Digital Career Institute

Python Course - Collections

Digital Career Institute

Goal of the Submodule

The goal of this submodule is to help the learners work with collections in Python. By the end of this submodule, the learners should be able to understand

- What is a collection
- Which are the built-in collection types in Python
- The characteristics and differences between each collection type.
- How to choose the right data type for each situation.
- The additional data types provided by the `collections` Python module.
- How to use counters, ordered dictionaries, chain maps and named tuples.

Topics

- Collections and iterables.
  - Types of collections: linear, associative and graphs.
  - Linear collections: types and I/O methods.
  - Associative collections: sets and arrays.
- Lists.
- Tuples.
- Sets.
- Dictionaries.
- Operations with iterables.
  - Iterating iterables.
  - Iterable functions.
- The `collections` Python module.
  - Counters.
  - Ordered dictionaries.
  - Chain maps.
  - Named tuples.

Collections

What is a Collection?

Digital Career Institute

Source: https://unsplash.com/photos/JG35CpZLfVs

What is a Collection?

An **accumulation** of objects.

A **group** of objects.

An **aggregate** of objects.

What is a Collection?

What is a Collection?

A text string is also a collection, of characters.

What is a Collection?

In computer programming a collection is a type of **iterable**.

An iterable is any type, which value can be decomposed into different members, and its members can be returned one at a time (i.e. in `for` loops).

Storing Collections: Linear

Objects of a collection can be stored in a line, one after the other.

These are called **Linear Collections**.

Examples:

- A queue of customers.
- A list of students.
- The steps in a recipe.
- The names of one's children.

Storing Linear Collections

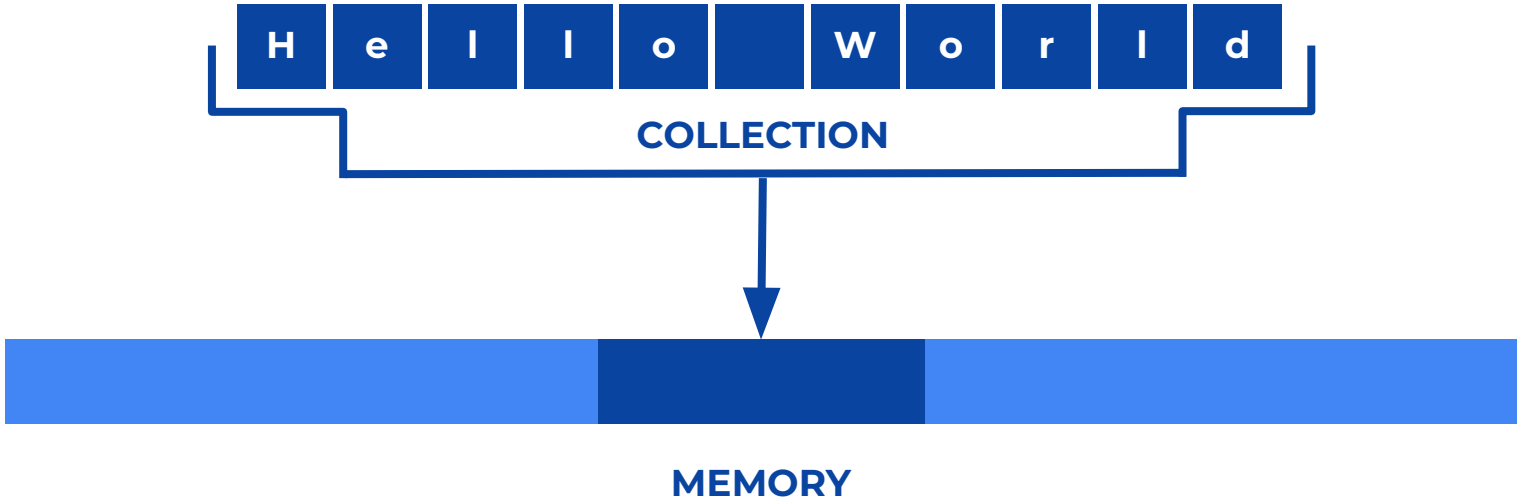
Placing the objects in line implicitly defines an order.

A different order of the same elements is a different sequence.

A linear collection is called
a **sequence** In Python.

