A_PDF_THAT_GOT_MUCH_LONGER_THAN_I_EXPECTED_IT_TO,_MUCH_LIKE_THIS_TITLE_HAS.pdf

EVERYTHING (with a name) is an object.

So you have a list and some numbs, strings (and so on):

This is a class, but is still an object.

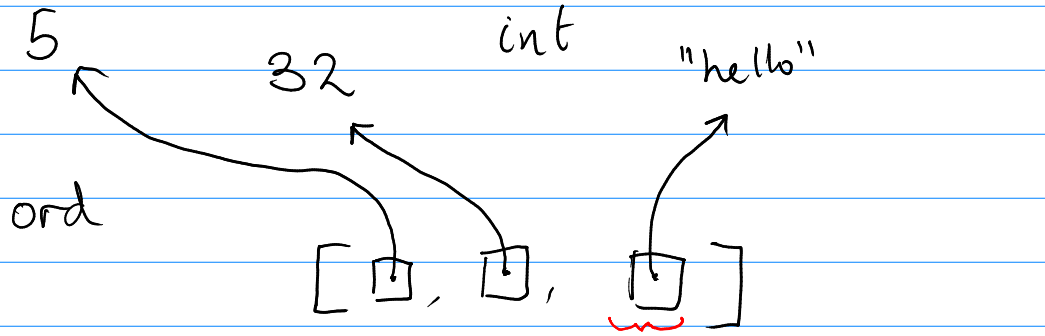
5 32 int "hello"

This is a function. Still an object.

ord

[5, 32, "hello"]

However, the list actually only references the other items:



Think of the arrows as numbers that say where to find what you want.

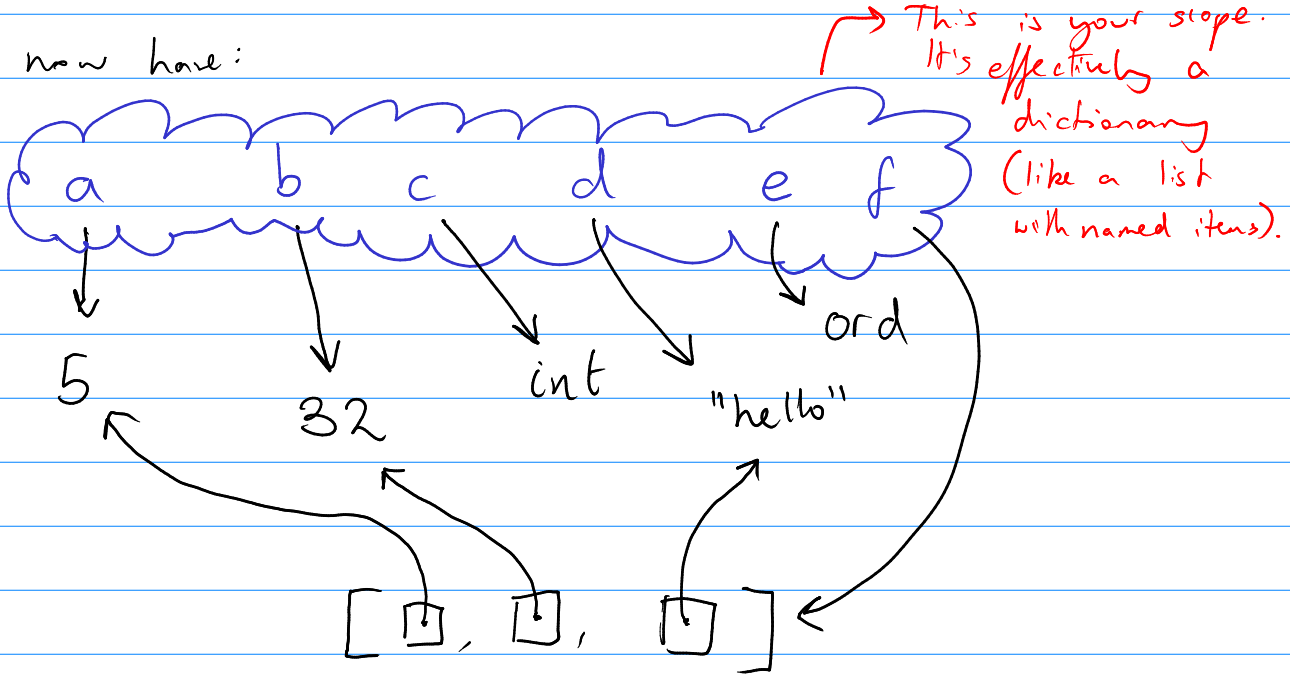
So let's say you have

```
a = 5
b = 32
c = int
d = "hello"
e = ord
f = [a, b, d]
```

