ESTRUCTURA DE DATOS

TALLER DE CONOCIMIENTO

MAYERLIS CARDENAS QUINTERO

ACTIVIDAD N°8

* Que es son las estructuras de datos

es una forma particular de organizar información en un computador para que pueda ser utilizada de manera eficiente. Diferentes tipos de estructuras de datos son adecuados para diferentes tipos de [aplicaciones](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_inform%C3%A1tica), y algunos son altamente especializados para tareas específicas.

Las estructuras de datos son medios para manejar grandes cantidades de información de manera eficiente para usos tales como grandes [bases de datos](https://es.wikipedia.org/wiki/Base_de_datos) y servicios de [indización](https://es.wikipedia.org/wiki/Indizaci%C3%B3n) de [Internet](https://es.wikipedia.org/wiki/Internet). Por lo general, las estructuras de datos eficientes son clave para diseñar [algoritmos](https://es.wikipedia.org/wiki/Algoritmo) eficientes.

* Tipos de estructuras de datos

Las **estructuras de datos estáticas** son aquellas en las que el tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse dicho tamaño durante la ejecución del programa, mientras que una **estructura de datos dinámica** es aquella en la que el tamaño ocupado en memoria puede modificarse durante la ejecución del programa.

Cada tipo de estructura dependerá del tipo de aplicación que se requiera. Una típica dentro de las **estructuras de datos estáticas** son los **arrays**

* Estructuras de datos en Python

Python incluye varias estructuras de datos de programación estándar, como list, tuple, dict, y set, como parte de sus tipos incorporados. Muchas aplicaciones no requieren otras estructuras, pero cuando lo hacen, la biblioteca estándar proporciona versiones poderosas y bien probadas que están listas para ser usadas.

* Defina los tipos de estructuras de datos en Python

Las listas son secuencia mutables, usadas normalmente para almacenar colecciones de elementos homogéneos (Donde el grado de similitud de los mismo depende de la aplicación).

*class***list**([*iterable*])

Las listas se pueden construir de diferentes formas:

* Usando un par de corchetes para definir una lista vacía: []
* Usando corchetes, separando los elementos incluidos con comas: [a], [a, b, c]
* Usando una lista intensiva o por comprensión: [x for x in iterable]
* Usando el constructor de tipo: list() o list(iterable)

Las tuplas son secuencias inmutables, usadas normalmente para almacenar colecciones de datos heterogéneos (Como las duplas o tuplas de dos elementos producidas por la función básica [enumerate()](https://docs.python.org/es/3/library/functions.html" \l "enumerate" \o "enumerate)). También son usadas en aquellos casos donde se necesite una secuencia inmutable de datos heterogéneos (Como por ejemplo permitir el almacenamiento en un objeto de tipo [set](https://docs.python.org/es/3/library/stdtypes.html#set) o [dict](https://docs.python.org/es/3/library/stdtypes.html" \l "dict" \o "dict)).

*class***tuple**([*iterable*])

Las tuplas se pueden construir de diferentes maneras:

* Usando un par de símbolos de paréntesis, para indicar una tupla vacía: ()
* Usando una coma al final, para crear una tupla de un único elemento: a, o (a,)
* Separando los elementos por comas: a, b, c o (a, b, c)
* Usando la función básica [tuple()](https://docs.python.org/es/3/library/stdtypes.html" \l "tuple" \o "tuple) built-in: tuple() o tuple(iterable)

Tipos Mapa — [dict](https://docs.python.org/es/3/library/stdtypes.html" \l "dict" \o "dict)

*class***dict**(*\*\*kwargs*)

*class***dict**(*mapping*, *\*\*kwargs*)

*class***dict**(*iterable*, *\*\*kwargs*)

Retorna un diccionario creado a partir de un parámetro opcional por posición, y por una serie de parámetros por nombre, también opcionales.