Storing Linear Collections

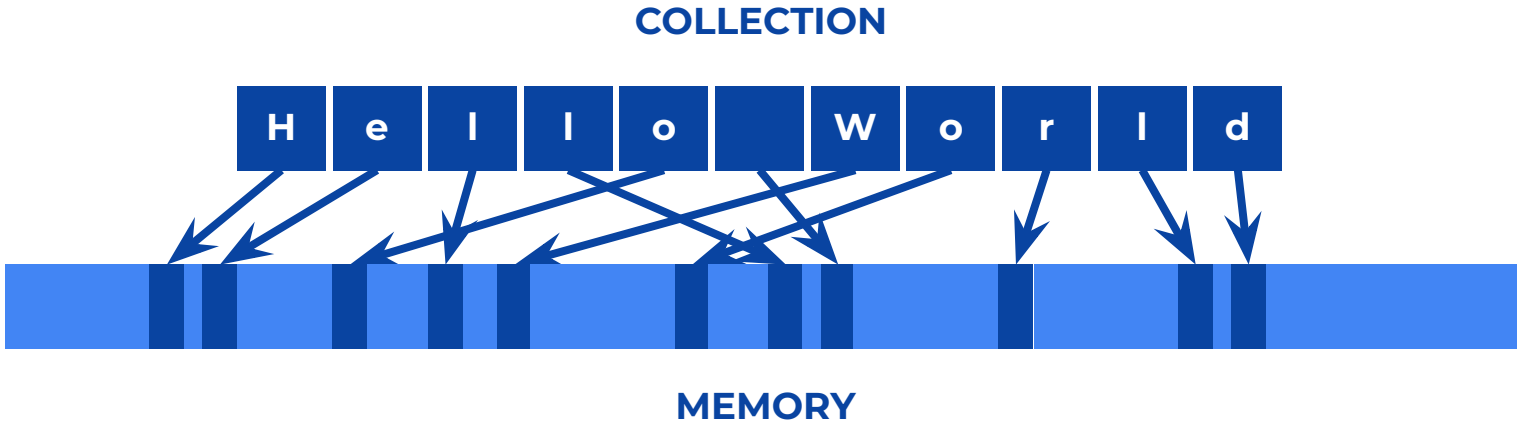
To identify the order, the objects are often stored using contiguous spaces in memory.
Then, the order of the sequence is **implicit** in memory.



Most language data types use this approach.

Storing Linear Collections

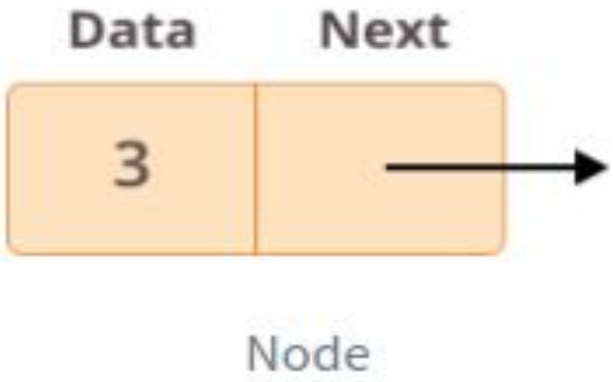
Items can also be stored in different locations in memory.
Then, the order of the sequence must be made **explicit**.



To identify the order, each item will hold its value
and a link to the next item in the sequence.
These are called **Linked Lists** and are not natively implemented in Python.

