Implementing value class other funⁿ

A) c = value(2.0) + value(3.0) = value(5.0)

this is correct

c = value(2.0) + 3.0 ⇒ this will give error.

b/z, value.add(float), but add funⁿ in value class expect both args to be value object

here, self = value, other = float ✓

so, we have to convert float to a value object. (see code)

---

c = 2.0 + value(3.0)

float — value.

c = float.add(value)

self — other

(so, we will get error, we cannot sum float & value)

To resolve this issue, radd dunder method.

$c = \underbrace{2.0}_{self} + \underbrace{value(3.0)}_{other}$

float.add(value)

So, use radd.

$\in \Rightarrow \underbrace{value}_{self}.radd(\underbrace{float}_{other})$

(we know that, $a+b = b+a$)

(see code)

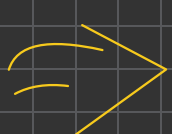
---

for --add-- we have (+) as a symbol,

for --radd-- do we have any other symbol?

No, if add fun^n not work, then radd is automatically called,

---



Similarly multiplication will be exactly similar to add.

A) c = value(2.0) * value(3.0)

c = value(6.0) ✓ works

(value.mul(value)

B) c = value(2.0) * 3.0 (error)

(value.mult(float)

(error will be float has no object data)

to correct it we need to convert float → value.

C) c = 2.0 * value(3.0)

float.mul(value) (err. cannot

self sum other float & value)

value.rmul(float)

self other

(but we know that, a*b = b*a)

(see code)

a) = a = Value (2.0) (this is correct)

a = -Value (2.0) (this will give err, as we have not implement --neg--)

a = -Value (2.0)

Value.neg()

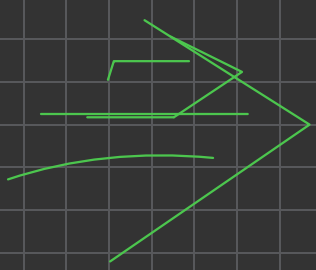
↓ seq.

-value (2.0) ≈ value (2.0) * (-1)

↓↓

Value.mul(-1)

this is double.



pow = ?

c = (Value(2.0)) ** 3 (this will give error)

--pow-- → dunder fun^n

pow → int / float

Value.pow(int/float)

(Value(2.0).pow(3))

(Value(2.0)) ** 3

3 → ** 3 → Value(8.0, (2.0,), (**3))

Value(2.0) →