Working With Data ▌ *Option*

Option

- A type that may be one of two things
  - Some data of a specified type
  - Nothing
- Used in scenarios where data may not be required or is unavailable
  - Unable to find something
  - Ran out of items in a list
  - Form field not filled out

Definition

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

Example

```rust
struct Customer {
    age: Option<i32>,
    email: String,
}

let mark = Customer {
    age: Some(22), email: "mark@example.com".to_owned(),
};
let becky = Customer {
    age: None, email: "becky@example.com".to_owned(),
};
match becky.age {
    Some(age) => println!("customer is {:?} years old", age),
    None => println!("customer age not provided"),
}
```

Example

```rust
struct GroceryItem {
    name: String,
    qty: i32,
}

fn find_quantity(name: &str) -> Option<i32> {
    let groceries = vec![
        GroceryItem { name: "bananas".to_owned(), qty: 4, },
        GroceryItem { name: "eggs".to_owned(), qty: 12, },
        GroceryItem { name: "bread".to_owned(), qty: 1, },
    ];
    for item in groceries {
        if item.name == name {
            return Some(item.qty);
        }
    }
    None
}
```

Recap

- *Option* represents either some data or nothing
  - *Some(variable_name)*
    - Data is available
  - *None*
    - No data is available
- Useful when needing to work with optional data
- Use *Option<type>* to declare an optional type