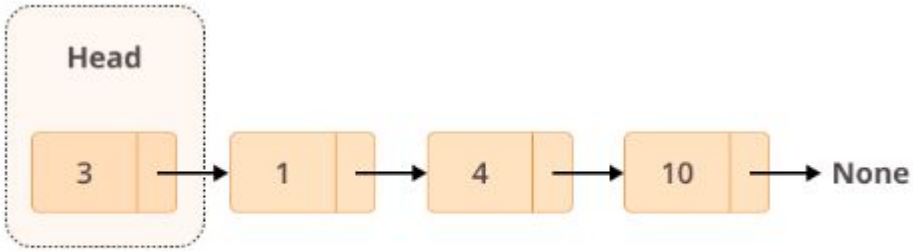
Storing Linear Collections

1. Data contains the value to be stored in the node.
2. Next contains a reference to the next node on the list.



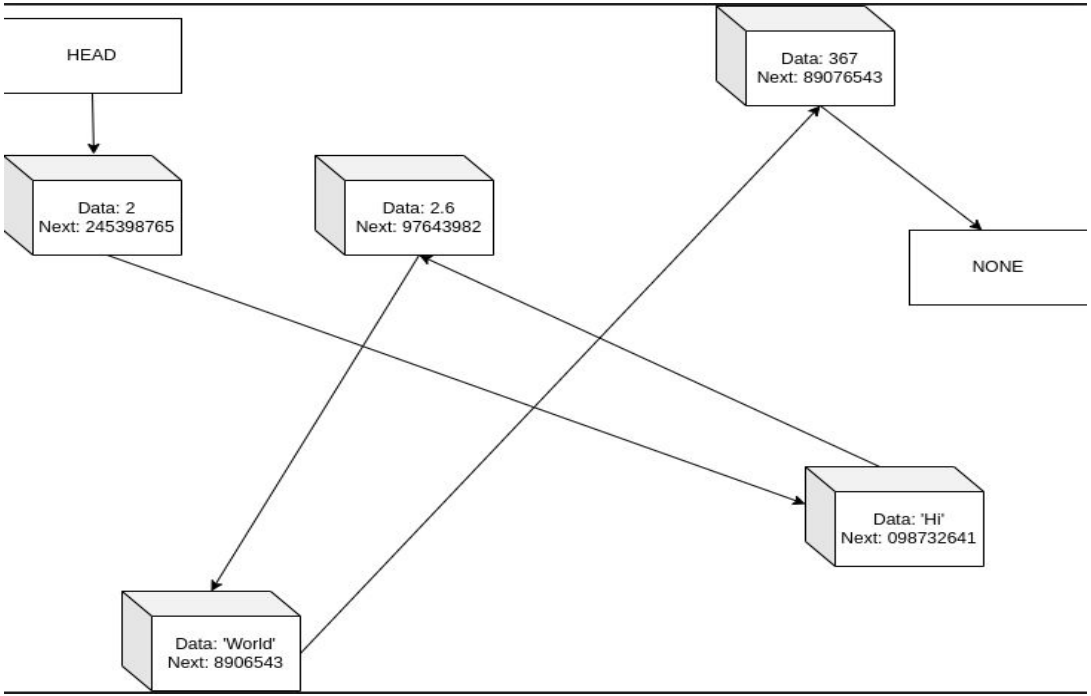
Storing Linear Collections

A linked list is a collection of nodes. The first node is called the `head`, and it's used as the starting point for any iteration through the list. The last node must have its `next` reference pointing to `None` to determine the end of the list. Here's how it looks:



Linked List

Storing Linear Collections



Using Collections

In computer science, collections are objects with properties and methods.

READ	ADD	REMOVE	SIZE

Different types of collections may have different methods,
but reading, adding and removing elements are common.

Adding is commonly referred as **Push**, removing as **Pop**,
reading as **Lookup**, and the size as **Length**.

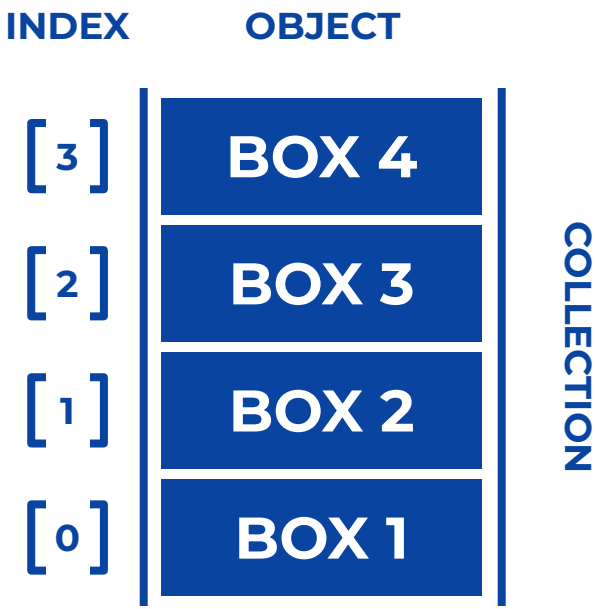
Reading Linear Collections

The objects in a linear collection can be accessed using numeric indexes that identify their position in the sequence.

```
collection[0] = Box 1
```

Indexes start at 0 and increase by 1.

They are not permanently associated to the object. If `Box 1` is removed, the rest of the boxes drop one position and the object associated with the index 0 will be `Box 2`.

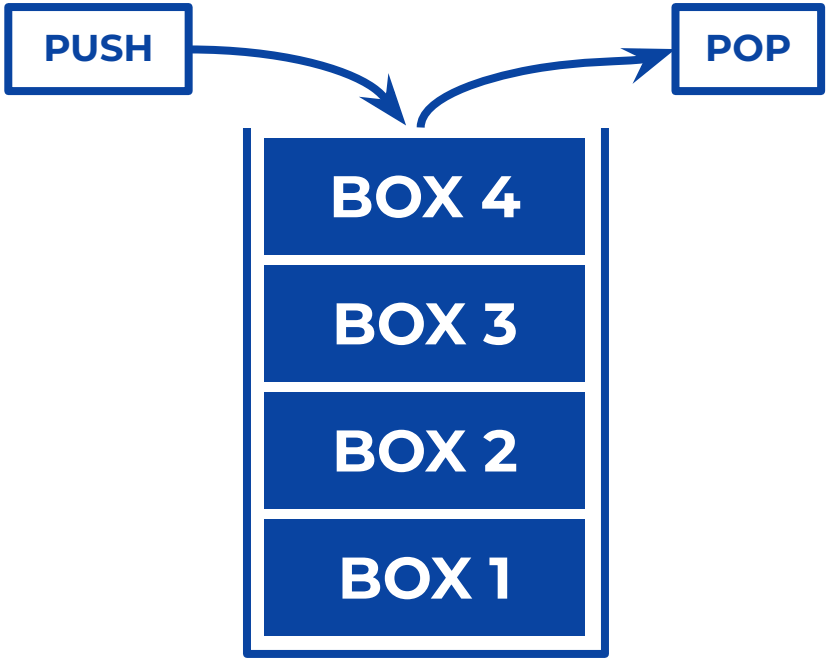


Types of Linear Collections: Stacks

A pile of items is called a **stack** in computer science.

