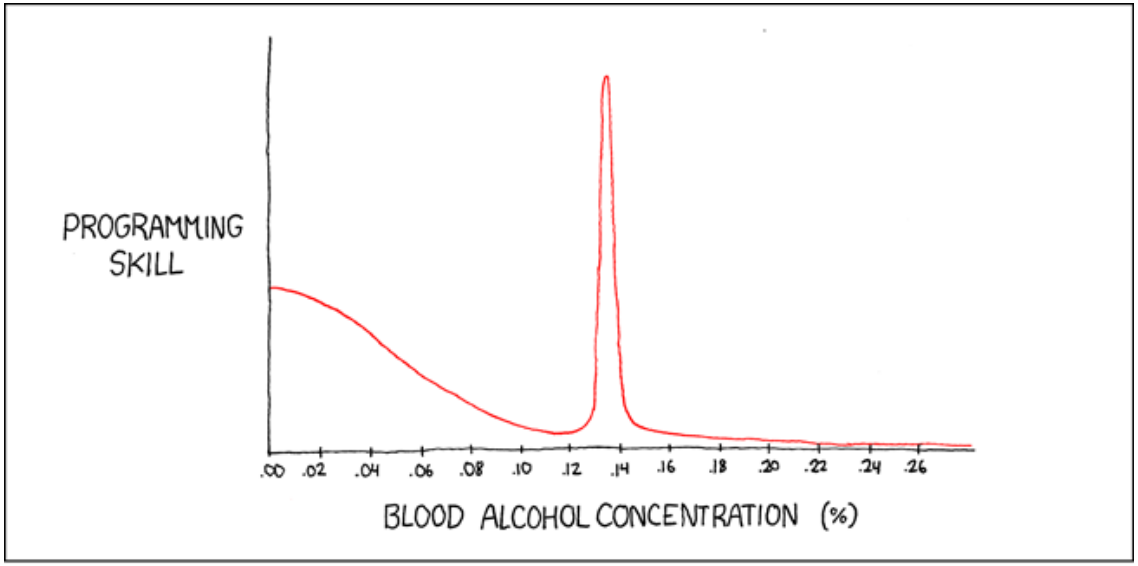
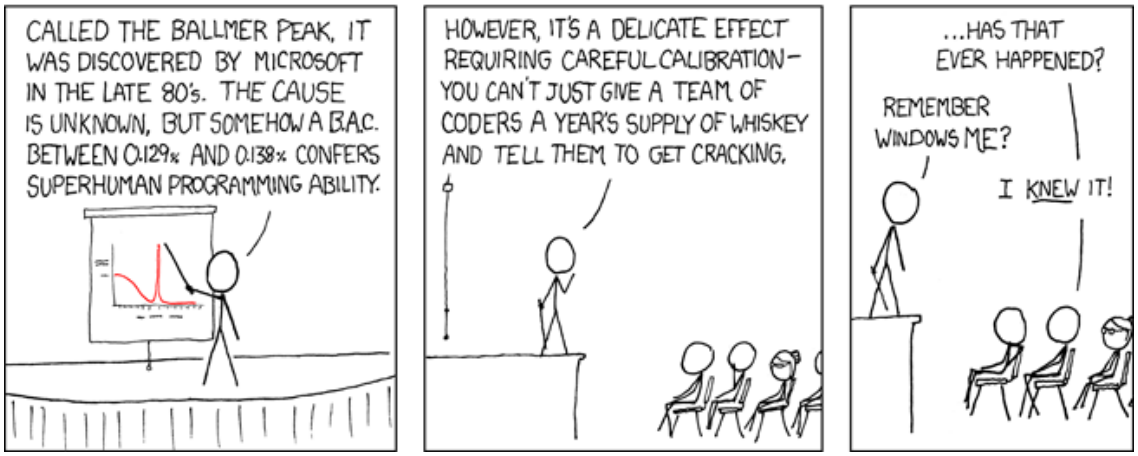
WTF is an **Object**?

A (brief) introduction to objects in python.





