# Fiche Débutant Rust (Jour 1)

---

## 1. Variables et mutabilité

### `let`

- Permet de **déclarer une variable**.
- Par défaut, les variables sont **immuables** (on ne peut pas changer leur valeur).

```
let x = 5; x = 6; // Erreur ! x est immuable
```

```
let x = 5;
x = 6; // ❌ Erreur ! x est immuable
```

### `let mut`

- `mut` rend une variable **mutable**, c'est-à-dire **modifiable**.

```
let mut y = 10;
y = 15; // ✅ Ok
```

```
let mut y = 10; y = 15;
```

### `const`

- Une **constante** ne change **jamais** et doit être **typée explicitement**.

```
const PI: f64 = 3.1415;
```

```
const PI: f64 = 3.1415;
```

---

# 2. Types de données de base

## `i32`, `u32` (entiers)

- `i32` : entier **signé** (peut être négatif ou positif)
  - Plage : de -2_147_483_648 à 2_147_483_647
- `u32` : entier **non signé** (uniquement positif)
  - Plage : de 0 à 4_294_967_295

```rust
let a: i32 = -42;
let b: u32 = 42;
```

```
let a: i32 = -42; let b: u32 = 42;
```

## `f64` (float 64 bits)

- Nombre **à virgule flottante**, précision double.
- Rust utilise `f64` par défaut pour les floats.

```rust
let temperature: f64 = 36.6;
```

```
let temperature: f64 = 36.6;
```

## `bool`

- Représente un **booléen** : `true` ou `false`.

```rust
let actif: bool = true;
```

```
let actif: bool = true;
```

## `char`

- Un **caractère unicode** (entre `'`), peut contenir des emojis, lettres, symboles.

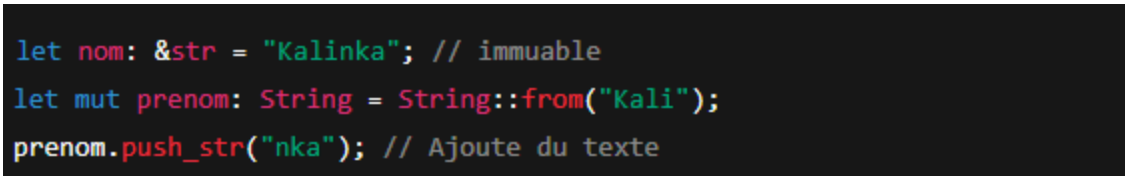
```rust
let lettre: char = 'A';
let emoji: char = '😄';
```

```
let lettre: char = 'A'; let emoji: char = '😄';
```

### `&str` vs `String`

- `&str` : une **chaîne de caractères statique** (non modifiable).
- `String` : une **chaîne dynamique**, modifiable et allouée sur le tas (heap).

```
let nom: &str = "Kalinka"; // immuable
let mut prenom: String = String::from("Kali");
prenom.push_str("nka"); // Ajoute du texte
```



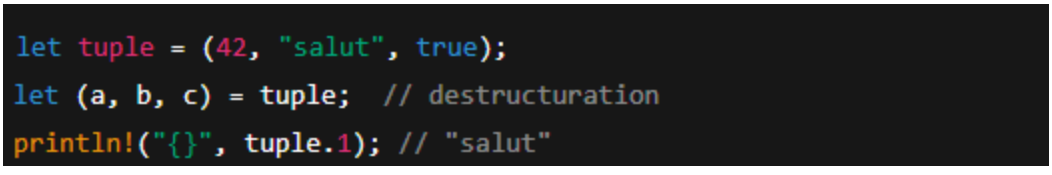

---

## 3. Structures de données simples

### Tuple `(valeur1, valeur2, ...)`

- Un **groupe de valeurs** de types **différents**.

```
let tuple = (42, "salut", true);
let (a, b, c) = tuple;  // destructuration
println!("{}", tuple.1); // "salut"
```




### Tableau `[valeur; taille]`

- Une séquence **fixe de valeurs du même type**.

```
let notes = [10, 12, 14]; let premier = notes[0]; // 10
```

```
let notes = [10, 12, 14];
let premier = notes[0]; // 10
```

### Vecteur `vec![]`

- Une **liste dynamique** (tu peux ajouter/enlever des éléments).

```
let mut liste = vec![1, 2, 3]; liste.push(4); // [1, 2, 3, 4]
```

```
let mut liste = vec![1, 2, 3];
liste.push(4); // [1, 2, 3, 4]
```

---

## 4. Fonctions

```
fn bonjour(nom: &str){
    println!("Salut, {}", nom);
 }

 fn addition(a: i32, b: i32) -> i32 {
      a + b
}
```

```
fn bonjour(nom: &str) {
    println!("Salut, {}", nom);
}

fn addition(a: i32, b: i32) -> i32 {
    a + b
}
```

- `fn` : mot-clé pour définir une fonction.
- `-> i32` : le type **retourné** par la fonction.
- Les **arguments** doivent être typés.

---

# 5. Contrôle de flux

## Condition

```
let age = 20; if age >= 18 {     println!("Majeur"); } else {     println!
("Mineur"); }
```

```rust
let age = 20;
if age >= 18 {
    println!("Majeur");
} else {
    println!("Mineur");
}
```

## Boucles

```
// Boucle for
for i in 0..3 {
    println!("{}", i); // Affiche 0, 1, 2
}

// Boucle while let mut n = 0;
while n < 3 {
    println!("{}", n);
    n += 1;
}
```

```rust
// Boucle for
for i in 0..3 {
    println!("{}", i); // Affiche 0, 1, 2
}

// Boucle while
let mut n = 0;
while n < 3 {
    println!("{}", n);
    n += 1;
}
```

## 6. Entrée / Sortie de base

```
use std::io;

fn main() {
    let mut entree = String::new();
    io::stdin().read_line(&mut entree).expect("Erreur de lecture");
    println!("Tu as écrit : {}", entree);
}
```


---

## 7. Compilation & Exécution

- Créer un projet :

`cargo new mon_app cd mon_app`

- Compiler et exécuter :

`cargo run`

---

## À retenir pour aujourd'hui

| Concept | À quoi ça sert ? |
|---|---|
| `let` | Déclare une variable immuable |
| `let mut` | Déclare une variable **modifiable** |
| `i32`, `u32` | Entiers (signés, non signés) |

| Concept | À quoi ça sert ? |
| --- | --- |
| `f64` | Nombre décimal |
| `String` | Chaîne de texte modifiable |
| `tuple` | Regrouper plusieurs types différents |
| `fn` | Créer une fonction |
| `if/else`, `for`, `while` | Contrôle du déroulement du programme |