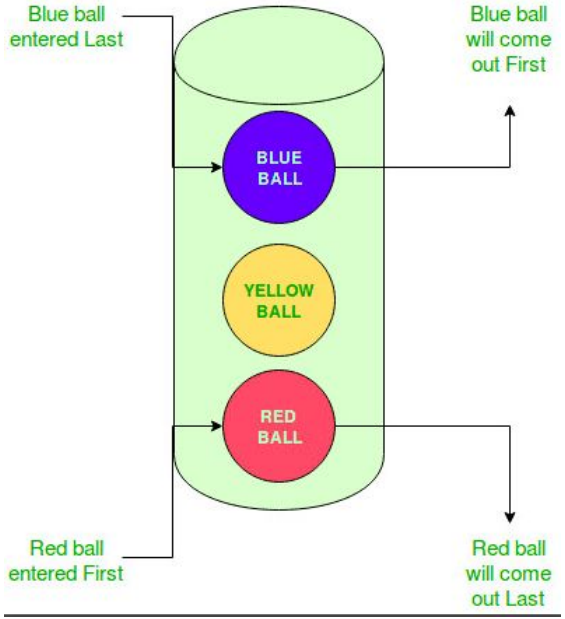
Stacks are linear collections where items can be added at one end and removed from the same end.

This method of manipulating linear collections is called **LIFO** (Last In, Frist Out).



Types of Linear Collections: Stacks

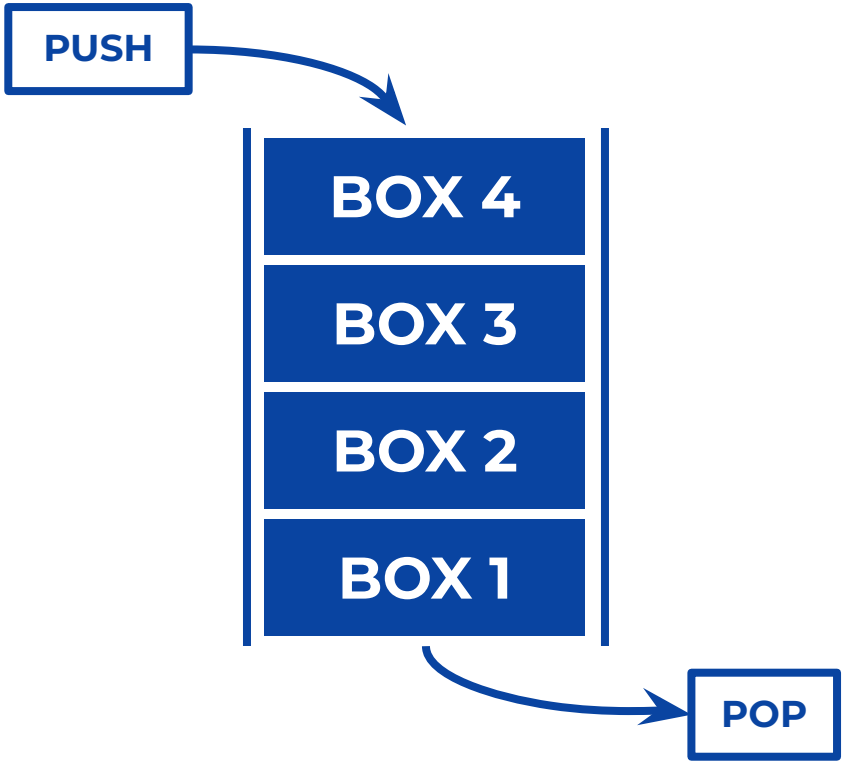
LIFO is an abbreviation for **Last in, first out** is the same as first in, last out (FILO). It is a method for handling data structures where the **last element** is processed first and the **first element** is processed last.



