Unit Testing

Live Examples

Testing Doubles

- **Never** use == to compare two Doubles
- Check if the difference between the Doubles is less than a *small* value
  - Small enough to not interfere with your logic
  - Large enough to ignore truncation errors

```
val b: Double = 0.1
val c: Double = b * 3
val expected: Double = 0.3
assert(c == expected)
```

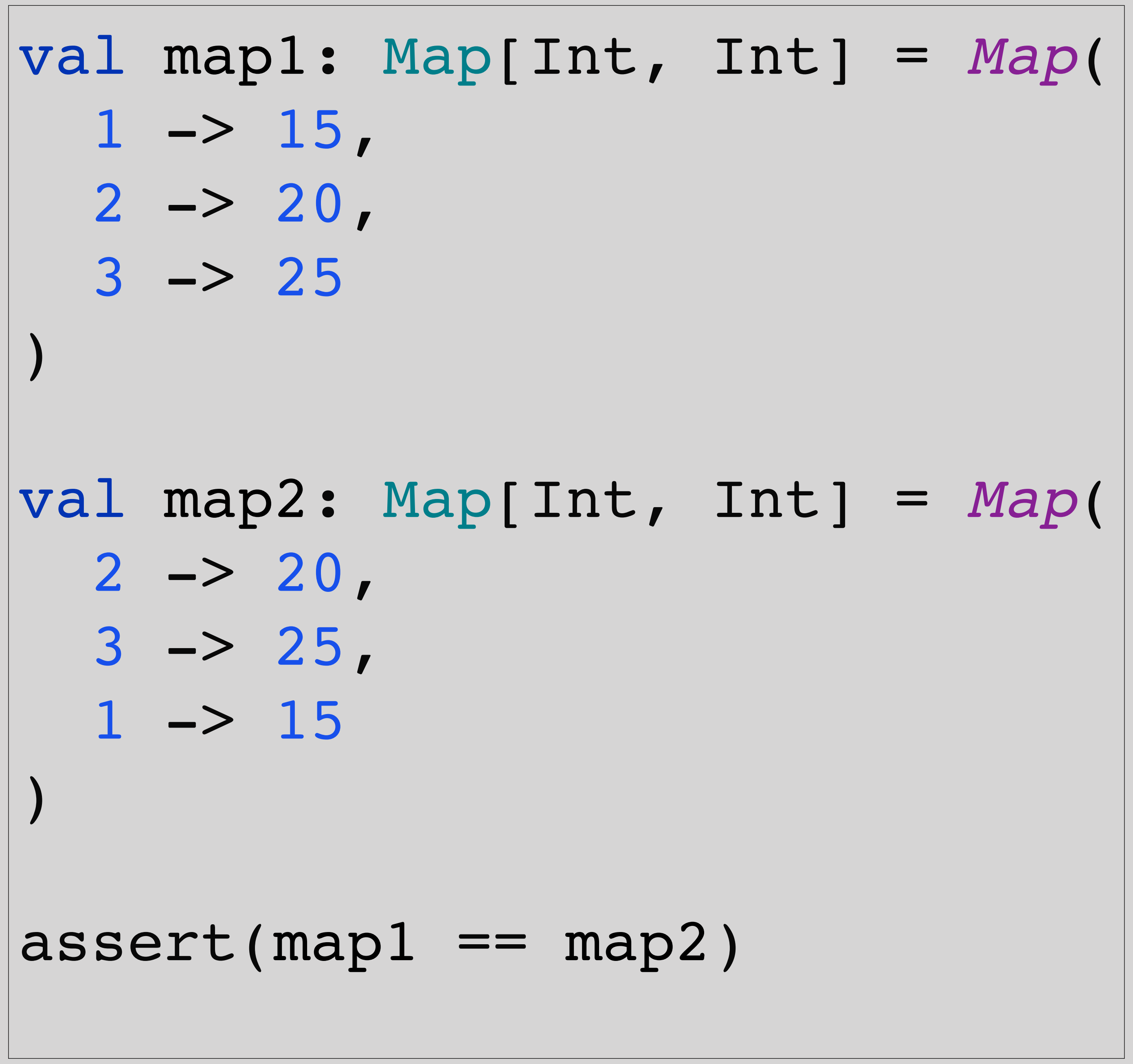
➡ **Fails**

```
val epsilon: Double = 0.00000001
val b: Double = 0.1
val c: Double = b * 3
val expected: Double = 0.3
assert(Math.abs(c - 0.3) < epsilon)
```

➡ **Passes**

Testing Maps

- Use == to compare two Maps
- Scala will check if each key exists in both maps and map to the same value in both
- Order does not matter in key-value data structures