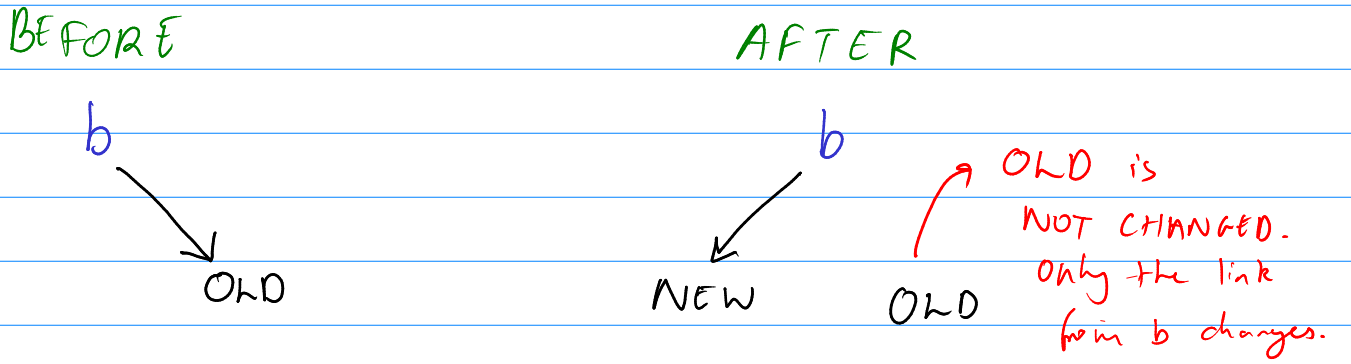
This is NOT the same as f = [5, 32, "hello"], which would create new objects.*

* well, actually it might not, but that's because they are immutable...

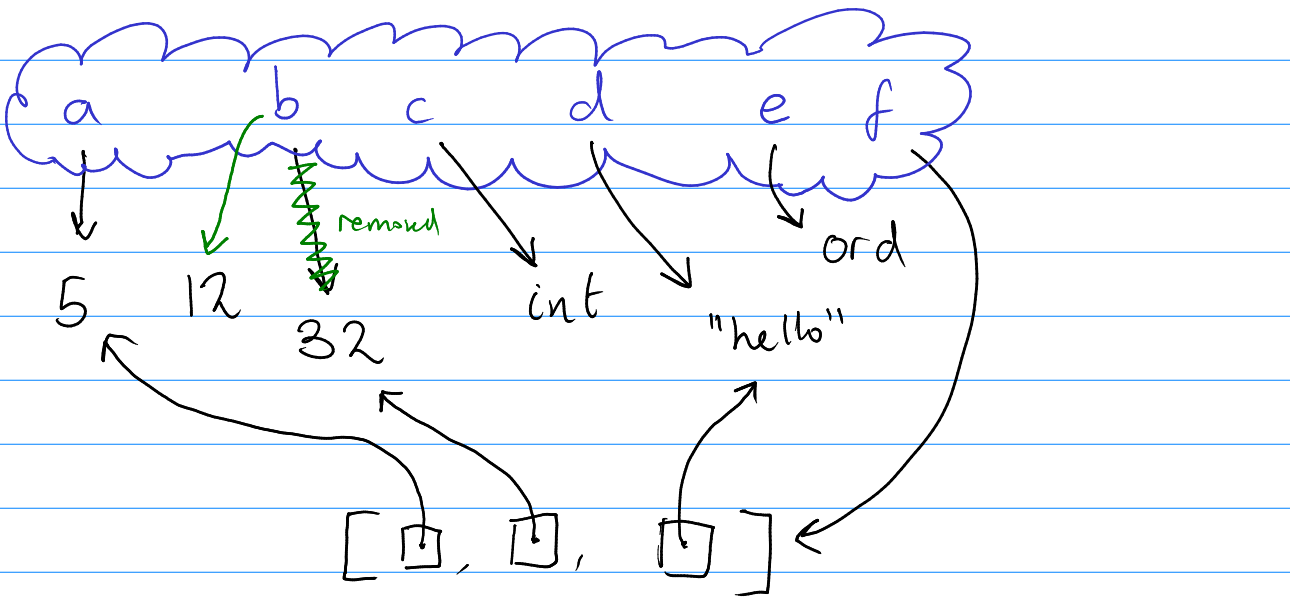
You now have:



So what does `b = 12` do?

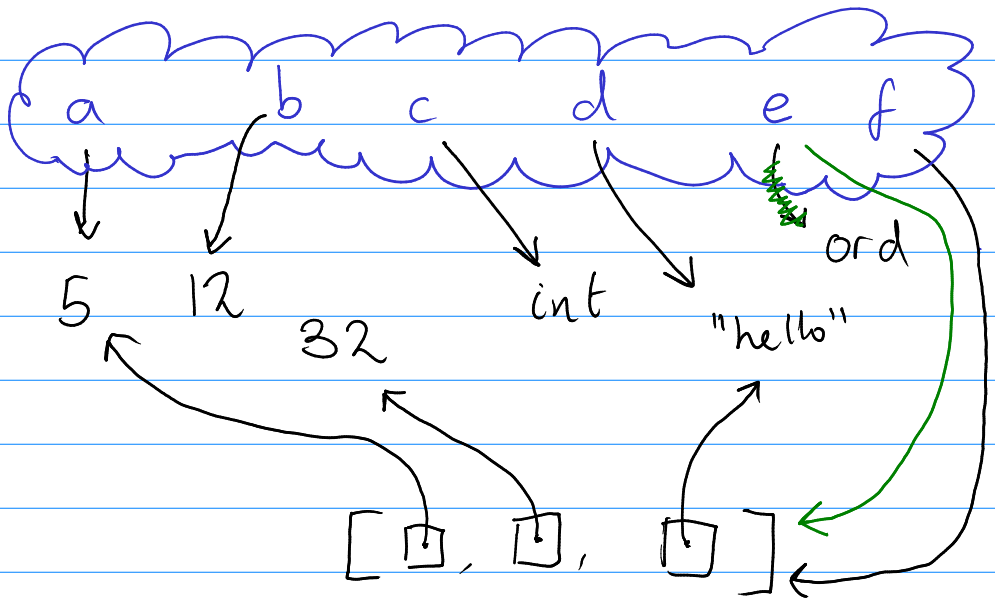


Or,

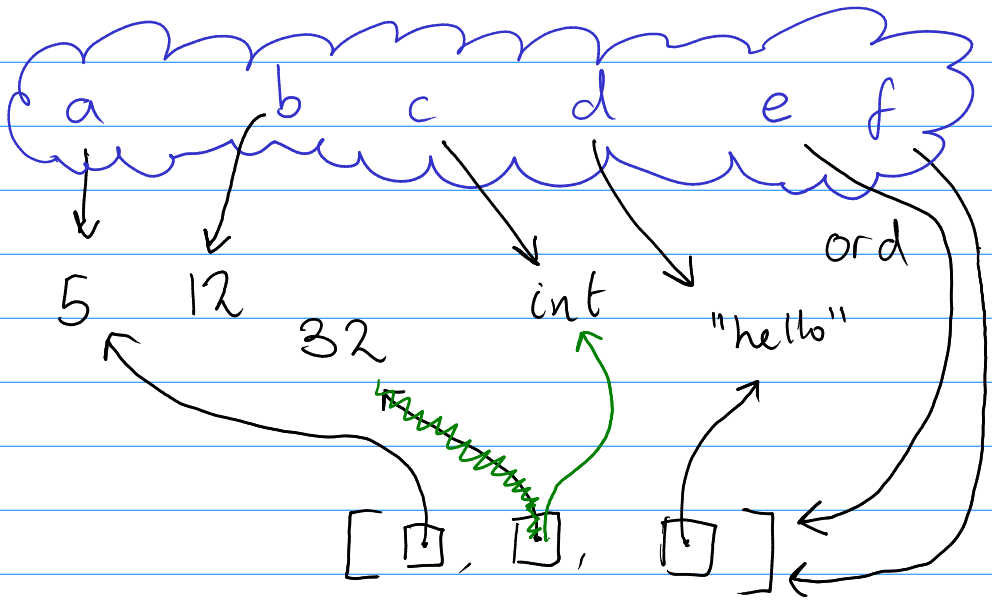


See how the list is unchanged?

What about e = f ?



And now f[1] = c ?



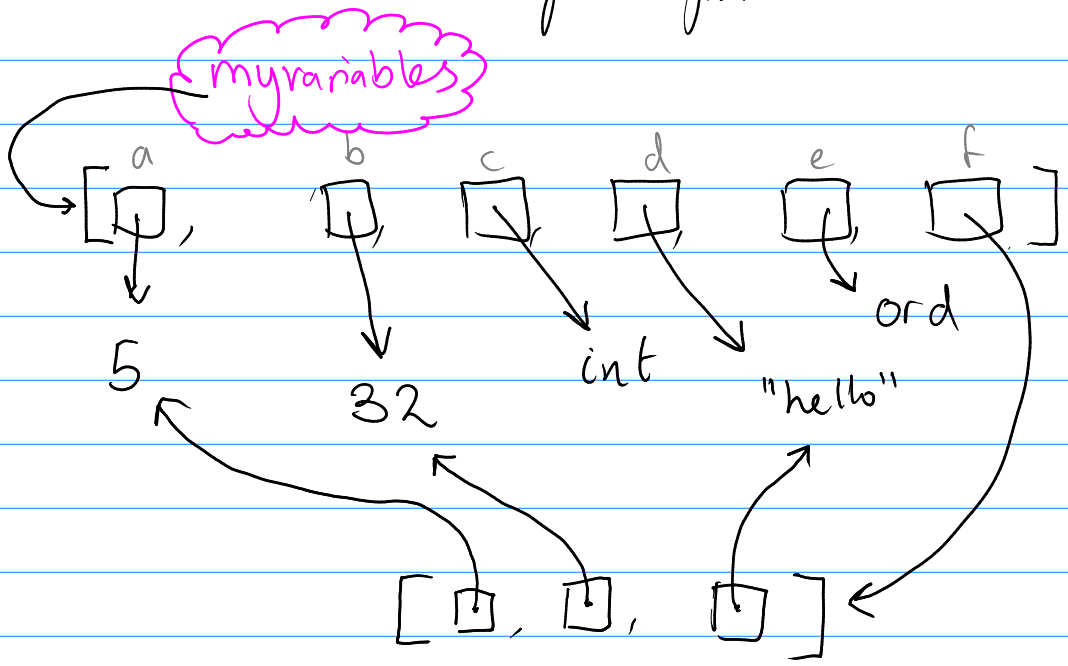
It looks like e changed as well!

That's because f and e pointed at the same thing, and f[1] = X changes the thing f points to, not the pointer† itself!

†Well, references. 'Pointer' means something specific.

