A briefcase has a 4-digit rolling-lock. Each digit is a number from 0-9 that can be rolled either forwards or backwards.

Create a function that returns the smallest number of turns it takes to transform the lock from the current combination to the target combination. One turn is equivalent to rolling a number forwards or backwards by one.

To illustrate:

* current-lock: 4089
* target-lock: 5672

What is the minimum number of turns it takes to transform 4089 to 5672?

def min\_turns(current, target):

turns = 0

for i in range(4):

current\_digit = int(current[i])

target\_digit = int(target[i])

clockwise\_distance = (current\_digit - target\_digit + 10) % 10

counterclockwise\_distance = (target\_digit - current\_digit + 10) % 10

min\_distance = min(clockwise\_distance, counterclockwise\_distance)

turns += min\_distance

return turns

We define a function called min\_turns that takes two arguments: current\_lock (the current lock combination) and target\_lock (the target lock combination).

# Inside the function, we initialize a variable turns to keep track of the total number of turns required to reach the target lock combination. We start with turns = 0.

# We use a for loop to iterate through each digit position (from left to right) in the lock combination. In your case, the lock has 4 digits, so we iterate from i = 0 to i = 3.

# For each digit position, we extract the current digit from current\_lock and the corresponding digit from target\_lock. We convert these digits to integers for numerical comparison.

# Next, we calculate the clockwise and counterclockwise distances between the current digit and the target digit. Clockwise distance is the number of turns needed to move from the current digit to the target digit by rolling forward, and counterclockwise distance is the number of turns needed to move in the opposite direction (rolling backward).

# To ensure that we always take the shorter path, we calculate both clockwise and counterclockwise distances and take the minimum of the two using the min function.

# We add the minimum distance calculated in step 6 to the turns variable to accumulate the total number of turns needed to reach the target digit for the current position.

# We repeat steps 4-7 for all four digit positions in the lock combination, and the turns variable accumulates the total minimum number of turns needed to transform the entire lock from the current combination to the target combination.

# Finally, we return the turns value, which represents the minimum number of turns needed to reach the target lock combination.

# In the example you provided with current\_lock = "4089" and target\_lock = "5672", the function calculates the minimum number of turns required to change 4089 to 5672, and the result is printed as "Minimum turns: 13," indicating that it takes 13 turns to reach the target lock combination.