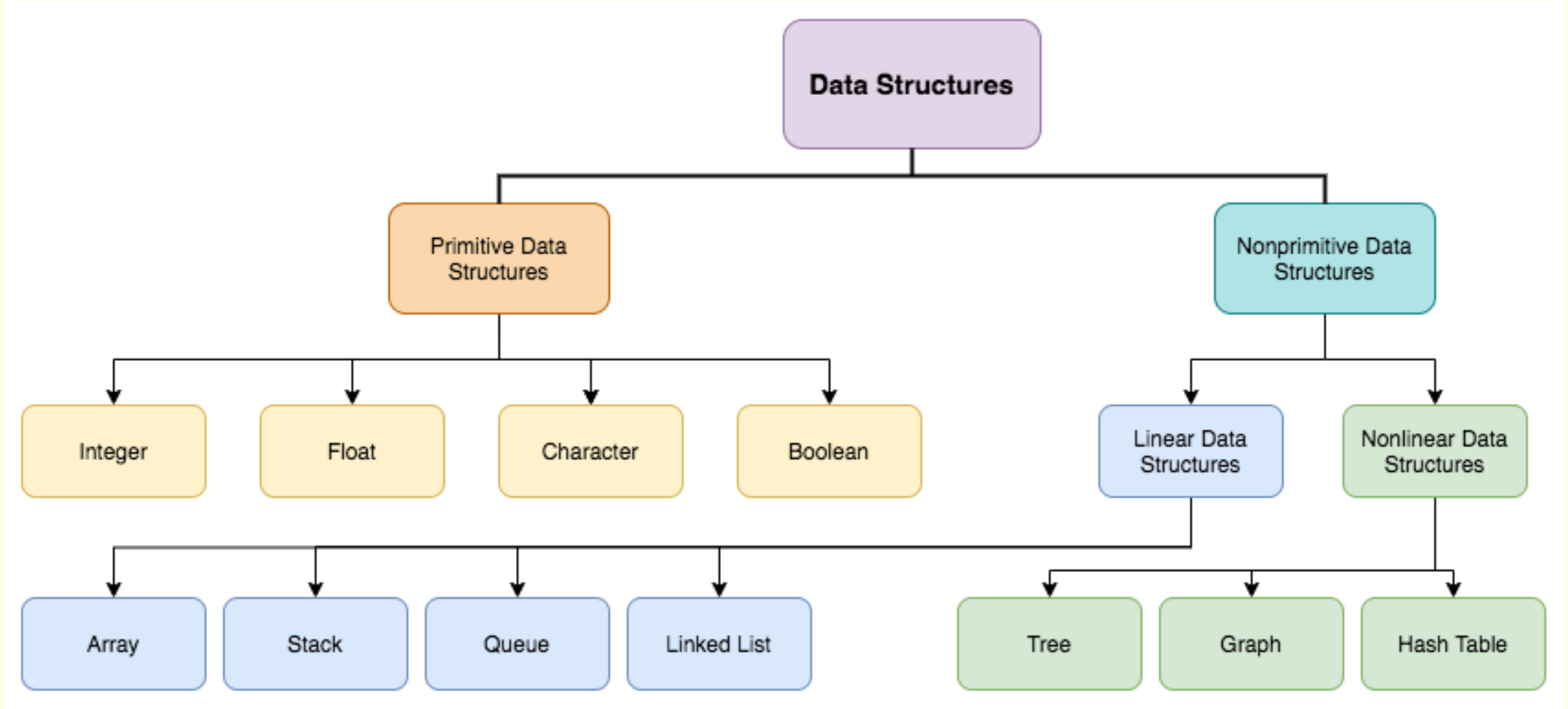

Data Structures
Primitive Data Structures
Nonprimitive Data Structures
Integer
Float
Character
Boolean
Linear Data Structures
Nonlinear Data Structures
Array
Stack
Queue
Linked List
Tree
Graph
Hash Table

Static VS dynamic structures

Static Data Structure	**Dynamic Data Structure**
A static data structure has a fixed size that is determined at the time of creation and cannot change during execution. For example: Array (mutable – can be changed) Tuple (Immutable – can not be changed) Advantages: • The program/memory management system can allocate a fixed amount of memory at compile time. No overhead for resizing. • Faster access: Direct access to elements via index. Disadvantages: • Fixed size: Wastes memory if not fully used, or may not be large enough. • No flexibility: Cannot grow or shrink as needed during runtime. *"No overhead for resizing."* This means that with **static data structures**, since the size is fixed and set at **compile time**, the program doesn't have to do any **extra work** to change the size during runtime.	A dynamic data structure can grow or shrink at runtime, allocating and deallocating memory as needed. For example: List, Linked List, Stack, Queue, Binary tree, Graph Advantages: • Flexible size: Adjusts to the program's needs. • Efficient memory use: Only uses memory when needed. • Inserting, merging and deleting of items is very easy and requires little processing power. Disadvantages: • More complex to implement (especially pointer management). • Slower access: No direct access like with arrays. • Overhead: Extra memory needed for pointers or management.