Los diccionarios se pueden construir de diferentes formas:

* Usando una lista separada por comas de pares key: value entre llaves:{'jack': 4098, 'sjoerd': 4127} or {4098: 'jack', 4127: 'sjoerd'}
* Usando una comprensión de diccionario: {}, {x: x \*\* 2 for x in range(10)}
* Usando un constructor de tipo: dict(), dict([('foo', 100), ('bar', 200)]), dict(foo=100, bar=200)

## Conjuntos — [set](https://docs.python.org/es/3/library/stdtypes.html#set), [frozenset](https://docs.python.org/es/3/library/stdtypes.html" \l "frozenset" \o "frozenset)

Un objeto de tipo conjunto o set es una colección no ordenada de distintos objetos [hashable](https://docs.python.org/es/3/glossary.html" \l "term-hashable). Los casos de uso habituales incluyen comprobar la pertenencia al conjunto de un elemento, eliminar duplicados de una secuencia y realizar operaciones matemáticas como la intersección, la unión, la diferencia o la diferencia simétrica (Para otros tipos de contenedores véanse las clases básicas [dict](https://docs.python.org/es/3/library/stdtypes.html" \l "dict" \o "dict), [list](https://docs.python.org/es/3/library/stdtypes.html" \l "list" \o "list), y [tuple](https://docs.python.org/es/3/library/stdtypes.html" \l "tuple" \o "tuple), así como el módulo [collections](https://docs.python.org/es/3/library/collections.html" \l "module-collections" \o "collections: Container datatypes)).

En la actualidad hay dos tipos básicos de conjuntos: [set](https://docs.python.org/es/3/library/stdtypes.html#set) y [frozenset](https://docs.python.org/es/3/library/stdtypes.html" \l "frozenset" \o "frozenset). La clase [set](https://docs.python.org/es/3/library/stdtypes.html#set) es mutable, es decir, el contenido del conjunto puede ser modificado con métodos como add() y remove(). Como es mutable, no tiene un valor de *hash* y no pueden ser usados como claves de diccionarios ni como elementos de otros conjuntos. La clase [frozenset](https://docs.python.org/es/3/library/stdtypes.html" \l "frozenset" \o "frozenset) es inmutable y [hashable](https://docs.python.org/es/3/glossary.html" \l "term-hashable), es decir, que sus contenidos no pueden ser modificados después de creados. Puede ser usado, por tanto, como claves de diccionario o como elemento de otro conjunto.

Se pueden crear conjuntos no vacíos (*sets*, no *frozensets*) escribiendo una lista de elementos separados por comas, entre llaves, por ejemplo {'jack', 'sjoerd'}, además de con el constructor de la clase [set](https://docs.python.org/es/3/library/stdtypes.html#set).

El constructor para ambas clases se usa de la misma forma:

*class***set**([*iterable*])

*class***frozenset**([*iterable*])

Retorna un nuevo *set* o *frozenset*, tomando los elementos a partir de *iterable*. Los elementos de un conjunto tienen que tener la propiedad de ser [hashable](https://docs.python.org/es/3/glossary.html" \l "term-hashable). Para representar conjuntos anidados, o conjuntos de conjuntos, los conjuntos interiores tienen que ser instancias de [frozenset](https://docs.python.org/es/3/library/stdtypes.html" \l "frozenset" \o "frozenset). Si no se especifica el parámetro *iterable*, se retorna un conjunto vacío.

Los conjuntos (*sets*) se pueden construir de diferentes formas:

* Usando una lista de elementos separados por coma entre corchetes: {'jack', 'sjoerd'}
* Usando un *set comprehention*: {c for c in 'abracadabra' if c not in 'abc'}
* Usando un constructor de tipo: set(), set('foobar'), set(['a', 'b', 'foo'])
* Realice un gráfico o dibujo de los tipos de estructuras de datos

**Arrays**

¿Qué es un array en programación? Un array es un tipo de dato estructurado que permite almacenar un conjunto de datos homogéneo y ordenado, es decir, todos ellos del mismo tipo y relacionados. Su condición de homogéneo, indica que sus elementos están compuestos por el mismo tipo de dato, y su condición de ordenado hace que se pueda identificar del primer al último elemento que lo compone.

