Arrays

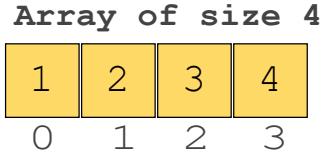
This lesson introduces you to a compound data type, Arrays.

We'll cover the following

- What Is an Array?
- Define an Array
- Access an Element of an Array
- How to Make an Array Mutable?
- Print the Array
- Get the Length of the Array
- Get Slice
  - Syntax
- Quiz

What Is an Array?

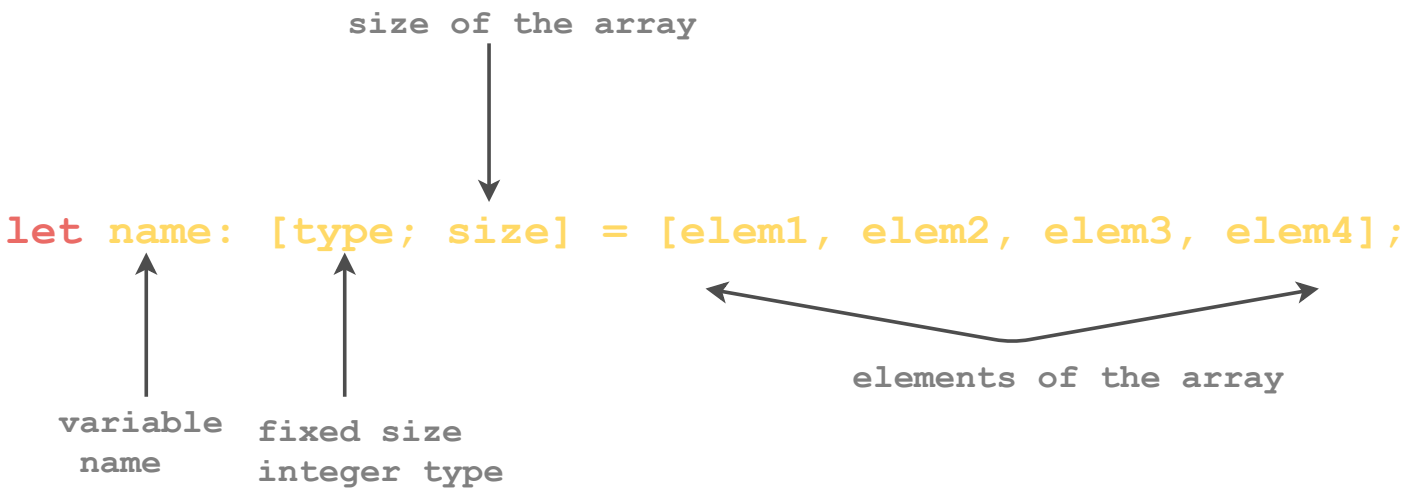
An array is a **homogenous sequence of elements**. Being a compound type, it is used when the collection of values of the same type are to be stored in a single variable. In Rust, an array can only be of a fixed length. Like all other languages, each element in the array is assigned an index. By default, the first element is always at index 0.



> **Note:** By default, arrays are immutable.

Define an Array

To define an array in Rust, we have to define the type and size of the array. To initialize an array, the array elements are enclosed in square brackets `[]`. The following illustration explains the concept:




```
#[allow(unused_variables, unused_mut)]
fn main() {
   //define an array of size 4
   let arr:[i32;4] = [1, 2, 3, 4];
   // initialize an array of size 4 with 0
   let arr1 = [0 ; 4];
}
```

- The array `arr` declaration on **line 4** declares an array with elements $1, 2, 3, 4$.
- The array `arr1` declaration on **line 6** implicitly determines the data type (integer) from the value 0 and 4 is the size of the array. So, this becomes an array consisting of 4 zeros.

Access an Element of an Array

Any value of the array can be accessed by writing the array name followed by the index number enclosed within square brackets `[ ]`.

```
fn main() {
   //define an array of size 4
   let arr:[i32;4] = [1, 2, 3, 4];
   //print the first element of array
   println!("The first value of array is {}", arr[0]);
   // initialize an array of size 4 with 0
   let arr1 = [0; 4];
   //print the first element of array
   println!("The first value of array is {}", arr1[0]);
}
```

How to Make an Array Mutable?

