**Algoritmo:** Secuencia finita de Instrucciones, cada una de las cuales tiene un significado preciso y puede ejecutarse con una cantidad finita de esfuerzo en un tiempo finito.

¿Como medimos el tiempo de ejecución? 🡪 Alternativa empírica: Contamos con el módulo TIME que ofrece dos funciones:

* Clock: nro de segundos que la CPU dedico a la ejecución del programa.
* Time: Retorna el nro de segundos que transcurrieron desde la medianoche del 01/01/1970.
* Los resultados obtenidos con estas funciones dependen de la maquina donde se ejecuten.
* Entonces el tiempo de ejecución depende de: los datos de entrada, la calidad del código generado por el compilador y la naturaleza y rapidez de las instrucciones de maquina usadas en ejecución.
* Tiempo de ejecución = T(n) 🡪Depende del tamaño de la entrada 🡪Máximo valor del tiempo de ejecución para entradas de tamaño n. 🡪No es posible expresarlo en una unidad de tiempo.
* Operación Elemental: Aquellas que el procesador realiza en un tiempo acotado por una constante.
* Si sabemos que vamos a tener entradas grandes, entonces conviene optar por el programa cuyo T(n) tiene la menor velocidad de crecimiento.
* Un programa es eficiente si su velocidad de crecimiento es pequeña.
* Tipos de O(n):
  + O(1): Complejidad constante.
  + O(log(n)): Complejidad logarítmica.
  + O(n): Complejidad lineal.
  + O(N^2): Complejidad cuadrática.
  + O(n^3): Complejidad cubica.
  + O(2^n): Complejidad exponencial.
  + O(n^n): Iteraciones con cada elemento.

Búsqueda Lineal o Secuencial: Se realiza la búsqueda comparando con cada elemento del vector hasta el último.

Búsqueda Binaria: El vector debe estar ordenado de manera creciente o decreciente. Compara en primer lugar con la componente central del vector, y si no es igual al valor buscado, se reduce el intervalo de búsqueda a la mitad derecha o izquierda dependiendo de donde se pueda estar el valor a buscar. 🡪O(log(n))

Métodos de ordenación:

* Selección: Si el vector tiene N elementos, este método hace N-1 pasadas sobre el vector. Tiene O(n^2).
* Inserción: Inserta cada elemento del vector en un subvector ordenado, pero lo hace en el mismo vector. Asume que el primer elemento esta ordenado. O(n^2).

**TAD:**

* Define los valores que pueden ser representados en un objeto de ese tipo, como se almacenara en memoria y que operaciones se pueden realizar sobre él.
* Clasificación:
  + - Simples o Compuestos.
    - Homogéneos o Heterogéneos.
    - Estáticos o Dinámicos.
* Es una colección o grupo de datos organizados de tal forma que tengan asociados un conjunto de operaciones para poder manipularlos.
* **Una abstracción es un** proceso en el cual se extraen los rasgos esenciales de algo para representarlos de forma escrita.
* **Nivel Lógico:** Se define abstractamente la estructura de datos y las operaciones que pueden realizarse sobre ella. La descripción que se obtenga debe ser independiente del lenguaje de programación que se vaya a utilizar.
* **Nivel De Implementación**: Se decide el lenguaje de programación en el que se implementara la estructura. Se implementarán cada una de las operaciones del TAD.

