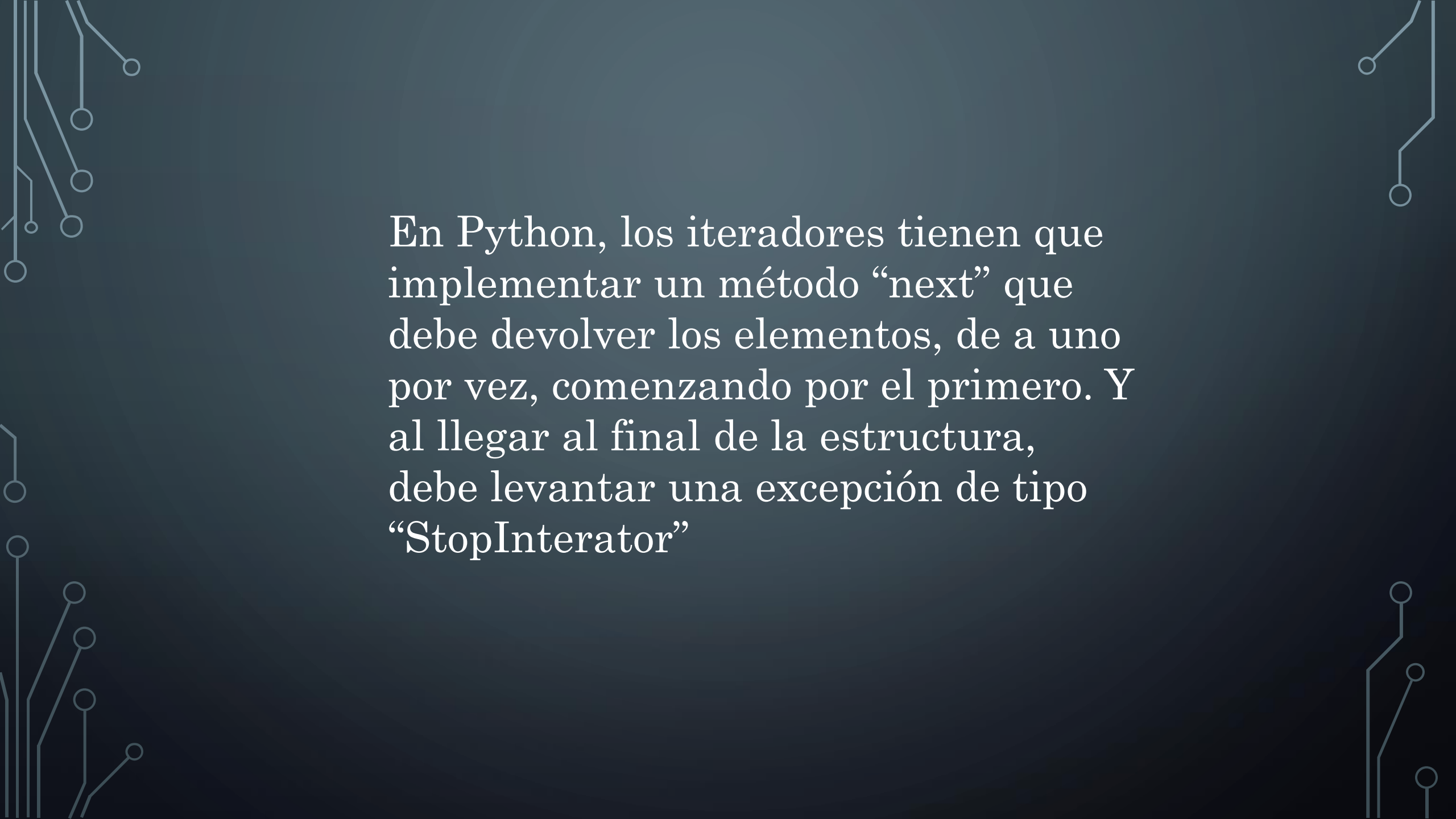


# Iterador



# Que es un iterador en los patrones de diseño?

El patrón Iterador es un mecanismo de acceso a los elementos que constituyen una estructura de datos para la utilización de estos sin exponer su estructura interna.