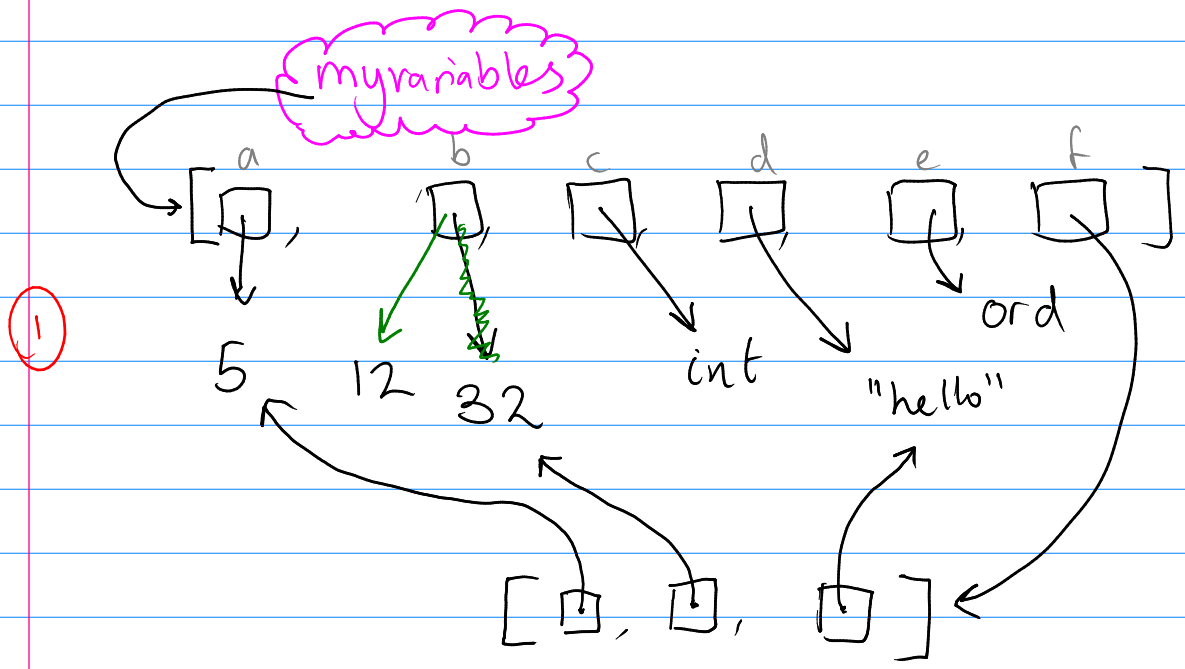
Let's abstract a, b, c, d, e and f.

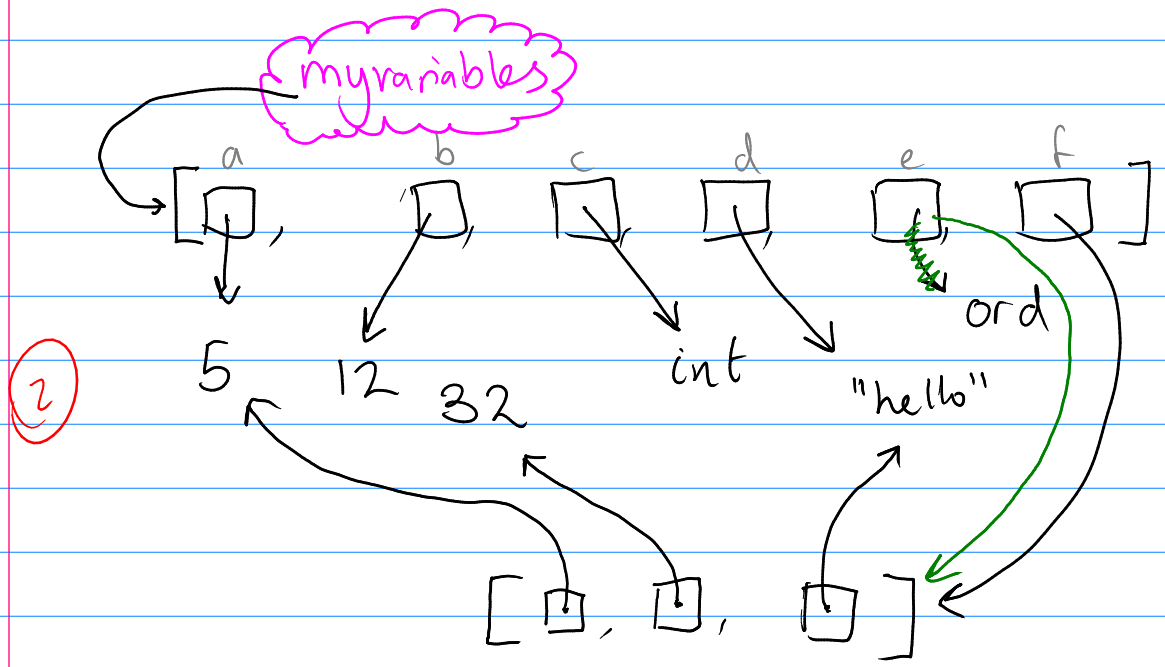
This will now be a list of length 7:

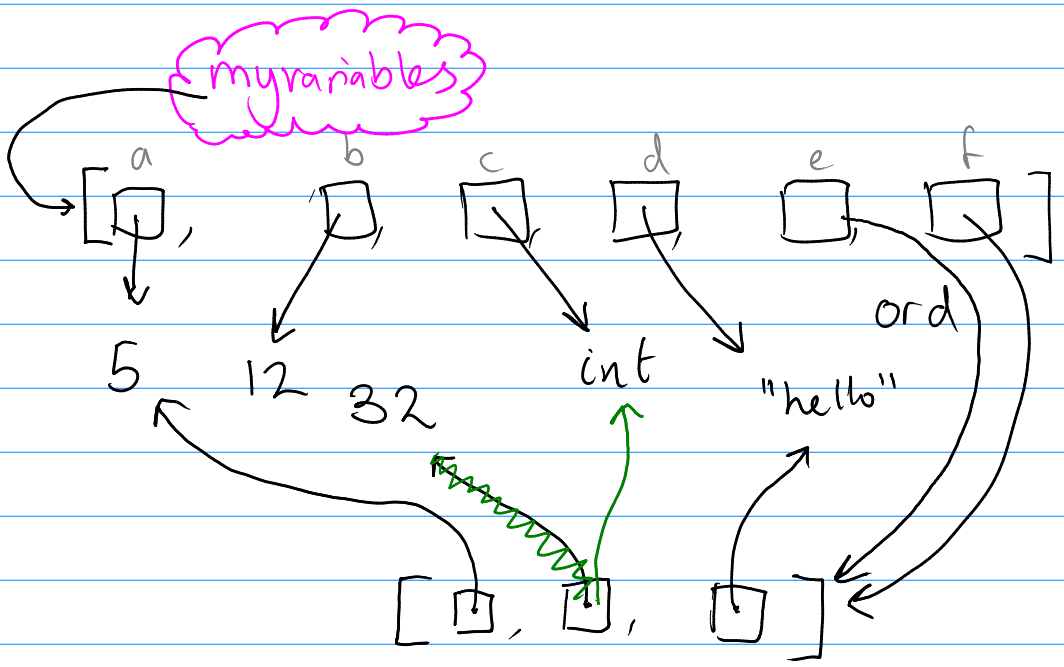


	OLD	NEW
①	b = 12	myvariables[1] = 12
②	e = f	myvariables[4] = myvariables[5]
③	f[1] = c	myvariables[5][1] = myvariables[2]

Let's see it in action


myvariables
a
b
c
d
e
f
5
12
32
int
"hello"
ord
1


myvariables
a
b
c
d
e
f
5
12
32
int
"hello"
ord
2



Notice anything? That's right: precisely the same thing happened.

What once seemed to be an inconsistency now makes sense.

There is one point:

a = b

DOES NOT MEAN "make the value of a into the value of b"

AND IT DOES NOT MEAN "make a point to b"

BUT IT DOES MEAN "make a point to what b points to".

(even with a=15, an int object is made, and then "15" points to it.)

In fact, Python has dictionaries which let you do:

`f["x"]` instead of `f[25]`

and you can even access some scopes with `globals()` and `locals()` and get back a dictionary! Try it!

There is one piece of voodoo: in-place operators.

`x += y` IS NOT `x = x + y`

... unless an object decides that it should be.

Thanks for reading