



El futuro digital
es de todos

MinTIC



Búsqueda



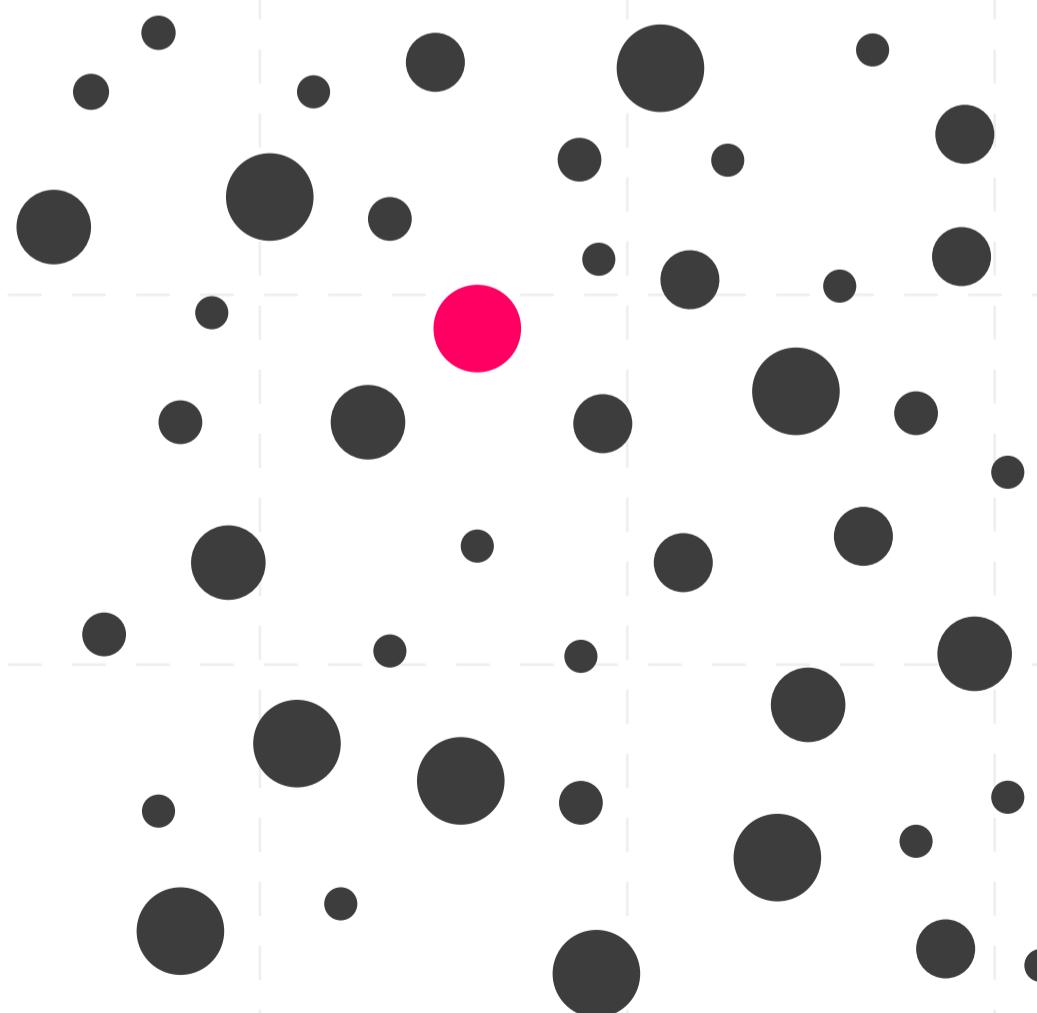
Universidad de Caldas

Hola:

Buscar es uno de esos elementos que cumple una condición que permite recorrer un conjunto de elementos.