The image features a dark blue background with white, stylized circuit board traces in the corners. These traces consist of straight lines and small circles, resembling electronic components or wiring. The traces are located in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

En Python, los iteradores tienen que implementar un método “next” que debe devolver los elementos, de a uno por vez, comenzando por el primero. Y al llegar al final de la estructura, debe levantar una excepción de tipo “StopInterator”

# Propósito

- \*Surge del deseo de acceder a los elementos de un contenedor de objetos (ejemplo una lista, arreglos, tablas de hash, listas enlazadas, etc..) sin exponer su representación interna.
- \*Es posible que se necesite más de una forma de recorrer la estructura siendo para ello necesario crear modificaciones en la clase.
- \*Diferentes iteradores pueden presentar diferentes tipos de recorrido sobre la estructura
- \*Los iteradores no tienen por qué limitarse a recorrer la estructura, sino que podrían incorporar otro tipo de lógica como filtrar ciertos elementos.

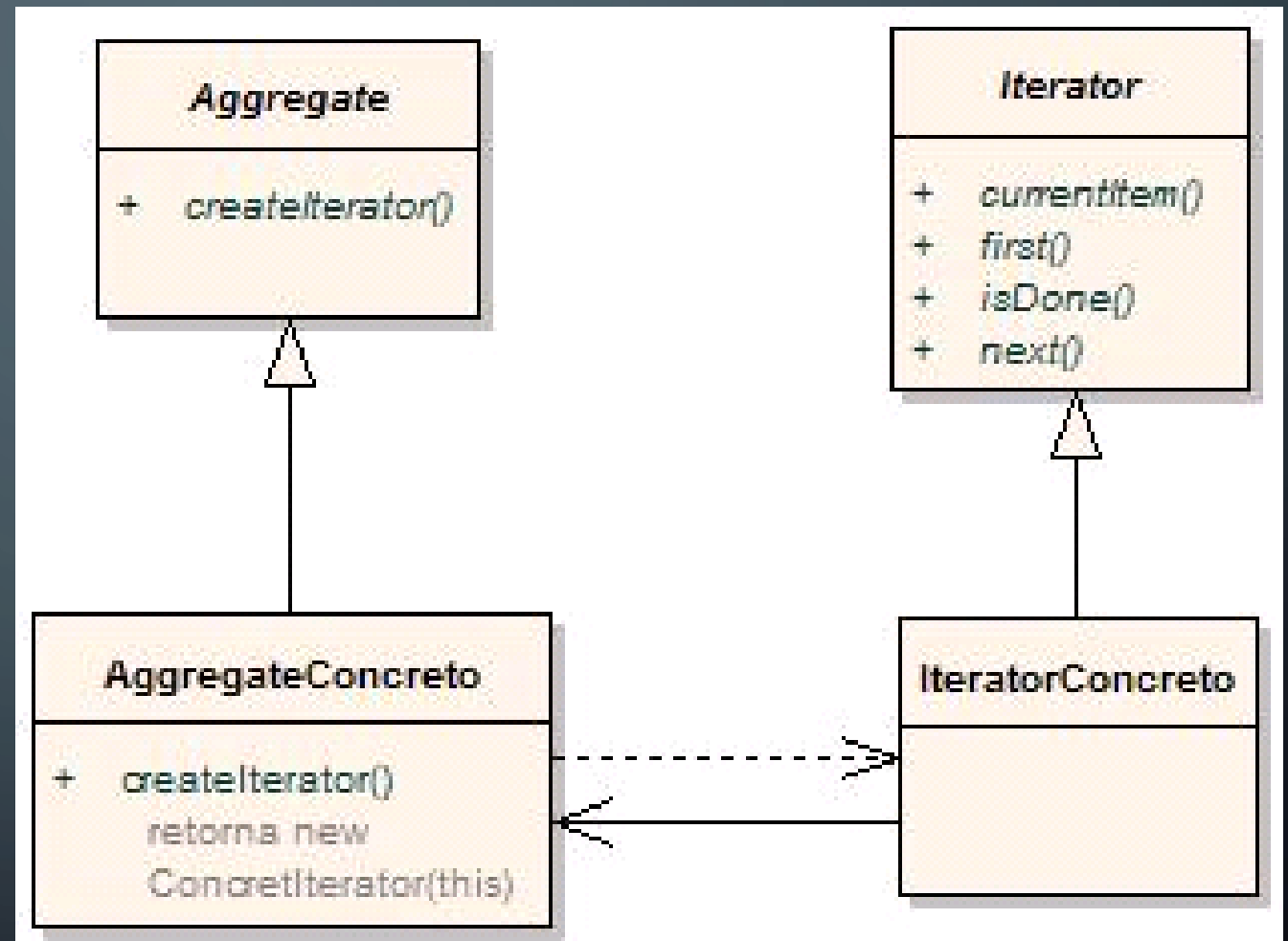
○ Agregado/Aggregate: define una interfaz para crear un objeto iterator.

