Today: C

Why C?

- mostly different paradigm from Scheme (imperative language)
- no garbage collection
- not object oriented
- different memory model than other Carleton CS languages

Two main different memory models that languages adopt

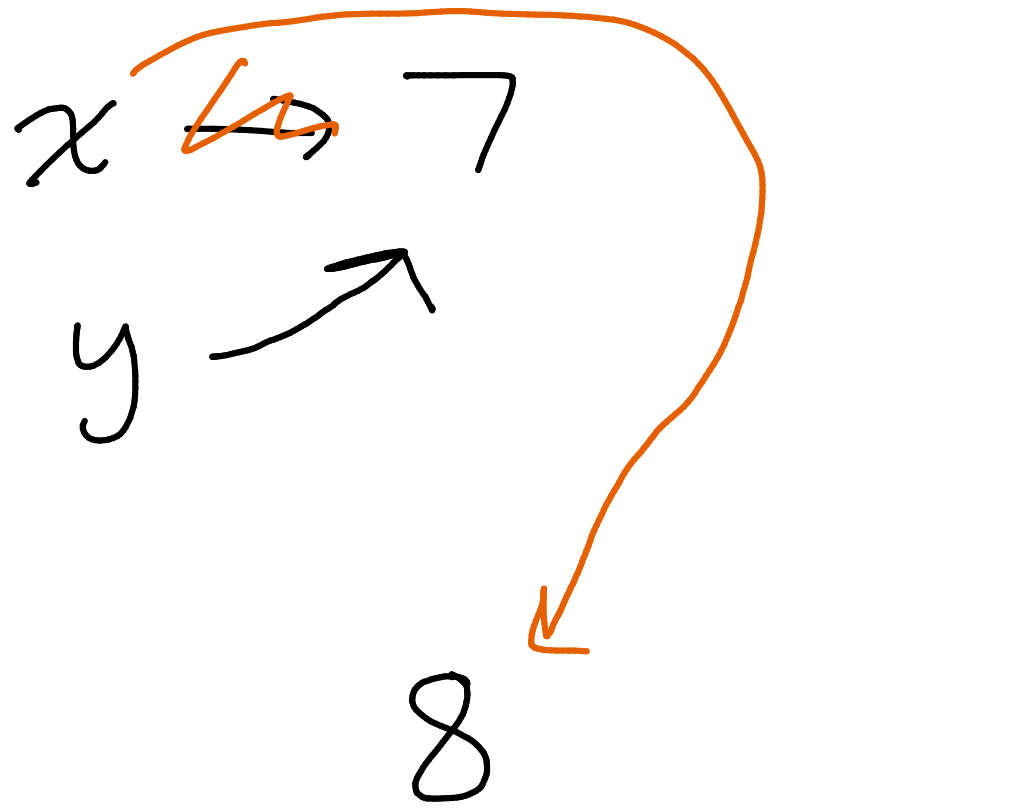
Reference model (Python, Scheme)

A varicble is a reference to an object/data

An assignment statement copies the reference, not the data

```
x = 7
y = x

x = x + 1
```



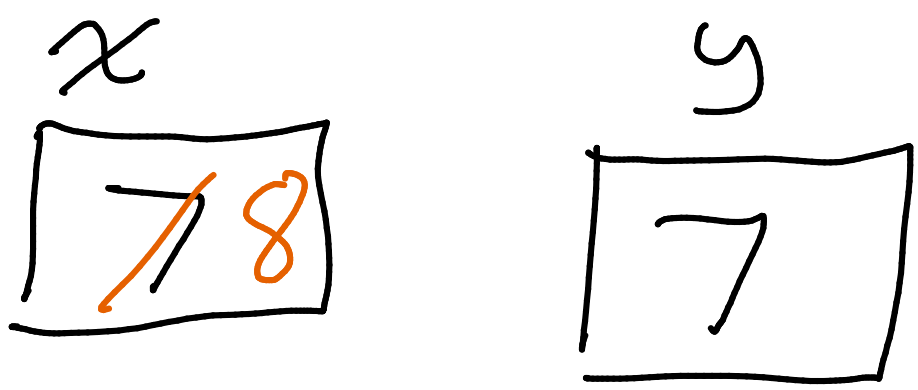
Value model (C)