Just like a variable becomes mutable by adding the `mut` keyword after let, the same goes for an array.

```
fn main() {
    //define a mutable array of size 4
    let mut arr:[i32;4] = [1, 2, 3, 4];
    println!("The value of array at index 1: {}", arr[1]);
    arr[1] = 9;
    println!("The value of array at index 1: {}", arr[1]);
}
```

Print the Array

The whole array can be traversed using a *loop* or the *debug trait*.

> The arrays elements can be traversed using loops, which will be discussed in chapter 7.

```
fn main() {
    //define an array of size 4
    let arr:[i32;4] = [1, 2, 3, 4];
    //Using debug trait
    println!("\nPrint using a debug trait");
    println!("Array: {:?}", arr);
}
```

Get the Length of the Array

To access the length of the array, use the built-in function `len`.

```
fn main() {
    //define an array of size 4
    let arr:[i32;4] = [1, 2, 3, 4];
    // print the length of array
    println!("Length of array: {}", arr.len());
}
```

Get Slice

Slice is basically a portion of an array. It lets you refer to a subset of a contiguous memory location. But unlike an array, the size of the slice is not known at compile time.

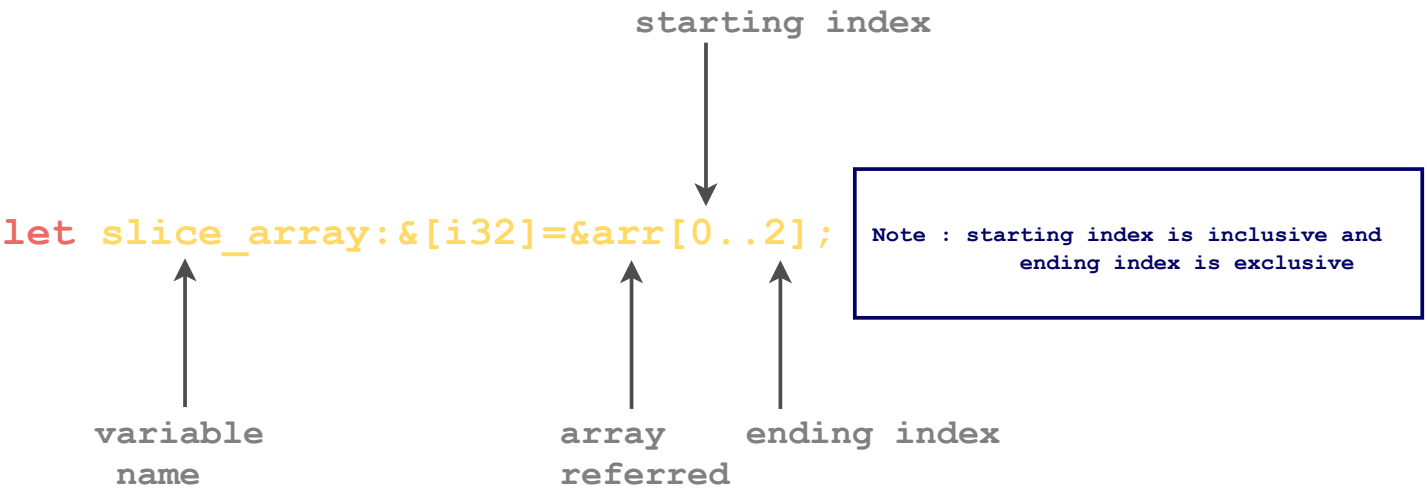
Syntax

A slice is a two-word object, the first word is a data pointer and the second word is a slice length.

> Data pointer is a programming language object that points to the memory location of the data, i.e., it stores the memory address of the data.

To declare an array slice, we need to specify the name of the source array and the range of elements to be included in the slice.

> **Note:** If the range of elements is not specified, it will consider the whole array as a slice.




```
fn main() {
    //define an array of size 4
    let arr:[i32;4] = [1, 2, 3, 4];
    //define the slice
    let slice_array1:&[i32] = &arr;
    let slice_array2:&[i32] = &arr[0..2];
    // print the slice of an array
    println!("Slice of an array: {:?}", slice_array1);
```

```
    println!("Slice of an array: {:?}", slice_array2);
}
```

Quiz

Test your understanding on arrays in Rust!

Quick Quiz on Arrays!

1 How can you define an array in rust?

2 What is the output of the following code?

```
let arr:[i32;4] = [1,2,3,4];
let slice_array:&[i32] = &arr[0..1];
println!("Slice of an array : {:?}", slice_array);
```

Retake Quiz

Now that you have learned about arrays, let’s move on to the next lesson “Tuples”.