Advent of Code 2025 😀

Contents

day 1

Problem summary

Given a **sequence** of dial **rotations** on a **circular scale from 0 to 99**, **starting at** position **50**, determine **how often** the dial **reaches position 0** after applying each rotation in order.

Mathematical formulation

- Let the dial positions be elements of the cyclic group $\mathbb{Z}_{100}$.
- Let the initial position be $x_0 = 50$.
- Let the sequence of n rotations be (d_i, k_i) with $d_i \in \{L, R\}$ and $k_i \in \mathbb{N}$.

Define the signed rotation

$$s_i = \begin{cases} -k_i \text{ if } d_i = L \\ k_i \text{ if } d_i = R \end{cases}$$

and the position update

$$x_i \equiv x_{i-1} + s_i \pmod{100}, \quad i = 1, \ldots, n.$$

The password is the number of indices i for which the dial reaches zero

$$P = |\{i \in \{1, \ldots, n\} : x_i = 0\}|.$$

Python3

python

```
>>> xs = '\nL68\nL30\nR48\nL5\nR60\nL55\nL1\nL99\nR14\nL82\n'
>>>
>>> temp, count = 50, 0
>>>
>>> list( map(int, xs.replace("L", "-").replace("R", "+").split())  )
[-68, -30, 48, -5, 60, -55, -1, -99, 14, -82]
```

↳ Define the signed rotation

$$s_i = \begin{cases} -k_i \text{ if } d_i = L \\ k_i \text{ if } d_i = R \end{cases}$$

```
>>>
>>> for r in map(int, xs.replace("L", "-").replace("R", "+").split()):
...     temp = (temp + r) % 100
```
↳ and the position update

$$x_i \equiv x_{i-1} + s_i \pmod{100}, \quad i = 1, \ldots, n.$$

```
...     if temp == 0: count += 1
```
↳ The password is the number of indices i for which the dial reaches zero

$$P = |\{i \in \{1, \ldots, n\} : x_i = 0\}|.$$

```
...
>>> print(count)
3
>>>
>>> (lambda xs: (
...     lambda pos, c:
...         ([(pos := (pos + r) % 100,
...            c := c + (pos == 0))
...           for r in map(int, xs.replace("L", "-").replace("R", "+").split())],
...          c)[1]
... )(50, 0))(xs)
3
```

typst

typst

```
#let day1(xs) = {
  let steps = xs.replace("L", "-").replace("R", "+").split().map(int)
  let result = steps.fold((50, 0), (acc, r) => {
    let pos = calc.rem((acc.at(0) + r), 100)
    let cnt = acc.at(1) + int(pos == 0)
    (pos, cnt)
  })
  result.at(1)
}
```

typst

```
#day1("L68 L30 R48 L5 R60 L55 L1 L99 R14 L82")
```

3

Docs

day1

Day 1: Secret Entrance

$(\mathrm{L68}, \mathrm{L30}, \mathrm{R48}, \mathrm{L5}, \mathrm{R60}, \mathrm{L55}, \mathrm{L1}, \mathrm{L99}, \mathrm{R14}, \mathrm{L82})^{\mathsf{T}} \to 3$

```
#day1("L68 L30 R48 L5 R60 L55 L1 L99
R14 L82")
```

```
3
```

Parameters

```
day1(xs: string) -> int
```

xs `string`

The argument

Part Two

py

```
>>> for r in map(int, xs.replace("L", "-").replace("R", "+").split()):
...     temp = (temp + r) % 100
...     if temp == 0: count += 1
```

↧

py

```
>>> temp, count = 50, 0
>>>
>>> for r in map(int, xs.replace("L", "-").replace("R", "+").split()):
...     if r >= 0:
...         for _ in range(r):
...             temp = (temp + 1) % 100
...             if temp == 0: count += 1
...     else:
...         for _ in range(r * (-1)):
...             temp = (temp - 1) % 100
...             if temp == 0: count += 1
...
>>> print(temp, count)
28 6
```

⇓

py

```
for r in map(int, xs.replace("L", "-").replace("R", "+").split()):
    for _ in range(abs(r)):
        temp = (temp + (1 if r > 0 else -1)) % 100
        count += temp == 0
```

Day 2: Gift Shop

py

```py
from day2 import xs

# [range(*list(map(int, e.split("-")))) for e in xs.split(",")]

ds = [range(*(lambda ys: [ys[0], ys[1] + 1])(list(map(int, e.split("-"))))) for e in xs.split(",")]


def p(x: int) -> int:
  x = str(x)

  if len(x) % 2 != 0:
    return 0

  # // ist leider nötig, aber muss eh das glieche wie / sein ...
  return 0 if x[:len(x)//2] == x[len(x)//2:] else int(x)

sum(sum(map(p, d)) for d in ds)
```