Types of Linear Collections: Queues

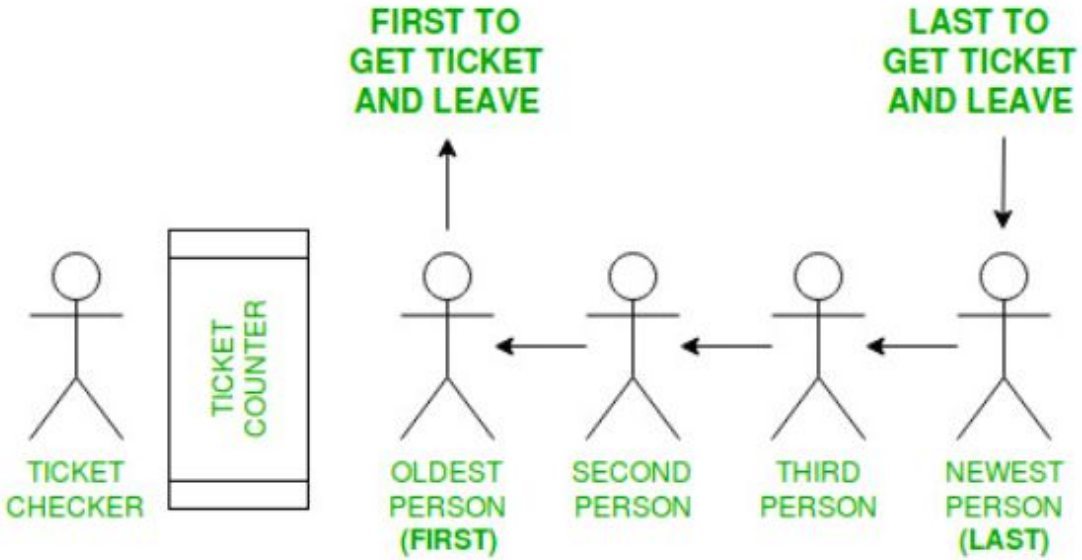
A **queue** is a linear collection that, instead, adds the new items on one end and removes them from the other.

This method of manipulating linear collections is called **FIFO** (First In, Frist Out).



Types of Linear Collections: Queues

FIFO is an abbreviation for **first in, first out**. It is a method for handling data structures where the **first element** is processed first and the **newest element** is processed last.



FIFO vs FILO

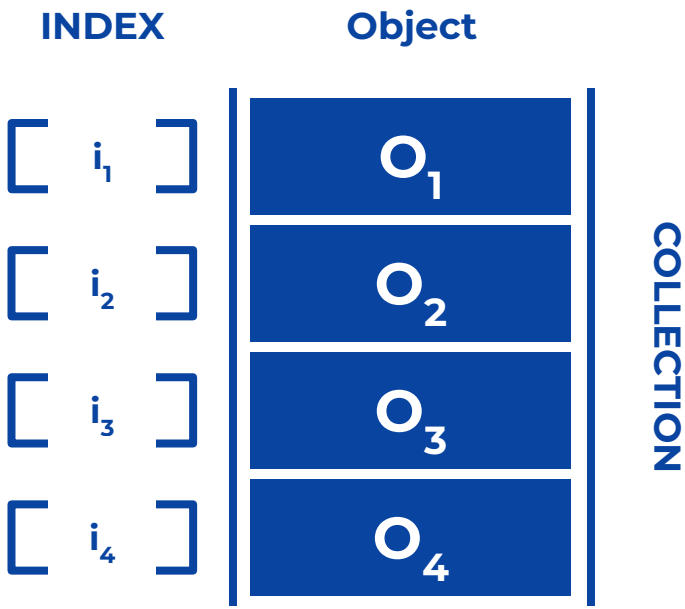
FIFO	LIFO
It stands for First-In-First-Out approach in programming.	It stands for Last-In-First-Out approach in programming.
In this, the new element is inserted below the existing element, So that the oldest element can be at the top and taken out first.	In this, the new element is inserted above the existing element, So that the newest element can be at the top and taken out first.
Therefore, the first element to be entered in this approach, gets out First.	Therefore, the first element to be entered in this approach, gets out Last.
In computing, FIFO approach is used as an operating system algorithm, which gives every process CPU time in the order they arrive.	In computing, LIFO approach is used as a queuing theory that refers to the way items are stored in types of data structures.
Time complexity of inserting element in FIFO is O(1).	Time complexity of inserting element in LIFO is O(1).
The data structure that implements FIFO is Queue.	The data structure that implements LIFO is Stack.

Storing Collections: Associative

Using linear collections implicitly defines an order, and adding or removing objects may change the index associated to each value.

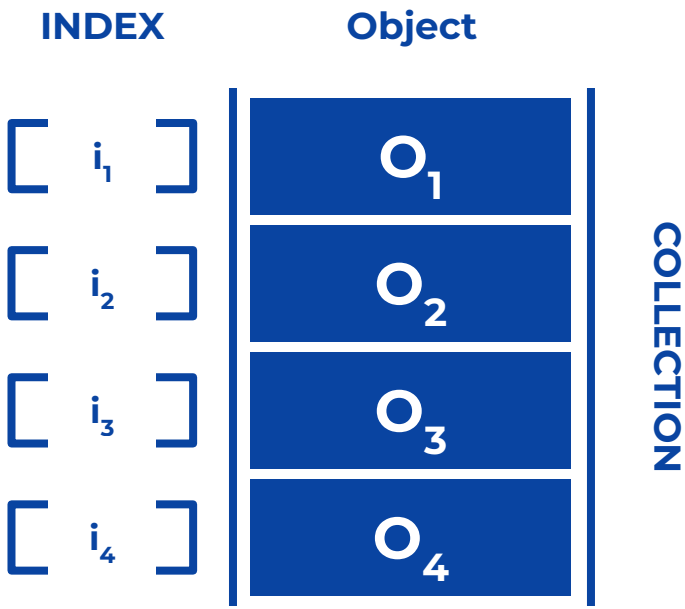
Associative Collections provide a method to assign each index to the same value, even if the rest of the objects are removed.

