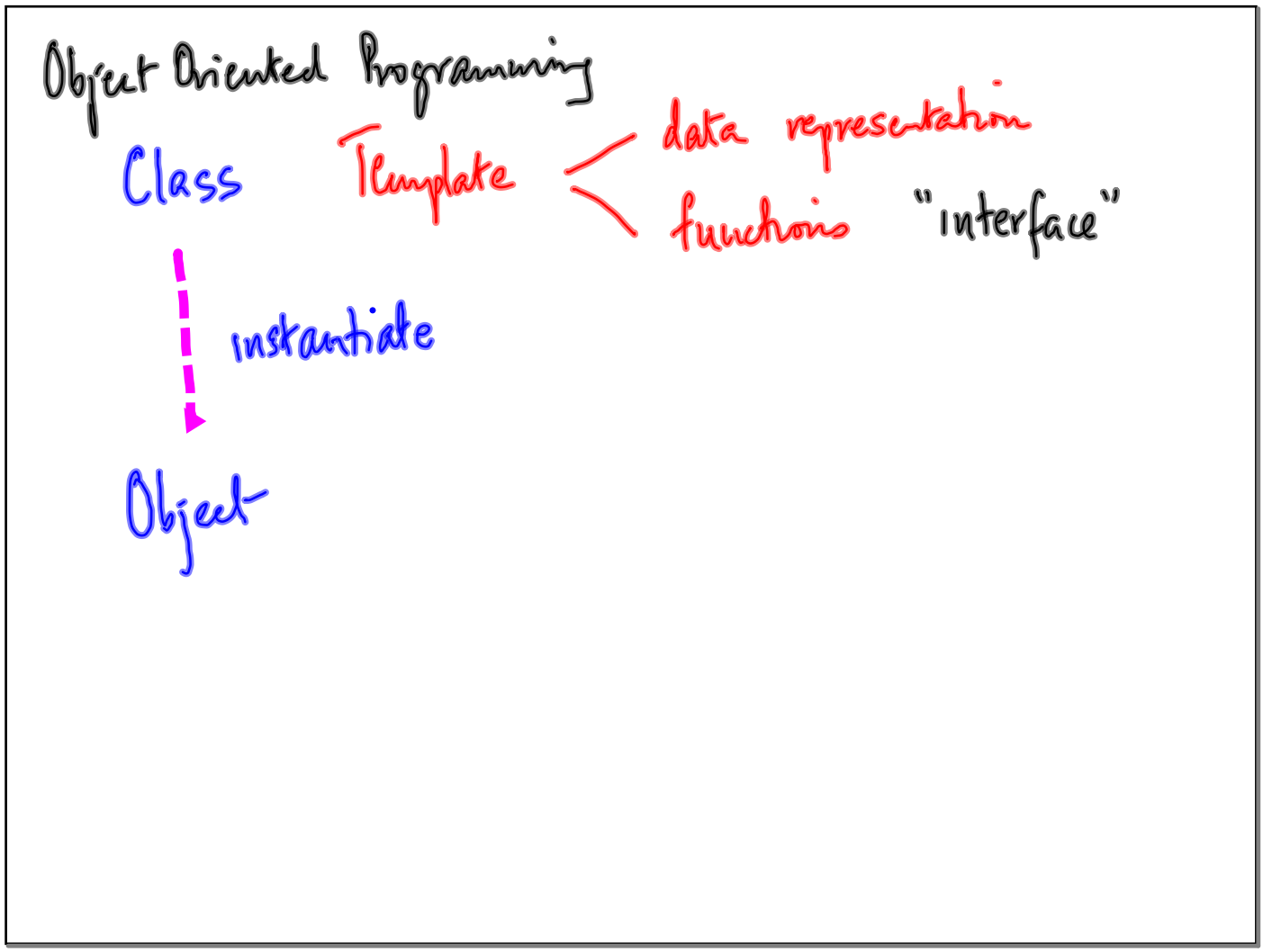
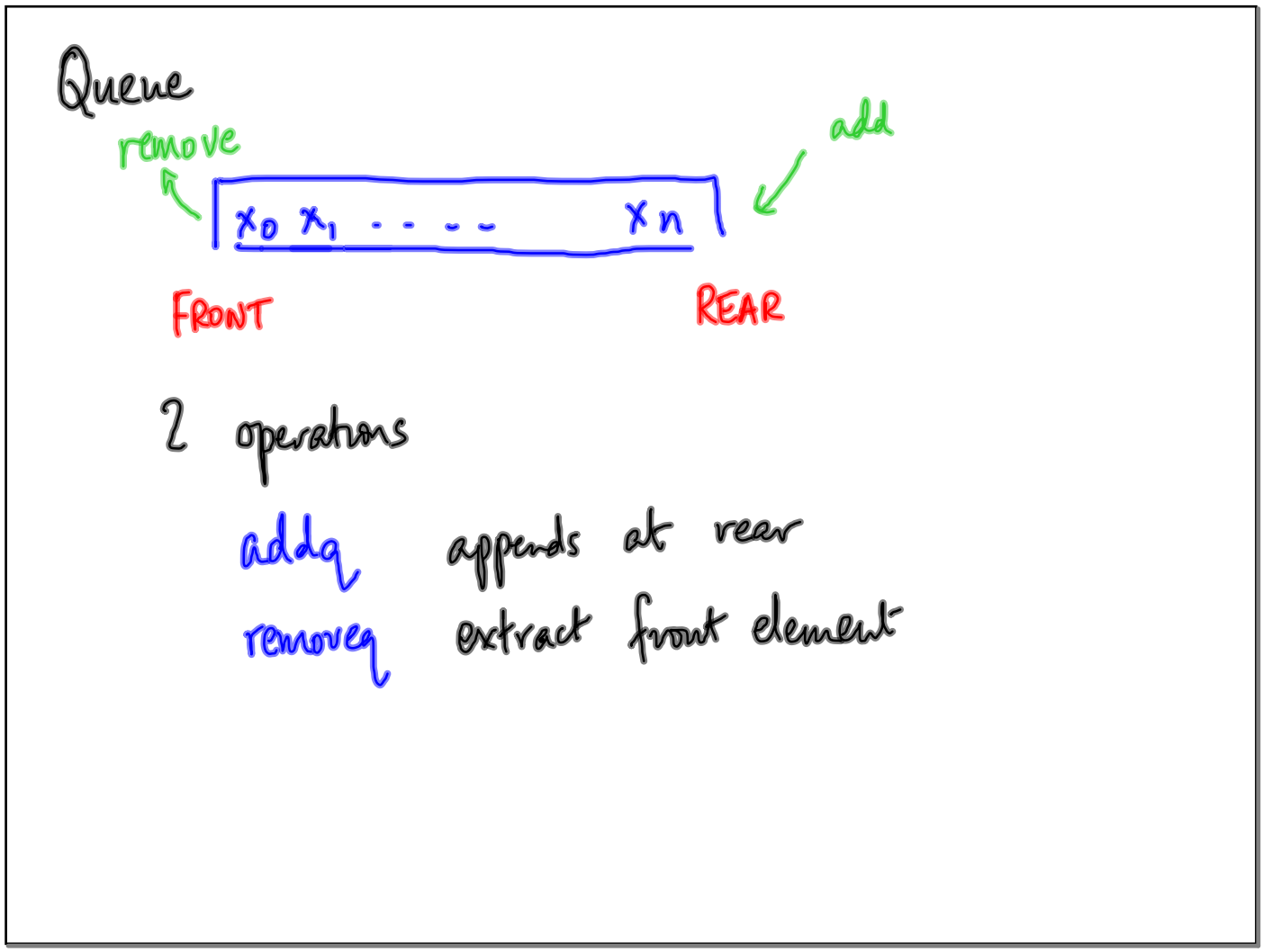
