# Solution Review: Integer Square Root

This lesson contains the solution review for the challenge to find the largest integer whose square is less than or equal to the given integer.

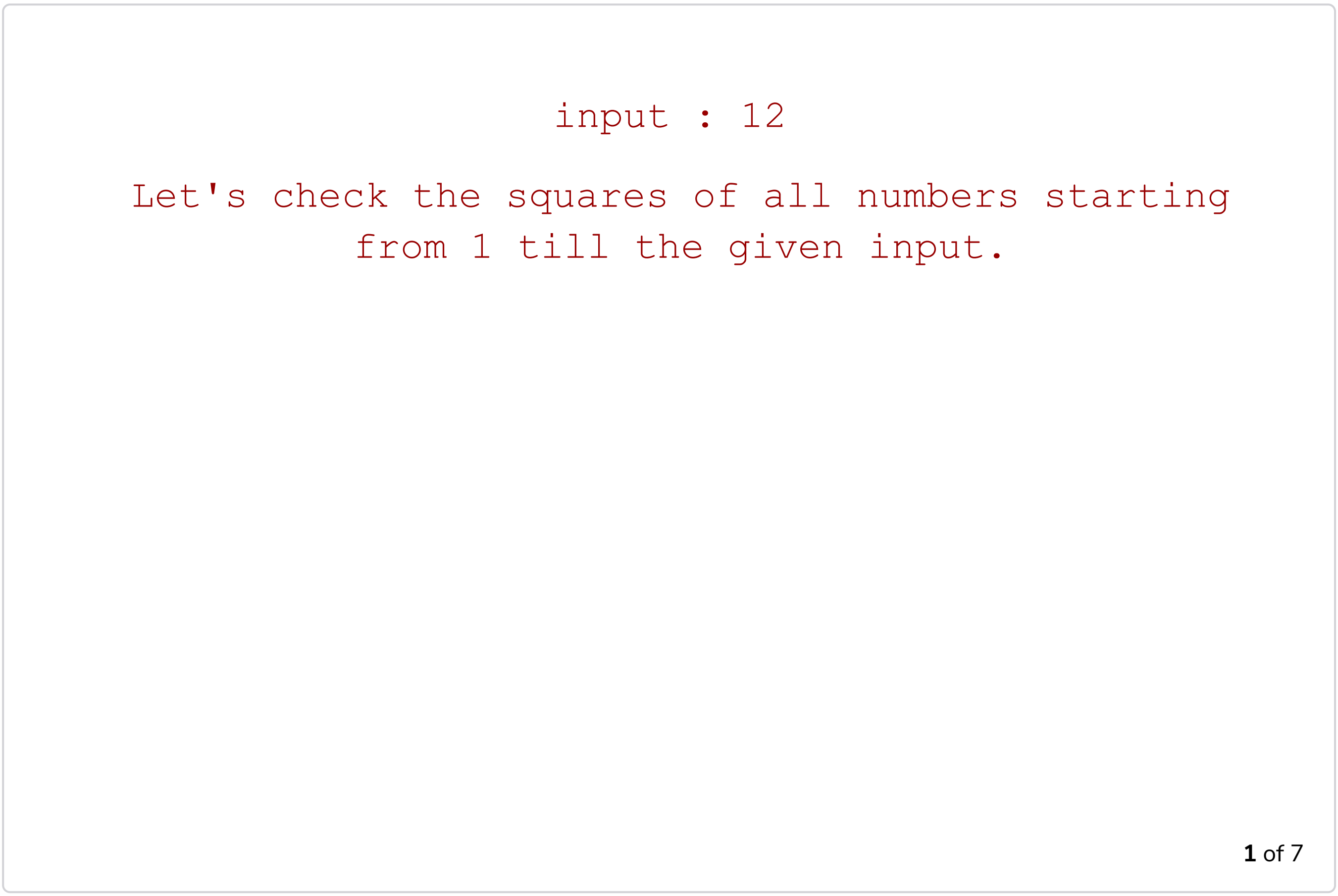
**We'll cover the following**

First of all, let's repeat the problem statement.

You are required to write a function that takes a non-negative integer, `k`, and returns the largest integer whose square is less than or equal to the specified integer `k`.

## Algorithm #

The naive approach to solve this problem is to start from `1` and check the square of every number up until `k`. Have a look at the slides below:



input : 12

Let's check the squares of all numbers starting from 1 till the given input.

**1** of 7

Now we want to improve things. Let’s think in terms of binary search. One thing that we can note from the above example is that if we are on integer 3, which squares up to 9, decreasing the integer 3 will not take us anywhere near 12. On the other hand, if we are on integer 4, which squares up to 16, increasing the number will not take us closer to 12. We can make use of this observation and tweak the binary search algorithm to solve our problem efficiently. This observation enables us to reduce our search span.

## Implementation #

Let’s have a look at the code below:

```
def integer_square_root(k):

    low = 0
    high = k

    while low <= high:
        mid = (low + high) // 2
        mid_squared = mid * mid

        if mid_squared <= k:
            low = mid + 1
        else:
            high = mid - 1

    return low - 1

k = 300
print(integer_square_root(k))
```

## Explanation #

On **lines 3-4**, `low` and `high` are initialized to `0` and `k`. The `while` loop on **line 6** will terminate when `low` becomes greater than `high`. In the next line, we calculate `mid` as we have always done while implementing the binary search. Additionally, we take the square of `mid` and store it in `mid_squared` on **line 8**. Now we need to compare `mid_squared` with `k`. If `mid_squared` is less than or equal to `k`, we have to discard all the numbers less than `mid`. Therefore, we set `low` equal to `mid + 1`. On the other hand, if `mid_squared` is greater than `k`, then all the numbers greater than `mid` will not be useful in our search, so we set `high` equal to `mid-1` (**line 13**). Finally, after the `while` loop terminates, `low - 1` will be the answer we are looking for, i.e., the largest integer whose square is less than or equal to `k`.

I hope you enjoyed this challenge. We have another waiting for you in the next lesson. All the best!