The index i_2 will always link to the object o_2, even if the object o_1 is removed from the collection.



Storing Collections: Associative

Examples:
- A dictionary.
- The shopping list.
- The ingredients in a recipe.
- A user profile.



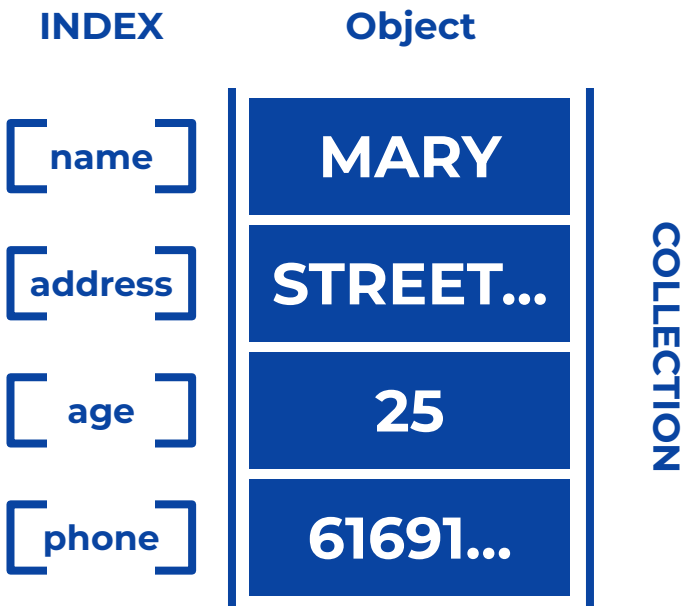
Types of Associative Collections: Arrays

Associative arrays are collections that use a key to index each one of its elements.

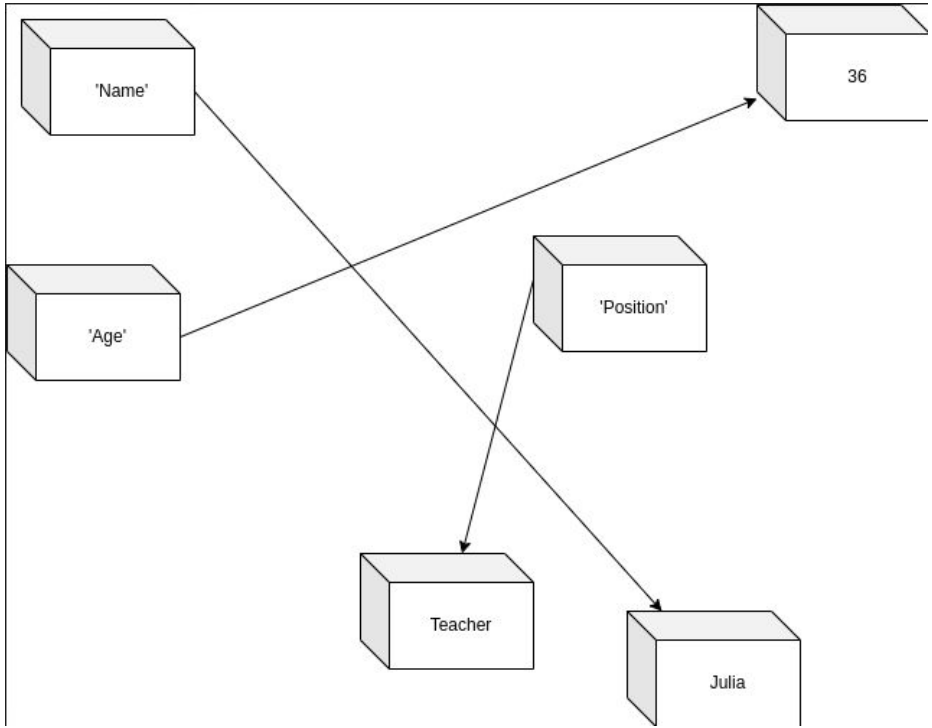
This key is usually a string that defines the value associated with it. It is also often called a collection of **key-value** pairs.

Associative arrays often have methods to see if a key exists, to remove a key, access all keys, ...

Example: a user profile.



