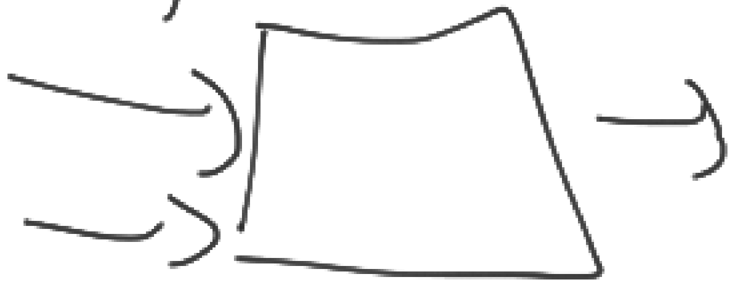
Abstraction

computation → function



+

Data

Encapsulation

- property

Object

· attributes

· methods

① Define

② Instantiate

③ Access

Dot operator

Avatar

Racing car

+

Game

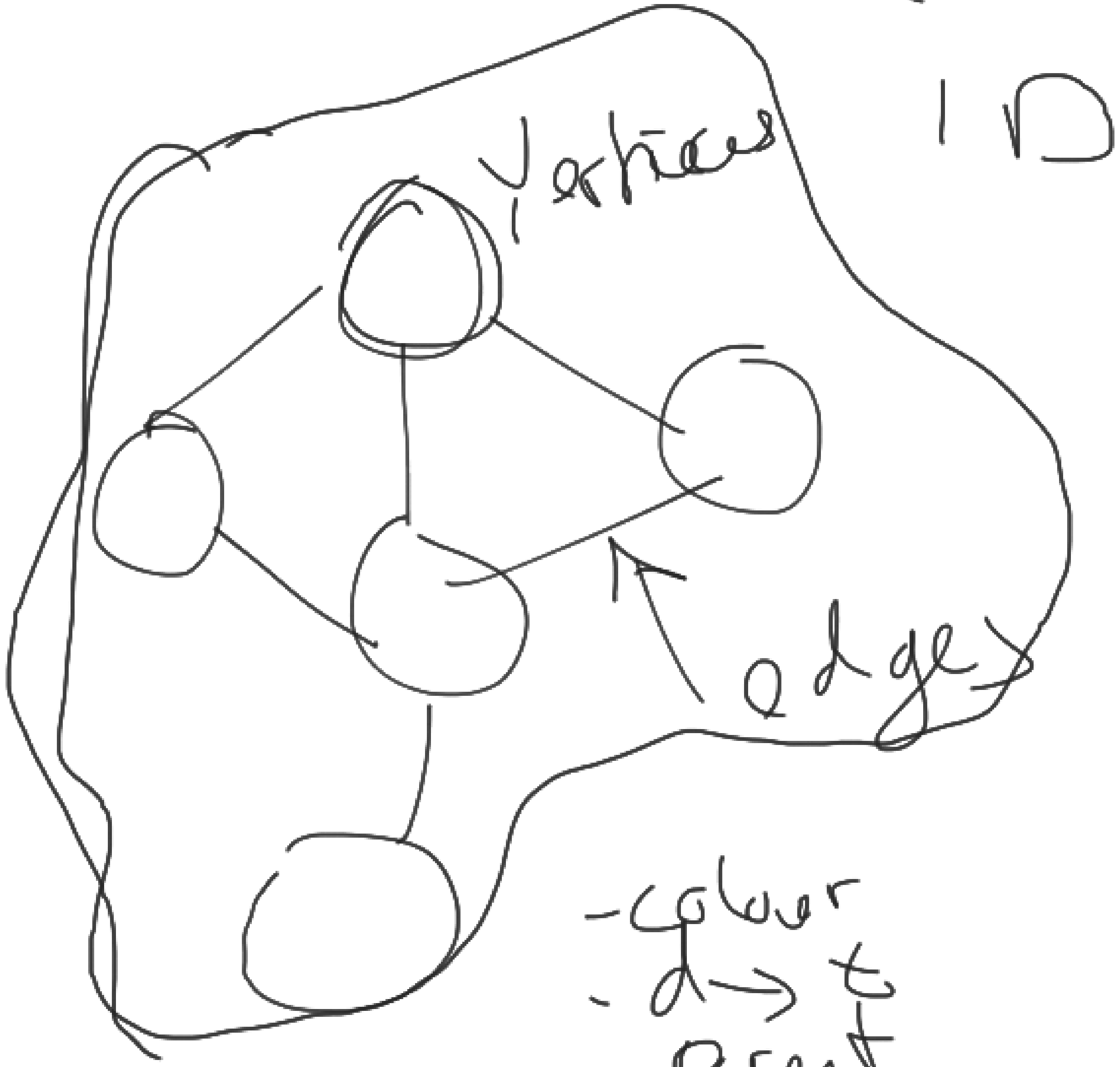
Calculator

Data Structure

ADT → Stack
Queues