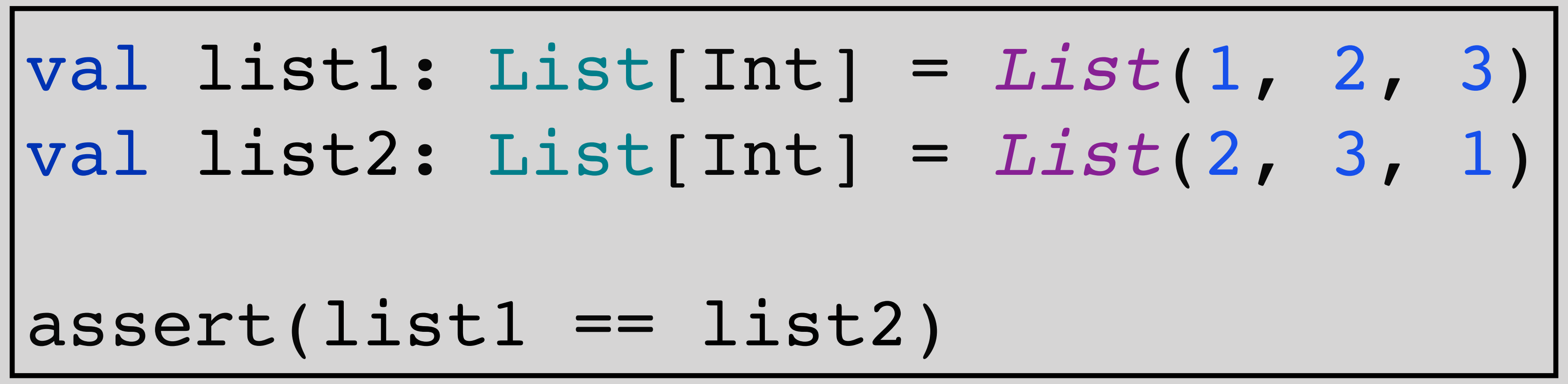


```scala
val map1: Map[Int, Int] = Map(
  1 -> 15,
  2 -> 20,
  3 -> 25
)

val map2: Map[Int, Int] = Map(
  2 -> 20,
  3 -> 25,
  1 -> 15
)

assert(map1 == map2)
```

Passes

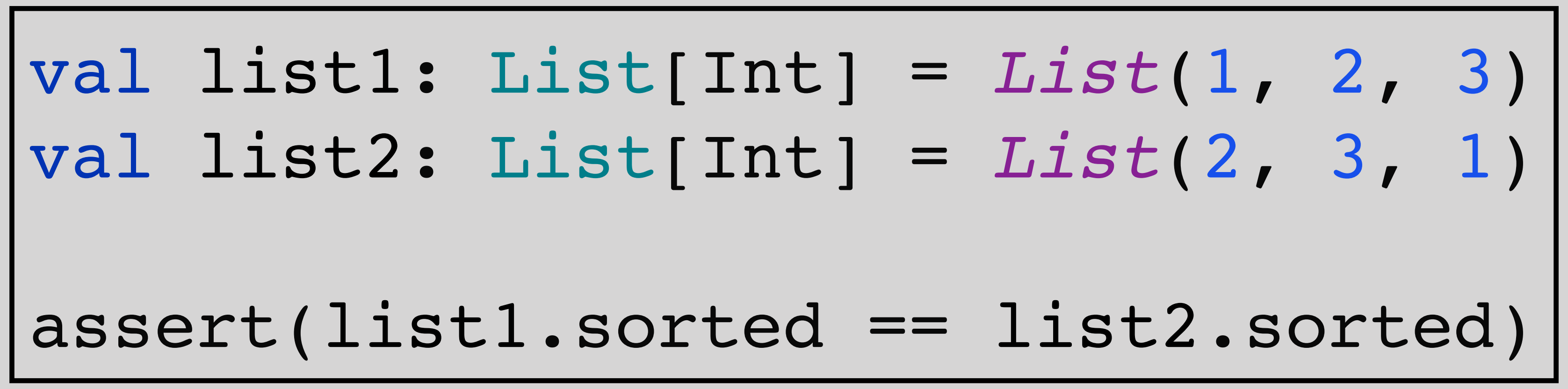
Testing Lists

- Use == to compare two Lists
- **Order matters in lists!**
    - Scala will check if both lists contain the same elements **in the same order**
- If you only care about the values, not the order, **sort** both lists before comparing



```scala
val list1: List[Int] = List(1, 2, 3)
val list2: List[Int] = List(2, 3, 1)

assert(list1 == list2)
```

Fails



```scala
val list1: List[Int] = List(1, 2, 3)
val list2: List[Int] = List(2, 3, 1)

assert(list1.sorted == list2.sorted)
```

Passes

Example

- **Testing**: Test a method that takes a String and returns a List of all the anagrams of the input

Recap of Anagrams

- Comparing Lists
  - Can use ==
  - Elements and order must match
  - Can sort the Lists if the order is not important
- It will not always be easy to know that a method is correct
  - My method should be very difficult for you to read at this point in your career
- How will you be confident that my code is correct on all inputs?
  - Thorough unit testing!
- How will you be confident that code you write is correct on all inputs?
  - Thorough unit testing!

Example

- **Testing**: Test a method that takes a List of Ints and returns a histogram of the Ints as a Map. The should map each unique integer that appears in the input List to the number of times that Int appears

- **Functionality**: Implement the histogram method

- Live Walkthrough

Example

- **Testing**: Test a method that takes a List of Ints and returns a List containing all of the most frequent values in the input

- **Functionality**: Implement this method

- Live Walkthrough