A quick **for** loop refresher.

```
myList = [1, 4, 2, 3, 5, 3, 2, 4, 3]
myCount = 0
for item in myList:
    if item == 3:
        myCount = myCount + 1

print(myCount)
```

A quick **for** loop refresher.

```
myList = [1, 4, 2, 3]
myNewList = []
for item in myList:
        myNewList.append(item**2)

print(NewList)
```

Python functions can have default options. (TempConverter in the homework)

```
getNumber(myList, n=0):
    return(myList[n])
```

```
myList = [3, 6, 23, 6, 1]

print(getNumber(myList, 2))

print(getNumber(myList, 1))

print(getNumber(myList))
```

You can **import** libraries or other files you have made.

Entire library	import math math.sqrt(10)	
Specific function		

You can **import** libraries or other files you have made.

Entire library	import math math.sqrt(10)	
Specific function	from math import sqrt sqrt(10)	

You can **import** libraries or other files you have made.

Notice no file extension (.py)

Entire library	import math math.sqrt(10)	import workshop1_homework workshop1_homework.factorial(10)
Specific function	from math import sqrt sqrt(10)	

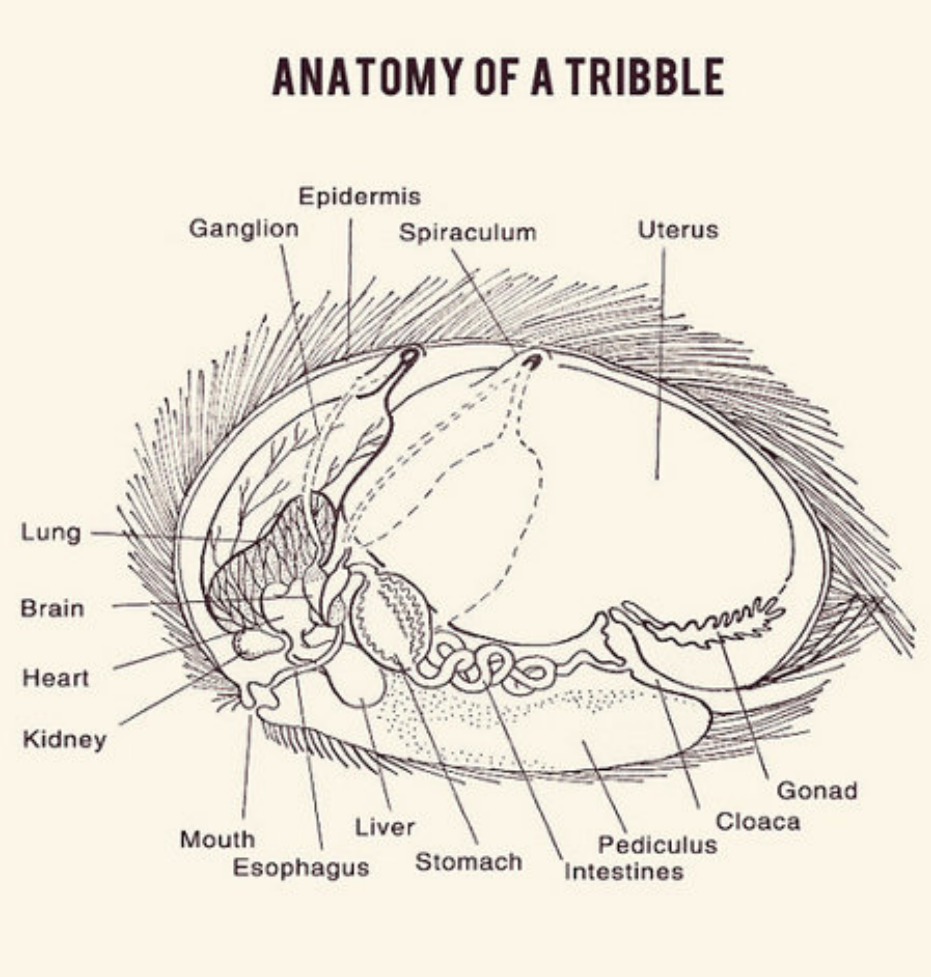
You can **import** libraries or other files you have made.

Entire library	import math math.sqrt(10)	import workshop1_homework workshop1_homework.factorial(10)
Specific function	from math import sqrt sqrt(10)	from workshop1_homework import factorial factorial(10)

Thinking about objects, what are some attributes and functions of Tribbles?

