

به راه Rust با یونس : جلسه سوم (فصل سوم کتاب اصلی Rust)

مدرس: یونس قربانی موضوع: مفاهیم رایج برنامه‌نویسی (Common Programming Concepts: Variables, Mutability, Functions, Control Flow)
مرجع: The Rust Programming Language (Latest Edition)

چک‌لیست تدریس در استریم (Presenter Checklist)

- 1. متغیرها و ثابت‌ها (let در برابر const) و تفاوت‌های بنیادین آن‌ها ✓
- 2. مفهوم Shadowing (سایه‌زنی متغیرها) و تفاوتش با mut ✓
- 3. انواع داده‌ها در Rust (Scalar Types: Integers, Floats, Booleans, Characters) ✓
- 4. انواع ترکیبی (Compound Types: Tuples و Arrays) و نحوه دسترسی به عناصر ✓
- 5. توابع در Rust (Function Signatures، پارامترها و نوع خروجی) ✓
- 6. تفاوت Statements (عبارت‌ها) و Expressions (عبارت‌های مقداردهی) و اهمیت نداشتن ; در انتهای تابع! ✓
- 7. کامنت‌ها و نحوه مستندسازی با /// ✓
- 8. کنترل جریان (Control Flow): شرط‌های if / else و استفاده از آن‌ها به عنوان Expression ✓
- 9. حلقه‌ها در Rust: حلقه بی‌نهایت loop، حلقه شرطی while و حلقه پیمایشی for ✓

تکه‌های کد کلیدی جلسه (Code Snippets)

1. متغیرها، سایه‌زنی و ثابت‌ها

```
1 fn main() {  
2     // 1. Mutability vs Immutability  
3     let mut x = 5;  
4     println!("The value of x is: {x}");  
5     x = 6;  
6     println!("The value of x is: {x}");  
7  
8     // 2. The data type must be explicitly specified and the value must be a  
9     compile-time constant expression  
10    const THREE_HOURS_IN_SECONDS: u32 = 60 * 60 * 3;  
11  
12    // 3. Shadowing (changing type or value while keeping the let property)  
13    let spaces = "  ";  
14    let spaces = spaces.len(); // Here the type of spaces changed from String/&str  
    to usize!  
}
```

2. ساختار توابع و تفاوت Expression و Statement

```
1 fn main() {  
2     let y = {  
3         let x = 3;  
4     }  
}
```

```

4      x + 1 // Without a semicolon (;), so it's an expression and returns the
        value 4!
5      };
6
7      println!("The value of y is: {y}");
8      let result = plus_five(5);
9      println!("The result is: {result}");
10     }
11
12     // Function with a specified return type (note the absence of return and semicolon
        on the last line)
13     fn plus_five(x: i32) → i32 {
14         x + 5
15     }

```

۳. کنترل جریان و حلقه‌ها (if, loop, while, for)

```

1     fn main() {
2         // If as an Expression
3         let condition = true;
4         let number = if condition { 5 } else { 6 };
5
6         // Loop with return value
7         let mut counter = 0;
8         let result = loop {
9             counter += 1;
10            if counter == 10 {
11                break counter * 2; // Returning a value from inside the loop
12            }
13        };
14
15        // For loop over a collection (very safe and widely used)
16        let a = [10, 20, 30, 40, 50];
17        for element in a {
18            println!("the value is: {element}");
19        }
20
21        // Range-based for loop (like Python)
22        for number in (1..4).rev() {
23            println!("{number}!");
24        }
25        println!("LIFTOFF!!!");
26    }

```

🧠 تقلب‌نامه و نکات کلیدی برای دانشجویان (Key Concepts)

۱. متغیرها (let) در برابر ثابت‌ها (const)

- **ثابت‌ها (const):** در کل طول اجرای برنامه و در هر Scope معتبر هستند. نوع داده‌شان حتماً باید مشخص شود و مقدارشان باید در زمان کامپایل مشخص باشد (نمی‌توان نتیجه یک تابع دینامیک را به const داد). نامگذاری استاندارد: با حروف بزرگ (MAX_POINTS).