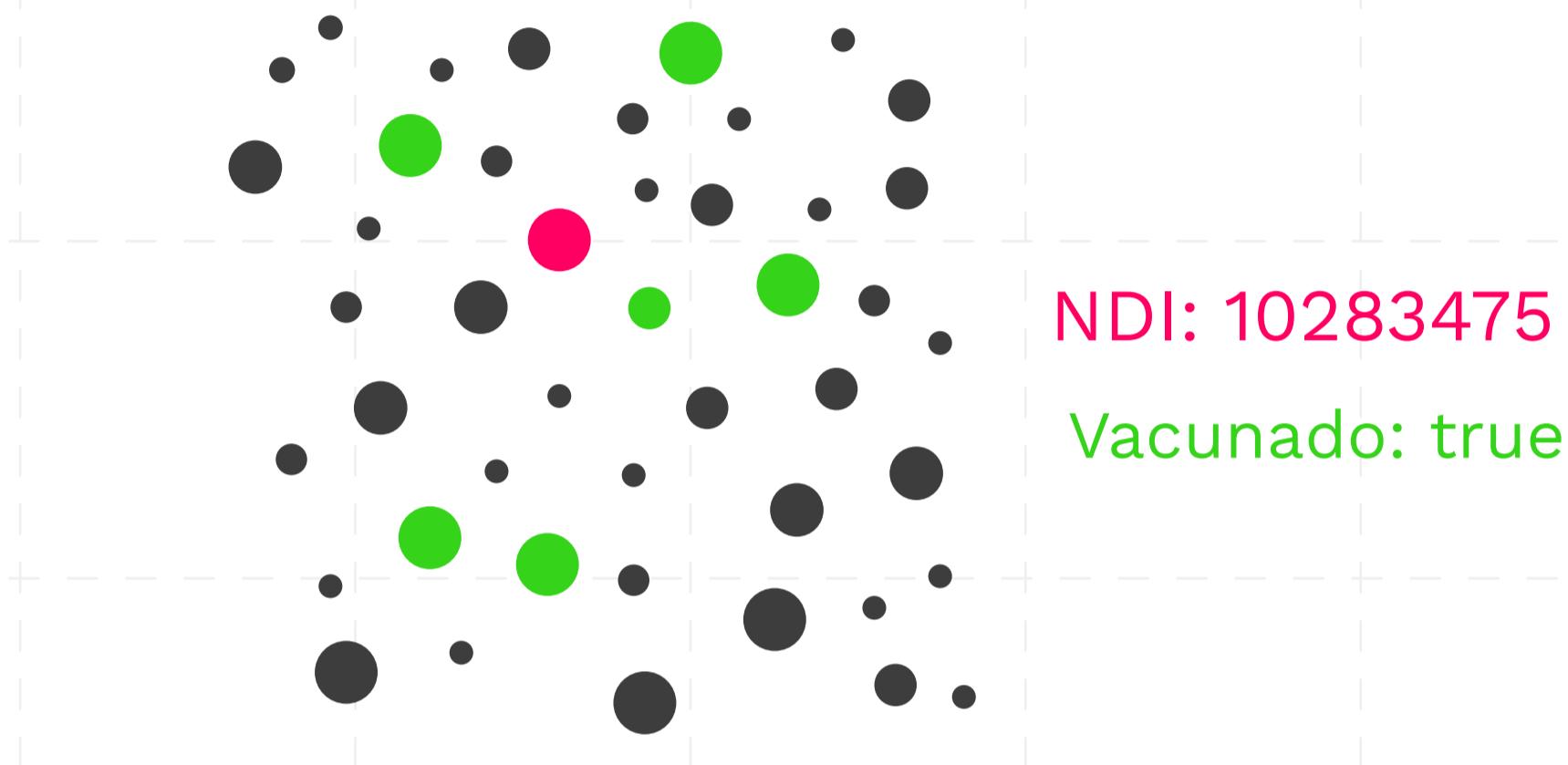
Supongamos que tenemos un conjunto de elementos en el que se requiere encontrar uno y solo un elemento que cumple una condición única. Esto generalmente se conoce como la **Llave principal del elemento**, que puede entenderse de manera análoga a nuestro documento de identidad, huellas digitales, claves, etc. Estas tienen que ser únicas, debido a que solo corresponden a una sola persona.