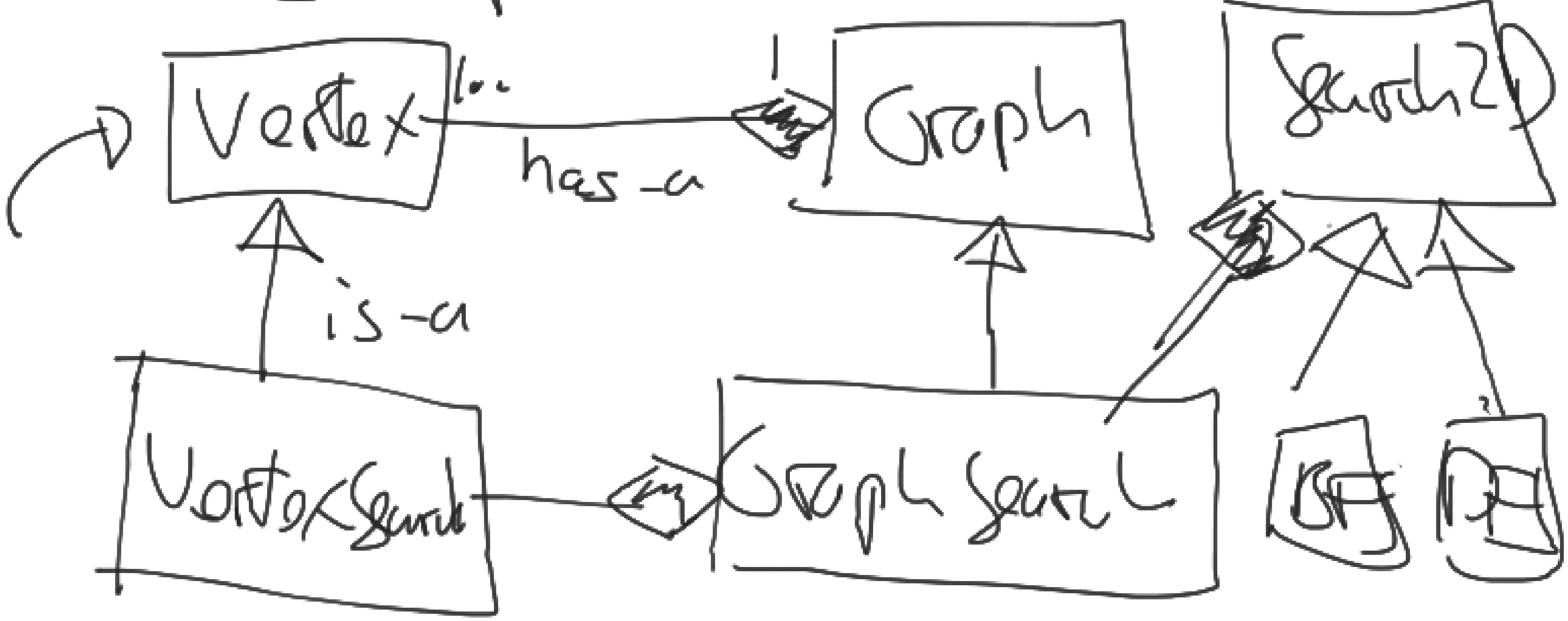
FILO LIFO
FIFO DRY

1D — 2D



-colour
-d → t
-parent
-f

Graph Search



Liskov Substitution

Open-close

Special Methods

__init__( ) : always called initialization

__str__( ) : return str

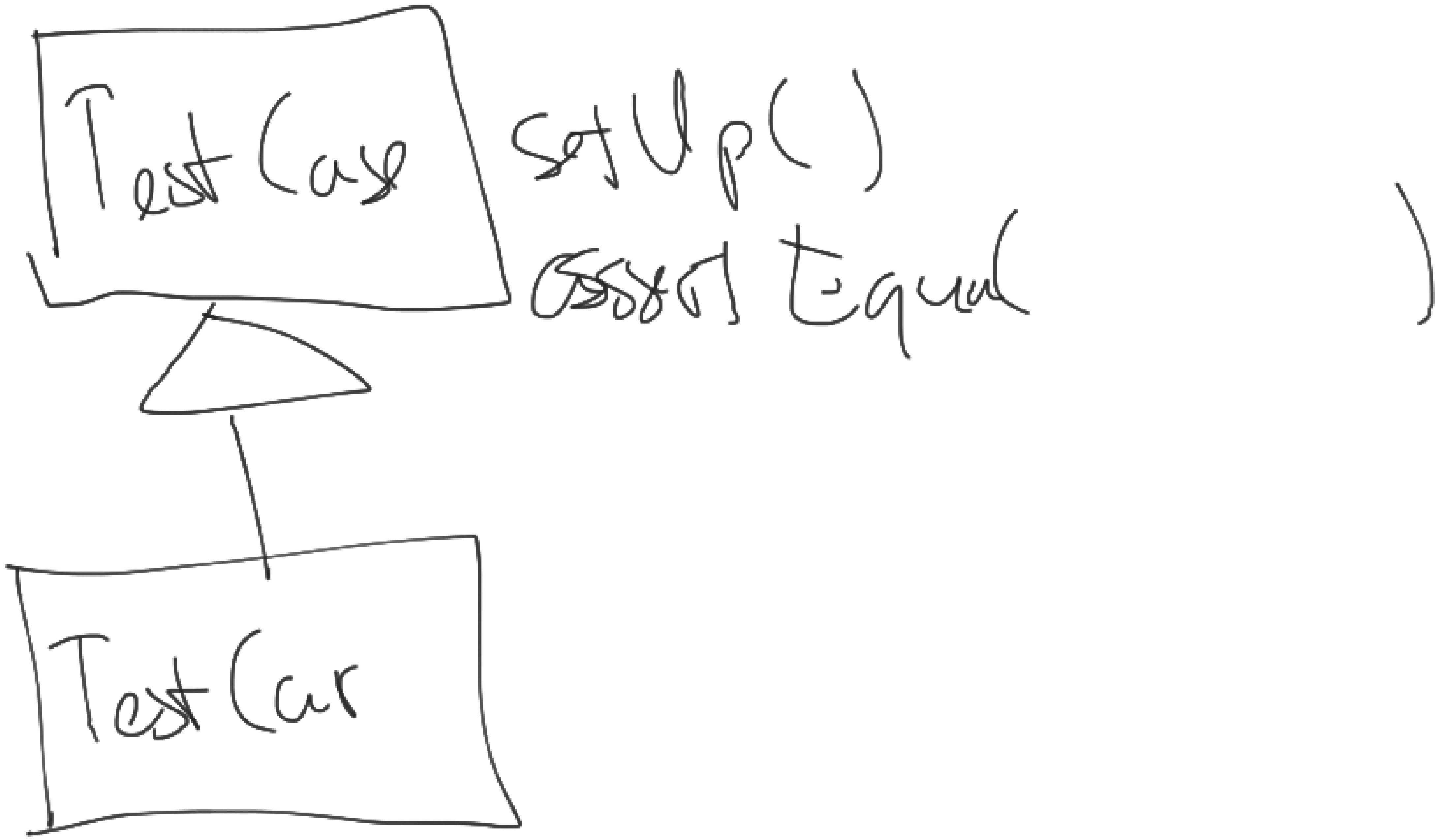
__iter__( ) : for_ in iterable :