Thinking about objects, what are some attributes and functions of Tribbles?

Thinking about objects, what are some attributes and functions of Tribbles?



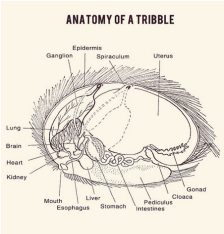
Thinking about objects, what are some attributes and functions of Tribbles?

color

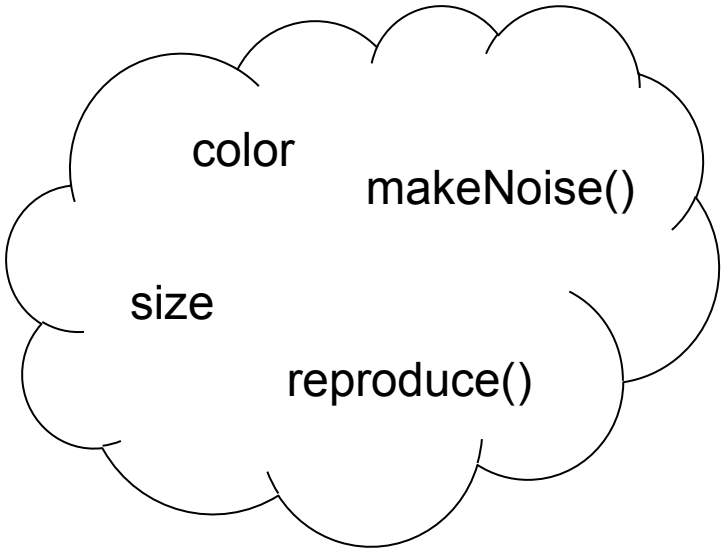
makeNoise()

size

reproduce()



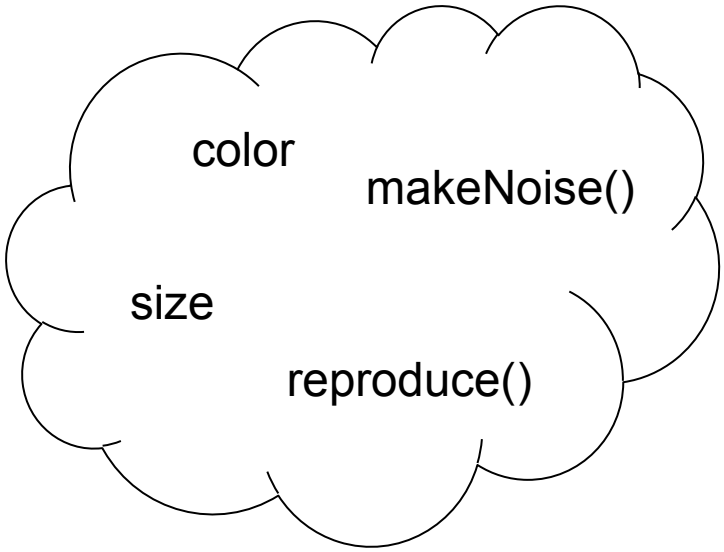
How would we tell our Tribble to do things?



myTribble = Tribble()

fred = Tribble()

How would we tell our Tribble to do things?



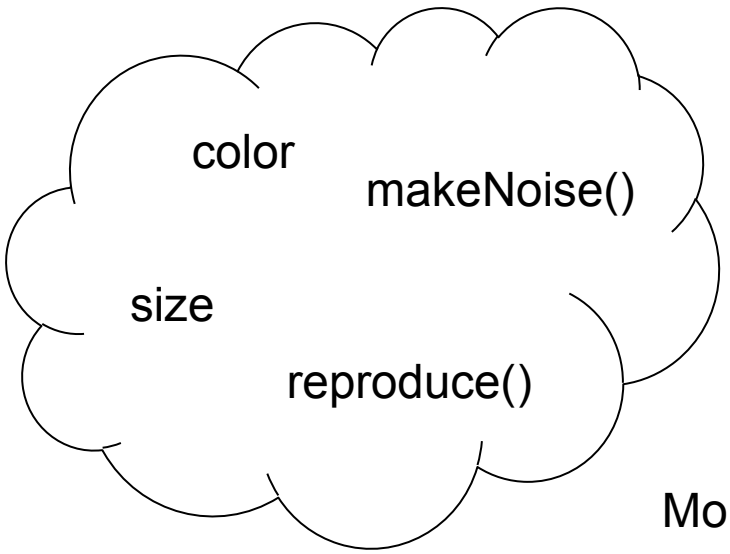
Tribble.color

Tribble.size

Tribble.makeNoise()

Tribble.reproduce()

How would we tell our Tribble to do things?



