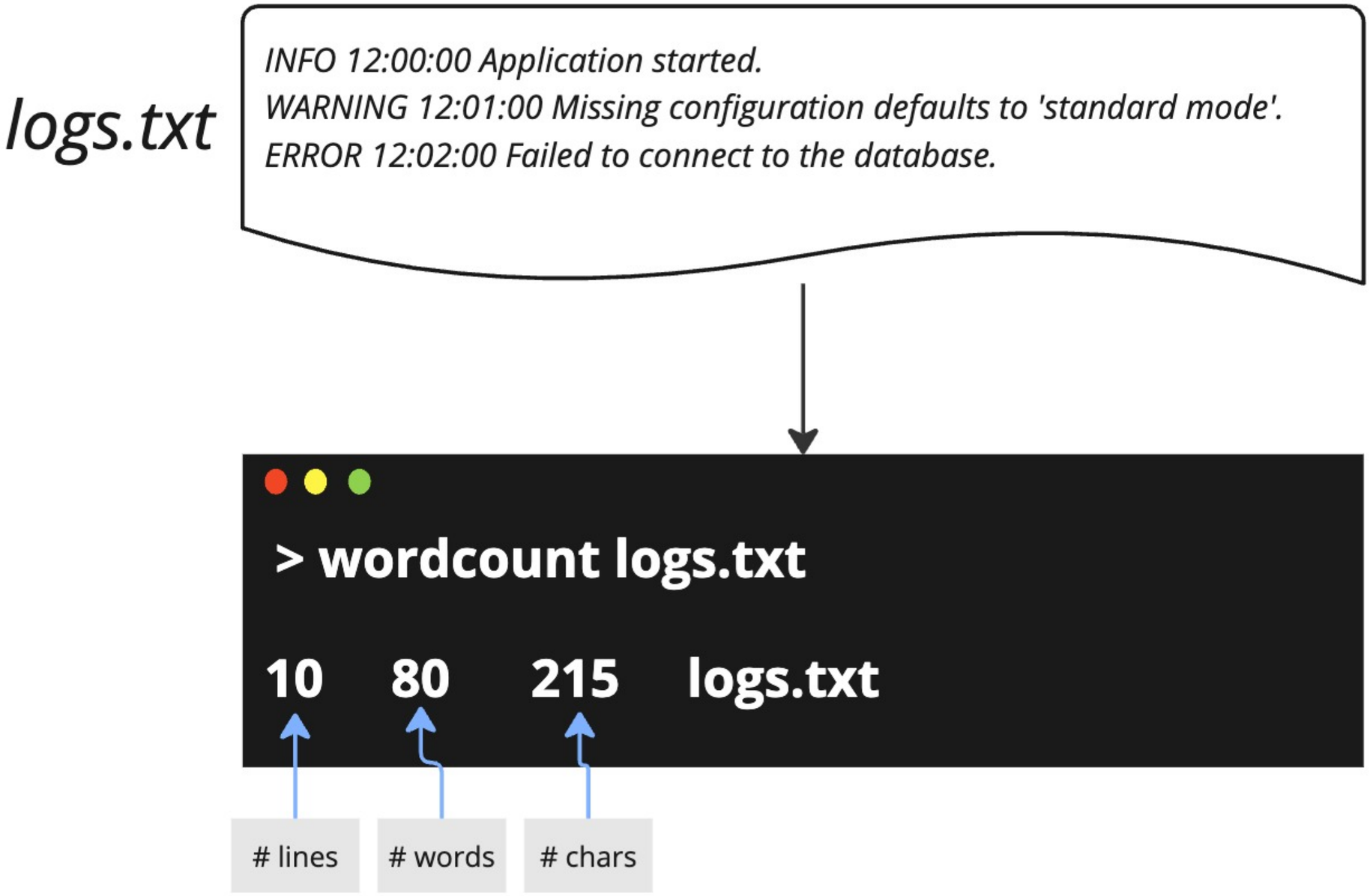

logs.txt
INFO 12:00:00 Application started.
WARNING 12:01:00 Missing configuration defaults to 'standard mode'.
ERROR 12:02:00 Failed to connect to the database.
> wordcount logs.txt
10 80 215 logs.txt
lines
words
chars

```
> wordcount logs.txt  --chars --words

100  450  logs.txt
```

```
> wordcount logs.txt  -cl

29  450  logs.txt
```

```
> wordcount logs.txt

29 100  450  logs.txt
```

```
> wordcount logs.txt example.txt notes.txt

29   100  450  logs.txt
15   90   300  example.txt
37   215   900 notes.txt
```

docs.rs

Crate documentation

Comments with '///' are used as documentation

Tells the compiler to generate extra code related to this struct

'short' adds a flag of '-n'
'long' adds a flag of '--name'

```rust
/// Simple program to greet a person
#[derive(Parser, Debug)]
#[command(version, about, long_about = None)]
struct Args {
    /// Name of the person to greet
    #[arg(short, long)]
    name: String,

    /// Number of times to greet
    #[arg(short, long, default_value_t = 1)]
    count: u8,
}
```

```
> cargo add clap --features derive
```

