Data Types

- Rust是statically typed language, 即compiler需要在编译的阶段就知道所有变量的类型。Rust可以通过给变量的赋值推断出来它输入什么类型，比如：

```
let mut guess = String::new();
```

  - 即，implicit和explicit annotatio:

```
fn main(){
    let t = true;    // inplicit

    let f: bool = false; // explicit
}
```

- 但其他没申明数据类型的情况就不能这样，比如：

```
let guess = "42".parse().expect("Not a number");
```

  - compiler是不知道要parse到什么类型的。

- 在表示数字的时候，可以人为地加上`_`分割一个数地数字从而更加易读。比如 123456 和 123_456 是一个意思
- Rust的char是4 bytes的！

Compound Type

- group multiple values into one type.主要有`tuples`和`arrays`
- **tuple:**

```
let tup: (u128, u128, u128) = (100, 100, 200);
```

  - 我们可以这样从tuple中取数(destructure)：

```
let tup: (u128, u128, u128) = (100, 100, 200);

let (x, y, z) = tup;

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

  - 另一种取数的方法则是用`.`来取数：

```
let first = tup.0;
let second = tup.1;
let third = tup.2;

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

- **Array:**
  - 可以inplicit声明，也可以explicit声明：

```
// array
let a = [1, 2, 3, 4, 5];

let a: [i32; 5] = [1, 2, 3, 4, 5];

let a = [3; 5];   // == let a = [3, 3, 3, 3, 3]
```

  - rust中array的取值也是用中括号：

```
let third = a[2];
println!("the third value of a is {}", third);
```