En este ejemplo podemos buscar un número de identificación específico y podemos encontrar a una persona dentro de ese grupo que cumple con esa condición.

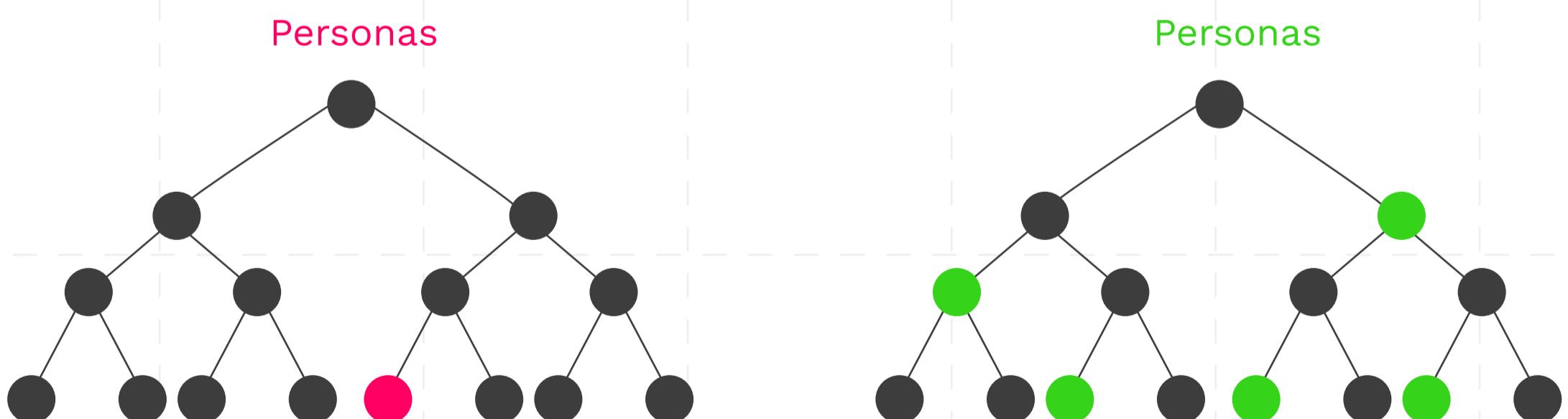


NDI: 10283475

En otro caso, lo que podría suceder es que al buscar en un grupo de elementos esa condición que estamos buscando la cumplan muchos de esos elementos y no solamente uno. En este caso, por ejemplo, podemos buscar si dentro de ese grupo de personas algunas ya están vacunadas, quienes son mayores de edad, quienes tienen un salario superior a un mínimo, etc.