Tribble.color

Tribble.size

Tribble.makeNoise()

Tribble.reproduce()

More specifically, if we wanted Fred's attributes:

fred.color

fred.size

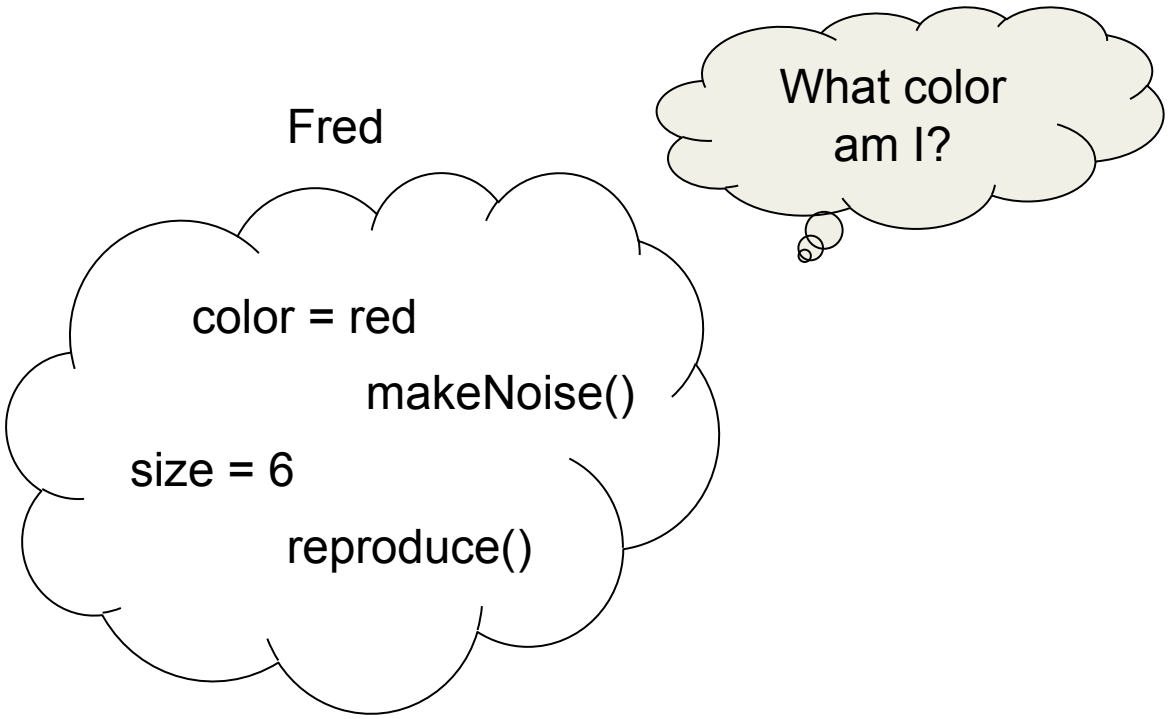
fred.makeNoise()

fred.reproduce()