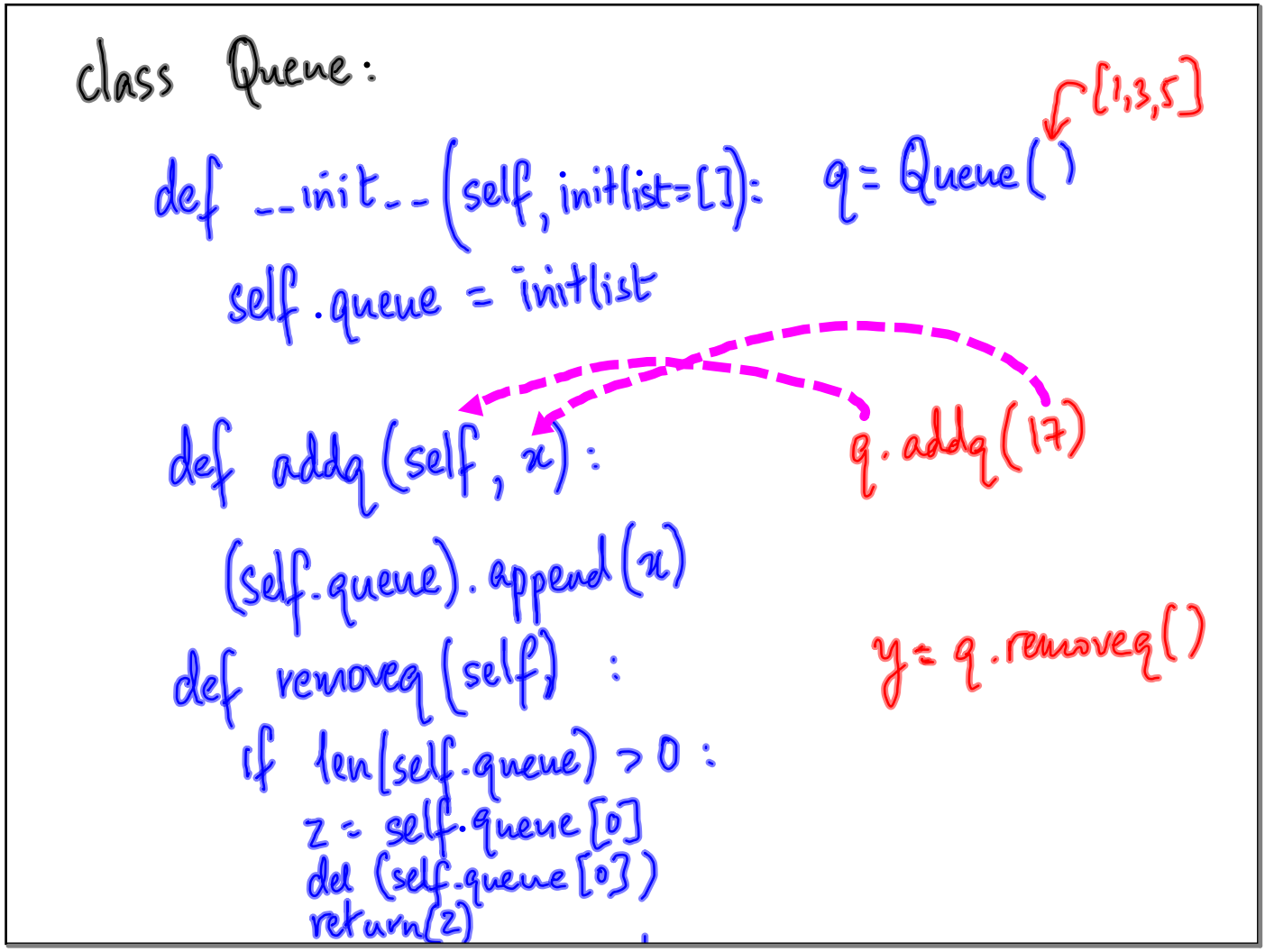

