3_Slices_fr

May 2, 2023

1 Le type slice

Les slices permettent de référencer une partie d'une collection (séquence continue d'éléments).

Les chaînes de littéraux sont des slices.

[ ]:
```
let s = "Hello, world!";
```

Le type de s est **`&str`** et c'est un slice. Les chaînes de littéraux sont donc immuables; **`&str`** est une référence immuable.

1.1 Slices des String

Un slice de String est une référence à une partie de d'un String :

[ ]:
```
{
    let s = String::from("hello world");
    let hello = &s[0..5];
    let world = &s[6..11];
}
```

En utilisant la syntaxe **[..]** on peut ne pas préciser le début de l'intervalle quand il commence à zéro. Les deux lignes suivantes sont équivalentes :

[ ]:
```
{
    let s = String::from("hello");
    let slice = &s[0..2];
    let slice = &s[..2];
}
```

La même règle s'applique pour la fin de l'intervalle. Les deux lignes suivantes sont équivalentes :

[ ]:
```
{
    let s = String::from("hello");

    let len = s.len();

    let slice = &s[3..len];
    let slice = &s[3..];
}
```

2 Exercices

Exercice 1: La fonction suivante prend comme paramètre une chaîne de caractères et doit retourner le premier mot rencontré. S'il n'y a pas d'espace dans le texte, la fonction retourne la chaîne de caractères dans sa totalité. Compléter le code pour obtenir ce résultat.

[ ]:
```
fn first_word(s: &String) -> &str {
    let bytes = s.as_bytes();

    for (i, &item) in bytes.iter().enumerate() { // itérer sur les bytes du␣
 ↪String
        if item == b' ' { //vérifier si le byte correspond à l'espace (on a␣
 ↪rencontré donc un mot)
            // TO DO (1) Ecrire le code qui retourne le premier mot
            return
        }
    }
    // TO DO (2) Ecrire le code qui retourne toute la chaîne de caractères

}

fn main() {
    let mut s = String::from("hello world");

    let word = first_word(&s);

    //TO DO (3) Décommenter la ligne suivante et essayer d'expliquer ce qui se␣
 ↪passe
    // s.clear();

    println!("the first word is: {}", word);
}
main();
```

Solution:

[ ]:
```
fn first_word(s: &String) -> &str {
    let bytes = s.as_bytes(); // convert our String to an array of bytes

    for (i, &item) in bytes.iter().enumerate() { // iterate on the String's␣
 ↪bytes
        if item == b' ' { //check if the byte is equal to blanc space, which␣
 ↪means we've got a word
            // TO DO (1) Write the statement that returns the first word
            return &s[0..i];
        }
    }
    // TO DO (2) Write the statement that returns the entire string
```

```
    &s[..]
}

fn main() {
    let mut s = String::from("hello world");

    let word = first_word(&s);

    //TO DO (3) uncomment the following line and try to explain what happens
    // s.clear(); // this empties the String, making it equal to ""

    println!("the first word is: {}", word);
}
main();
```

L'erreur à la ligne (3) : quand on a une référence immuable vers une donnée, on ne peut pas prendre une deuxième référence mutable vers cette donnée. `clear` veut tronquer le String, il essaye de prendre une référence mutable et cela provoque l'erreur.

Exercice 2: Re-écrire le code du jeu fizz buzz en utilisant la syntaxe "if in a let" et un seul println!

[ ]:
```
fn main() {
}
main();
```

Solution:

[ ]:
```
fn main() {
    for number in 1..20 {
        let value: String;
        let result = if number % 3 == 0 {
            if number % 5 == 0 {"Fizz Buzz"} else { "Fizz"}
        } else if number % 5 == 0 {
            "Buzz"
        } else {
            value = number.to_string();
            &*value
        };
        println!("Result {}", result);
    }
}
main();
```