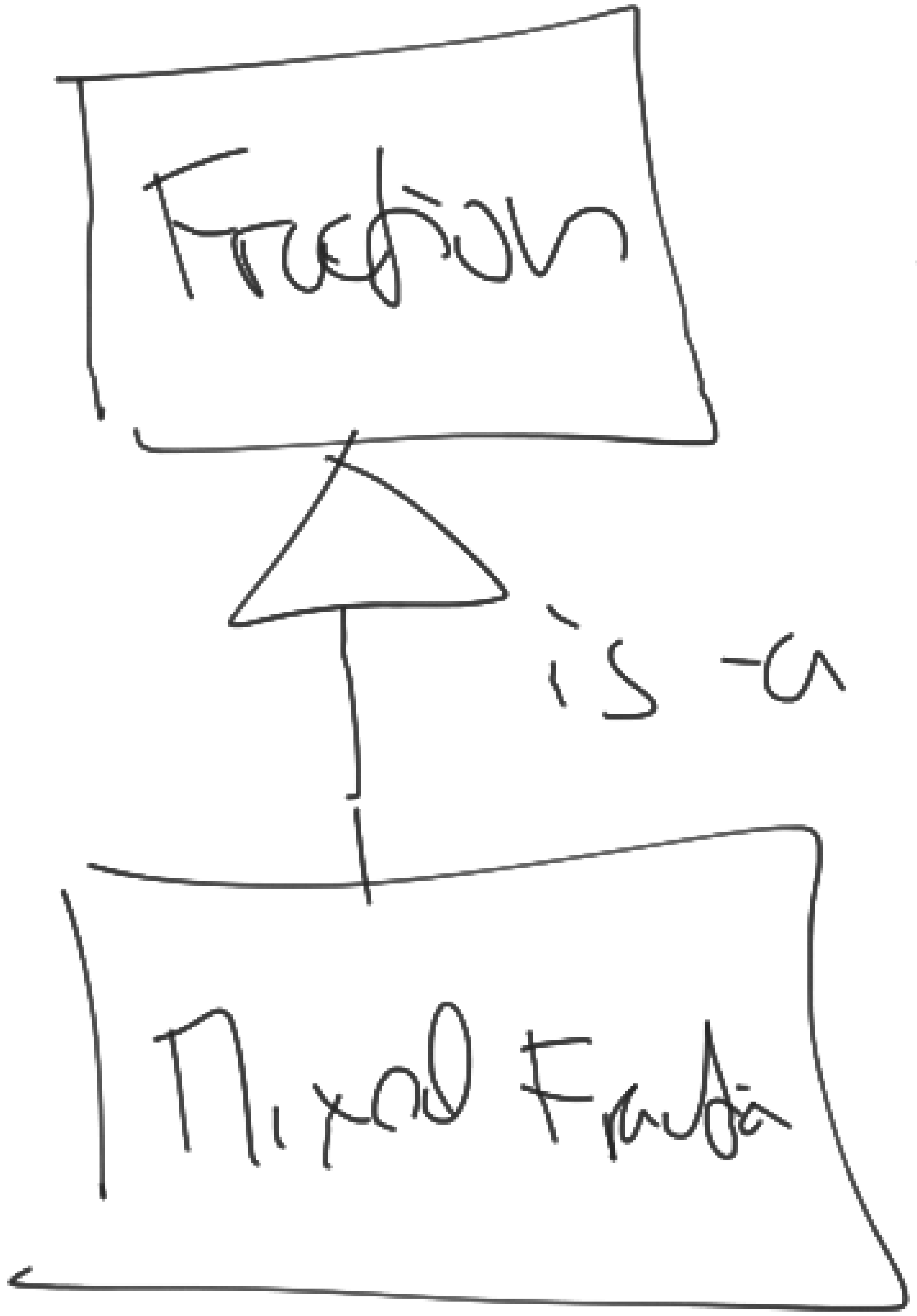
__contains__( ) : item in object

__add__(self, other) : ↑ a + b

__sub__(self, other) : a - b

__eq__( ) : a == a


TestCase
setUp( )
assertEqual(   )
TestCar



$$\frac{a}{b} + \frac{c}{d}$$

$$= \underline{\underline{\frac{a \cdot d + b \cdot c}{b \cdot d}}}$$

$$\boxed{\frac{1}{2}}$$ 0.5 float

$$\frac{1}{3} \qquad \frac{3}{0} \Rightarrow \frac{3}{.1} = \}$$

$$1\frac{1}{2} = \frac{3}{2}$$

numerator

denominator

$$\frac{2}{4} \rightarrow \frac{1}{2}$$

$$\frac{1}{2} + \frac{2}{3} = \frac{3+4}{6} = \frac{7}{6} \}$$

$$\frac{1}{3} + \frac{1}{6} = \frac{6+3}{\cancel{6}\,18} = \frac{9}{18} = \frac{1}{2}$$

gcd(9, 18) = 9

$1\frac{2}{3}$ ⇒ MixedFraction (2, 3, 1) → $\frac{5}{3}$

$\frac{2}{3}$ ⇒ MixedFraction (2, 3)

get_three_num ( ) → (2, 3, 1)

$$\frac{5}{3} + \frac{1}{2} = \frac{10+3}{6} = \frac{13}{6}$$

$1\frac{1}{3}$

[3, 2, 1, 5, 6]

ADT

↳ append()

