通用的编程概念

- 1. 变量与可变性
- 2. 数据类型
 - 1) 标量类型
 - 2) 复合类型
- 3. 函数
- 4. 注释
- 5. 控制流

变量与可变性

△声明变量用 let 关键字 默认情况下,变量是不可变的(immutable)一旦被绑定到某个值上后 就不可以被重新赋值 如果重新给他赋值 编译就会报错

报错: cannot assign twice to immutable variable

△声明变量时 在变量前面加上 mut 就可以使变量可变

变量与常量

- 常量(constant),常量在绑定值以后也是不可变的,但是它与不可变的变量有很多区别:
 - 不可以使用 mut, 常量永远都是不可变的
 - 声明常量使用 const 关键字,它的类型必须被标注
 - 常量可以在任何作用域内进行声明,包括全局作用域
 - 常量只可以绑定到常量表达式,无法绑定到函数的调用结果或只能在运行时才能计算出的值
- 在程序运行期间,常量在其声明的作用域内一直有效
- 命名规范: Rust 里常量使用全大写字母,每个单词之间用下划线分开,例如: - MAX POINTS
- 例子: const MAX_POINTS: u32 = 100_000;

Shadow

```
main.rs x

src > main.rs > (m) x

1     fn main() {
2         let k = 5;
3         let x = x + 1;
4         let x = x * 2;
5
6         println!("The value of x is {}", x);
7     }
8
```

用后定义的同名变量隐藏前面的变量

Shadowing (隐藏)

- 可以使用相同的名字声明新的变量,新的变量就会 shadow(隐藏)之前声明的同名变量
 - 在后续的代码中这个变量名代表的就是新的变量
 - (例子)
- shadow 和把变量标记为 mut 是不一样的:
 - 如果不使用 let 关键字,那么重新给非 mut 的变量赋值会导致编译时错误
 - 而使用 let 声明的同名新变量, 也是不可变的
 - 使用 let 声明的同名新变量,它的类型可以与之前不同

简而言之就是如果使用 mut 定义变量的话 后续只能重新为变量赋值 不能改变它的类型 而用 shadow 相当于重新定义了一个新的变量 好处也是非常直接的