A variable is a named container (box) for an object / data

An assignment statement copies the object/data from one container to another

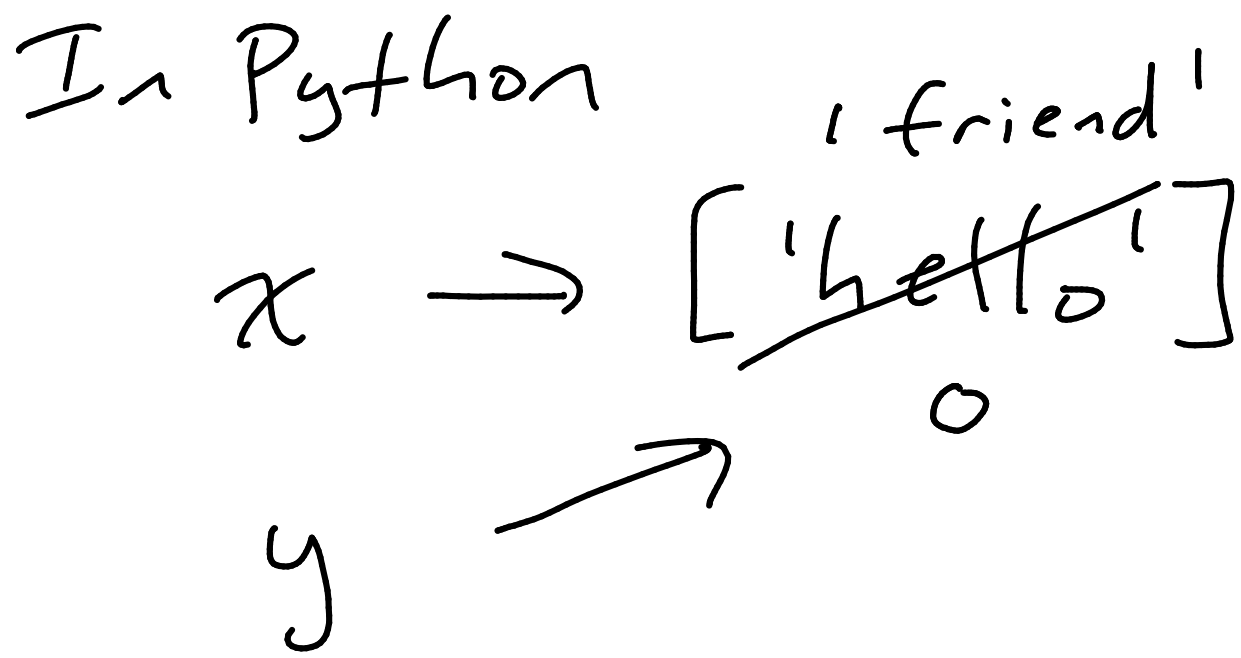
```
x = 7
y = x

x = x + 1
```



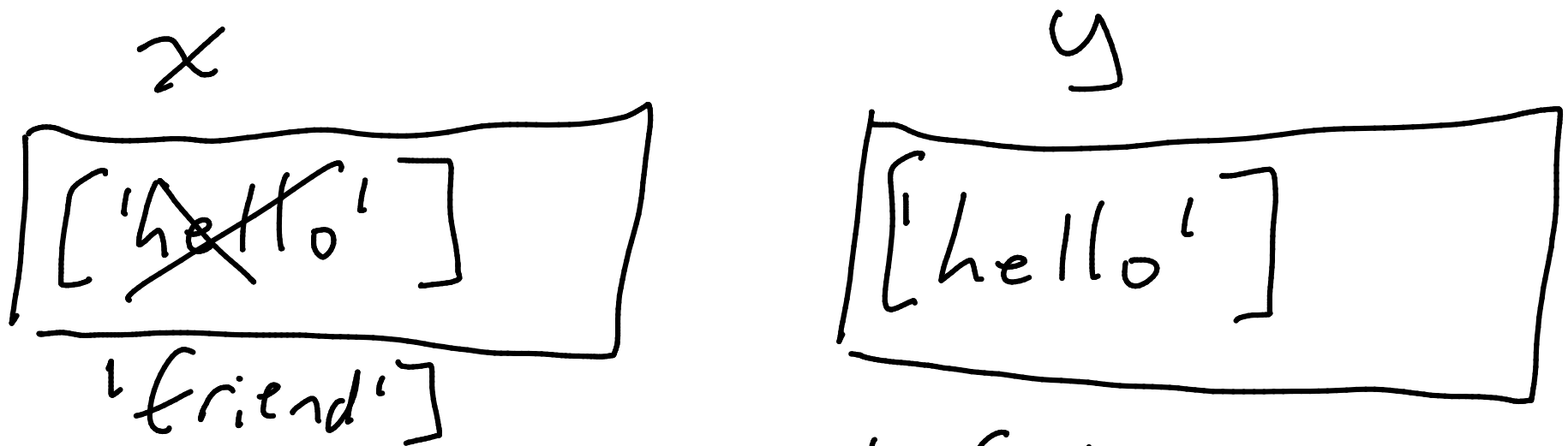
```
x = ['hello']
y = x
x[0] = 'friend'
print(y)
```

