函数

Main 函数作为入口

声明函数用 fn 关键字

规范：snack case
蛇形命名法相关**命名法**：
https://blog.csdn.net/weixin_45129599/article/details/128802967?ops_request_misc=&request_id=&biz_id=102&utm_term=snake%20case&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-8-128802967.142^v101^pc_search_result_base8&spm=1018.2226.3001.4187

一些编程语言和框架强烈推荐或者约定使用 snake case 风格。例如，Rust、Python、Ruby 等都鼓励使用 snake case。Snake case 是一种命名规范风格，通常用于变量名、函数名和文件名等标识符。Snake case 的特点是将单词用下划线连接，并全部小写。这种风格使得标识符更易读，也更符合一些编程语言和工具的命名约定。

CamelCase

驼峰命名法 每个单词以大写字母开头

小驼峰命名 就是第一位单词首字母小写；后面每个单词的字母首字母，都是大写。

大驼峰命名 就是每一位单词的第一个字母都是大写

区别 就是 第一位是大写还是小写进行区分。

```
利用 ffxi game  或者 FFXI GAME 进行命名
用这2个 进行组合成驼峰命名
ffxiGame : 小驼峰命名
FfxiGame : 大驼峰命名
```

snake_case 蛇形命名法

```
依旧以 ffxi game 或者 FFXI GAME 进行命名
ffxi_game 这种就是蛇形命名法。
```

spinal-case 脊柱命名法

```
依旧以 ffxi game 或者 FFXI GAME 进行命名
ffxi-game 这就是脊柱命名法
```

具体代码

```rust
fn main() {
    println!("hello world");
    another_function();
}

fn another_function() {
    println!("Another function");
}

```

函数的参数

Parameter 定义的参数 也就是形式参数
Argument 传进来的参数 实参
函数的签名中 必须声明每个参数的类型

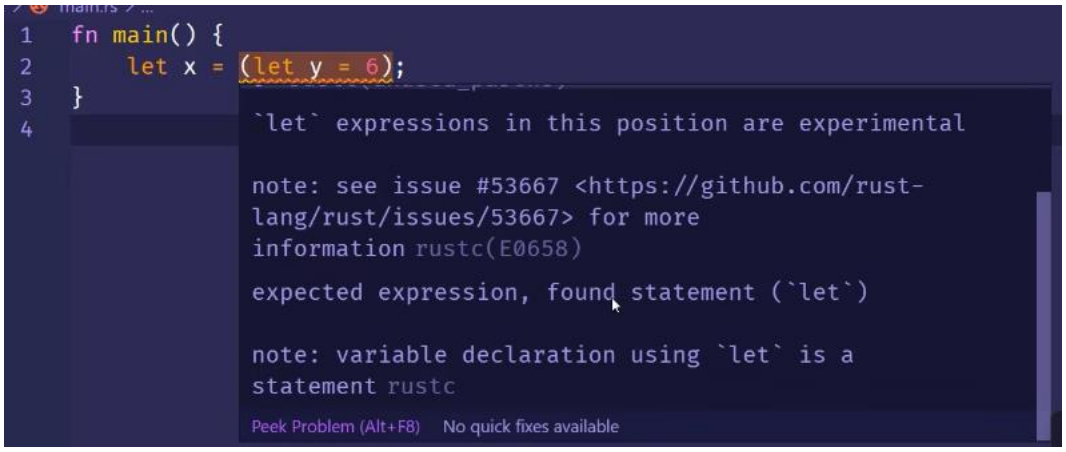
函数体中的语句与表达式

- 函数体由一系列语句组成，可选的由一个表达式结束
- Rust 是一个基于表达式的语言
- 语句是执行一些动作的指令
- 表达式会计算产生一个值
- （例子）
- 函数的定义也是语句
- 语句不返回值，所以不可以使用 let 将一个语句赋给一个变量（例子）

语句和表达式

1）语句是执行一些操作但不返回值的代码单元。例如，声明变量、赋值、函数调用、宏调用等都是语句。函数的定义也是语句
语句不返回值，所以**不可以使用 let 将一个语句赋给一个变量**

```rust
// 声明变量的语句
let x = 5;

// 表达式语句（函数调用）
println!("Hello, World!");

// if 语句
if x > 0 {
    println!("x is positive");
} else {
    println!("x is non-positive");
}

// 函数定义语句
fn my_function() {
    // 函数体
}
```



2） 表达式是计算并产生一个值的代码单元。表达式可以是一个常量、变量、运算、函数调用等。每个表达式都有一个类型，并产生一个值。任何单个的常量或者常量的计算结果都是表达式

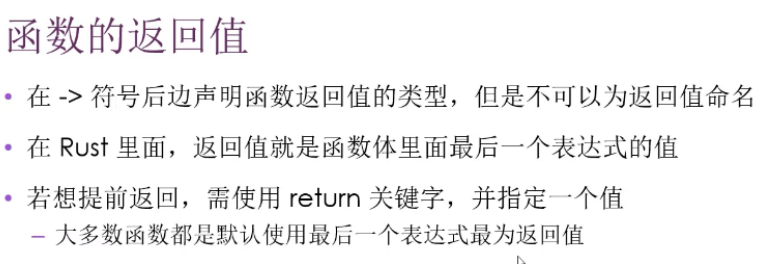
```rust
src > main.rs > ...
fn plus_five(x: i32) -> i32 {
    x + 5
}

fn main() {
    let x = plus_five(6);

    println!("The value of x is: {}", x);
}
```

注意这段代码中 x+5 是表达式类型 后边不加逗号 加逗号就是语句
如图中块表达式:

```rust
let y = {
    let x = 1;
    x + 3
};
```

括号中接的是**表达式 x+3 放在块最后 而且不加分号 表示这个块最后的返回值**
如果加分号 就变成语句了（而语句就没有返回值了 硬说有也可以 返回一个空的 tuple->//() 如下图 y ）

函数的返回值



函数中最后一个表达式就是他的返回值

注释

```rust
// This is a function
fn five(x: i32) -> i32 {
    x + 5
}

/* dsfsfsadf
sadfdsfds */
// This is main function
// The entry point
fn main() {
    // call five()
    let x = five(6); // 5 + 6

    println!("The value of x is: {}", x);
}
```

很像 c 语言 注释写法也差不太多
还有一种比较特别的文档注释