# Updating a String

This lesson will teach to update a string in Rust.

**We'll cover the following**

An existing string can be updated by appending a character or a string.

> **Why not make a new String rather than updating an existing one?**
>
> Updating an existing String is useful when you want to make changes to an existing String at run time rather than compile one like, in situations where changes are made to the String on a condition.

## Push a Single Character #

There are cases when it is required to update a string by pushing a single character. One example is to create a string which contains a single character repeated N times on a particular condition. Rust helps you do it by using the `push` method.

**Steps to push a character to a String:**

- Make a mutable string variable.
- To push a single Unicode character to a String object, pass a character within the `push()` built-in method.

  The following code shows how to do it!

```
fn main() {
  // define a String object
  let mut course = String::from("Rus");
  // push a character
  course.push('t');
  println!("This is a beginner course in {}.", course);
}
```

There are cases when it is required to grow a String by concatenating a new String to an existing String. Rust helps you do it by using the `push`, `+` operator and the `format!` macro method.

## Push a String

Rust helps you to grow a String object using a `push_str` method.

**Steps to push a String to a String:**

- Make a mutable String variable.
- To push a string to a growable string variable, pass a character within the `push_str()` built-in method.

  The following code shows how to do it!

```
fn main() {
  // define a string object
  let mut course = String::from("Rust");
  // push a string
  course.push_str(" Programming");
  println!("This is a beginner course in {}.", course);
}
```

## Concatenation Using `+` Operator

A String can be concatenated to another String using the `+` operator.

> **Note:** The right-hand-side operand is to borrowed while concatenating using `+` operator.

The following code shows how to do it!

```
#[allow(unused_variables, unused_mut)]
fn main(){
   // define a String object
   let course = "Rust".to_string();
   // define a String object
   let course_type = " beginner course".to_string();
   // concatenate using the + operator
   let result = course + &course_type;
   println!("{}", result);
}
```

## Format Macro

To add two or more String objects together, there is a macro called `format!`. It takes variables or values and merges them in a single String.

> **Note:** The `format!` macro allows concatenating in the desired order by passing a positive integer number within the placeholder. If the number is not mentioned it will concatenate in the order of the values written.

> To display the result of `format!` macro, the result is to be saved in a variable.

The following code shows how to do it!

```
fn main(){

   let course = "Rust".to_string();
   let _course_type = "beginner course".to_string();
   // default format macro
   let result = format!("{} {}", course, _course_type);
   // passing value in the placeholder in the format macro
   let result = format!("{1} {0}", course,_course_type);
   println!("{}", result);
}
```

## Quiz

Test your understanding of updating a String in Rust.

Quick Quiz on Updating a String!

1

Which of the following methods cannot be used for concatenating a string with another string?

2

What is the output of the following code?

```
fn main() {
  let mut s = String::from("Learn ");
  s.push('P');
  s.push_str ("rogramming");
  println!("{}!", s);
  let res= format!("{}{}",s," in Rust");
  println!("{}!", res);
}
```

3

What is the output of the following code?

```
fn main() {
  let mut s = "Learn ";
  s.push( 'P' );
  s.push_str("rogramming");
  println!("{}!", s);
  let res = format!("{}{}", s, " in Rust");
  println!("{}!", res);
}
```

Retake Quiz

Now that you have learned to update a string, let’s look at slicing a string in the next lesson.