# Data Structures and Algorithms By Umar Saad

# Table of contents

# Strategy for solving a problem

1. State the problem clearly
2. Come up with some example in/outputs, trying to cover all edge cases
3. Come up with a solution
4. Implement the solution using 3. and fix bugs if any
5. Understand the complexity and identify inefficiencies
6. Fix the inefficiency, repeat steps 3-6

The problem at hand:

Alice has some cards with numbers written on them. She arranges the cards in decreasing order, and lays them out face down in a sequence on a table. She challenges Bob to pick out the card containing a given number by turning over as few cards as possible.Write a function to help Bob locate the card.

1a. Stating the problem clearly

Essentially the problem is we need to find the specific card’s location **(index)** which holds the target **(target)** from a list of numbers **(nums)** that are in decreasing order with the least amount of card checks possible.

1b. Analysing the inputs and Outputs

Inputs

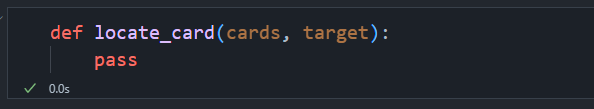
* List of the card: **Nums**
* The number we want to find: **Target**

Ouputs

* The index of the card which holds target: **Position**

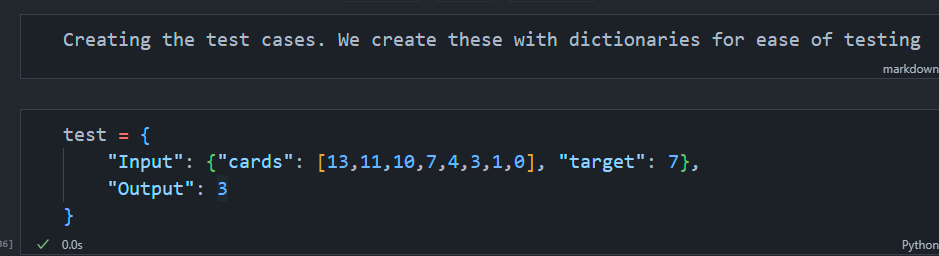
1c. Now we can frame the function for out solution

* Ensure we use proper naming conventions for functions and variables
* Ensure we understand the question (ask)

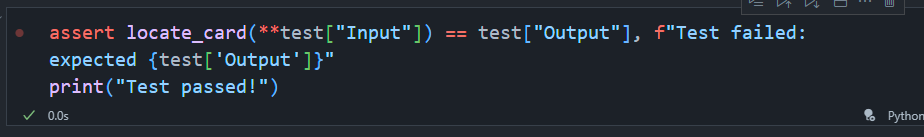


2. Creating Example input and outputs (Test Cases)

Now create the test case expected inputs and outputs as dictionaries



So we can test our function with



However our function should be able to handle ANY set of valid inputs, some of the potential edge cases are:

* Target is in the beginning, middle or end
* The list only contains one element, which is target
* The list doesn't contain target
* The list contains repeating numbers
* The target appears more than once

To solve these test cases we create a list called tests which holds each test for the edge cases

Stopped at 25:18