Programming 101: Introduction to Rust with D&D Character Building

Objective

This document introduces you to basic programming concepts in Rust while creating a Dungeons & Dragons (D&D) character. You will learn to work with variables, data types, and basic operations in Rust by building elements of a D&D character sheet.

What is a Variable?

A **variable** is like a container that holds a value. Think of it as a labeled box where you can store information about your character, such as their name, class, or strength.

Declaring Variables

In Rust, you use the `let` keyword to declare variables. When declaring a variable, you can specify its type, such as a whole number for an attribute like strength or a text string for the character's name.

```
1       let strength: u8 = 15; // Declares an integer variable named strength
2       let name: &str = "Arin"; // Declares a string variable named name
```

Basic Data Types in Rust

In Rust, you'll use data types to specify what kind of information a variable holds. Here are a few common types we'll use for our D&D character:

- **Integer (`u8`)**: Represents a whole number (from 0 to 255), ideal for scores like strength or intelligence.
- **Boolean (`bool`)**: Represents a true or false value, which could be used for states like `is_alive`.
- **String (`&str`)**: Represents text, useful for names, classes, and descriptions.

Task: Create Basic Character Attributes

In this task, you will declare variables for some basic character attributes. Define the following variables:

- `name`: A string holding your character's name.
- `class`: A string representing your character's class (e.g., "Wizard", "Fighter").
- `strength`, `dexterity`, `intelligence`: Integers holding attribute scores between 1 and 20.

Example Code

Here's how you might declare these variables in Rust:

```
fn main() {
    let name: &str = "Arin";
    let class: &str = "Fighter";
    let strength: u8 = 16;
    let dexterity: u8 = 12;
    let intelligence: u8 = 14;

    println!("Character Name: {}", name);
    println!("Class: {}", class);
    println!("Strength: {}", strength);
    println!("Dexterity: {}", dexterity);
    println!("Intelligence: {}", intelligence);
}
```

Performing Basic Operations

In D&D, each attribute score has a modifier. A modifier is calculated by the formula:

$$\text{modifier} = \frac{\text{score} - 10}{2}$$

We'll use Rust to calculate the modifier for each attribute.

Task: Calculate Modifiers

Using the formula above, write a function that calculates the modifier for a given attribute score.

```
fn calculate_modifier(score: u8) -> i8 {
    ((score as i8 - 10) / 2)
}

fn main() {
    let strength: u8 = 16;
    let dexterity: u8 = 12;
    let intelligence: u8 = 14;

    println!("Strength Modifier: {}", calculate_modifier(strength));
    println!("Dexterity Modifier: {}", calculate_modifier(dexterity));
    println!("Intelligence Modifier: {}", calculate_modifier(intelligence));
}
```

Testing Your Code

To test your code, make sure you're in the correct project directory and run the following command:

```
cargo run
```

You should see output similar to:

```
Character Name: Arin
Class: Fighter
Strength: 16
Dexterity: 12
```

```
Intelligence: 14
Strength Modifier: 3
Dexterity Modifier: 1
Intelligence Modifier: 2
```

Additional Exercises

Now that you understand variables and basic operations, try the following exercises:

- Change the character's class and attribute scores. Rerun the program to see how modifiers change.
- Add more attributes, such as `wisdom` and `charisma`, and calculate their modifiers.
- Modify the code to display a short character description based on their class and attributes.

Next Steps

Once you're comfortable declaring variables and performing calculations, you'll be ready to move on to more complex tasks, such as querying additional character information from the D&D API. Each task will build on your knowledge to create a richer character experience.