Array

- Static (Fixed size)
- Same data type for all elements
- Stores data linearly

- Elements are stored in contiguous memory locations, allowing fast sequential access.
- Contents are mutable - The values of elements can be changed,
- Memory allocated at compile time

Array

Example in pseudocode:

Array birdName(4)

birdName[0] = "pigeon" #the index here is 0

birdName[1] = "robin" #the index here is 1

birdName[2] = "blackbird" #the index here is 2

Result: birdName = [“pigeon”, “robin”, “blackbird”, “”]

numSpecies = len(birdName) #will assign 4 to numSpecies.

2-dimensional arrays

An array can have two or more dimensions.

Imagine a 2-dimensional array called numbers, with 3 rows and 4 columns.

Elements in the array can be referred to by their row and column number, so that:

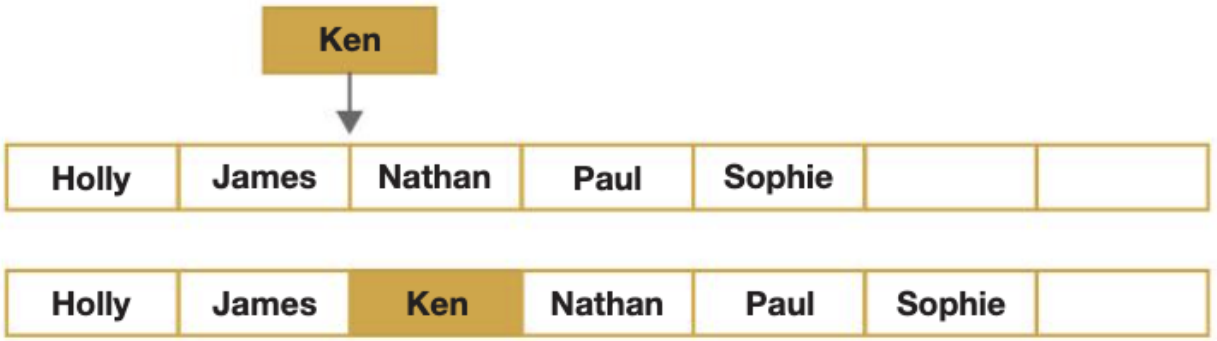
Example in pseudocode:

Array numbers (3,4)

numbers[1,3] = 8 in the example below.

	Column 0	Column 1	Column 2	Column 3
Row 0	1	2	3	4
Row 1	5	6	7	8
Row 2	9	10	11	12

Inserting a new item in an array



The steps are as follows:

- Test for list already full, print message if it is and quit
- Determine where new item needs to be inserted
- Starting at the end of the list, move other items along one place
- Insert new item in correct place

Deleting a name from the array

Suppose the name Ken is to be deleted from the array:

The names coming after Ken in the list need to be moved up to fill the gap.

Q: Why not simply leave the array element names[2] blank after deleting Ken?

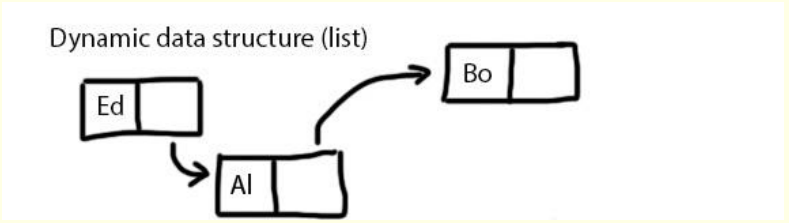
It's not a good idea to leave blank spaces in the list because then, functions like length(list) will not work correctly.

First, items are moved up to fill the empty space by copying them to the previous spot in the array:

Finally the last element, which is now duplicated, is replaced with a blank.

List

- Dynamic size (can grow and shrink)
- Can store different data types in the same list
- Mutable (elements can be changed, added, removed)
- Memory allocated at run time
- List values are stored non-contiguously. This means they do not have to be stored next to each other in memory.



Example

Example showing the use of a dynamic list structure in Python:

```
numbers = [] #Initalise list
numbers.append(12) #Add a value
numbers.append(8)
numbers.append(5)
```

Result: numbers = [12,8,5]

```
print(numbers.pop()) #Remove item
```

- Result: numbers = [12,8]

Tuple

- A tuple cannot be changed at runtime // a tuple is immutable
- Can store different data types
- Cannot be modified (no adding, removing, or changing elements)

Example:

• colour = (255, 0, 0) # Red colour