Object Oriented Programming
Class
Template
data representation
functions
"interface"
instantiate
Object


Queue
remove
add
x_0 x_1 - - - - x_n
FRONT
REAR
2 operations
addq appends at rear
removeq extract front element


class Queue:
def __init__(self, initlist=[]):
self.queue = initlist
def addq(self, x):
(self.queue).append(x)
def removeq(self):
if len(self.queue) > 0:
z = self.queue[0]
del (self.queue[0])
return(z)
q = Queue()
[1,3,5]
q.addq(17)
y = q.removeq()

Examining the contents of an object

str(x) ⟶ string representation of x

Special function __str__ which is called when we use str() on an object

```
def __str__(self):
    return (str(self.queue))
```

What about

```
q = Queue([1,3,5])
print(q.queue)
```

This works, but it shouldn't, in principle!

```
q.queue.append(77)
```

Also "works"!

Need a separation of private & public

Typically

Data representation is private

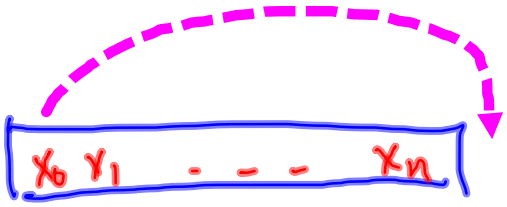
Interface functions are public

Python has no way to specify this

Everything is public!

