121. Best time to buy and Sell Stock

You are given an array prices where prices[i] is the price of a given stock on the i^{th} day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.

Example 1:

Input: prices = [7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Example 2:

Input: prices = [7,6,4,3,1]

Output: 0

Explanation: In this case, no transactions are done and the max profit = 0.

Constraints:

- 1 <= prices.length <= 10^5
- 0 <= prices[i] <= 10^4

Code:

```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        min_price=prices[0]
        max_profit=0
        for price in prices:
            profit=price-min_price
            max_profit=max(max_profit,profit)
            min_price=min(min_price,price)
        return max_profit
```

To maximize profit, we need to find the maximum difference between two elements in the array such that the larger element appears after the smaller element. This essentially means finding the lowest valley followed by the highest peak in the array.

We can do this by iterating through the array and keeping track of the minimum price seen so far. Then, for each element, we calculate the profit if we were to sell on that day (current price minus minimum price seen so far). If this profit is greater than the current maximum profit, we update the maximum profit.

The time complexity of the given code is O(n), where n is the number of elements in the prices list.