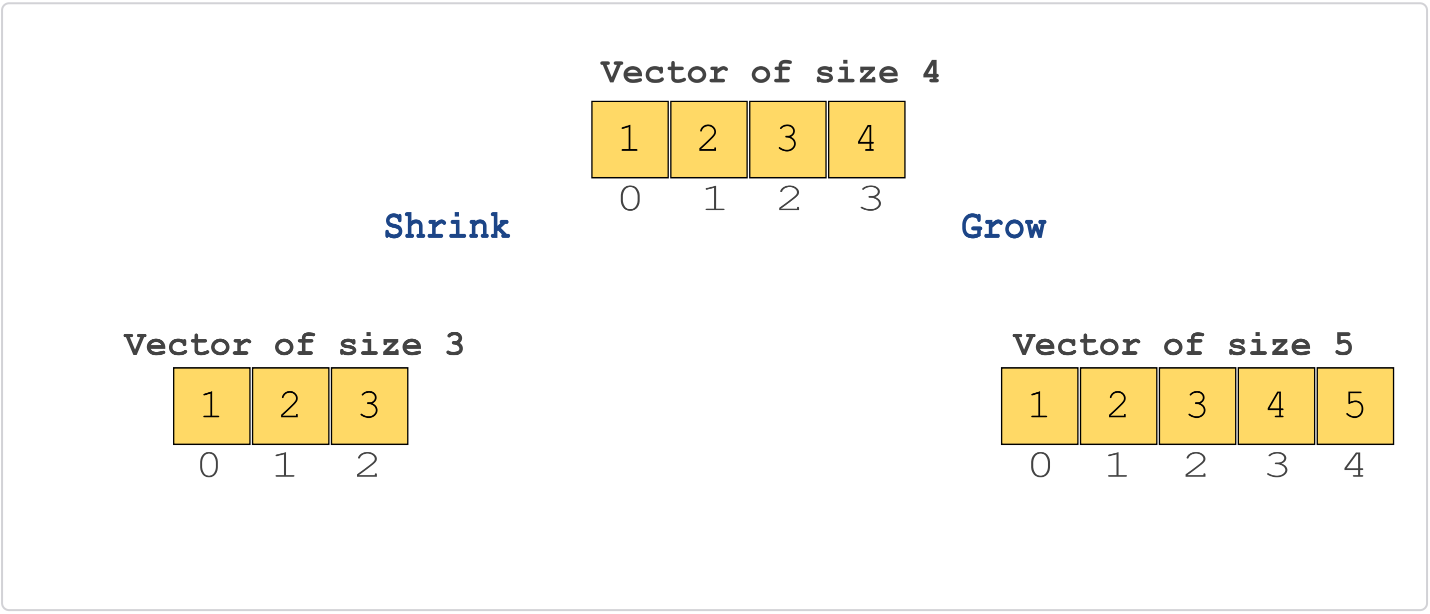
Introduction to Vectors

This lesson discusses vectors in Rust.

We'll cover the following

What are Vectors?

Vectors are resizable arrays meaning(they can grow or shrink in size).

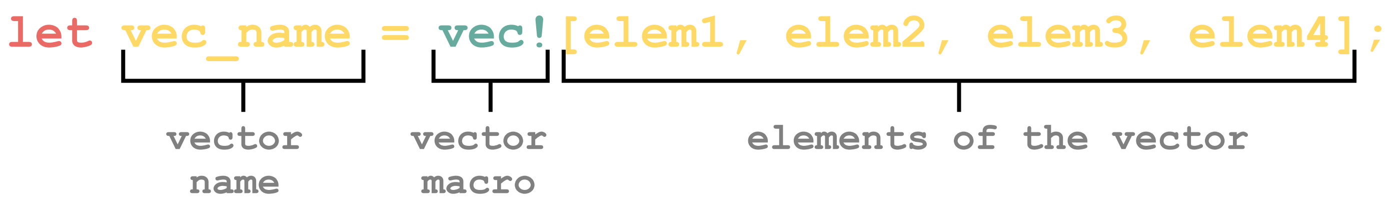


Create Vectors

There are two ways to create a vector:

