

Fundamentals | Loop Labels

■ Loop Labels

- ◆ Loops can be annotated with a label for control flow
- ◆ Allows changing flow control to an outer loop
 - *break*
 - *continue*
- ◆ Useful when working with nested loops

■ Syntax

'ident: loop {}

'ident: for x in y {}

'ident: while true {}

■ Example – *break*

```
let matrix = [  
    [2, 4, 6],  
    [8, 9, 10],  
    [12, 14, 16],  
];  
  
'rows: for row in matrix.iter() {  
    'cols: for col in row {  
        if col % 2 == 1 {  
            println!("odd: {}", col);  
            break 'rows;  
        }  
        println!("{}", col);  
    }  
}
```

■ Example – *continue*

```
type UserInput<'a> = Result<&'a str, String>;  
'menu: loop {  
    println!("menu");  
    'input: loop {  
        let user_input: UserInput = Ok("next");  
        match user_input {  
            Ok(input) => break 'menu,  
            Err(_) => {  
                println!("try again");  
                continue 'input;  
            }  
        }  
    }  
}
```

■ Recap

- ◆ Loop labels can be applied to any type of loop
 - *loop*, *while*, *for*
- ◆ Control can be directed to outer loops using loop labels
 - **Break** will exit the specified loop
 - **Continue** will execute the specified loop

```
'ident: loop {}
```