Table of Contents

1. Day 1 1

2. Day 2 1

3. Day 3 3

Code Bytes Daily Challenge Explanation (Python)

Robertus Diawan Chris

1. Day 1

The challenge for day one is reverse string from an input.

On this challenge i use slice string to reverse string (to print the string text backward). Slice string have this syntax `[start:stop:step]` and giving no value as start and stop indicates default to 0 as start and string length as stop, and also if we assign step to "-1" denotes starting from the very end.

So for this challenge the main code is `string[::-1]`. I solve this challenge using external file as an input, the code is below:

```
f = open('input', 'r')

string = f.readlines()

for s in string:
    print(s[::-1])
```

inside input file there's a string

Cat

The Daily Byte

civic

and using readlines method we can read each sentence in different lines.

2. Day 2

The challenge for day two is palindrome case. It's basically check if a sentence read the same forward and backward.

For this challenge i solve it by comparing the actual string and the reverse string version, if it's the same then it's polindrome. I turn all the string into lowercase and remove all symbols before comparing actual string and reverse string. Here is the code: `from fire import Fire`

```
def main(input):
    input_without_symbols = input.replace(',', '').replace(':', '').re-
place(' ', '').replace('.', '')

    print(f'{input} -> true') if input_without_symbols.lower() == in-
put_without_symbols.lower()[::-1] else print(f'{input} -> false')

if __name__ == '__main__':
    Fire(main)
```

I use fire python module to give an input in command line interface, you could give the list directly inside the code if you want. I'm still learning about regular expression so i'm not sure how to implement regex in this case for now.

3. Day 3

The challenge for day three is how to determine if a robot vacuum cleaner return to the original position.

U -> up

D -> down

L -> left

R -> Right

So first thing first we need to know that to back to original position we need to define the variable of the moves and the oposite of the move, for example, the oposite of up is down so if the robot going up then the we're gonna add the value of up-down variable and if the robot going down then we're gonna minus the value of up-down variable so if the robot back to original position then the value of up-down varible should be 0.

With that in mind, here's the code:

```
from fire import Fire

def main(moves):
    UD = 0 #up and down variable
    LR = 0 #left and right variable

    for m in moves:
        if m == 'U':
            UD += 1
        elif m == 'D':
            UD -= 1
        elif m == 'L':
            LR += 1
        elif m == 'R':
            LR -= 1

    print(UD == 0 and LR == 0)

if __name__ == '__main__':
    Fire(main)
```