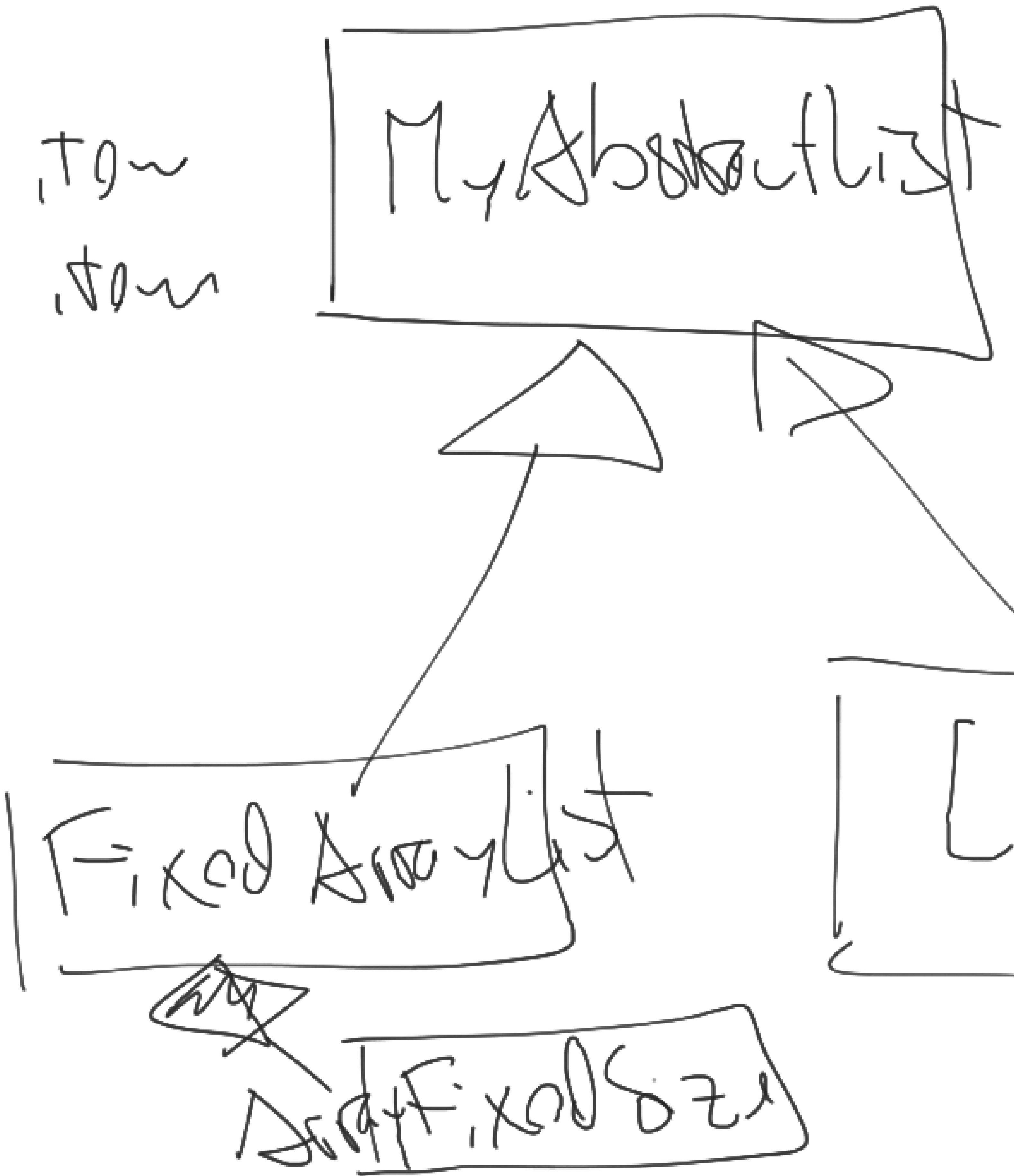
a[idx] get

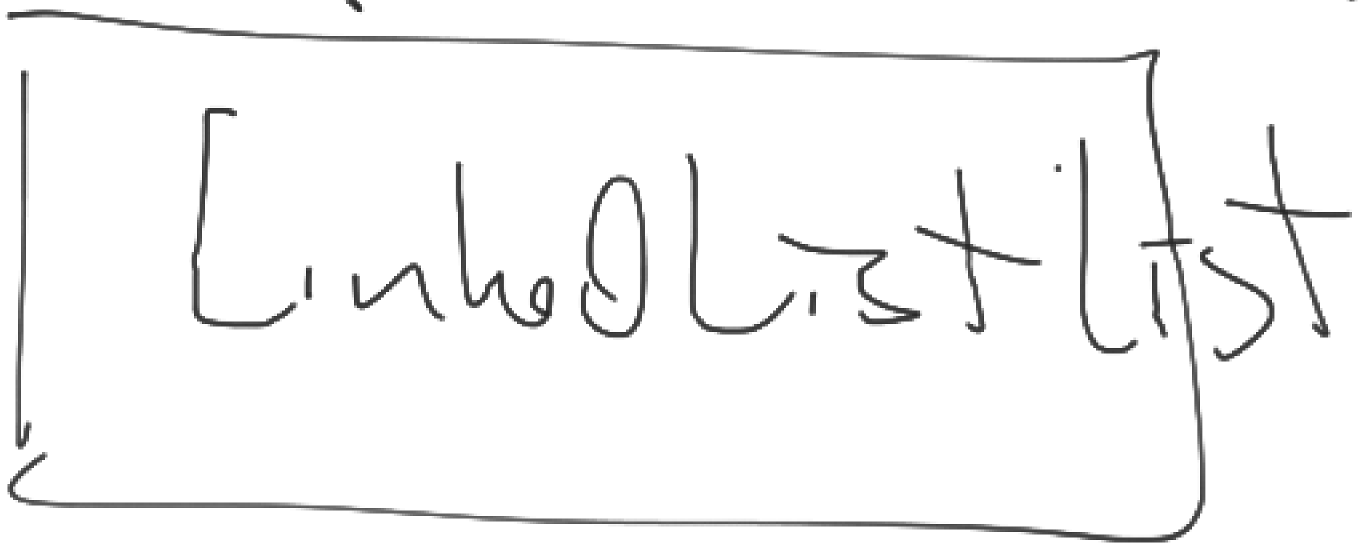
a[idx] = val set

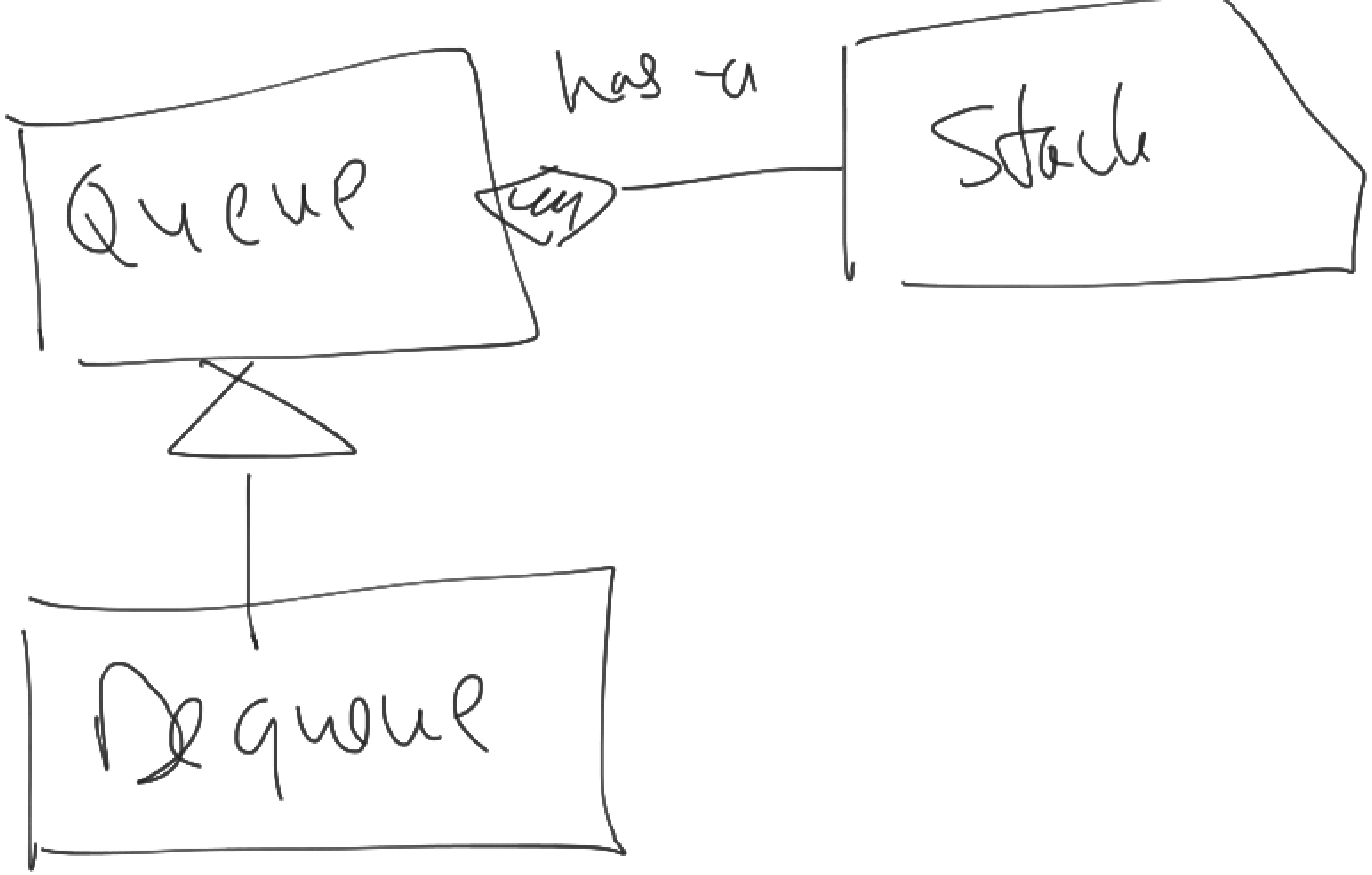
a.insert(idx, item)