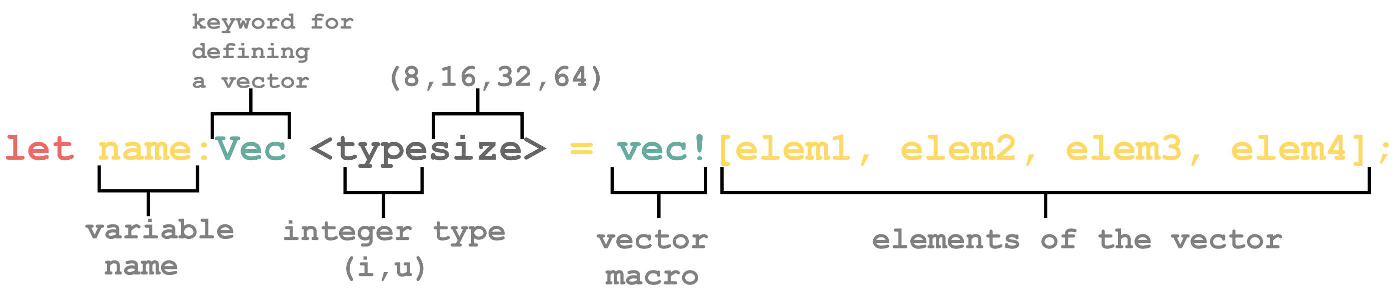
Syntax

To create a vector write the vector macro (`vec!`) followed by the elements of the vector enclosed in square brackets



Define a vector

It is optional to define the type and size of the vector enclosed within angular brackets. Use the vector macro(`vec!`) before defining the elements of the vector.


keyword for defining a vector
(8,16,32,64)
let name:Vec <typesize> = vec![elem1, elem2, elem3, elem4];
variable name
integer type (i,u)
vector macro
elements of the vector

Define a vector

```
fn main() {
   //define a vector of size 4
   let my_vec = vec![1, 2, 3, 4, 5];
   //print the vector
   println!("{:?}", my_vec);
}
```

> **Note:** Like arrays can be displayed on the screen using the `println!()` macro.

Access an Element of a Vector

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

```
fn main() {
   //define a vector of size 4
   let my_vec = vec![1, 2, 3, 4, 5];
   //access a particular value
   println!("{}", my_vec[0]);
}
```

> **Note:** If you try to access an index that does not exist, the compiler will give out of bound access error, ✕.

This is illustrated in the code below:

```
fn main() {
   //define a vector of size 4
   let my_vec = vec![1, 2, 3, 4, 5];
   //access a particular value
   eprintln!("{}", my_vec[9]);
}
```

To cater to out of bound exceptions, you can use a `None` keyword.

```
fn main() {
    let my_vec = vec![1, 2, 3,4,5];
    match my_vec.get(9) {
        Some(x) => println!("Value at given index:{}", x),
        None => println!("Sorry, you are accessing a value out of bound")
    }
}
```

