1. Using let keyword
   let x = 5;
2. Using mut keyword
   let mut x = 5;
3. The compiler will throw an error
4. Constants are immutable and declared with const. They must have a fixed type.
5. Shadowing is declaring a new variable with the same name as a previous one.
6. Yes
7. Rust doesn't allow uninitialized variables
8. let creates an immutable variable, let mut creates a mutable variable.
9. Yes
10. Yes (constants & static variables)
11. ref is using while pattern matching
    & is used to create a reference.
12. Each value in Rust has a single owner and we can transfer ownership
13. Stack: Stores data in fixed size.
    Heap: Stores data dynamically.
14. closures are functions that can capture variables from their surrounding environment
15. Aliases are references to the same memory location.
    They will not own those values but can access it without modifying it.
16. loop {
    println!("loop");
    }
17. Use break keyword
18. Use the continue keyword in the end of the loop, before the next iteration code.
19. Use loop inside another loop and put brackets to divide it
20. Use label name for the loop and use the name with break keyword
21. let v = vec![1, 2, 3];
    for i in &v {
    println!("{}", i);
    }
22. Didn't understand clearly
23. iter(): Creates an iterator over references
    iter_mut(): Creates an iterator over mutable references
24. Use iterators or avoid unnecessary allocations
25. For loops are more optimized
26. True