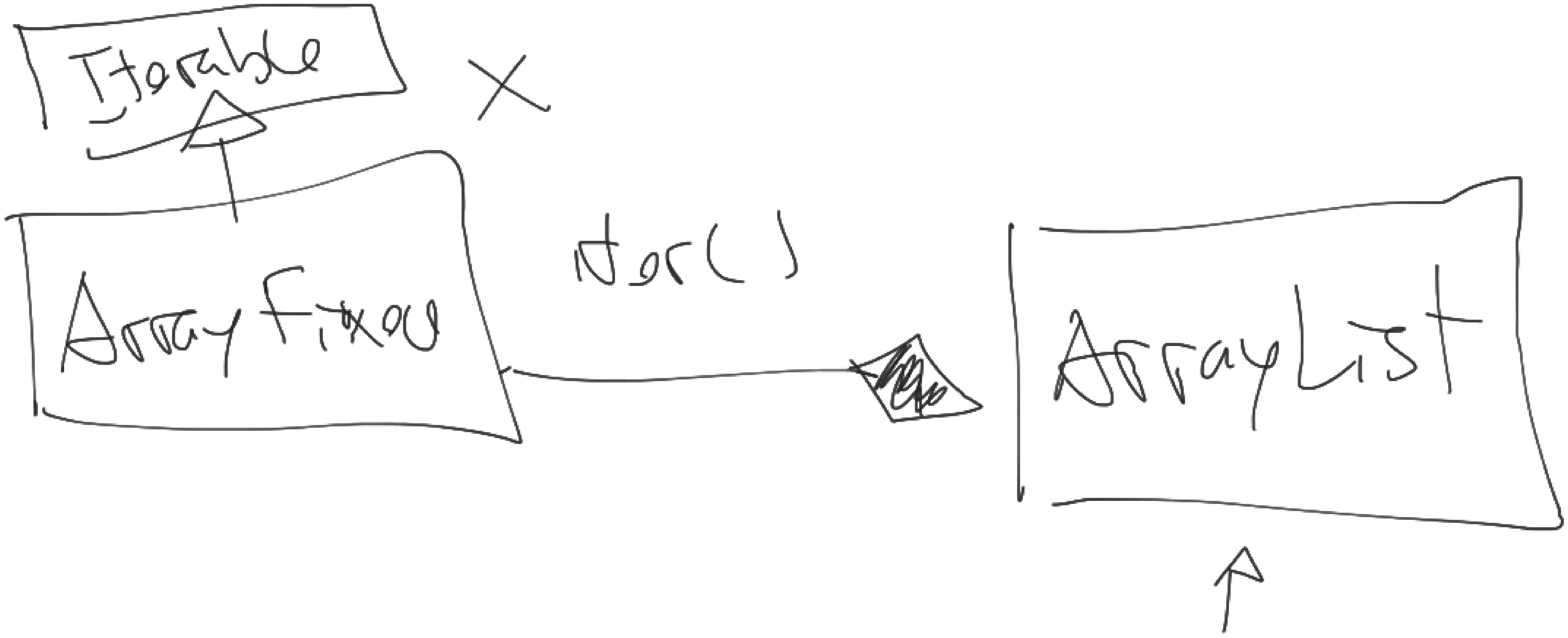
get item

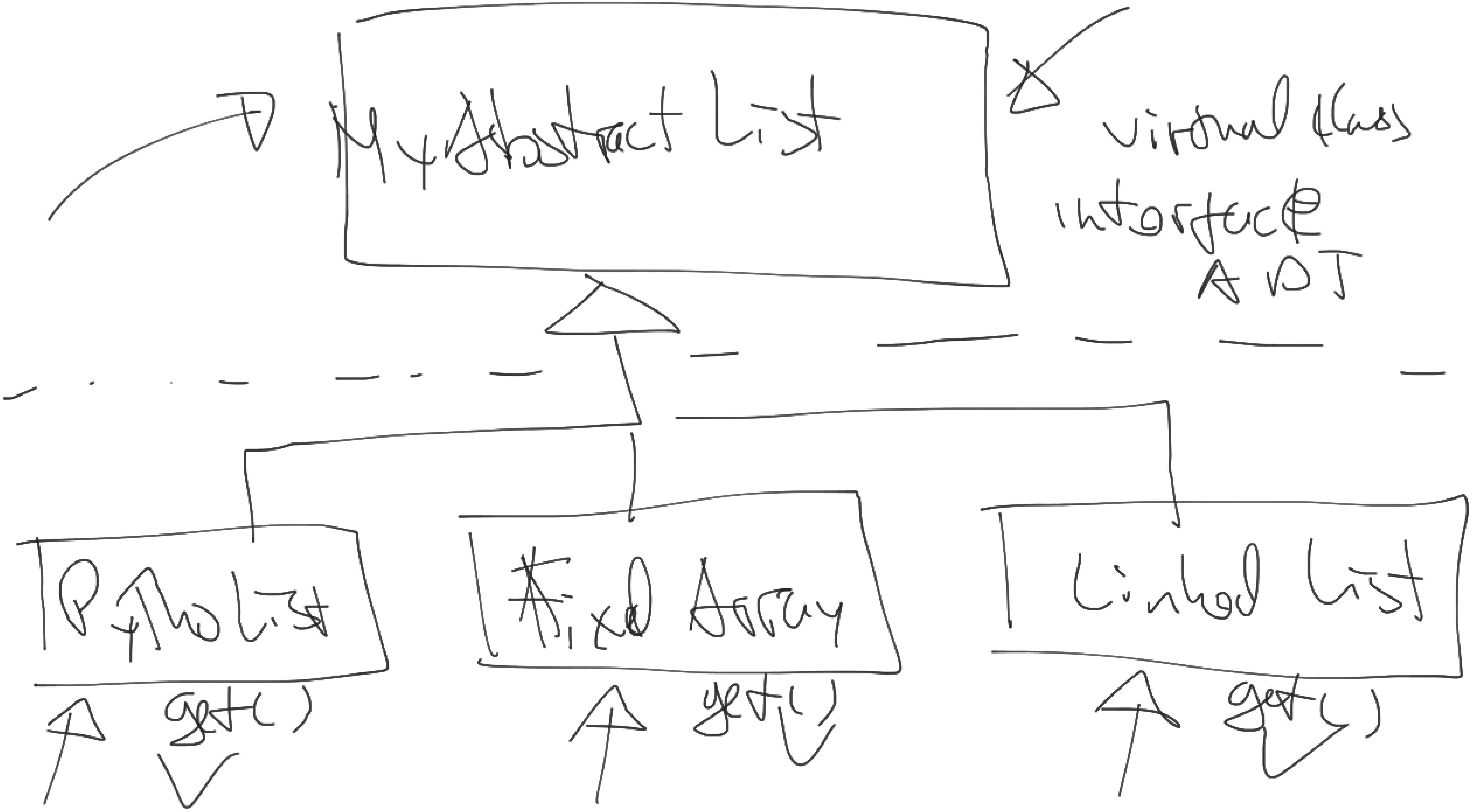
set item





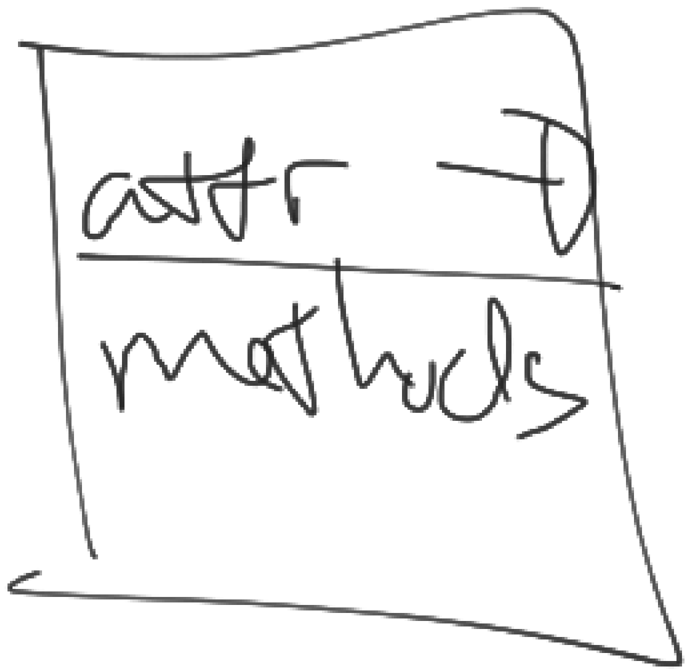

has -a
Queue
Stack
Dequeue

