check that for us!

Assume that the board comes in the form of a 3x3 array, where the value is 0 if a spot is empty, 1 if it is an "X", or 2 if it is an "O", like so:

```
[[0, 0, 1],
[0, 1, 2],
[2, 1, 0]]
```

We want our function to return:

- -1 if the board is not yet finished (there are empty spots),
- 1 if "X" won,
- 2 if "O" won,
- 0 if it's a cat's game (i.e. a draw).

2/21/2020 Node HW: 9 – Telegraph

You may assume that the board passed in is valid in the context of a game of Tic-Tac-Toe.

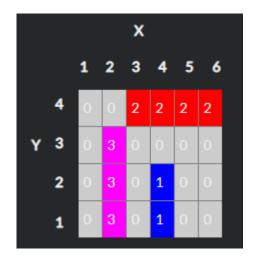
# Task 2:

# **Battle ships: Sunk damaged or not touched?**

Task

Your task in the kata is to determine how many boats are sunk damaged and untouched from a set amount of attacks. You will need to create a function that takes two arguments, the playing board and the attacks.

## **Example Game**



The board

Boats are placed either horizontally, vertically or diagonally on the board. 0 represents a space **not** occupied by a boat. Digits 1–3 represent boats which vary in length 1-4 spaces long. There will always be at least 1 boat up to a maximum of 3 in any one game. Boat sizes and board dimensions will vary from game to game.

#### Attacks

2/21/2020

Attacks are calculated from the bottom left, first the X coordinate then the Y. There will be at least one attack per game, and the array will not contain duplicates.

```
[[2, 1], [1, 3], [4, 2]];

First attack `[2, 1]` = `3`

Second attack `[1, 3]` = `o`

Third attack `[4, 2]` = `1`
```

#### **Function Initialization**

### **Scoring**

• 1 point for every whole boat sank.

2/21/2020 Node HW: 9 – Telegraph

- 0.5 points for each boat hit at least once (**not** including boats that are sunk).
- -1 point for each whole boat that was not hit at least once.

### **Sunk or Damaged**

- 'sunk' = all boats that are sunk
- 'damaged' = all boats that have been hit at least once but not sunk
- `notTouched/not\_touched` = all boats that have not been hit at least onceOutput

You should return a hash with the following data

```
`sunk`, `damaged`, `notTouched`, `points`
```

### **Example Game Output**

In our above example..

- First attack: `boat 3` was damaged, which increases the `points` by `o.5`
- Second attack: miss nothing happens
- Third attack: `boat 1` was damaged, which increases the `points` by `o.5`
- `boat 2` was untouched so `points -1` and `notTouched +1`
- No whole boats sank

### Return Hash

2/21/2020 Node HW: 9 – Telegraph

```
{ sunk: 0, damaged: 2 , notTouched: 1, points: 0 }
```

## Task 3:

## Random user

Сделайте приложение с помощью API randomuser.me.

В приложении выполните запрос 1000 человек и выведите на экран такую статистику:

- Самый молодой юзер (возраст, имя, фамилия)
- Самый старший юзер (возраст, имя, фамилия)
- Количество мужчин в выборке (количество)
- Количество женщин в выборке (количество)
- Средний возраст мужчин в выборке (возраст)
- Средний возраст женщин в выборке (возраст)
- Город с наибольшим количеством юзеров в выборке (город, количество)