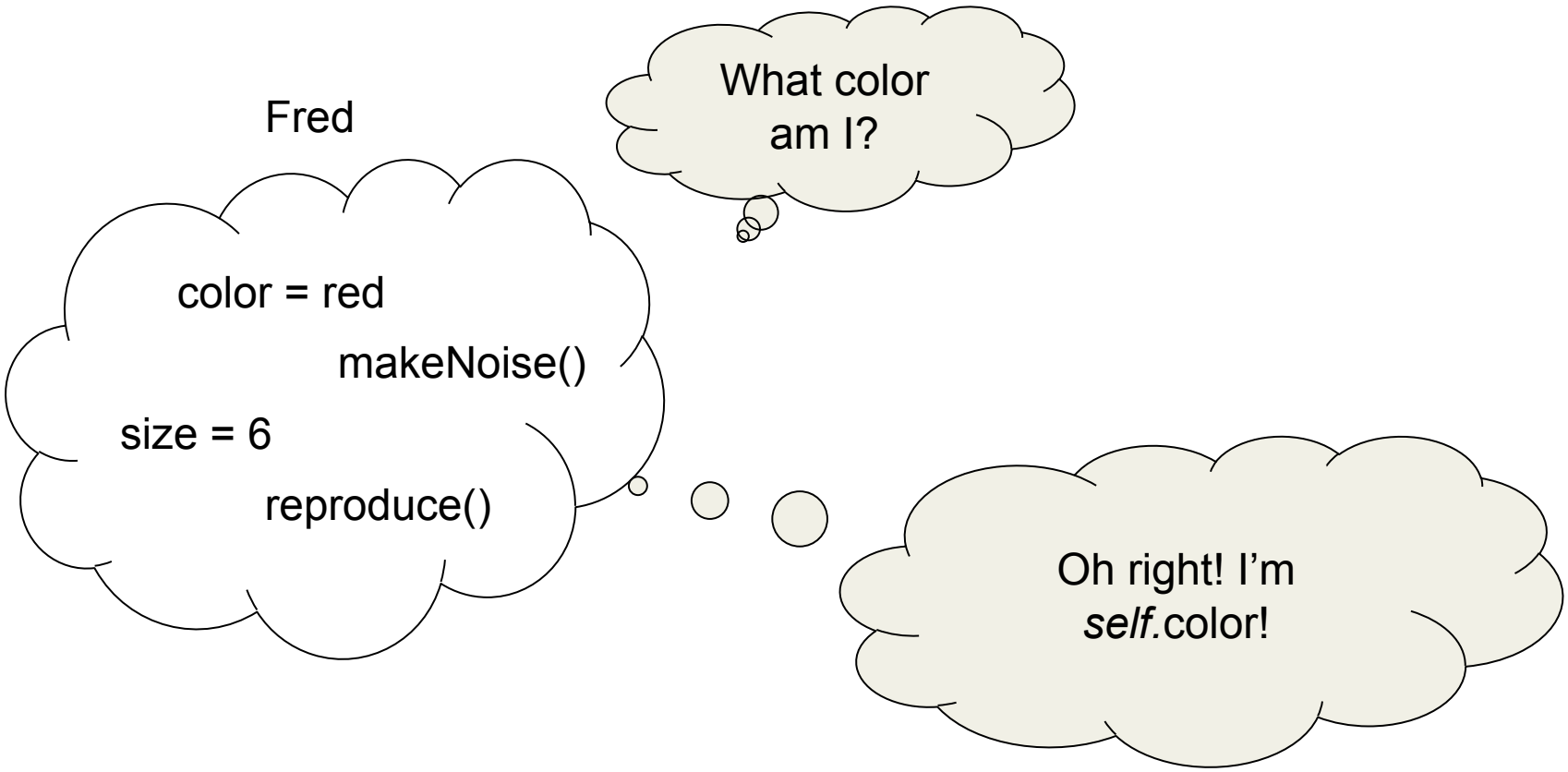
myTribble = Tribble()

fred = Tribble()

