Control Flow

这章其实比较简单了，就给一些例子吧：

- **If statement:**

```
fn main() {
    let number :i32 = 3;

    if number < 5{
        println!("The condition is true");
    }else{
        println!("The condition is false");
    }
}
```

  - 但Rust并不会把非boolean类型的值，转成bool，所以下面的程序会报错：

```
fn main() {
    let number :i32 = 3;
    if number{
        println!("The condition is true");
    }
}
```

output:

```
$ cargo run
   Compiling Control_Flow v0.1.0
(/home/ubuntu/Desktop/Rust_Learning/3_Common_Programming_Concept/Control_Flo
w)
error[E0308]: mismatched types
 --> src/main.rs:4:8
  |
4 |     if number{
  |        ^^^^^^ expected `bool`, found `i32`

For more information about this error, try `rustc --explain E0308`.
error: could not compile `Control_Flow` due to previous error
```

  - If- else if - else:

```
fn main() {
    let number :i32 = 3;

    if number < 5{
        println!("The condition is true");
    }
    else if number > 7{
        println!("The condition is greater than 7");
    }
    else{
        println!("The condition is false");
    }
}
```

- 将if-else作为一个expression (回顾上一章Function,什么时expression):

```
let word = if 1<2 {5} else {6};
println!("The word now is {}", word);
```

注意: `let word = if 1<2 {5} else {"Hello"};`是不行的，以为rust在compile的时候就知道word是什么数据类型。

Loops

- Rust有一个很好的点是，它可以给loop命名，用`` `loop_name `` 的形式。这样就可以像汇编一样跳到某一个指定的loop位置，这在C++，Python中是不具备的。

```
fn loop_function(){
    let mut count = 0;
    'counting_up: loop{
        println!("count = {}", count);
        let mut remaining = 10;

        loop{
            println!("remaining = {}", remaining);
            if remaining == 9{
                break;
            }
            if count == 2{
                break 'counting_up;
            }
            remaining -= 1;
        }

        count += 1;
    }
}
```

output:

```
count = 0
remaining = 10
remaining = 9
count = 1
remaining = 10
remaining = 9
count = 2
remaining = 10
```

- 同样，loop也是可以作为expression而返回值的。事例如下：

```
let mut counter = 0;
let result = loop{
    counter += 1;
    if counter == 10{
        break counter * 2
    }
};

println!("the result is {}", result);
```

Conditional Loops

- 用while实现有条件的循环。

```
let mut number = 3;

while number != 0{
    println!("{}!", number);
    number -= 1;
}

println!("LIFTOFF!!!");
```

For

- Rust中的`for`像python里的for一样强大。它可以遍历数组，可以在一个range中取值。
- Rust遍历数组：

```
let a = [10, 20, 30, 40, 50];

for element in a{
    println!("the value is {}", element);
}
```

- Rust在Range中取值：

```
for number in (1..4).rev(){
    println!("{}!", number);
}
println!("LISTOFF!!!");
```

output:

```
1!
2!
3!
LISTOFF!!!
```

- 注意这个range和python的range一样是左闭右开的。
- `.rev()`是让这个range反转。