# Rust 编程语言入门


杨旭，微软 MVP
Rust、Go 开发者

# 13.5 迭代器

# 什么是迭代器

- 迭代器模式：对一系列项执行某些任务
- 迭代器负责：
  - 遍历每个项
  - 确定序列（遍历）何时完成
- Rust 的迭代器：
  - 懒惰的：除非调用消费迭代器的方法，否则迭代器本身没有任何效果。
  - （例子）

# 13.5 迭代器（1）

- Iterator trait 和 next 方法

# Iterator trait

- 所有迭代器都实现了 Iterator trait
- Iterator trait 定义于标准库，定义大致如下：
- ```
  pub trait Iterator {
      type Item;

      fn next(&mut self) -> Option<Self::Item>;

      // methods with default implementations elided
  }
  ```
- type Item 和 Self::Item 定义了与此该 trait 关联的类型。
  - 实现 Iterator trait 需要你定义一个 Item 类型，它用于 next 方法的返回类型（迭代器的返回类型）。
  ```

# Iterator trait

- Iterator trait 仅要求实现一个方法：next
- next：
  - 每次返回迭代器中的一项
  - 返回结果包裹在 Some 里
  - 迭代结束，返回 None
- 可直接在迭代器上调用 next 方法
- （例子）

# 几个迭代方法

- iter 方法：在不可变引用上创建迭代器
- into_iter 方法：创建的迭代器会获得所有权
- iter_mut 方法：迭代可变的引用

再见