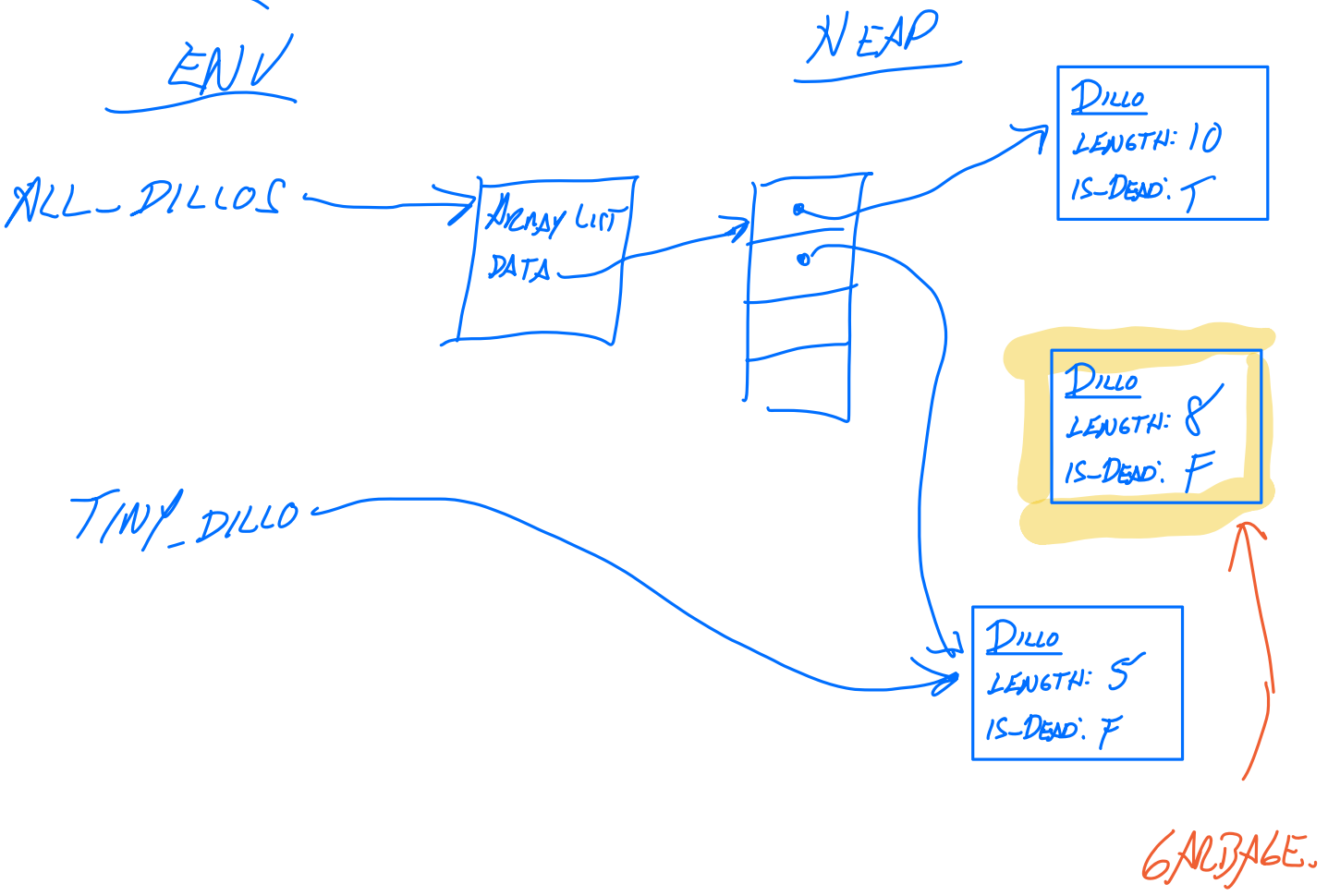
Lecture 35: Garbage Collection

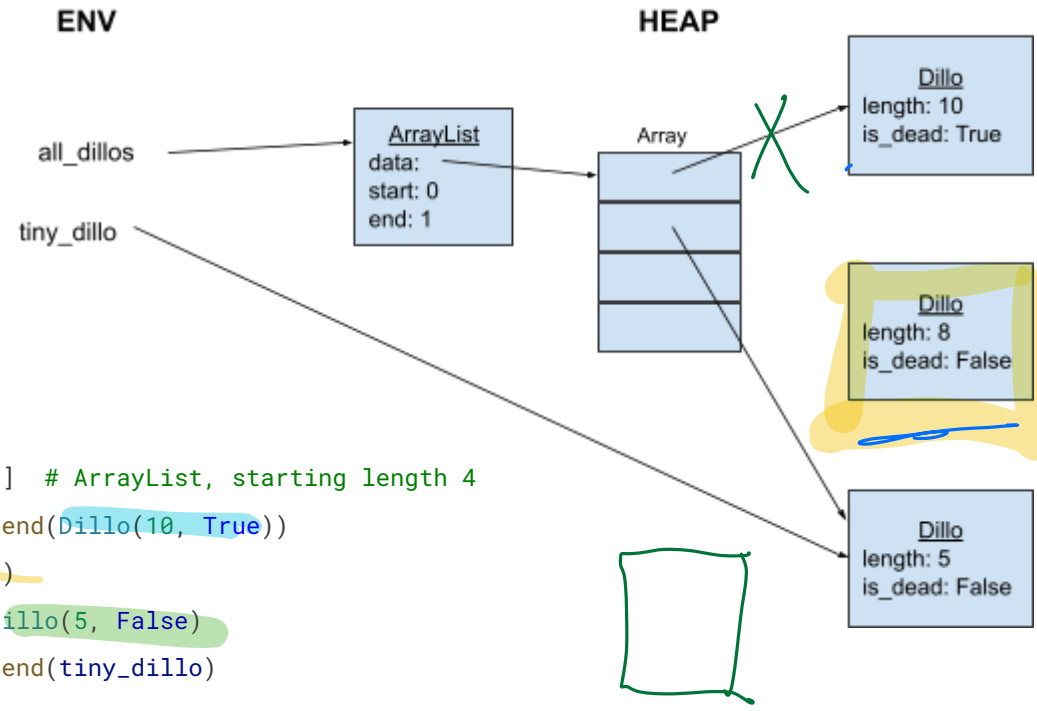
```
class Dillo:
    def __init__(self, length: int, is_dead: bool):
        self.length = length
        self.is_dead = is_dead

all_dillos = []  # ArrayList, starting length 4
all_dillos.append(Dillo(10, True))
Dillo(8, False)
tiny_dillo = Dillo(5, False)
all_dillos.append(tiny_dillo)
```



Garbage: data in heap that the program cannot use

Garbage collection: process of finding garbage objects and removing them




```
all_dillos = []  # ArrayList, starting length 4
all_dillos.append(Dillo(10, True))
Dillo(8, False)
tiny_dillo = Dillo(5, False)
all_dillos.append(tiny_dillo)
```

`all_dillos.pop(0) # Take off first element`

env		GARBAGE		ADDR	heap
all_dillos	------> @1001	NO	@	1001	ArrayList(data:@1002, start:0, end:1, size:2, cap:4)
tiny_dillo	------> @1008	NO	@	1002	@1006
		NO	@	1003	@1008
		NO	@	1004	
		NO	@	1005	
		NO	@	1006	Dillo(length: 10, is_dead: True)
		YES	@	1007	Dillo(length: 8, is_dead: False)
		NO	@	1008	Dillo(length: 5, is_dead: False)
			@	1009	free
			@	1010	free
			@	1011	free
			@	1012	free

Of algorithms we've seen, what is this similar to?

SPACES NOT BEING USED!

How it really works: GC uses DFS for each name in the environment

