Tip: everything is more difficult in the last half of the term. If you are stuck on a problem for a significant amount of time, try walking away and go engage in something you enjoy, then come back to it. Your brain will still be working on it without you actively thinking and your new approach will likely be better than your last.
Arrays

• An order arrangement of related items
• Colloquially called lists
  • Caution: lists are an actual data structure that behave differently from arrays
• Examples
  • Array of numbers such as in a gradebooks
  • Strings -> array of characters
Creating 1D Arrays (Statistically)

```cpp
int grades[5];
```

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Access Each Element:

Array name represents:

Initial Values:
Initializing and Populating Static 1D Arrays

Declaration

    int grades[5] = {0,0,0,0,0};

Individual Elements

    grades[0] = 0;
    grades[1] = 0;
    grades[2] = 0;
    grades[3] = 0;
    grades[4] = 0;
Populating 1D Arrays with Loops

```c
int grades[5];
for(int i = 0; i < 5; i++)
    grades[i] = 0;
Or
int i = 0;
while (i < 5){
    grades[i] = 0;
    i++;
}
```
Read and Print

```cpp
int amount = 5;
int grades[amount];
for(int i = 0; i < amount; i++) {
    cout << "Please input a grade: ";
    cin >> grades[i];
}
for(int i=0; i<amount; i++)
    cout << "Grade " << i << ": " << grades[i] << endl;
```
Static vs. Dynamic Arrays

- Static: use when the size will not change
  ```csharp
  int grades[5];
  ```
- Dynamic: use when you do not know how big the array needs to be at compile
  ```csharp
  int grades = new int[5];
  ```
Demo