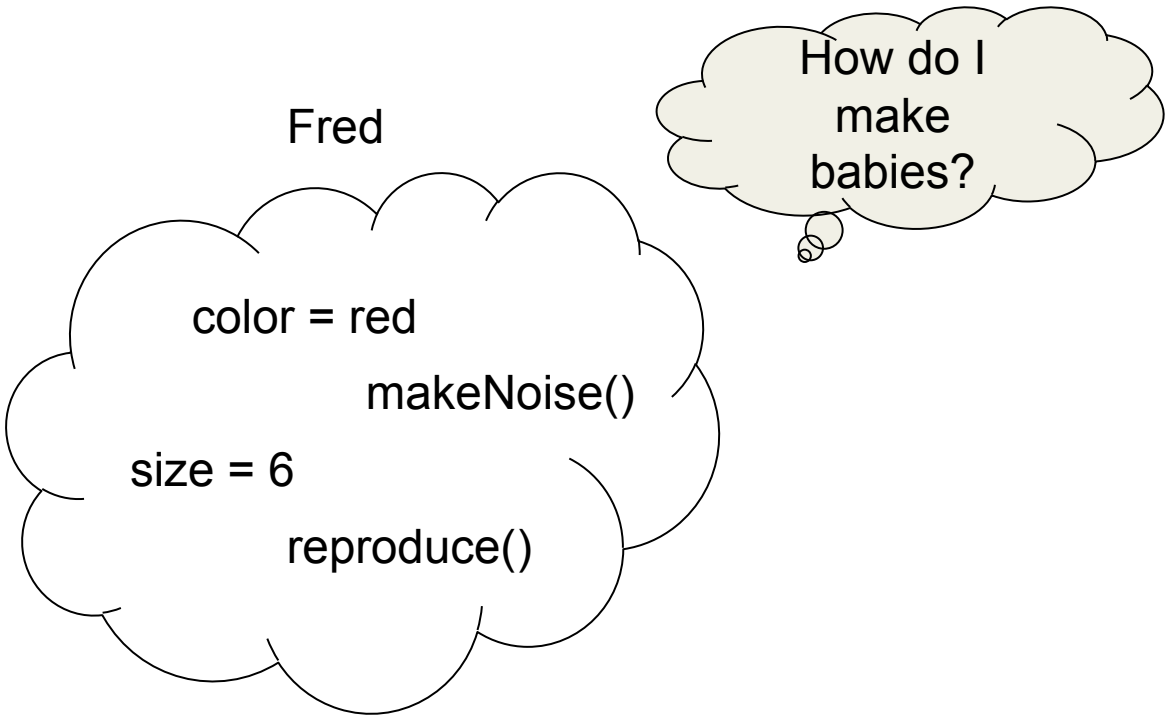
But how does a particular Tribble refer to his own attributes?



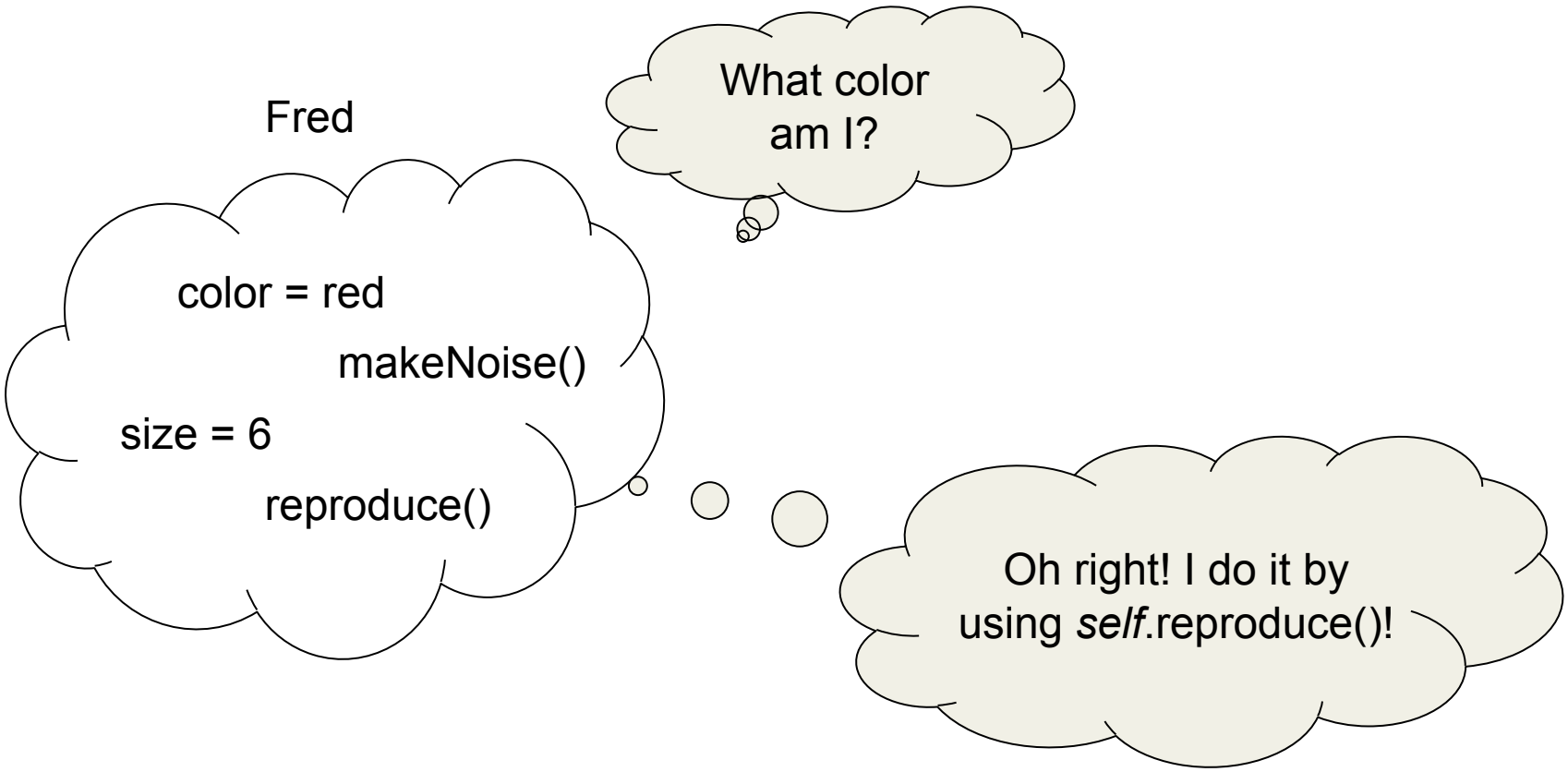
But how does a particular Tribble refer to his own attributes?