- **Mark each location you find as "not garbage"**
- **Anything not marked must be garbage => can remove it**

Bonus: what would happen if we resized the arraylist to size 8?

env			heap		
all_dillos	------> @1001		@	1001	ArrayList(data:@1002, start:0, end:1, size:2, cap:4)
tiny_dillo	------> @1008		@	1002	@1006
			@	1003	@1008
			@	1004	
			@	1005	
			@	1006	Dillo(length: 10, is_dead: True)
			@	1007	~~Dillo(length: 8, is_dead: False)~~
			@	1008	Dillo(length: 5, is_dead: False)
			@	1009	~~free~~
			@	1010	~~free~~
			@	1011	~~free~~
			@	1012	~~free~~
			@	1013	~~free~~
			@	1014	~~free~~
			@	1015	~~free~~
			@	1016	~~free~~
			@	1017	free
			@	1018	free
			@	1019	free
			@	1020	free

What Generates Garbage?

Example: Find the average of a list of positive numbers

What does this have to do with garbage?

```
# Example 1
nums = [67, 45, 0, 66, -21, 50]
pos_nums = [x for x in nums if x > 0]
avg_val = sum(pos_nums) / len(pos_nums)
print(avg_val)
```

Even if you never use pos_nums again, it's still reachable in the environment => it is not garbage

```
# Example 2
def avg_pos(numlist: list[int]) -> float:
    pos_nums = [x for x in numlist if x > 0]
    avg_val = sum(pos_nums) / len(pos_nums)
    return avg_val

avg_val = avg_pos(nums)
print(avg_val)
```

WILL GET CLEANED UP WHEN FUNCTION RETURNS!