Variables Constants and Shadowing

Variables (Statically typed language)

1. All variables must be initialized to some value

2. Implicit Type Assignment: (rust will figure out the type)

    - let <variable_name> = <string, int, etc.>

3. Explicit Type Assignment: (you will add on the type)

    - let <variable_name>: =
    - Example of type:
        - i32
        - u32

4. How to do a simple print:

    - Using formatted strings:
        - println!("<variable_name> is: {}", <variable_name>);

5. Variable info:

6. All variables are immutable by default

    - Making it mutable:
        - let **mut** <variable_name> = <string, int, etc>
        - Ex.
            - let mut y = 5;
            - y = 10;
            - println!("y = {}", y);
        - WARNING: Cargo gives a warning about this because we are resetting the variable without using the intialized value. It is ok, but Cargo recognizes this as "dumb coding"
            - What it wants us to do is the following:
                - let `mut` y = 10;
                - println!("y = {}", y);
            - This will lead to another warning, cause the value y is not changing at all now.
            - Which means you can just make it immutable from the start.

7. All variables can be redefined:

    - Unlike in other languages, you can redefine a variable using the `let` keyword
        - ex.
            - let x = 5;
            - let x = 2.1 + x; //x is now 7.1, the expression is first evaluated * //and then it is assigned.

8. Variable scopes:

- You can create a scope for a variable using the `{}` at any time
- Within the a function you can make a scope
  - Ex.

  -
    ```
        fn foo(){
            let x = 5;
            {
            let x = 2;//x is 2
            }
            //x is stil 5
            }
    ```

9. Constants:

   1. //constants need to be in CAPITAL_SNAKE_CASE and the type must be defined `const SECOND_IN_MINUTE: u32 = 60;` println!("SECOND_IN_MINUTE = {}", SECOND_IN_MINUTE); //const SECOND_IN_MINUTE: u32 = 35; //error you cannot redefine a const