Iterator: define la interfaz para acceder y recorrer los elementos de un agregado.

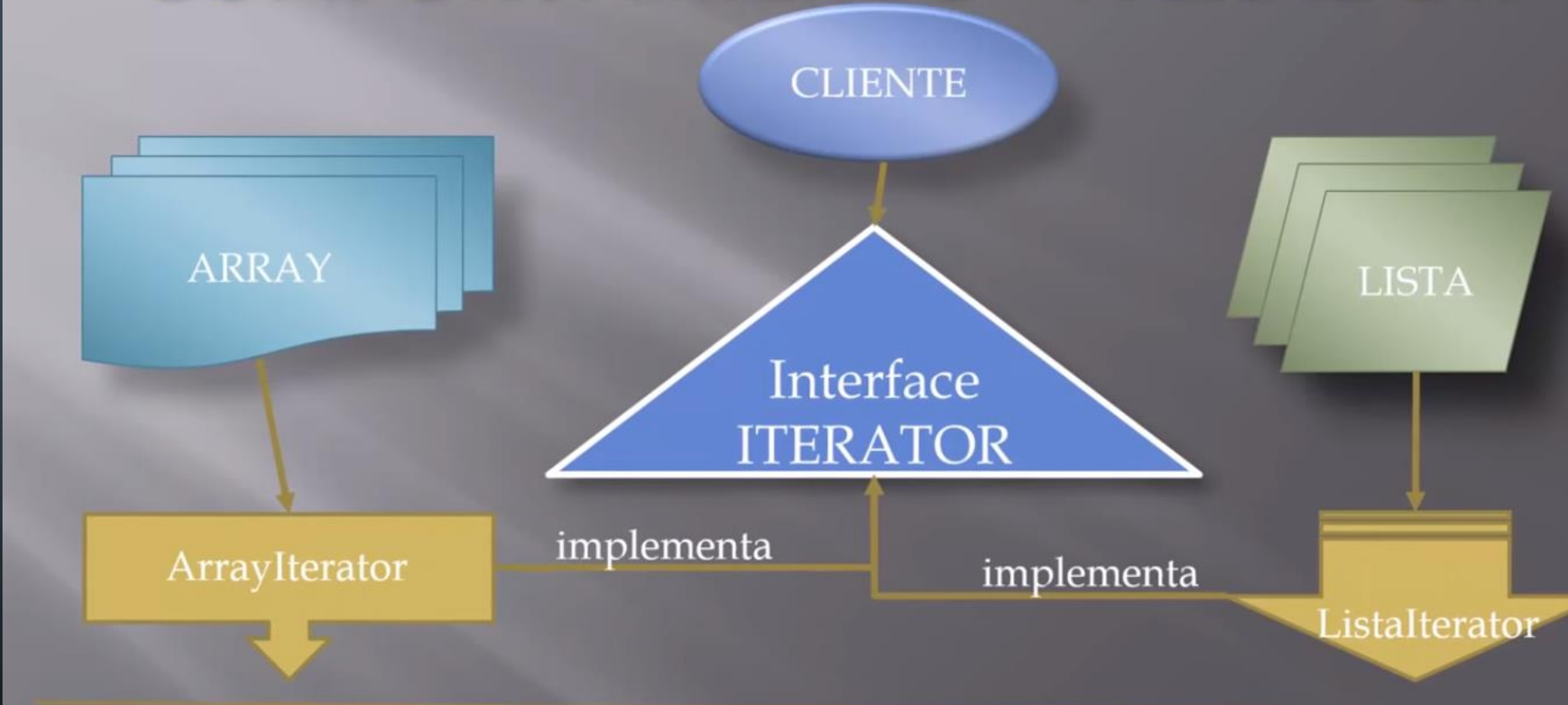
IteradorConcreto: implementa la interfaz del iterador y guarda la posición actual del recorrido en cada momento.

