Option and Result Enums

Learn to Code with Rust / Section Review

The **Option** Enum

- The **Option** enum models a scenario where a type could be a present value or an absent one.
- The **Some** variant has an associated piece of data of generic type **T**.
- The **None** variant holds no associated data.

```
pub enum Option<T> {
    None,
    Some(T),
}
```

The **unwrap**, **expect**, and **unwrap_or** Methods

- The **unwrap** method attempts to extract the value from the **Some** variant.
- If the **Option** variant is **None**, the **unwrap** method will cause a panic at runtime.
- The **expect** method is identical but allows the customization of the printed error message.
- The **unwrap_or** method returns a fallback value in the case of a **None** variant.

The **Result** Enum

- The **Result** enum models the outcome of an evaluation that can produce either a success or an error.
- The **Ok** variant has an associated piece of data of generic type **T**.
- The **Err** variant has an associated piece of data of generic type **E**.
- The **Result** enum implements the **Debug** trait.

```
pub enum Result<T, E> {
    Ok(T),
    Err(E),
}
```

The **unwrap**, **expect**, and **unwrap_or** Methods

- The **unwrap** method attempts to extract the value from the **Ok** variant.
- If the **Result** variant is **Err**, the **unwrap** method will cause a panic at runtime.
- The **expect** method is identical but allows the customization of the printed error message.
- The **unwrap_or** method returns a fallback value in the case of a **Err** variant.

Extra Tidbits

- The variants are included in the Rust prelude, so **Some, None**, **Ok**, and **Err** can be written without their enum prefix.
- The **match** keyword and **Result/Option** enums are a popular combination.