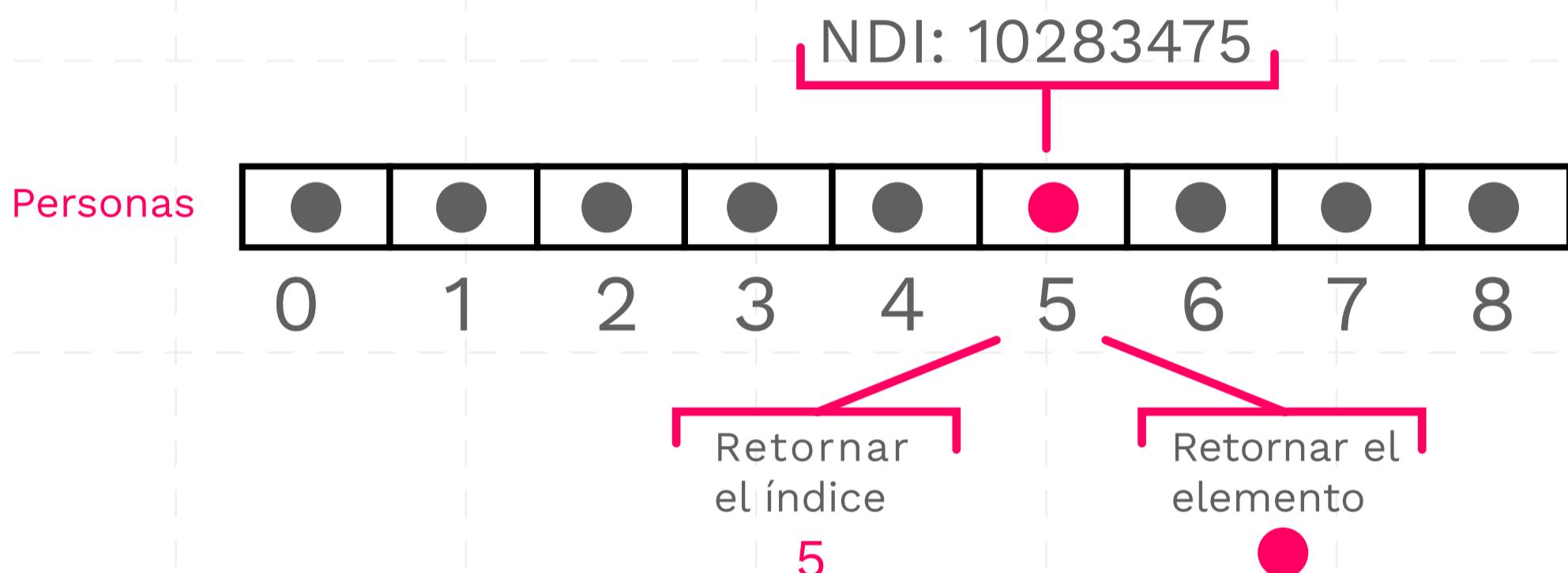
La estructura de datos que representa la información direccionará el algoritmo o los programas que van a realizar la búsqueda. No es lo mismo hacer una búsqueda en una lista, que hacerlo por ejemplo una estructura como esta que se llama árbol en la que cada círculo tiene dos ramas en las cuales se deben buscar los elementos que se buscan.



