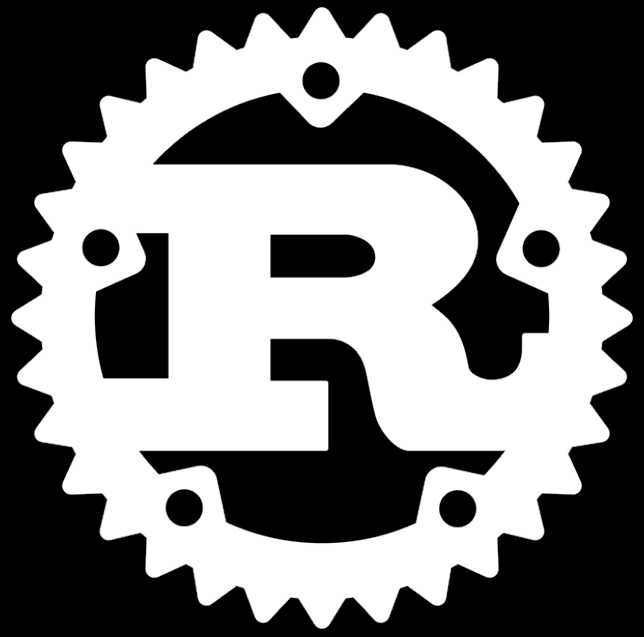


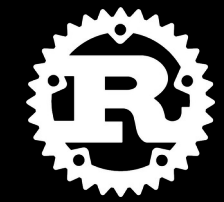
Formation Rust

Les fondamentaux

Pourquoi apprendre le Rust ?

- Intégré dans le noyau Linux
- Syntaxe haut niveau, possibilité de programmer en bas niveau
- Vous tenez à votre RAM
- Vous voulez devenir Nazi
- Look at this cute crab <3

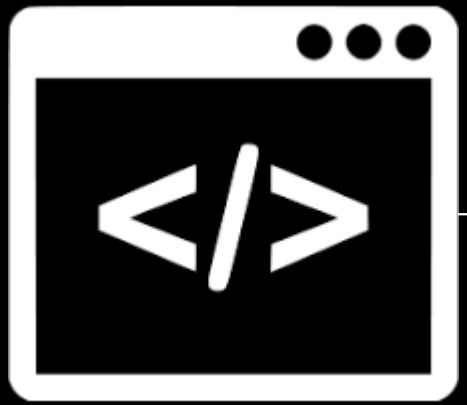
3 types de langages



Compilateur

Executable

Execution

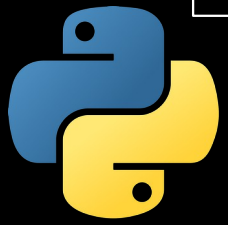


Compilateur

Bytecode

JVM

Hello world



Interpreteur

Petit point nomenclature

Item	Convention
Crates	unclear
Modules	`snake_case`
Types	`UpperCamelCase`
Traits	`UpperCamelCase`
Enum variants	`UpperCamelCase`
Functions	`snake_case`
Methods	`snake_case`
General constructors	`new` or `with_more_details`
Conversion constructors	`from_some_other_type`
Macros	`snake_case!`
Local variables	`snake_case`
Statics	`SCREAMING_SNAKE_CASE`
Constants	`SCREAMING_SNAKE_CASE`
Type parameters	concise `UpperCamelCase`, usually single uppercase letter: `T`
Lifetimes	short `lowercase`, usually a single letter: `'a`, `'de`, `'src`

Syntaxe de base

```rust
fn main() {
    // Ceci est un commentaire
    /*
    Ceci est un
    commentaire multiligne
    */

    let foo: i32 = 42;  // Variable immutable
    let mut bar = 21;   // Variable mutable

    bar = foo-bar;

    println!("foo = {}; bar = {}", foo, bar); // Ceci est une macro

    if dis_bonjour("fdp") { // if + appel de fonction
        println!("Bébou");
    }

    let mut ctr = 0;
    while ctr<10 {
        ctr += 1;
    }

    let notes = [10, 15, 8, 13, 69420];
    for note in notes {
        println!("Un élève à eu : {note}");
    }
```

```rust
fn dis_bonjour(nom: &str) -> bool {
    println!("Bonjour, {}", nom);
    nom == "Loulou"
}
```

Les types par défault

- Entiers : u8, u16, u32, u64, u128, i8, i16, i32, i64, i128
- Flottants : f32, f64
- Texte : char, &str, String
- Collections : Tuple, Array, Vec, Range, RangeInclusive

1er Exercice : Qui réussit son CF ?

```
60      let mut read = String::new();
61      println!("Entrez votre note");
62      io::stdin()
63          .read_line(&mut read)
64          .expect("Failed to read line");
65
66      let note: f32 = read.trim()
67          .parse()
68          .expect("Please type a number!");
```

Les types personalisés

```
13 struct Color(u8, u8, u8);
14
15 struct Point {
16     x : f32,
17     y : f32
18 }
19
20 enum ResultatCF {
21     Echec,
22     Reussite(f32)
23 }
```

```
53     let _c = Color(255, 127, 63);
54
55     let _origine = Point {
56         x : 0.,
57         y : 0.
58     };
```

```
enum Option<T> {
    None,
    Some(T)
}
```

```
78     match res {
79         ResultatCF::Echec => {println!("You failed");},
80         ResultatCF::Reussite(n) => {println!("GGEZ B), j'ai eu {n}");}
81     }
```

2ème Exercice : SAT

```rust
pub enum LogExp<'a> {
    T,
    F,
    X(usize),
    Non(&'a LogExp<'a>),
    Et(&'a LogExp<'a>, &'a LogExp<'a>),
    Ou(&'a LogExp<'a>, &'a LogExp<'a>),
    Oux(&'a LogExp<'a>, &'a LogExp<'a>)
}
```

It’s KAHOOT time !