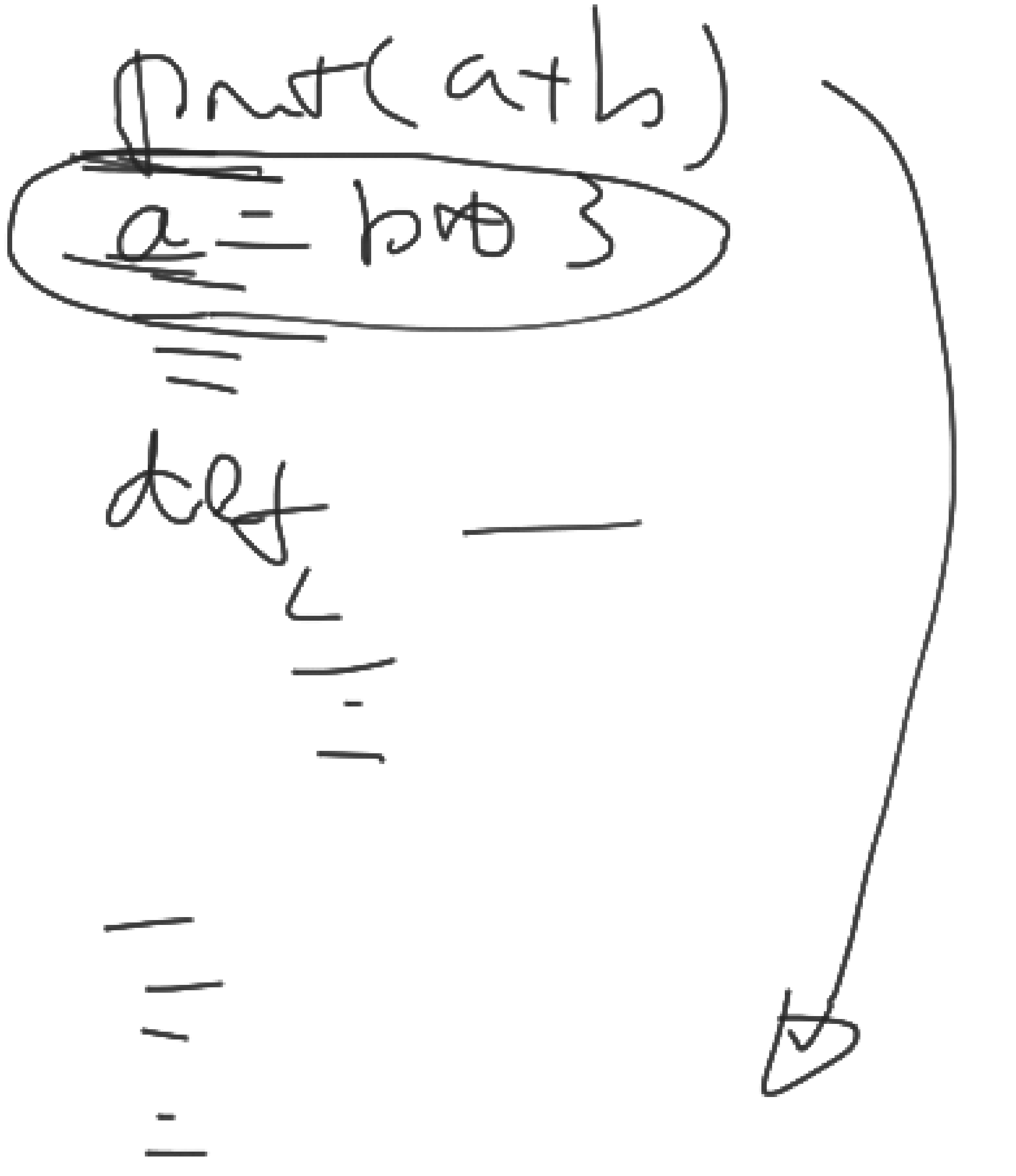

Iterable
X
ArrayFixed
iter()
ArrayList


MyAbstract List
virtual class
interface
ADT
PythoList
get()
Fixed Array
get()
Linked List
get()