Can use class functions internally

Add rotateq()




```
y = self.removeq()
self.addq(y)
```

like `q.removeq()`

Handle empty queue case appropriately

Can also define auxiliary local functions

```
def reverseq(self):
    def f(l):
        l.reverse()
    f(self.queue)
```

"normal" local function definition

Can use class functions as "globally" defined fns

```
q = Queue(l)
q.addq(v)
```

implicitly bound to self

```
q = Queue(l)
Queue.addq(q, v)
```

explicitly pass a parameter for self

raise ExceptionName("message")

generates an exception that can be caught & handled by a caller

Back to OO

Classes, Objects

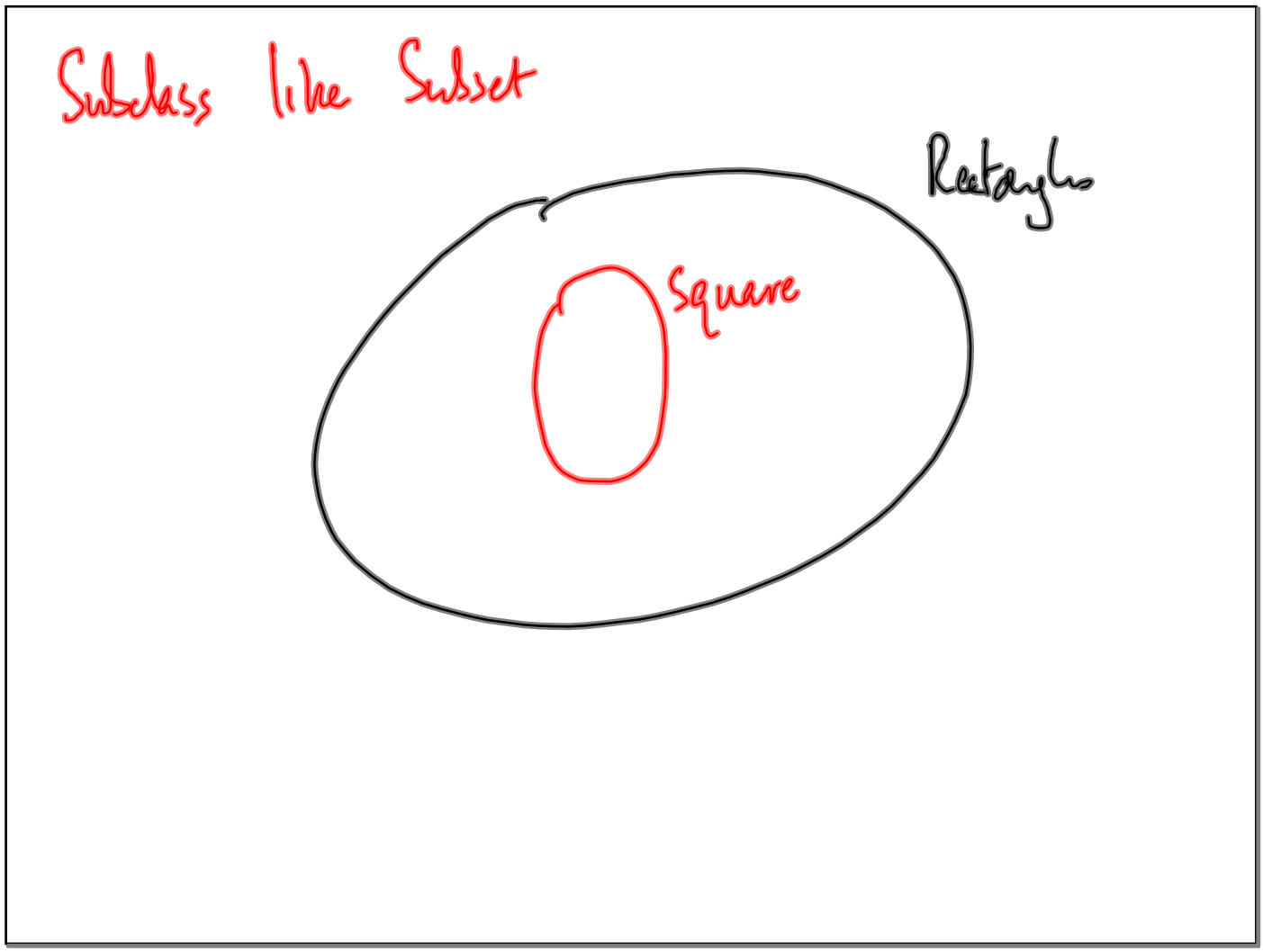
Hierarchy of classes

Define a special case of a known class

```
class Rectangle:
    |
class Square:
```

