# Rust : Tests

## Qualité de Développement — R5.A.08

C. Raïevsky

2026-2027

IUT A Valence
Université Grenoble Alpes

Département Informatique

*BUT Informatique 3ème année*

# Tests en Rust

- Les tests sont du code Rust standard
- Pas de bibliothèque externe
- Intégrés dans les outils : `cargo test`

**Deux principaux types de tests supportés**

- Tests unitaires
- Tests d'intégrations

# Un test est une fonction

## Fonction

- Annotée : `#[test]`
- Le test passe si la fonction se termine
- Le test échoue si elle panique

```
1 #[test]
2 fn it_works() {
3     let result = add(2, 2);
4     assert_eq!(result, 4);
5 }
```

## Trois assertions :

- `assert!(e);` → panique si 'e' est faux
- `assert_eq!(e1, e2);` → panique si 'e1' et 'e2' sont différents
- `assert_ne!(e1, e2);` → panique si 'e1' et 'e2' sont égaux

# Organisation des tests

## Tests unitaires

- Placés au plus près du code qu'ils testent
- Dans un sous module dédié
- Dans le même fichier ou dans un module à part

## Tests d'intégration

- Au niveau du ***package***
- Dans des modules dédiés

# Tests unitaires

## Exemple minimal (`cargo new --lib`)

```rust
pub fn add(left: usize, right: usize) -> usize {
    left + right
}

#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn it_works() {
        let result = add(2, 2);
        assert_eq!(result, 4);
    }
}
```

# Tests unitaires - Utilisation de l'opérateur ?

```
1  fn sqrt(number: f64) -> Result<f64, String> {
2      if number >= 0.0 { Ok(number.powf(0.5)) }
3      else {Err("square root of negative".to_owned())}
4  }
5
6  #[cfg(test)]
7  mod tests {
8      use super::*;
9      #[test]
10     fn test_sqrt() -> Result<(), String> {
11         assert_eq!(sqrt(4.0)?.powf(2.0), x);
12         Ok(())
13     }
14 }
```

# Tests unitaires - *should_panic*

```
1 #[test]
2 #[should_panic]
3 fn test_any_panic() {
4     division(1, 0);
5 }
```

# Tests d'intégration

## À la racine du *package*

- Dans un répertoire dédié
- Compilés et exécutés uniquement lors des tests
- Répartition dans des fichiers au choix

```
adder
├── Cargo.lock
├── Cargo.toml
├── src
│   └── lib.rs
└── tests
    ├── integration_tests.rs
    └── other_tests.rs
```

# Tests d'intégration

**Exemple - `integration_tests.rs`**

```
1 use adder;
2
3 #[test]
4 fn it_adds_two() {
5     assert_eq!(4, adder::add(2, 2));
6 }
```

# Les exemples sont des tests

**Les exemples de code présents dans la documentation**

- Sont exécutés lors de la phase de test
- Permettent d'assurer la cohérence entre la documentation et le code.

# Tests dans la documentation

## Exemple comme test

```
1 /// A wonderful function that computes the sum of its arguments.
2 /// # Example usage:
3 /// ```
4 /// let sum = package::add(11, 31);
5 /// assert_eq!(sum, 42);
6 /// ```
7 pub fn add(left: usize, right: usize) -> usize {
8     left + right
9 }
```

Le code des lignes 4 et 5 va être passé comme un test lors d'un "`cargo test`"