Procedural → functions

Object Oriented

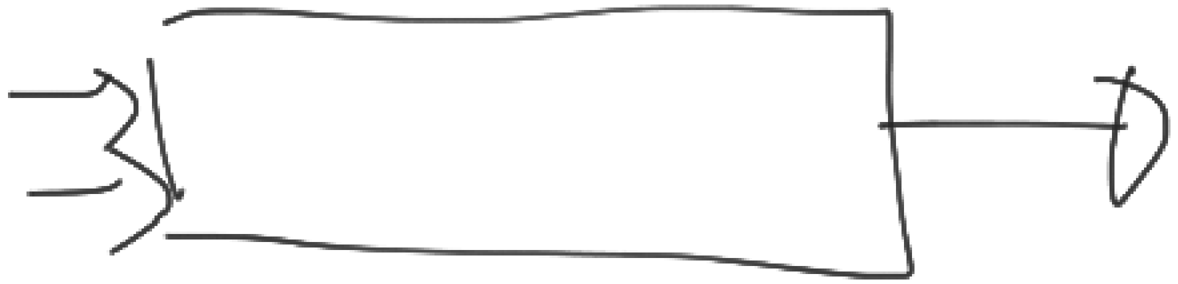




side effect

functional programming

↳ pure function



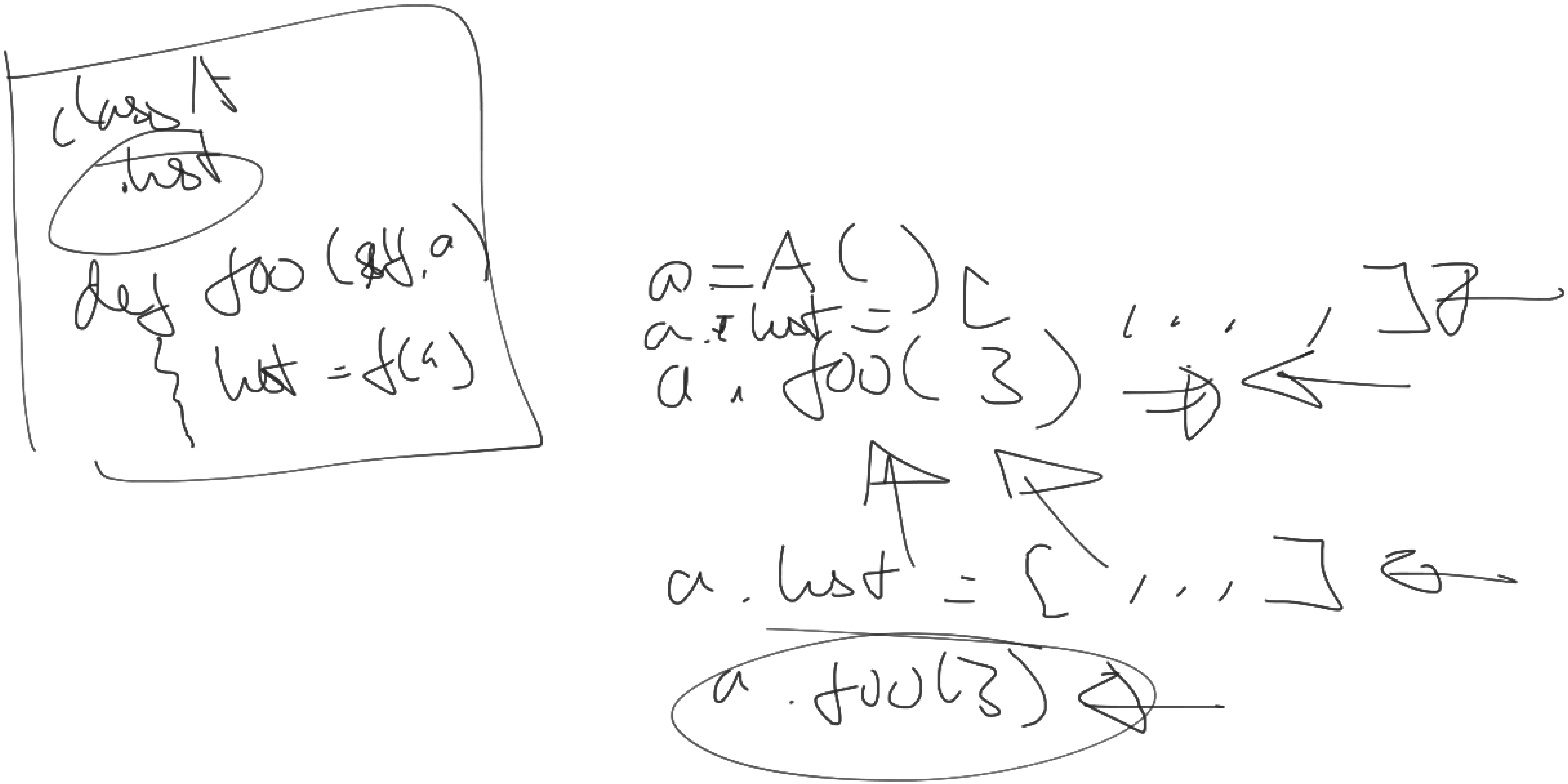
testing

formal method

$\boxed{f(x)}$

```
Initial guess
Check is close to a solution ?  ①
If not :
    update my guess  ②
else
    done
```

iterative



```
class A
    .lst
    def foo(self, a)
        lst = f(a)
```

```
a = A()
a.lst = [ , , , ]
a.foo(3)

a.lst = [ , , , ]
a.foo(3)
```

map(f, [1, 2, 3, 4])

↳ higher order

f(1) →

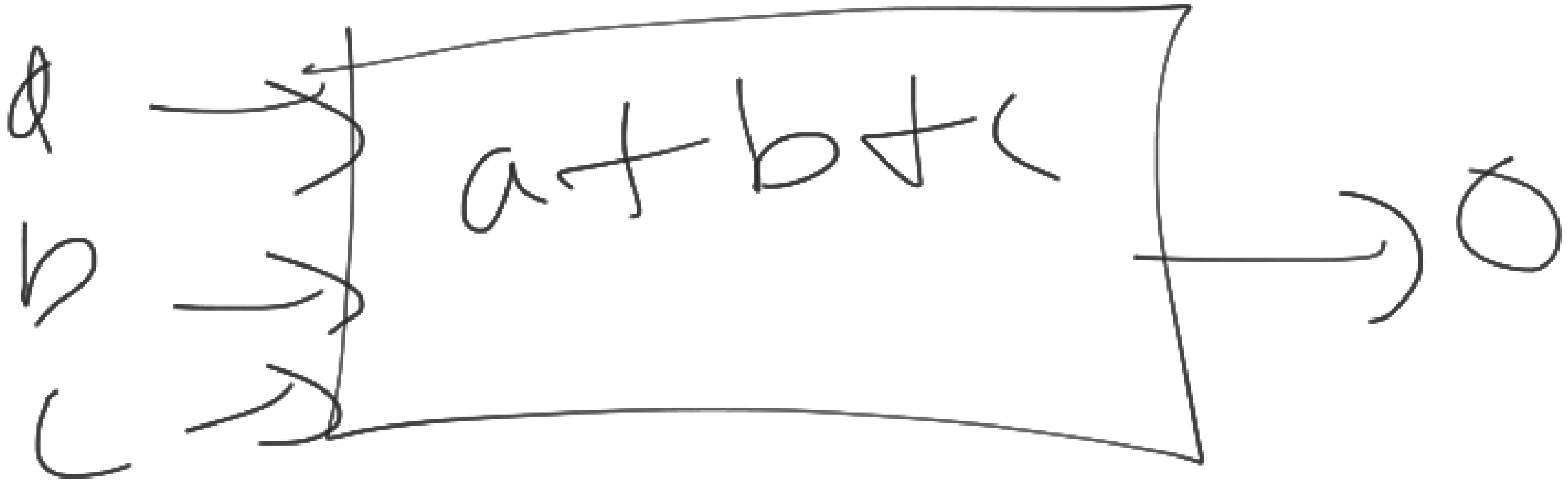
f(2) →

f(3) →

f(4) →

If-else in one line

$$\underset{\text{True}}{\boxed{\text{expression}}} \quad \text{if} \quad \boxed{\text{condition}} \quad \text{else} \quad \underset{\text{False}}{\boxed{\text{expression}}}$$



$$o = f(\underset{\uparrow}{i_1}, \underset{\uparrow}{i_2}, \underset{\uparrow}{i_3})$$