In Python



print(y) prints ['friend']

If this were a value model language
(If this were C)



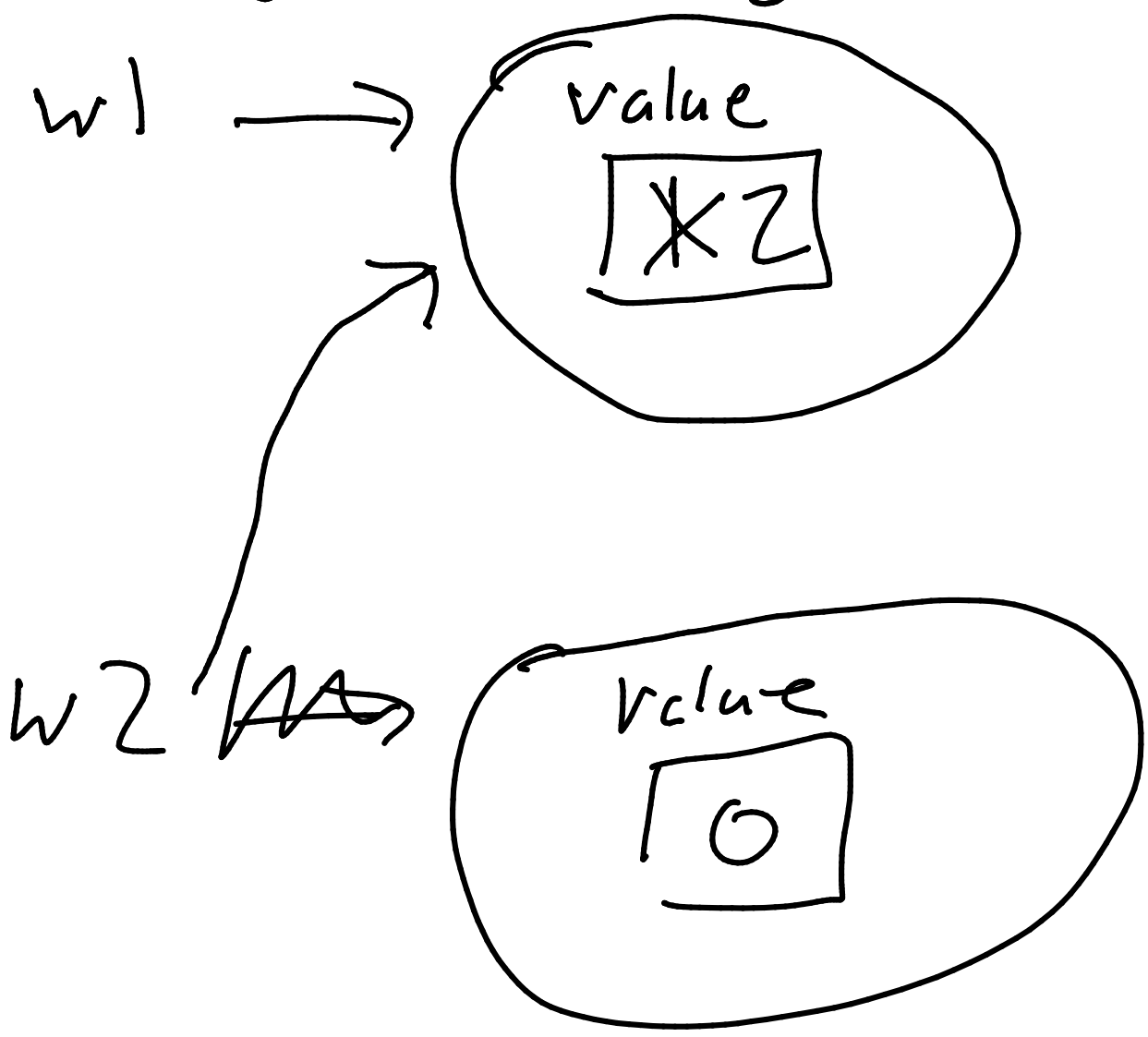
```java
// ContainerExample.java
class Container {
    public int value;
}


public class ContainerExample {

    public static void main (String[] args) {

        Container w1 = new Container();
        Container w2 = new Container();
        w1.value = 1;
        w2 = w1;                    1                1
        System.out.println(w1.value + " " + w2.value);
        w2.value = 2;
        System.out.println(w1.value + " " + w2.value);
    }                               2                2
}
```

Java is a reference model language w/ regards to objects



Java is a value lang w/ regard to primitives

(Python is same)

```c
#include <stdio.h>

struct Container {
    int value;
};

int main() {
    struct Container w1;
    struct Container w2;
    w1.value = 1;
    w2 = w1;
    printf("%i %i\n", w1.value, w2.value);
    w2.value = 2;
    printf("%i %i\n", w1.value, w2.value);
}
```