En este caso buscar todos los elementos en esta estructura no será igual el algoritmo que busca en este tipo de estructuras como el que busca en listas. La estructura de datos condiciona el algoritmo de datos que vamos a utilizar para buscar el elemento.

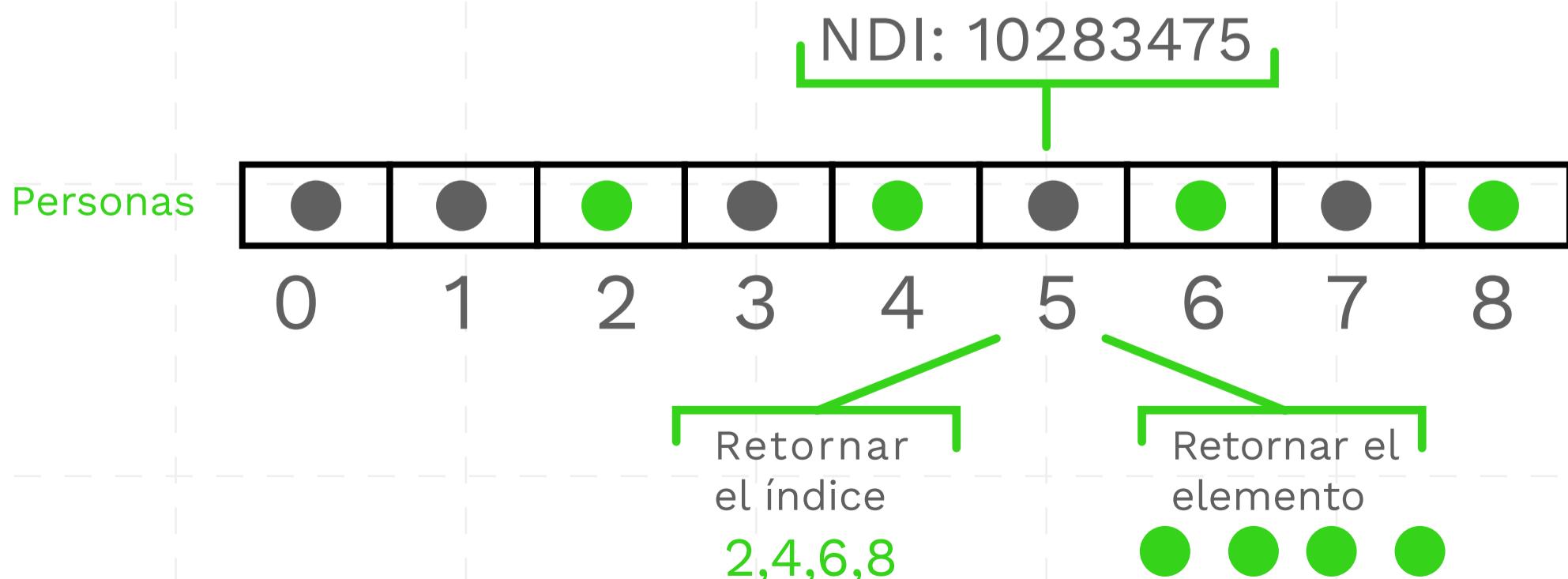
Un algoritmo de búsqueda es aquel que obtiene información almacenada, es decir, que cumple una condición en una estructura de datos.

