🧠 Topic 5: Iterators

🔹 Definition

An **iterator** is an object that allows you to traverse through all the elements of a collection (like lists or tuples), one element at a time.

🔹 Terminologies

- **Iterable:** Object that can return an iterator (like a list).
- **Iterator:** Object that keeps track of the current position in iteration.
- **next():** Built-in function to get the next element.
- **iter():** Function to create an iterator object.

🔹 Example + Outputs

```
numbers = [1, 2, 3]
it = iter(numbers)

print(next(it))
print(next(it))
print(next(it))
```

Output:

1

2

3

Looping automatically uses iterators:

```
for x in [10, 20, 30]:
    print(x)
```

Output:

10

20

30

🔹 **Benefits**

✅ Memory efficient.
✅ Easy to handle sequences.
✅ Automatically used in loops.

🔹 **Challenges**

Solved Challenge 1:

```
nums = [4,5,6]

it = iter(nums)

print(next(it))
```

Solved Challenge 2:

```
text = "Hi"

for ch in text:

    print(ch)
```

Your 18 Challenges:

1. Create iterator from list.
2. Get first two values with next().
3. Loop through tuple.
4. Create iterator from string.
5. Convert range() to iterator.
6. Use next() manually.

7. Create custom iterator class.
8. Stop iteration gracefully.
9. Use iter() in for loop.
10. Iterate through dictionary keys.
11. Iterate through values.
12. Create infinite iterator (careful).
13. Use iterator to sum elements.
14. Count vowels using iterator.
15. Iterate through nested list.
16. Reverse iterate through list.
17. Iterate even indexes only.
18. Iterate until condition met.