DECIMALS

MATHEMATICAL OPERATIONS

Some arithmetic operators don't work the same as floats or integers

`//` and `%` → also `divmod()`

The `//` and `%` operators still satisfy the usual equation: `n = d * (n // d) + (n % d)`

But for integers, the `//` operator performs floor division → `a // b = floor(a/b)`

For Decimals however, it performs truncated division → `a // b = trunc(a/b)`

negative numbers!

```
10 // 3 → 3      Decimal(10) // Decimal(3) → Decimal(3)
-10 // 3 → -4    Decimal(-10) // Decimal(3) → Decimal(-3)
```

Boils down to the algorithm used to actually perform integer division

– ——————

- figure out the sign of the result
- use absolute values for divisor and dividend
- keep subtracting **b** from **a** as long as **a >= b**
- return the signed number of times this was performed

— res is + `10 - 3 = 7` `7 >= 3` `7 - 3 = 4` `4 >= 3` `4 - 3 = 1` `1 < 3 - STOP` return 3

—— res is – `10 - 3 = 7` `7 >= 3` `7 - 3 = 4` `4 >= 3` `4 - 3 = 1` `1 < 3 - STOP` return -3

this is basically the same as truncating the real division

`trunc(10/3) → 3` `trunc(-10/3) → -3`

But `n = d * (n // d) + (n % d)` is still satisfied

`n = -135, d = 4`

	Integer	Decimal
`-135 // 4`	-34	-33
`-135 % 4`	1	-3
`d * (n // d) + (n % d)`	`4 * (-34) + 1 = -135`	`4 * (-33) + (-3) = -135`

Other Mathematical Operations

The Decimal class defines a bunch of various mathematical operations, such as sqrt, logs, etc.

But not all functions defined in the math module are defined in the Decimal class

E.g. trig functions

We can use the math module,

but Decimal objects will first be cast to floats

– so we lose the whole precision mechanism that made us use Decimal objects in the first place!

Usually will want to use the math functions defined in the Decimal class if they are available

```
decimal.getcontext().prec = 28

x = 0.01
x_dec = Decimal('0.01')

root = math.sqrt(x)
root_mixed = math.sqrt(x_dec)
root_dec = x_dec.sqrt()

print(format(root, '1.27f'))                    → 0.100000000000000005551115123
print(format(root_mixed, '1.27f'))              → 0.100000000000000005551115123
print(root_dec)                                 → 0.1

print(format(root * root, '.27f'))              → 0.010000000000000001942890293
print(format(root_mixed * root_mixed, '.27f'))  → 0.010000000000000001942890293
print(root_dec * root_dec)                      → 0.01
```

Code