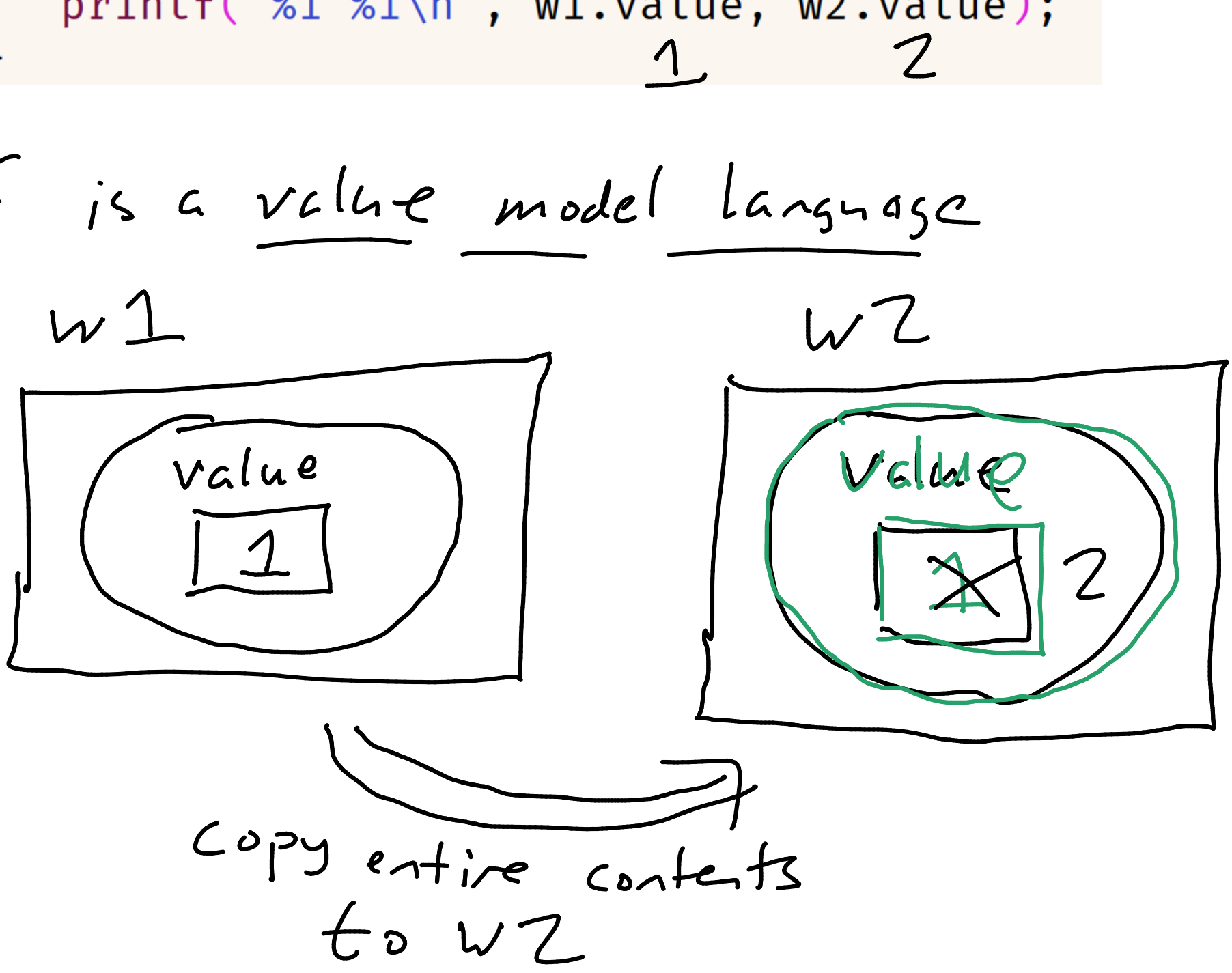
1 1

1 2

2

1

C is a value model language



```c
#include <stdio.h>
#include <stdlib.h>

struct Container {
    int value;
};

int main() {
    struct Container *w1 = malloc(sizeof(struct Container));
    struct Container *w2 = malloc(sizeof(struct Container));
    (*w1).value = 1;
    w2 = w1;
    printf("%i %i\n",(*w1).value,(*w2).value);
    (*w2).value = 2;
    printf("%i %i\n",(*w1).value,(*w2).value);
}
```

"pointer"

w1 is of type struct Container *

COPY pointer

*w1 => follow the pointer to the memory (got on water slide)

2 2

struct Container *w1