**Especificación del TAD:**

* TAD nombre\_del\_TAD
* Igualdad observacional 🡪Cuando dos mismas estructuras son iguales.
* Usa🡪Se definen los tipos de datos que usaremos en la especificación del TAD.
* Parámetro Formal🡪El tipo de los elementos que contiene la estructura🡪Cuando es heterogénea se pone “a”.
* Géneros🡪Es el nombre con el que hacemos referencia a la instancia del TAD🡪Ej: Pila<a>.
* Observadores Básicos🡪Son las operaciones que no cambian a la estructura🡪Se pueden usar restricciones dependiendo el caso, pueden ser Pre{….} o Post{…}.
* Generadores🡪Son operaciones que generan una nueva estructura del mismo tipo que se trabaja🡪Siempre hay dos generadores: Vacío, y Con Elementos🡪También se utiliza la restricción Post{..}
* Otras Operaciones🡪Van todas las operaciones que no sean ni Observadores Básicos ni Generadores. Estas si pueden modificar la estructura.
* Axiomas🡪Se explica que hacen los Observadores Básicos, los Generadores y las Otras Operaciones, una por una.
* Exporta🡪Se ponen todas las operaciones separadas por “,”

Las operaciones que **no tienen sentido** son las que no cumplen con la lógica de la estructura. Por Ejemplo: Acceder por posición a un conjunto.

Las operaciones **no indispensables** son las que, dada una operación, se puede hacer lo mismo manualmente. Por Ejemplo: Vaciar una lista = Vaciarla manualmente sacando elemento a elemento.

**Restricción de acceso:** No poder pedir elementos que no sean de los extremos

En Python **el tipado es dinámico**, ya que los tipos de las variables se determina en tiempo de ejecución.

La **variable** es usada como nombre de un objeto. Un **objeto** es todo valor que se pueda manipular en Python. Todo objeto tiene un TIPO.

Una **clase** es una forma de implementar un tipo. Toda clase define un tipo, pero no todo tipo está asociado a una clase. En Python toda clase define un tipo y todo tipo está asociada a una clase.

Luego de crear un objeto de una clase dada, Python llama al método \_\_init\_\_ de esa clase para que lo inicialice.

Podemos indicar con un “\_” adelante del nombre para indicar que eso es un detalle de implementación privado y no debe usarse.

La **referencia** hace que los dos objetos siempre tengan los mismos elementos.

Un dataclass es una clase pensada para almacenar datos, similar a un registro (Structs). Los tipos que se definen con Dataclass permiten agrupar información relacionada en una sola entidad u objeto. Y esta entidad permite definir nuevos tipos de datos que están compuestos por valores de tipos mas simples. 🡪from dataclasses import dataclass. Para definir una dataclass, usamos @dataclass sobre la clase que queramos usar.

Para usar un tipo de datos “Cualquier Tipo” se pone: **from typing import Any**

**Recursión**: Cuando se obtienen soluciones a problemas donde una función se llama a si misma para resolver el problema, se tiene una Función Recursiva. Para un mismo algoritmo la recursión permite una expresión mas clara y sintética, lo que simplifica la compresión y el mantenimiento. A su vez, las soluciones recursivas son mas costosas en tiempo y memoria (Menos Eficientes). 🡪Búsqueda Binaria

Arboles:

* Cada elemento de un árbol (Nodo) se puede relacionar con cero o mas elementos a quien llamamos “Hijos”.
* Si el árbol no esta vacío, hay un único elemento al cual se le llama RAIZ y no tiene padre, es decir, no es hijo de ningún otro.
* Todo otro elemento del árbol posee un único padre.
* Si un nodo no tiene hijos, se lo llama “Hoja” o “Nodo Terminal”
* El enlace entre dos Nodos se lo llama “Arista”
* La raíz de un árbol se define como el nodo que no tiene Ascendientes.
* Se denomina “Grado De Un Nodo” a la cantidad de hijos directos que tiene un nodo.
* Se denomina “Nivel De Un Nodo” al numero de descendientes que debe recorrerse desde la raíz del árbol hasta dicho nodo. El nivel de la raíz es 0.
* El nivel máximo en que se encuentran los elementos de un árbol se denomina “Profundidad” o “Altura”.

Índices: Colocar índices a un archivo es tener una lista ordenada que sirva de referencia para acceder en forma más rápida a la información buscada. En caso de agregar o borrar un elemento, tenemos que actualizar los índices.