

Advent of Code 2025 in Typst 😊

Contents

Advent of Code 2025 in Typst 😊	1
day 1	1
Problem summary	1
Mathematical formulation	1
Mathematische Formulierung	1
Python3	1
typst	3

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

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

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

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

Mathematische Formulierung

Die Drehscheibe besitzt die Positionen des zyklischen Raums \mathbb{Z}_{100} . Der Anfangswert ist $x_0 = 50$.

Die Eingabe sei eine Folge von ganzen Zahlen r_1, r_2, \dots, r_n , wobei jede Zahl bereits ein Vorzeichen trägt. Ein Eintrag ist negativ, falls die ursprüngliche Richtung L war, und positiv, falls die ursprüngliche Richtung R war.

Die Aktualisierung der Position erfolgt durch

$$x_i \equiv x_{i-1} + r_i \pmod{100} \quad \text{für } i = 1, \dots, n.$$

Das Passwort ergibt sich aus der Anzahl der Schritte, in denen die Position den Wert null annimmt:

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

Python3

python

```

1 >>> from main import xs
2 >>>
3 >>> xs
4 '\nL68\nL30\nR48\nL5\nR60\nL55\nL1\nL99\nR14\nL82\n'
5 >>>
6 >>> xs.split()
7 ['L68', 'L30', 'R48', 'L5', 'R60', 'L55', 'L1', 'L99', 'R14', 'L82']
8 >>>
9 >>> xs.replace("L", "+")
10 '\n+68\n+30\nR48\n+5\nR60\n+55\n+1\n+99\nR14\n+82\n'
11 >>>
12 >>> xs.replace("L", "+").replace("R", "-")
13 '\n+68\n+30\n-48\n+5\n-60\n+55\n+1\n+99\n-14\n+82\n'
14 >>>
15 >>> xs.replace("L", "+").replace("R", "-").split()
16 ['+68', '+30', '-48', '+5', '-60', '+55', '+1', '+99', '-14', '+82']
17 >>>
18 >>> [int(e) for e in xs.replace("L", "+").replace("R", "-").split()]
19 [68, 30, -48, 5, -60, 55, 1, 99, -14, 82]
20 >>>
21 >>> list(map(int, xs.replace("L", "+").replace("R", "-").split()))
22 [68, 30, -48, 5, -60, 55, 1, 99, -14, 82]
23 >>>
24 >>> (50 + 68) % 100
25 18
26 >>>
27 >>> (50 - 68) % 100
28 82
29 >>>
30 >>> list(map(int, xs.replace("L", "-").replace("R", "+").split()))
31 [-68, -30, 48, -5, 60, -55, -1, -99, 14, -82]
    ~> Define the signed rotation


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


32 >>>
33 >>> for r in map(int, xs.replace("L", "-").replace("R", "+").split()):
34 ...     print(temp)
35 ...     temp = (temp + r) % 100
    ~> and the position update


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


36 ...     print(temp)
37 ...     if temp == 0: count += 1
    ~> The password is the number of indices  $i$  for which the dial reaches zero


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


38 ...     print(count)
39 ...
40 50
41 82
42 0
43 82
44 52
45 0
46 52
47 0
48 1

```

```

49 0
50 95
51 1
52 95
53 55
54 1
55 55
56 0
57 2
58 0
59 99
60 2
61 99
62 0
63 3
64 0
65 14
66 3
67 14
68 32
69 3
70 >>>
71 >>>
72 >>> (lambda xs: (
73 ...     lambda pos, c:
74 ...        ([(pos := (pos + r) % 100,
75 ...             c := c + (pos == 0))
76 ...             for r in map(int, xs.replace("L", "-").replace("R",
77 ...                 "+").split()))],
78 ...             c)[1]
79 ... )(50, 0))(xs)
80 3
81 >>>

```

typst

typst

```

1 #let count-zeros(xs) = {
2   let steps = xs.replace("L", "-").replace("R", "+").split().map(int)
3   let result = steps.fold((50, 0), (acc, r) => {
4     let pos = calc.rem((acc.at(0) + r), 100)
5     let cnt = acc.at(1) + int(pos == 0)
6     (pos, cnt)
7   })
8   result.at(1)
9 }

```

typst

```
1 #count-zeros("L68 L30 R48 L5 R60 L55 L1 L99 R14 L82")
```

3