

Crate | rand

# ■ rand

- ◆ Widely used random number generation crate
- ◆ Supports both simple & advanced configuration
- ◆ Features:
  - Picking values from a collection
  - Seeding
  - Cryptographically secure generators
  - Uniform & weighted distributions

# ■ Basic Usage – Prelude

- ◆ Imports common traits and types for working with *rand*

```
use rand::prelude::*;
```

## ■ Basic Usage

```
let number: u8 = random();  
let yes_no: bool = random();
```

# Basic Usage

```
let mut rng = thread_rng();  
let number = rng.gen_range(0..10);  
  
let letters = ['a', 'b', 'c'];  
let letter = letters.iter().choose(&mut rng);  
  
let mut letters = letters;  
letters.shuffle(&mut rng);
```

# ■ Seeding

- ◆ Seeding is not cryptographically secure!

```
use rand::prelude::*;  
use rand_pcg::Pcg64;  
use rand_seeder::Seeder;
```

```
let rng = Pcg64::seed_from_u64(10);  
let rng: Pcg64 = Seeder::from("seed value").make_rng();
```

# ■ Distributions

```
use rand::distributions::{Distribution, Uniform};  
use rand::prelude::*;
```

```
let range = Uniform::from(5..500);  
let mut rng = thread_rng();  
range.sample(&mut rng);
```

```
[dependencies]  
rand = "*"   
rand_distr = "*"
```

# ■ Recap

- ◆ *rand* crate provides random number generation
- ◆ *thread\_rng* uses thread-local RNG and can be called multiple times, or cached
  - Cryptographically secure
- ◆ Import the *rand prelude* for convenience
- ◆ The *choose* function will choose a random item from an iterator
- ◆ The *sample* function will sample a random number from a distribution