- **سایه‌زنی (Shadowing):** با استفاده مجدد از کلمه `let` می‌توانید متغیری با همان نام قبلی بسازید که حتی می‌تواند نوع داده متفاوتی داشته باشد. این با `mut` فرق دارد چون پس از انجام این کار، متغیر دوباره `immutable` می‌شود.

۲. انواع داده‌های اسکالر (Scalar Types)

زبان Rust یک زبان **Statically Typed** است (نوع تمام متغیرها باید در زمان کامپایل مشخص باشد)، اما کامپایلر اغلب می‌تواند نوع را حدس بزند (Type Inference).

- **اعداد صحیح (Integers):** `i8, i16, i32, i64, i128, isize` (منفی و مثبت) و `u8, u16, u32, u64, u128, usize` (فقط مثبت). نوع پیش‌فرض `i32` است.
- **اعداد اعشاری (Floats):** `f32` و `f64` (پیش‌فرض `f64` به دلیل دقت بالاتر و سرعت پردازشی مشابه در پردازنده‌های مدرن).
- **بولی (Boolean):** `bool` با مقادیر `true` و `false`.
- **کاراکتر (char):** برخلاف پایتون، کاراکتر در Rust یک کاراکتر **Unicode Scalar Value** چهار بایتی است (یعنی نه تنها حروف انگلیسی، بلکه ایموجی‌ها و حروف فارسی را هم به راحتی پوشش می‌دهد!). با تک‌کویشن `'z'` مشخص می‌شود.

۳. انواع ترکیبی (Compound Types)

- **تاپل (Tuple):** مجموعه‌ای با طول ثابت از انواع داده‌های مختلف.

```
1 let tup: (i32, f64, u8) = (500, 6.4, 1);
2 let (x, y, z) = tup; // Destructuring
3 let five_hundred = tup.0; // Dot notation
```

سکدنیا اب یسرتسد

- **آرایه (Array):** مجموعه‌ای با طول ثابت که تمام عناصر آن باید یک نوع داده داشته باشند. روی استک (Stack) ذخیره می‌شوند نه هیپ (Heap).

```
1 let a: [i32; 5] = [1, 2, 3, 4, 5];
2 let a = [3; 5]; // اب تسایواسم
```

۴. تفاوت Expressions و Statements (بسیار مهم)

- **Statements (بیانه‌ها):** دستوراتی که کاری انجام می‌دهند و **مقداری بر نمی‌گردانند** (مثل تعریف متغیر با `let x = 6`). اگر انتهای خط نقطه ویرگول `;` بگذارید، آن را به Statement تبدیل کرده‌اید.
- **Expressions (عبارت‌ها):** بخش‌هایی از کد که **یک مقدار را ارزیابی و برمی‌گردانند** (مثل `5 + 6` یا صدا زدن یک تابع).
- **نکته طلایی:** در Rust، بدنه توابع یا بلوک‌های کد `{ .. }` با آخرین Expression مخابره می‌شوند. اگر انتهای تابع `return x + 5` را بدون نقطه ویرگول بگذارید، یعنی دارید آن را به عنوان خروجی تابع برمی‌گردانید (معادل `return x + 5;`).

۵. کنترل جریان (Control Flow)

- **شرطها:** شرطها در `if` حتماً و حتماً باید یک `bool` باشند (مثل پایتون نیست که عدد صفر را `false` در نظر بگیرد). چون `if` یک Expression است، می‌توانید به راحتی از آن درون `let` استفاده کنید.
- **حلقه‌ها (`loop` , `while` , `for`):**
- حلقه `loop` تا ابد می‌چرخد مگر با `break` متوقفش کنید (امکان برگرداندن مقدار با `break val` وجود دارد).
- حلقه `for` امن‌ترین و محبوب‌ترین حلقه در Rust است چون خطاهای Index Out of Bounds رو به لطف سیستم امنیتی حذف می‌کند.