# Arrays

# Let's Talk Dorms

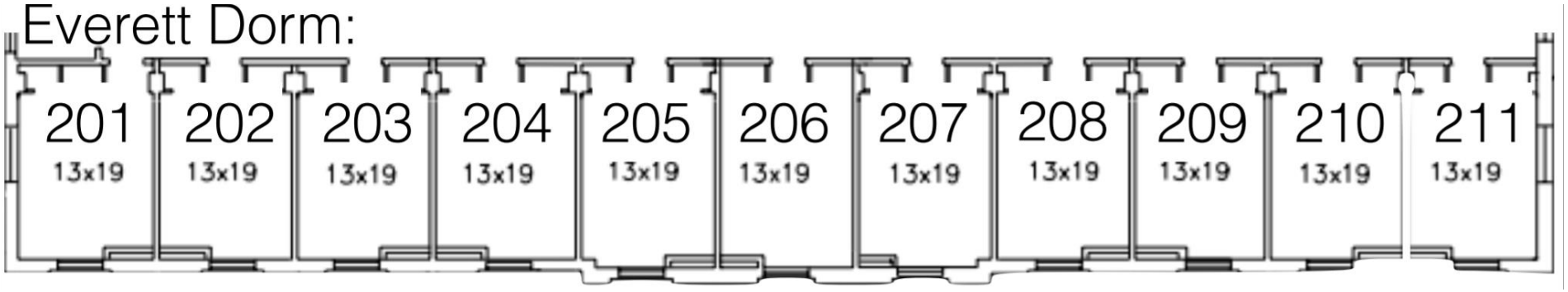
# Dorm Room Naming Proposal

- What if dorms were not numbered? What if they were *named*?

- Some dorm room *name* ideas:
  - Erudite Elephant
  - Silly Snake
  - Loud Liger
  - Ornery Onyx
  - Fuzzy Frog
  - Healthy Hippo
  - Petite Pig

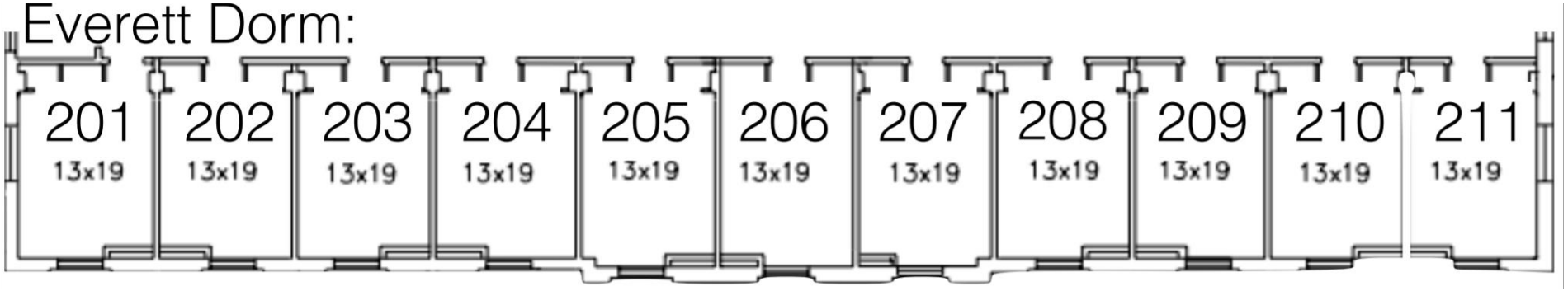
- *What benefits do dorm room numbers provide?*
  - If you can reason through this, you can reason through arrays.

# What are the benefits of a Dorm **Name + Number** address scheme?




- Naming is difficult. Numbering is very easy.
  - With a dorm room you only need to remember *one* name.
- Assigning numbers in order allows you to quickly locate a room.
- You can talk about entire *ranges* of rooms.
  - "Ok, I'll check rooms 100-110, you take rooms 111-120."