Búsqueda en una lista secuencial:



Si se encuentra el elemento que se está buscando se debe retornar el índice en el cual se encontró, pero también puede retornar el elemento porque no sabemos si este contiene más información como el nombre de la persona, dirección, correo electrónico, etc.

Veamos cuando varios elementos cumplen una condición:



Como en el ejemplo anterior es posible retornar los índices de los elementos que cumplen esa condición o retornar los elementos que requieran más información.

Esta es una lista que tiene solo los números de los documentos, en la que definimos una función que se llama **buscar**.

```
personas=['1001458257','100225897','10283475','35236587','568745218']
def buscar_id(id_s):
    for id in personas:
        if id==id_s:
            return id
    return None
```

0 1 1 1 1 0 1 0
0 1 1 1 1 0 1 0
0 1 1 1 1 0 1 0
0 1 1 1 1 0 1 0
0 1 1 1 1 0 1 0

Recorrido parcial

Se hace un recorrido parcial buscando el elemento que cumpla la condición, por ejemplo, un número de cédula específica y una vez la encuentre dejará de buscar.

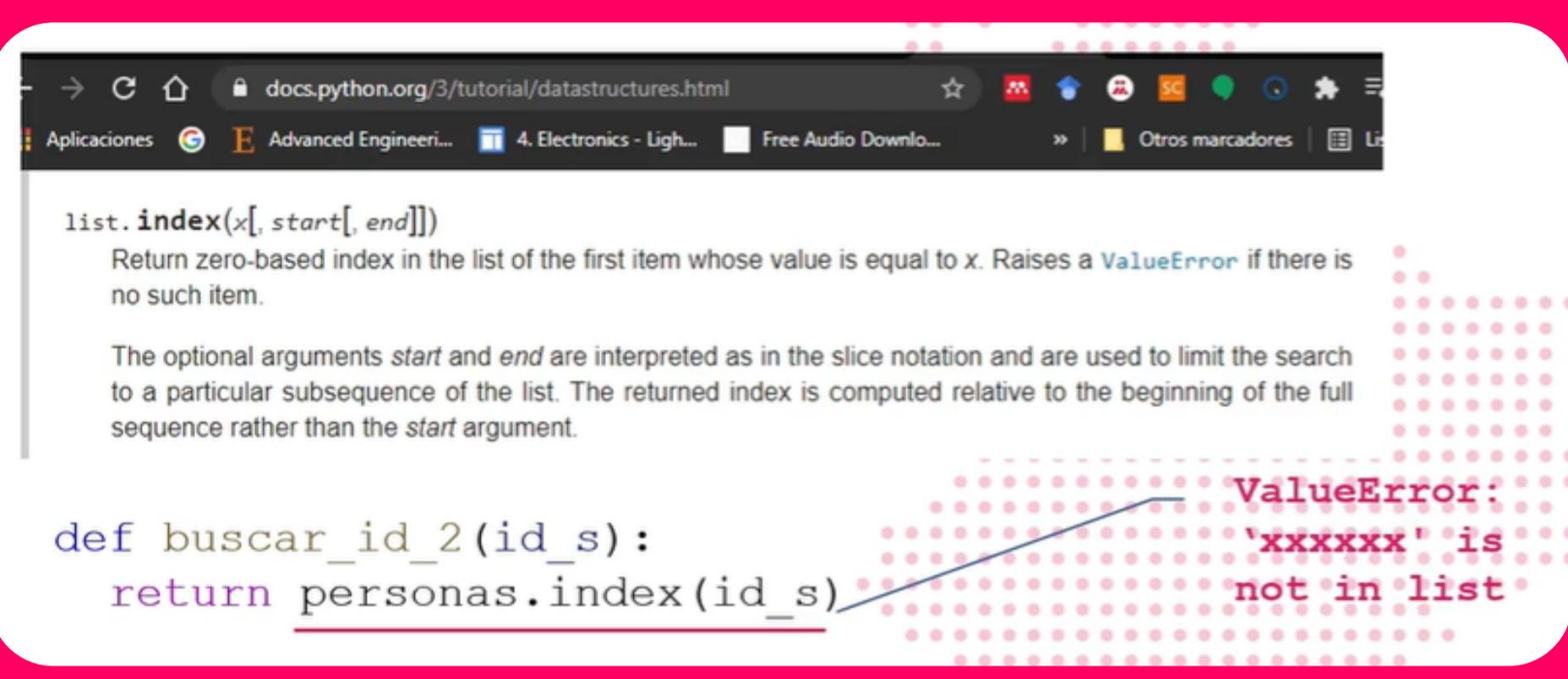
```
personas=[('1001458257',True), ('100225897',True), ('10283475',False),
          ('35236587',True), ('568745218',False)]

def vacunados():
    lista_vacunados=[]
    for p in personas:
        if p[1]==True:
            lista_vacunados.append(p)
    return lista_vacunados
```

0 1 0 1 0 1
0 1 0 1 0 1
0 1 0 1 0 1
0 1 0 1 0 1
0 1 0 1 0 1

En este caso se hace un recorrido completo por los elementos buscando cuántos cumplen la condición y al final obtendremos una sublista.

Con Python, la función list tiene una función que se llama index que nos retorna el índice, es decir, la posición en la cual se encuentra un elemento, sin embargo, si el elemento no se encuentra se genera un error que interrumpe la ejecución del programa.



list.**index**(*x*[, *start*[, *end*]])