Crate authors can choose to make some parts of their crate optional

Tell cargo you want to install extra features from a given crate by using the 'features' flag

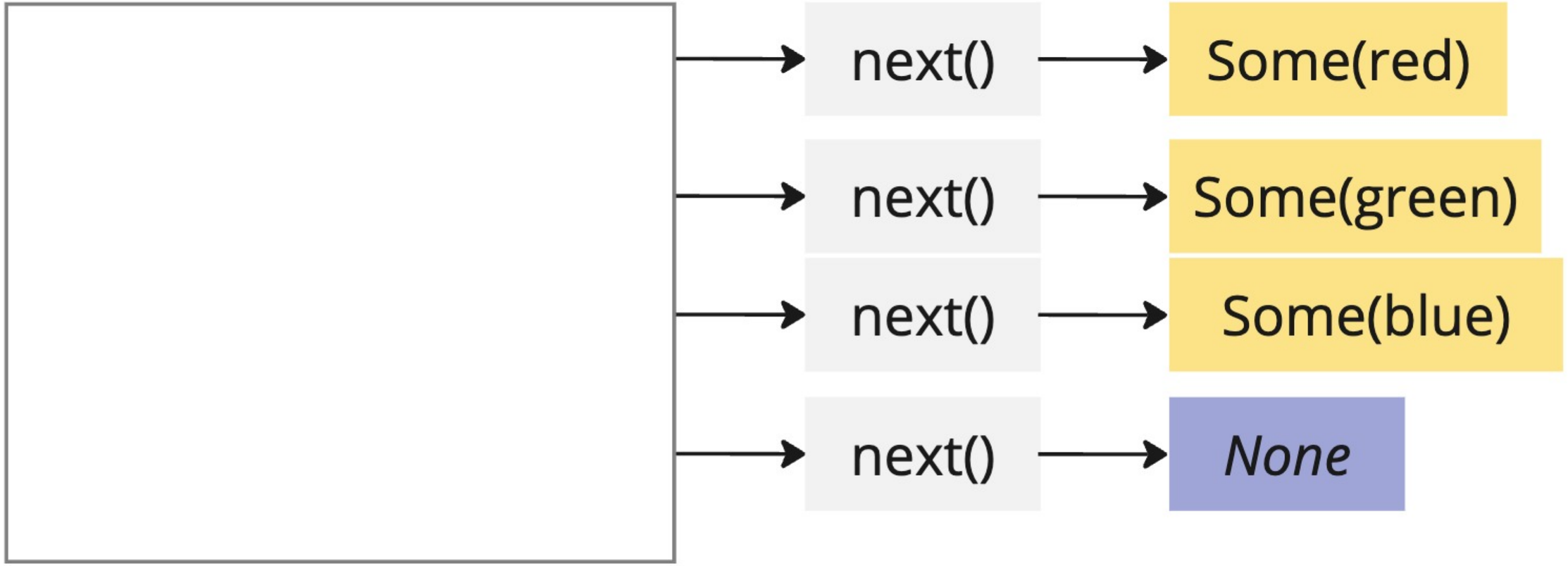
Nothing special about 'derive'. It is what the author decided to call this feature

```rust
env::args().collect::<Vec<String>>();
```