But how does a particular Tribble refer to his own attributes?



But how does a particular Tribble refer to his own attributes?



Referring to attributes within a Tribble

*self.*color
self.size

*self.*makeNoise()

*self.*reproduce()

Referring to attributes from outside a Tribble:

fred = Tribble()

fred.color
fred.size

fred.makeNoise()

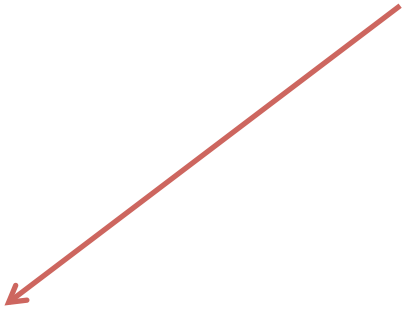
fred.reproduce()

Functions defined within classes:

```
class Tribble:

    …

    def reproduce(self):
        baby = Tribble()
        baby.color = self.color
        baby.size = 1
        return baby
```

Functions defined within classes:

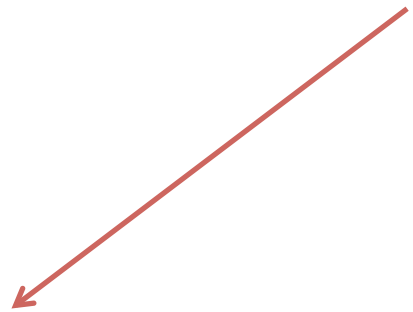
All functions defined within a class take *self* as the first argument.



```
class Tribble:

    …

    def reproduce(self):
        baby = Tribble()
        baby.color = self.color
        baby.size = 1
        return baby
```

Functions defined within classes:

```
class Tribble:

    …

    def reproduce(self):
        baby = Tribble()
        baby.color = self.color
        baby.size = 1
        return baby
```



All functions defined within a class take *self* as the first argument.

Otherwise, functions in classes act exactly like all other functions.

Initializing classes: using the __init__ function

```
class Tribble:

    def __init__(self):
        self.color = “red”
        self.size = 10
```

Initializing classes: using the __init__ function

All classes must define the __init__ function

```
class Tribble:

    def __init__(self):
        self.color = “red”
        self.size = 10
```

Initializing classes: using the __init__ function