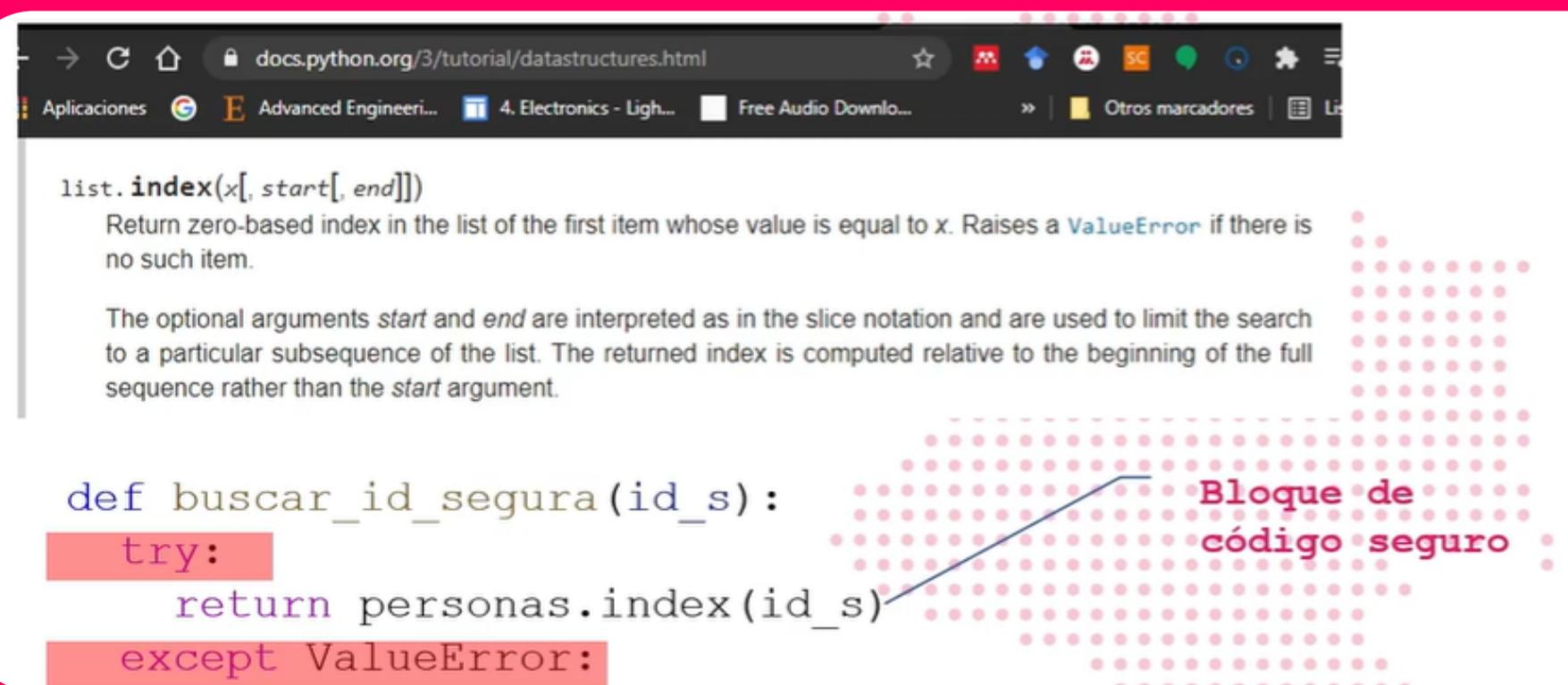
Return zero-based index in the list of the first item whose value is equal to *x*. Raises a `ValueError` if there is no such item.

The optional arguments *start* and *end* are interpreted as in the slice notation and are used to limit the search to a particular subsequence of the list. The returned index is computed relative to the beginning of the full sequence rather than the *start* argument.

```
def buscar_id_2(id_s):
    return personas.index(id_s)
```

ValueError:
`'xxxxxx'` is
not in list

Podemos implementar las instrucciones *try* y *except* que son un bloque de código seguro, es decir, si se presenta este tipo de error que termina con la ejecución, no lo haga.



list.**index**(*x*[, *start*[, *end*]])

Return zero-based index in the list of the first item whose value is equal to *x*. Raises a `ValueError` if there is no such item.

The optional arguments *start* and *end* are interpreted as in the slice notation and are used to limit the search to a particular subsequence of the list. The returned index is computed relative to the beginning of the full sequence rather than the *start* argument.

```
def buscar_id_segura(id_s):
    try:
        return personas.index(id_s)
    except ValueError:
```

Bloque de código seguro

En conclusión, vimos que la búsqueda es una de las funciones clásicas y básicas cuando se tiene un grupo de elementos con la que podemos hacer búsquedas secuenciales de manera parcial o completa. Vimos algunas características que nos facilitan la búsqueda en Python, en particular las función `index` y también vimos la búsqueda segura que nos ayuda a terminar una ejecución de una manera más amigable.





Universidad de Caldas