Model 1 Array

Consider a list of numbers:

```
lucky = [29, 16, 23, 47, 37]
```

The easiest way to store these numbers in memory is to put them side by side. For example, if the list were to begin at memory address 40:

Address:		40	41	
Value:	···	29	16	
Index:		0	1	2

When programming, we can access individual numbers by their ***index***. For example, the value of `lucky[1]` is `16`, and `lucky[5]` is out of range.

Questions (10 min) **Start time:** _____

1. In the diagram above, write the index below each value.

2. We call the beginning of the list the ***head*** and the end of the list the ***tail***.

a) What is the address of the head?
40

b) What is the address of the tail?
44

c) What is the index of the head?
0

d) What is the index of the tail?
4

3. What is the relationship between the index and the corresponding memory address?

The index is relative to the first address. It counts the number of cells over from the head.

4. Based on your answer to the previous question, why do indexes start at 0 instead of 1?

Starting at zero makes it easier to compute the memory address of list items. We simply add `head + index`.

5. There are (currently) five numbers in the list. Why is `lucky[5]` out of range?

The indexes range from 0 to 4.

6. The statement `lucky.insert(2, 13)` changes the list to `[29, 16, 13, 23, 47, 37]`. What is the address of the head and tail after inserting 13?

The head is 40 (unchanged), and the tail is 45.

7. In terms of memory operations, what does it take to insert values in the middle of an array?

All subsequent values in the array need to be shifted over by one memory cell, resulting in multiple LOAD and STORE operations.