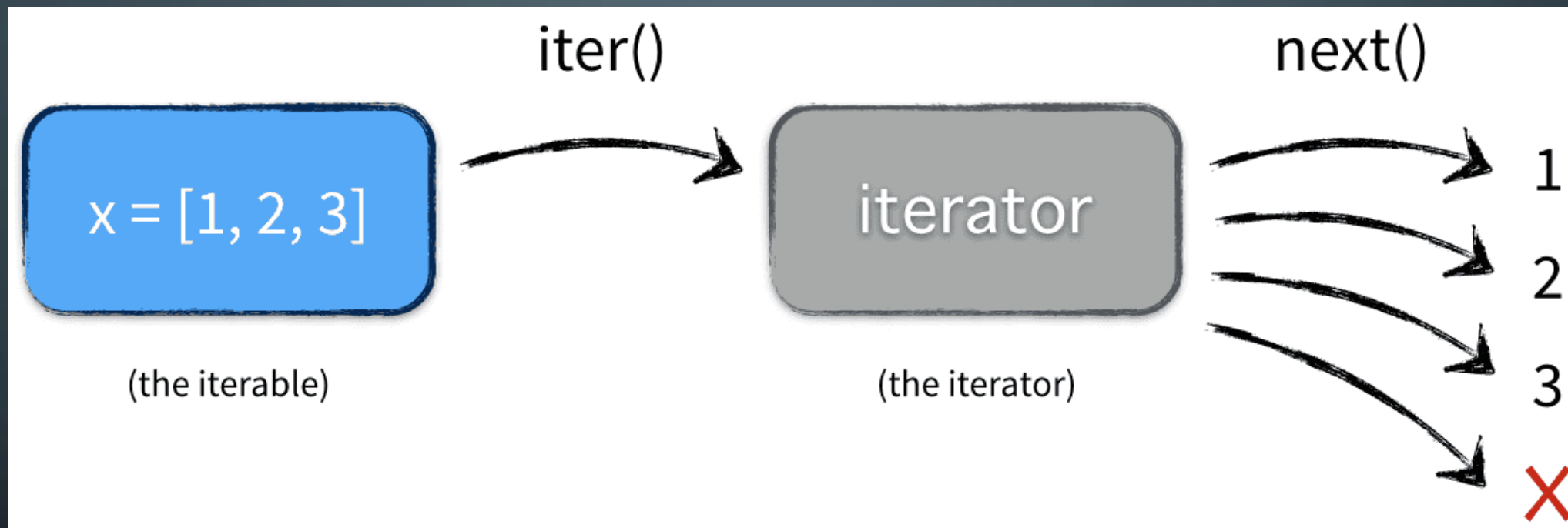
AgregadoConcreto: implementa la interfaz de creación de iteradores devolviendo una instancia del iterador concreto apropiado.

○ Cliente: solicita recorrer una colección y lo hace siguiendo los métodos otorgados por la interfaz Iterator.



# COMPORTAMIENTO - ITERADOR





# Generadores



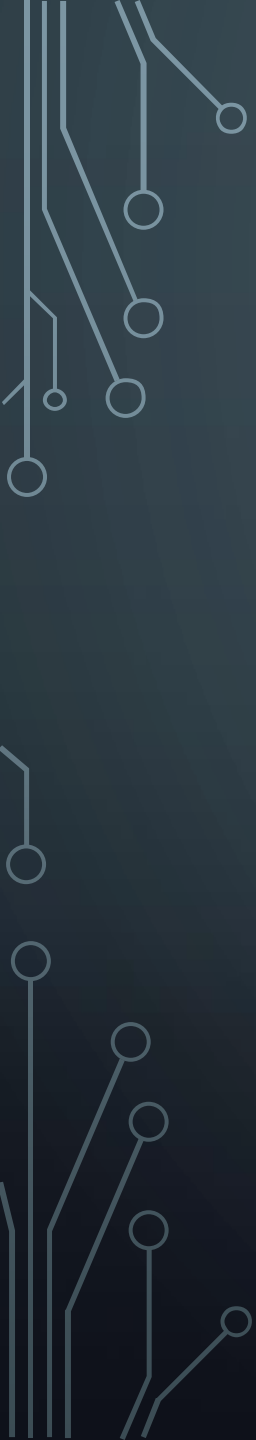


# Que es un Generador

Los generadores son objetos invocables, tales como las funciones y los métodos, los cuales implementan un método `__next__()`.

Extrae valores de una función y se almacenan en objetos iterables (que se pueden recorrer).

