Solution Review 2: Return an Array of Squares

This lesson gives a detailed solution review to the problem in the previous lesson.

We'll cover the following

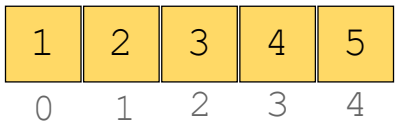
Solution:

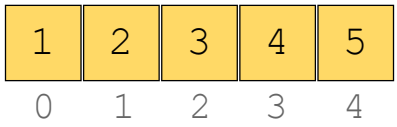
```
fn arr_square() -> [i32;5] {
    let mut square:[i32;5] = [1, 2, 3, 4, 5]; // mutable array
    for i in 0..5 {  // compute the square of each element
        square[i] = square[i] * square[i];
    }
    square
}
fn main(){
    println!("Updated Array : {:?}",arr_square());
}
```

Explanation

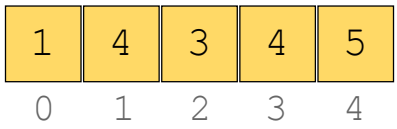
- On **line 2**, a mutable array `square` of **type** `i32` **and size** `5` is initialized with elements 1 , 2 , 3 , 4 ,5.
- On **line 3**, a `for` loop takes a variable `i` that iterates over the elements of the array `square` and squares each element and updates the `square` array on **line 4**.

The following illustration shows how the square of an element of an array is calculated using a `for` loop:

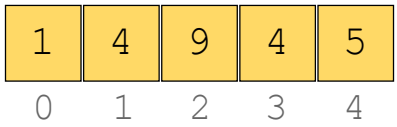




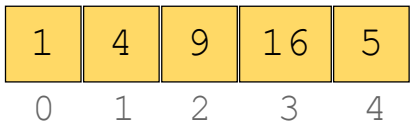

```
arr[0] = arr[0]*arr[0]
arr[0] = 1 * 1
```


```
arr[1] = arr[1]*arr[1]
arr[1] = 2 * 2
```

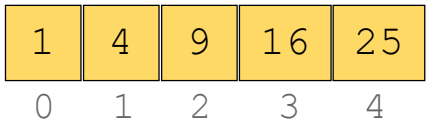




```
arr[2] = arr[2]*arr[2]
arr[2] = 3*3
```




```
arr[3] = arr[3]*arr[3]
arr[3] = 4*4
```

5 of 6




```
arr[4] = arr[4]*arr[4]
arr[4] = 5*5
```

6 of 6

You have now learned about functions that are invoked from main function or other functions. What if the function calls itself? Let's learn about recursive functions in the next lesson.