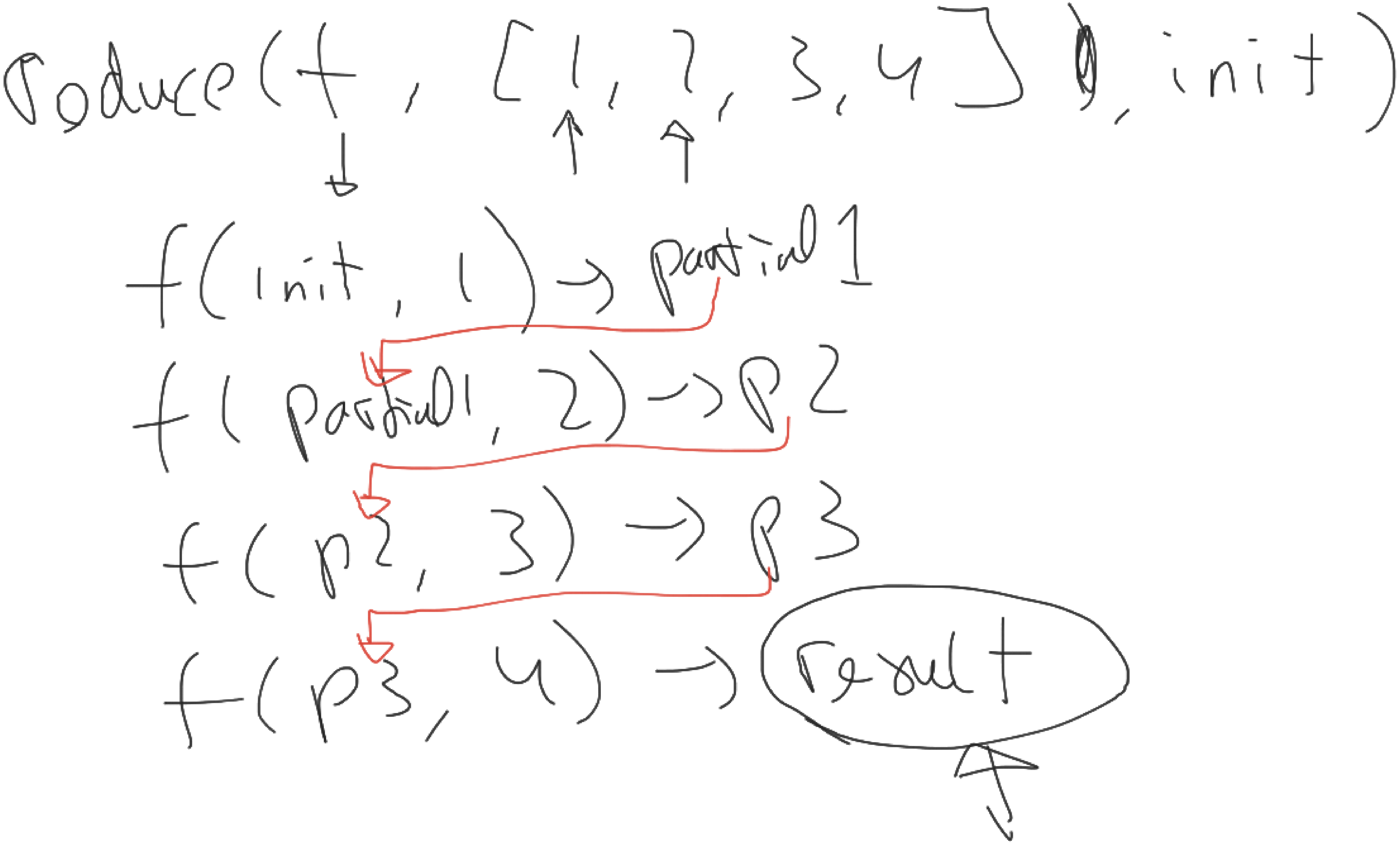
filter(f, [1, 2, 3, 4])

$f(1) \rightarrow T$ [1, 2, 4]

$f(2) \rightarrow T$

$f(3) \rightarrow F$ ✗

$f(4) \rightarrow T$



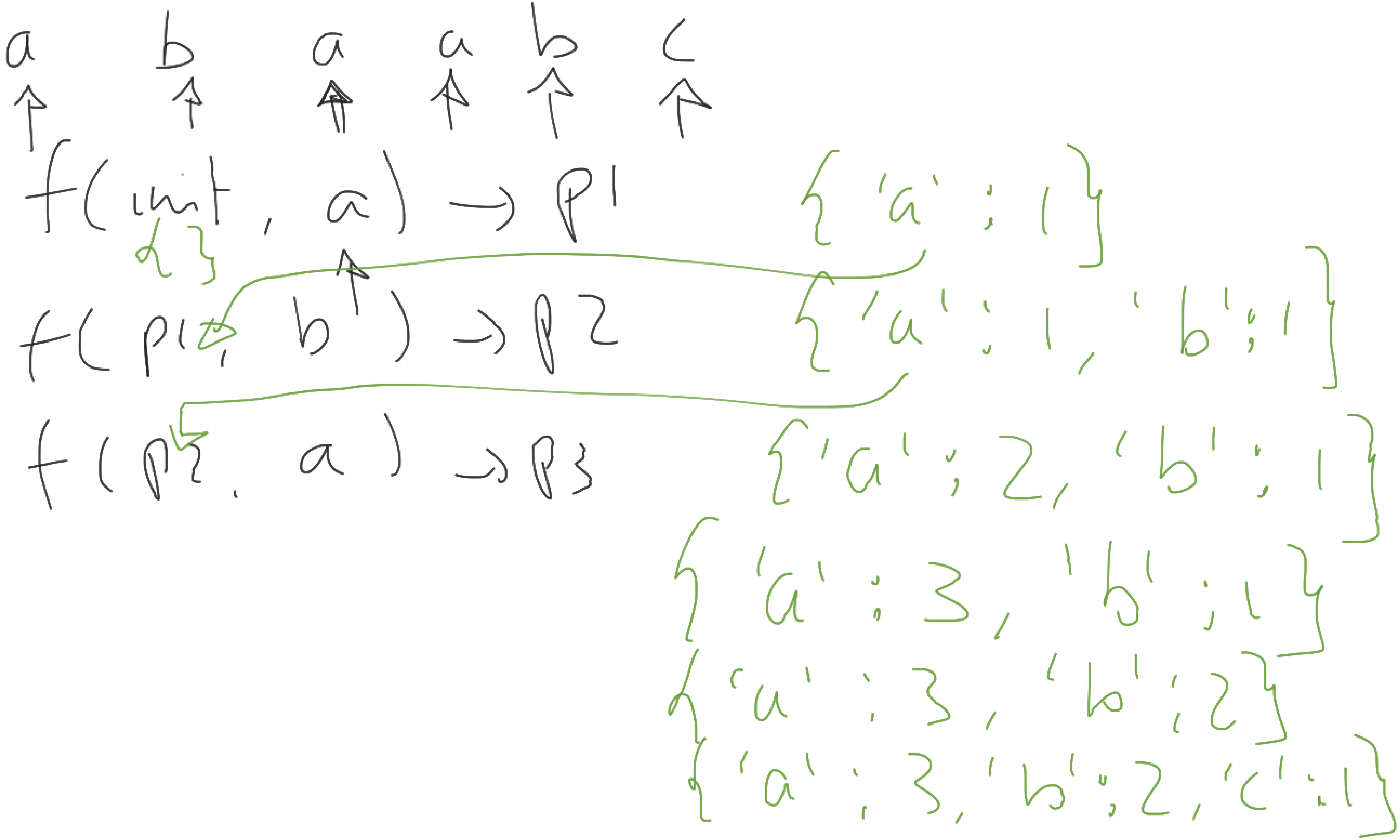
reduce(f, [1, 2, 3, 4] ~~0~~, init)

f(init, 1) → partial 1

f(partial1, 2) → p2

f(p2, 3) → p3

f(p3, 4) → result



a b a a b c

f(init, a) → p1 {'a': 1}

{}

f(p1, b) → p2 {'a': 1, 'b': 1}

f(p2, a) → p3 {'a': 2, 'b': 1}

{'a': 3, 'b': 1}

{'a': 3, 'b': 2}

{'a': 3, 'b': 2, 'c': 1}