Associative collection

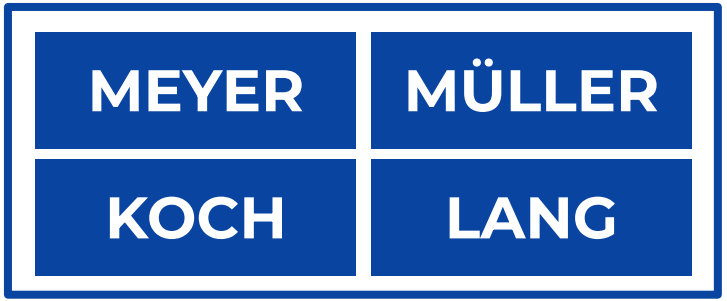


Types of Associative Collections: Sets

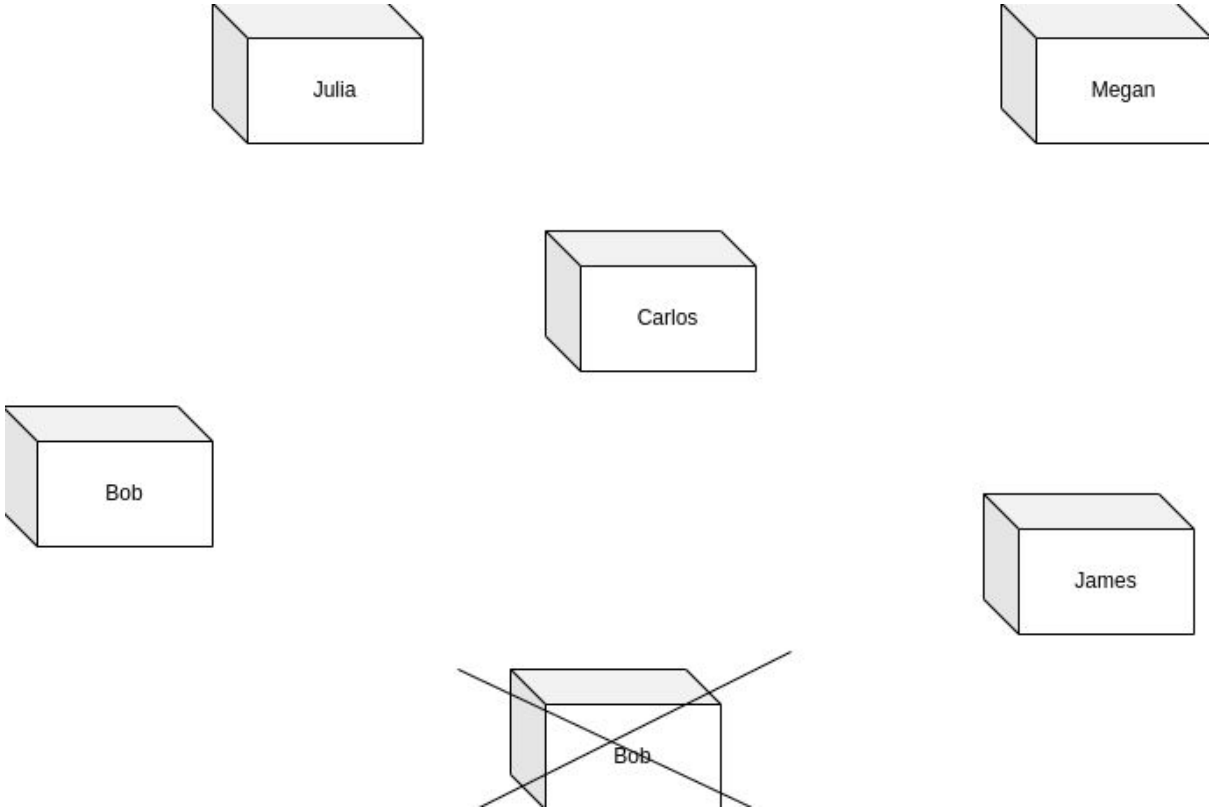
A **Set** is an unordered collection where duplicates are not allowed. They replicate the behavior of sets from **Set Theory**.

Sets often have methods to see if a value exists in the set and also provide specific methods to add and remove objects.

Example: german family names.



