COMPARISON OPERATORS

Categories of Comparison Operators

- binary operators
- evaluate to a `bool` value

Identity Operations	`is`	`is not`			compares memory address – any type
Value Comparisons	`==`	`!=`			compares values – different types OK, but must be compatible
Ordering Comparisons	`<`	`<=`	`>`	`>=`	doesn't work for all types
Membership Operations	`in`	`not in`			used with iterable types

Numeric Types

We will examine other types, including iterables, later in this course

Value comparisons will work with all numeric types

Mixed types (except complex) in value and ordering comparisons is supported

Note: Value equality operators work between floats and Decimals, but as we have seen before, using value equality with floats has some issues!

```
10.0 == Decimal('10.0')   → True
```

```
0.1 == Decimal('0.1')     → False

Decimal('0.125') == Fraction(1, 8)  → True

True == 1    → True

True == Fraction(3, 3)  → True
```

Ordering Comparisons

Again, these work across all numeric types, except for complex numbers

```
1 < 3.14 → True

Fraction(22, 7) > math.pi   → True

Decimal('0.5') <= Fraction(2, 3)   → True

True < Decimal('3.14')  → True

Fraction(2, 3) > False   → True
```

Chained Comparisons

```
a == b == c   →   a == b and b == c

a < b < c     →   a < b and b < c


1 == Decimal('1.0') == Fraction(1,1)     → True

1 == Decimal('1.5') == Fraction(3, 2)    → False


1 < 2 < 3     → 1 < 2 and 2 < 3   → True


1 < math.pi < Fraction(22, 7)

   → 1 < math.pi and math.pi < Fraction(22, 7)

   → True
```

Chained Comparisons

```
a < b > c      → a < b and b > c

5 < 6 > 2      → 5 < 6 and 6 > 2      → True

5 < 6 > 10     → 5 < 6 and 6 > 10     → False


a < b < c < d      → a < b and b < c and c < d

1 < 2 < 3 < 4      → 1 < 2 and 2 < 3 and 3 < 4      → True

1 < 10 > 4 < 5     → 1 < 10 and 10 > 4 and 4 < 5    → True
```

```
if my_min == cnt < val > other <= my_max not in lst:
    # do something
```

Code