**Estructura de datos dinámicas**

Por otro lado, vimos que en programación existen estructuras de datos dinámicas, es decir, una colección de elementos -nodos- que normalmente se utilizan para dejar asentados registros. A diferencia de un array que contiene espacio para almacenar un número fijo de elementos, una estructura dinámica de datos se amplía y contrae durante la ejecución del programa. Veamos algunos casos:

**Estructura de datos lineales**

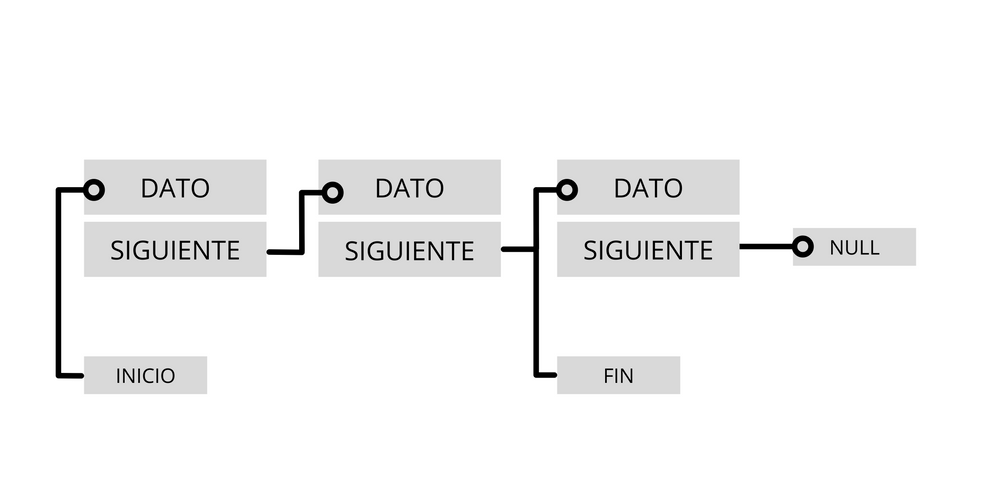
Las estructuras de datos lineales son aquellas en las que los elementos ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y un único predecesor, es decir, sus elementos están ubicados uno al lado del otro relacionados en forma lineal.

Hay tres tipos de estructuras de datos lineales:

Listas enlazadas

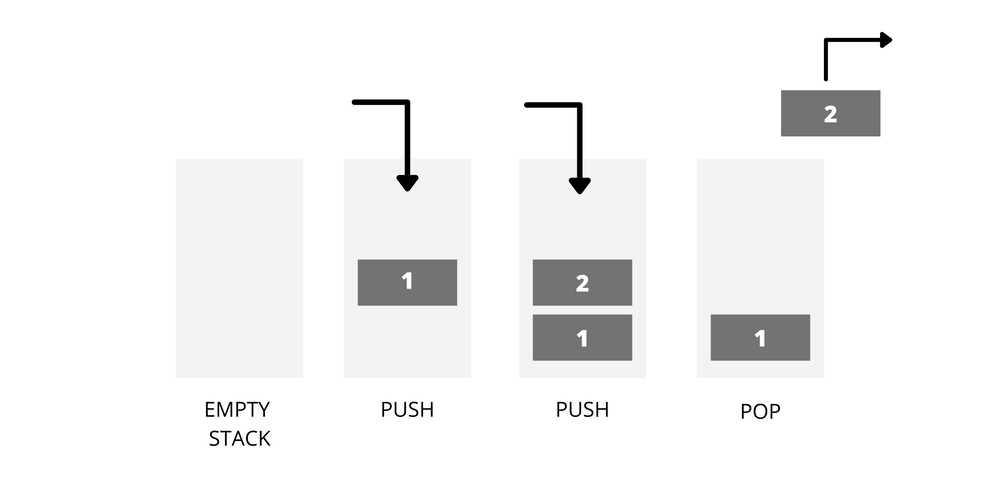
* Pilas
* Colas
* Listas enlazadas

En las estructuras de datos, las listas enlazadas se construyen con elementos que están ubicados en una secuencia. Aquí, cada elemento se conecta con el siguiente a través de un enlace que contiene la posición del siguiente elemento. De este modo, teniendo la referencia del principio de la lista podemos acceder a todos los elementos de la misma.



* **Pila**

La pila es un tipo especial de lista lineal dentro de las estructuras de datos dinámicas que permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo LIFO (del inglés Last In, First Out, es decir, último en entrar, primero en salir). ¿Cómo funciona? A través de dos operaciones básicas: apilar (push), que coloca un objeto en la pila, y su operación inversa, desapilar (pop), que retira el último elemento apilado.



* **Estructura de datos no lineales**

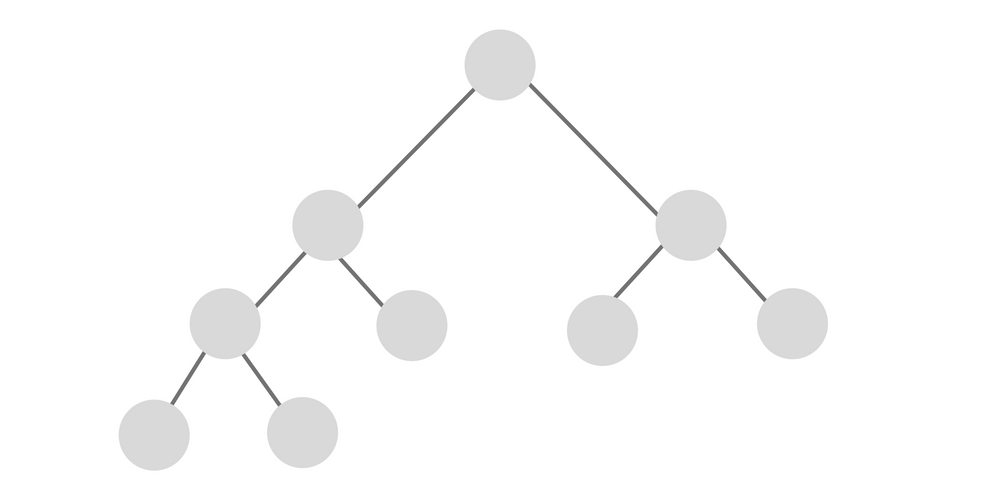
Las estructuras de datos no lineales, también llamadas multienlazadas, son aquellas en las que cada elemento puede estar enlazado a cualquier otro componente. Es decir, cada elemento puede tener varios sucesores o varios predecesores.

Existen dos tipos:

* Árboles
* Grafos
* Árboles

En estructura de datos, los árboles consisten en una estructura no lineal que se utiliza para representar datos con una relación jerárquica en la que cada elemento tiene un único antecesor y puede tener varios sucesores.

Los mismos se encuentran clasificados en: árbol general, un árbol donde cada elemento puede tener un número ilimitado de sub árboles y árboles binarios, que son una estructura de datos homogénea, dinámica y no lineal en donde a cada elemento le pueden seguir como máximo dos nodos.



* **Grafos**

Otro tipo de no lineal de estructura de datos en programación, son los grafos. Se trata de una estructura matemática formada por un conjunto de puntos —una estructura de datos— y un conjunto de líneas, cada una de las cuales une un punto a otro. Los puntos se llaman nodos o vértices del grafo y las líneas se llaman aristas o arcos.

* 