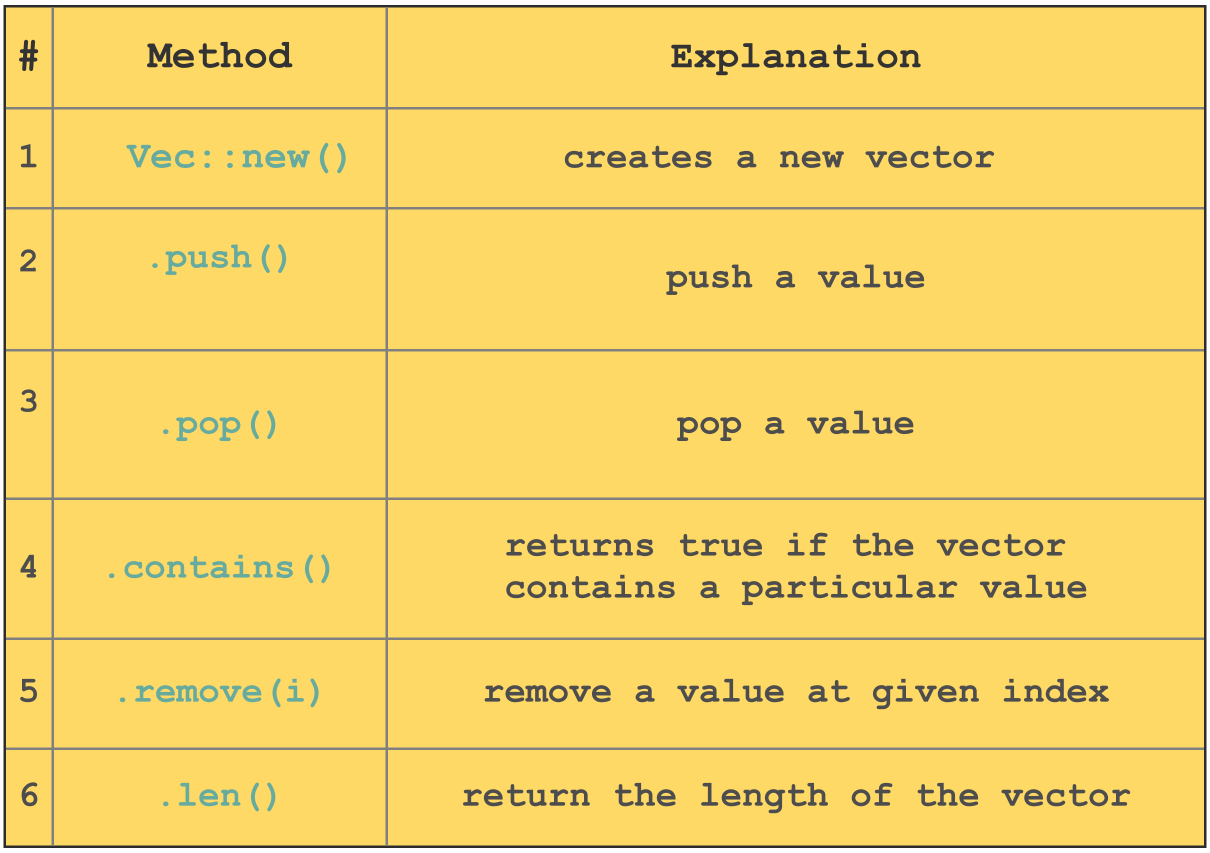
Print the Vector

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

```
fn main() {
    println!("Print using debug trait");
    let my_vec = vec![1, 2, 3,4,5];
    //using debug trait
    println!("Vector : {:?}", my_vec);
    println!("Print using for loop");
    // using loop
    let mut index = 0;
    for i in my_vec {
        println!("Element at index {}:{} ", index, i);
        index = index+1;
    }
}
```

Methods of Vectors

The methods of vectors are summarized in the chart below:



#	Method	Explanation
1	Vec::new()	creates a new vector
2	.push()	push a value
3	.pop()	pop a value
4	.contains()	returns true if the vector contains a particular value
5	.remove(i)	remove a value at given index
6	.len()	return the length of the vector

Vector Methods

The following code demonstrates each of the above methods:

```rust
fn main() {
   let mut my_vec = Vec::new();
   println!("Empty Vector : {:?}", my_vec);
   my_vec.push(1);
   my_vec.push(2);
   my_vec.push(3);
   println!("Pushed elements 1 , 2 , 3 : {:?}", my_vec);
   my_vec.pop();
   println!("Popped value: {}", 3);
   println!("Popped element at last index : {:?}", my_vec);
   my_vec.remove(1);
   println!("Removed value: {}", 2);
   println!("Removed element at index 1 : {:?}", my_vec);
   println!("Size of vector is :{}", my_vec.len());
   println!("Does my vector contains 1 : {}", my_vec.contains(&1));
}
```

> **Note:** When using the `.contains` function, consider borrowing the value. The reason will become clearer once we discuss different kinds of borrow operations in the later chapter.

Quiz

Test your understanding of basics of vectors in Rust.

Quick Quiz on Basics of Vectors!

1 Vectors are resizable arrays.

2 What is the output of the following code?

```
fn main() {
let my_vec = vec![1, 2, 3, 4, 5];
match my_vec.get(10) {
    Some(x) => println!("Value at given index:{}", x),
    None => println!("Sorry, you are accessing a value out of bound")
}
match my_vec.get(3) {
    Some(x) => println!("Value at given index:{}", x),
    None => println!("Sorry, you are accessing a value out of bound")
}

}
```

Retake Quiz

Now that you have learned the basics of vectors, let’s learn about the methods of vectors in the next lesson.