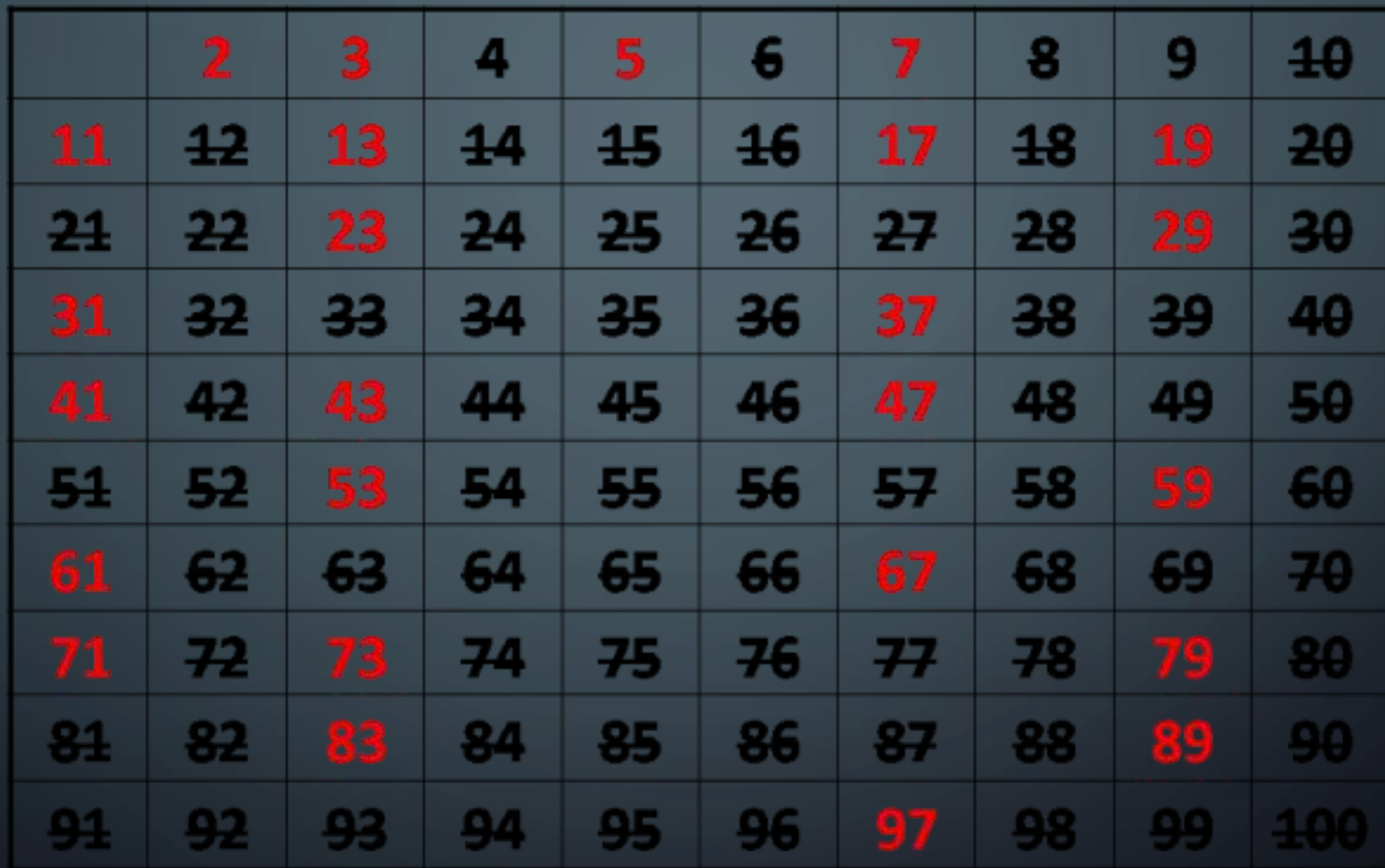
Los valores se almacena 1 en 1.

The image features a dark blue background with white, stylized circuit board traces. These traces are located in the corners and along the edges, with small circles at various points, resembling solder joints or vias. The traces are more prominent in the top-left, bottom-left, and bottom-right corners, with a few smaller ones in the top-right corner.

Son funciones que nos permitirán obtener sus resultados poco a poco. Es decir, cada vez que llamemos a la función nos darán un nuevo resultado. Por ejemplo, una función para generar todos los números primos que cada vez que la llamemos nos devuelva el siguiente número primo. ¿Podemos construir una función que nos devuelva todos los números pares? Esto no es posible si no usamos generadores. Como sabemos los números primos son infinitos(otro ejemplo son los pares).

# ¿PARA QUÉ SIRVEN LOS GENERADORES?

Como el propio nombre indica, para generar datos en tiempo de ejecución. Además también podemos acelerar búsquedas y crear bucles más rápidos.



	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100