Subclass

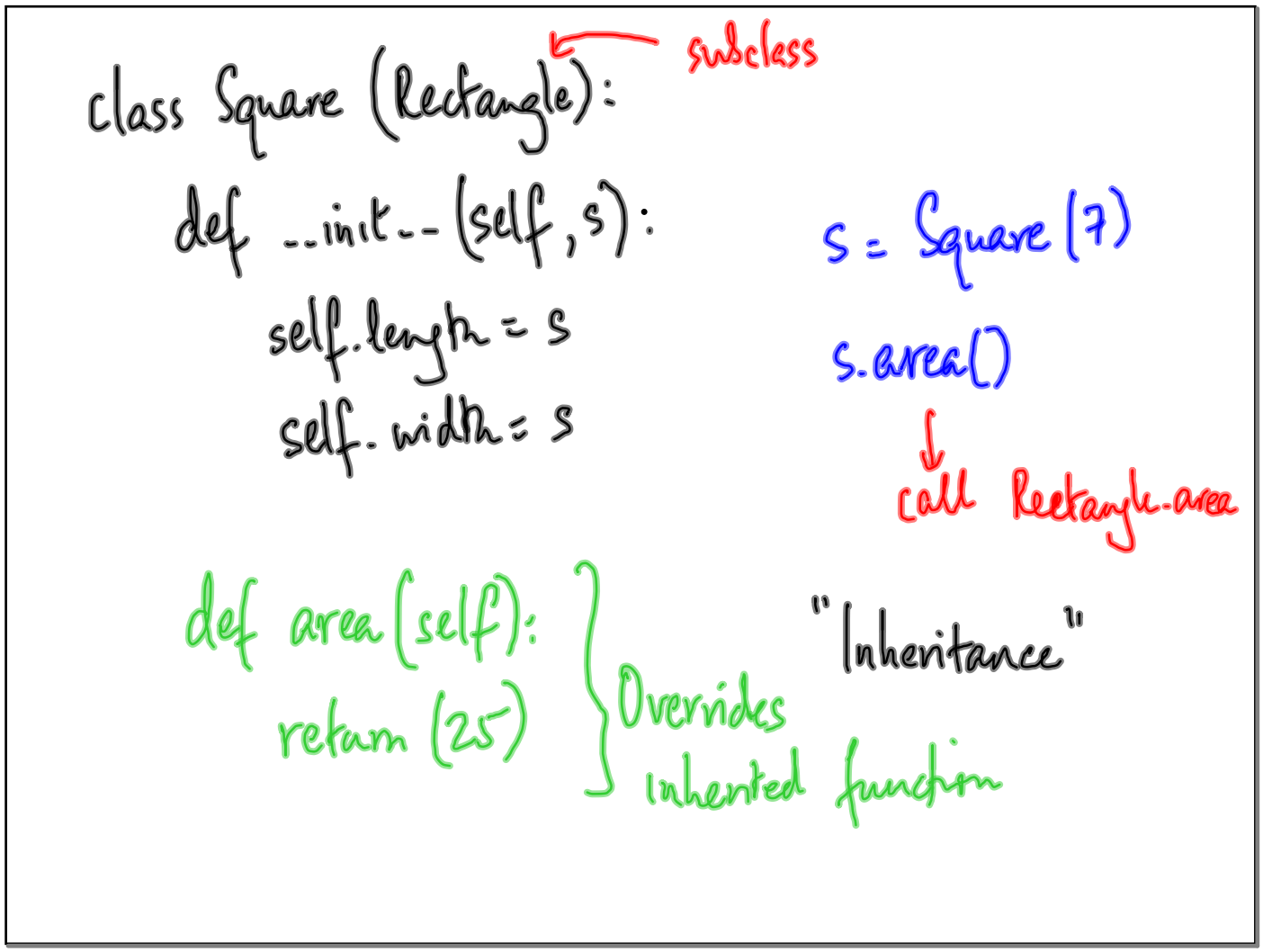
Every Square is a Rectangle!


Subclass like Subset
Rectangle
Square

```
class Rectangle:
    def __init__(self, l, h):
        self.length = l
        self.height = h

    def area(self):
        return(self.length * self.height)
```

```
r = Rectangle(4,7)
r.area()
28
```


class Square (Rectangle):
subclass
def __init__(self, s):
self.length = s
self.width = s
s = Square(7)
s.area()
call Rectangle.area
def area(self):
return (25)
Overrides inherited function
"Inheritance"

In particular, all exceptions are objects that inherit from Exception

```
class MyException(Exception):
    pass                          <-- do nothing!
    :
raise MyException(message)
```

```
try:
    :
except MyException:
    :
```

Classes & inheritance.

Double Ended Queue deque

rotate() fn

a Deque is same as Queue

```
def Deque(Queue):
    def __init__(self, initlist=[]):
        self.queue = initlist

    def removefront(self):
        y = self.removeq()
        return(y)
    def addrear(self, v):
        self.addq(v)

    def removerear(self):
        if ...
            y = self.queue[-1]
            del(self.queue[-1])
            return(y)
        :
        :
```

Implicitly

Deque.rotate() ≡ Queue.rotate()

by inheritance