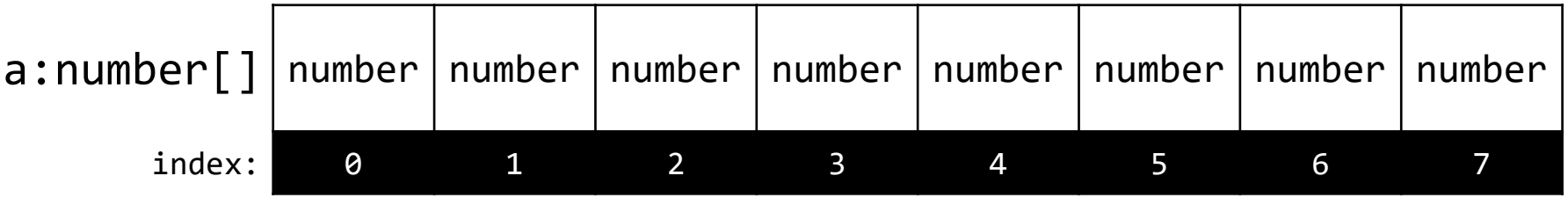
# Arrays are like Dorms for Data




| a:number[] | number | number | number | number | number | number | number | number |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

An **array** is a variable, with a **name**, that holds *many* values addressed by an **index** "room" number.

# **Arrays** provide uniform housing for **many values of the same type**.

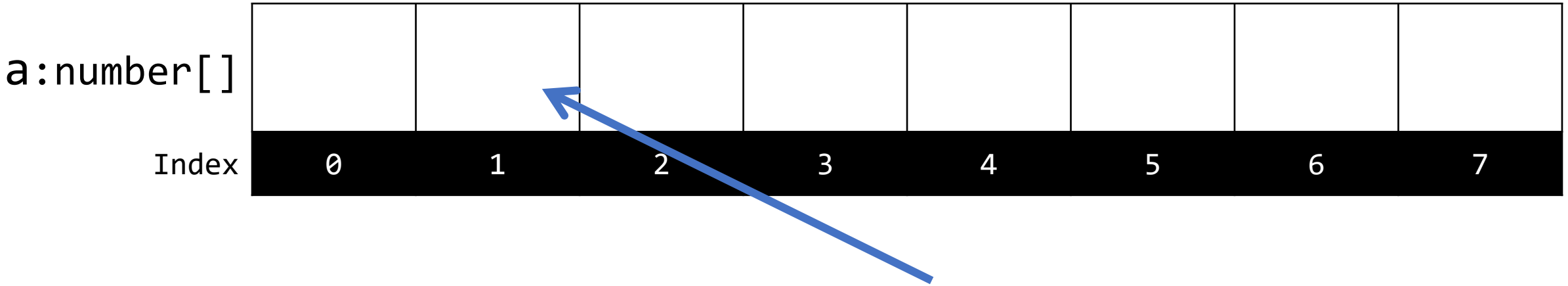



1. Each item in an array is called an **element**

2. An element is a single value **addressed by its index** ("Room #")

3. All elements in an array are of the **same *type***
   - An array of `numbers`, `strings`, `objects`, `etc`

# Elements are addressed by the array variable's **name** and **index**




1. Notation: **`arrayName[index]`**, i.e. **`a[1]`**

2. **Indexing starts at [0]** (not [1])
   - First index *always* 0
   - Last index *always* length of array – 1
   - This is a convention shared by most programming languages

# Declaring and Constructing **new** Arrays

1. You can **declare an array** of *any type* by placing an empty pair of square braces after the type:

   **`let <name>: <type>[];`** – array of <type>
   **`let ages: number[];`** – array of number values
   **`let words: string[];`** – array of strings

2. You **construct** an empty array by writing the empty square brackets

3. These two tasks are usually done at the same time:

```
let words: string[] = [];
```

# Array Operations Reference

| Operation | Form | Example |
|---|---|---|
| **Declaration** | `let <name>: <type>[];` | `let scores: number[];` |
| **Construction (Empty)** | `<name> = [];` | `scores = [];` |
| **Construction (Non-empty)** | `<name> = [<values>];` | `scores = [12, 0, 9];` |
| **# of Elements** | `<name>.length` | `scores.length` |
| **Access Element** | `<name>[<index>]` | `scores[0]` |
| **Assign Element** | `<name>[<index>] = <expression>;` | `scores[1] = 12;` |
| **Append Element** | `<name>[<name>.length] = <expr>;` | `scores[scores.length] = 13;` |