Set

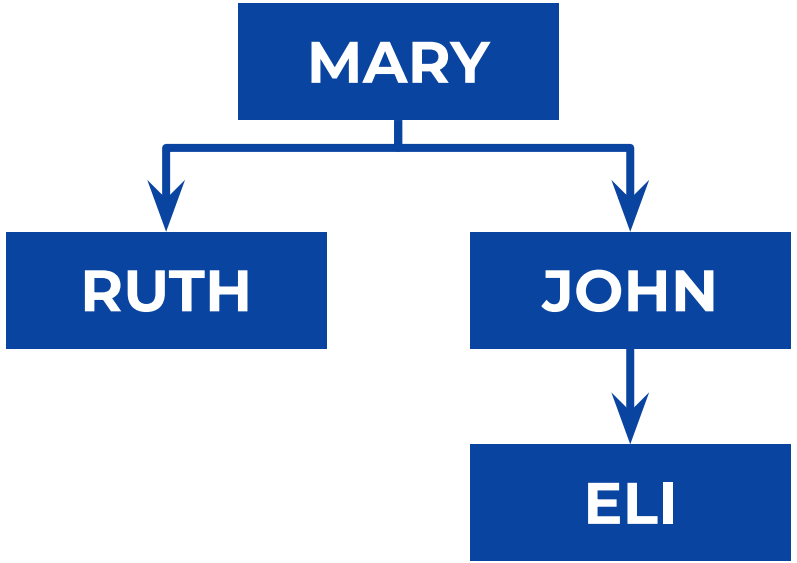


Storing Collections: Graphs

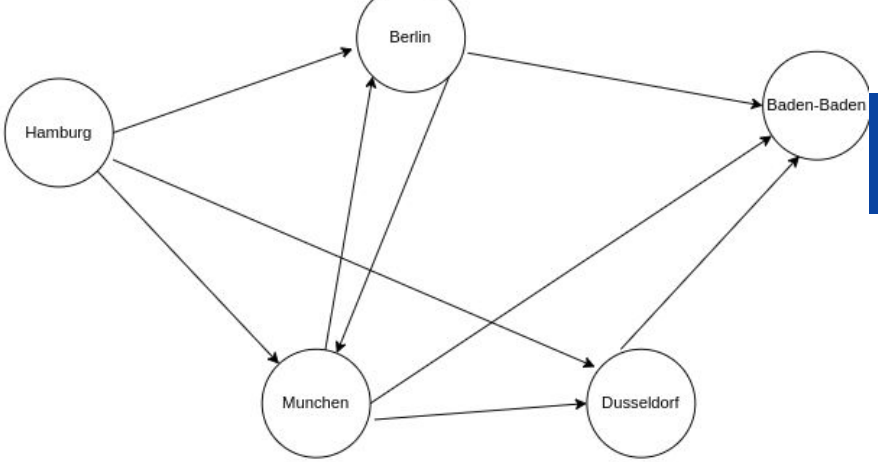
Graphs are collections of node objects, each of which refers to one or more other objects and has its own properties and methods to add, remove and read its elements.

Examples:

- The road network.
- The fiber network.
- The employees in a company.
- The genealogy.

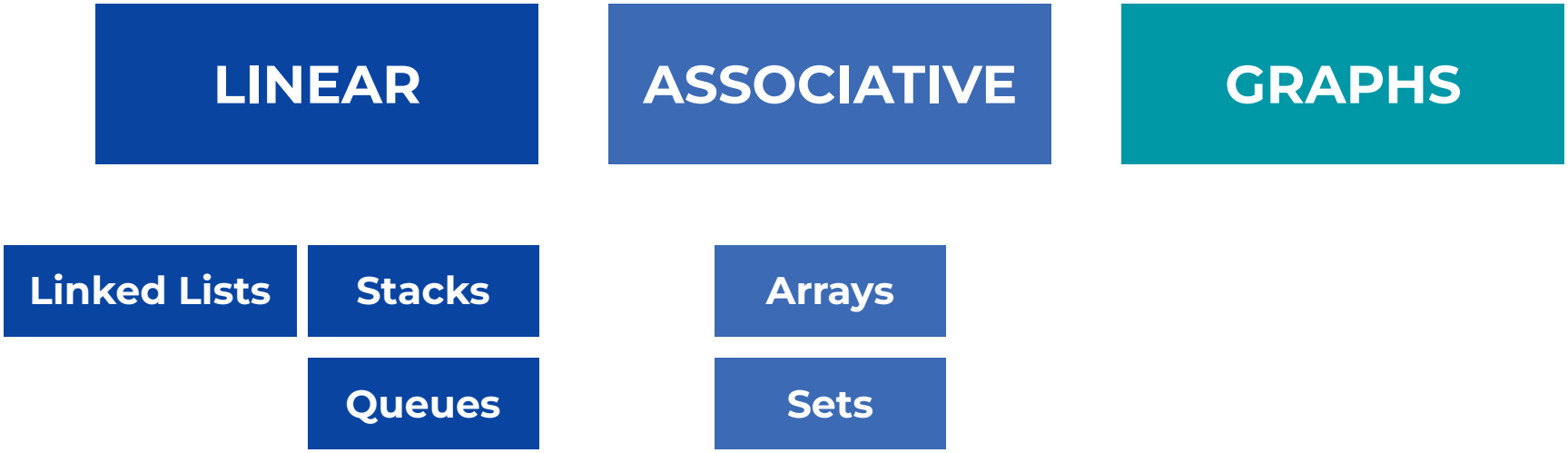


Graph



Hamburg	'Berlin', 'Dusseldorf', 'Munchen',
Berlin	'Munchen', 'Baden-Baden'
Munchen	'Berlin', 'Baden-Baden', 'Duseldorf'
Dusseldorf	Baden-Baden
Baden-Baden	

Types of Collections: Summary



We learned ...

- What is a collection.
- That there are different types of collections: linear, associative and graphs.
- That linear collections have an order, which can be implicit or explicit (linked lists).
- How to access and manipulate stacks and queues.
- That removing the first object of a linear collection will change the index associated to the rest of the objects.
- That associative collections use other kinds of indexes to access their objects.
- That sets do not allow duplicate values in the collection.