Gives us an ***iterator***

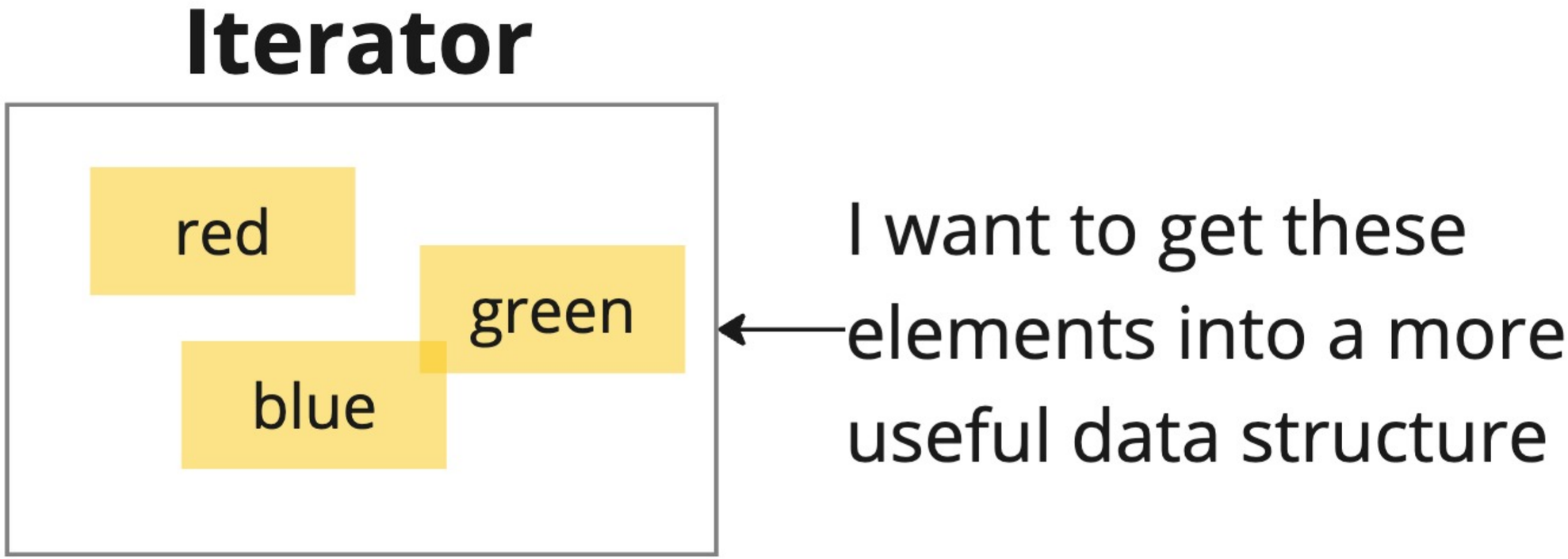
Iterators are the #1 tool we have for working with collections of data

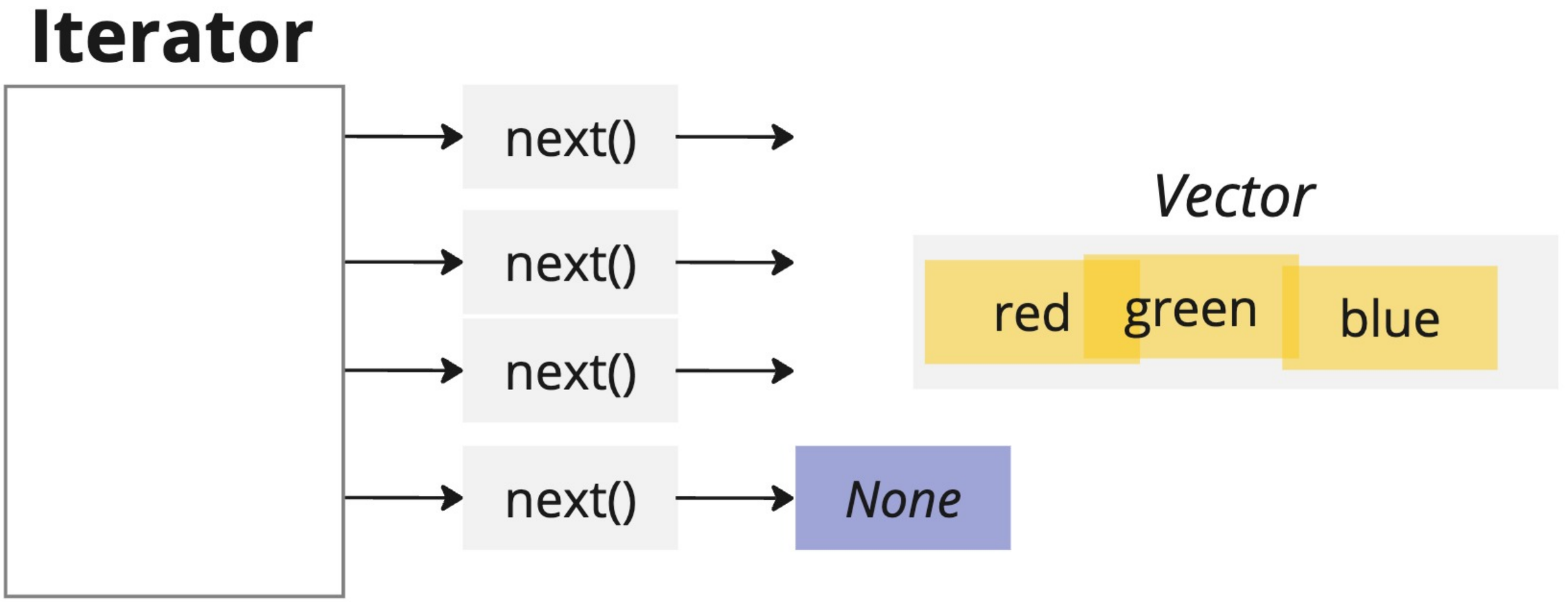
Iterator

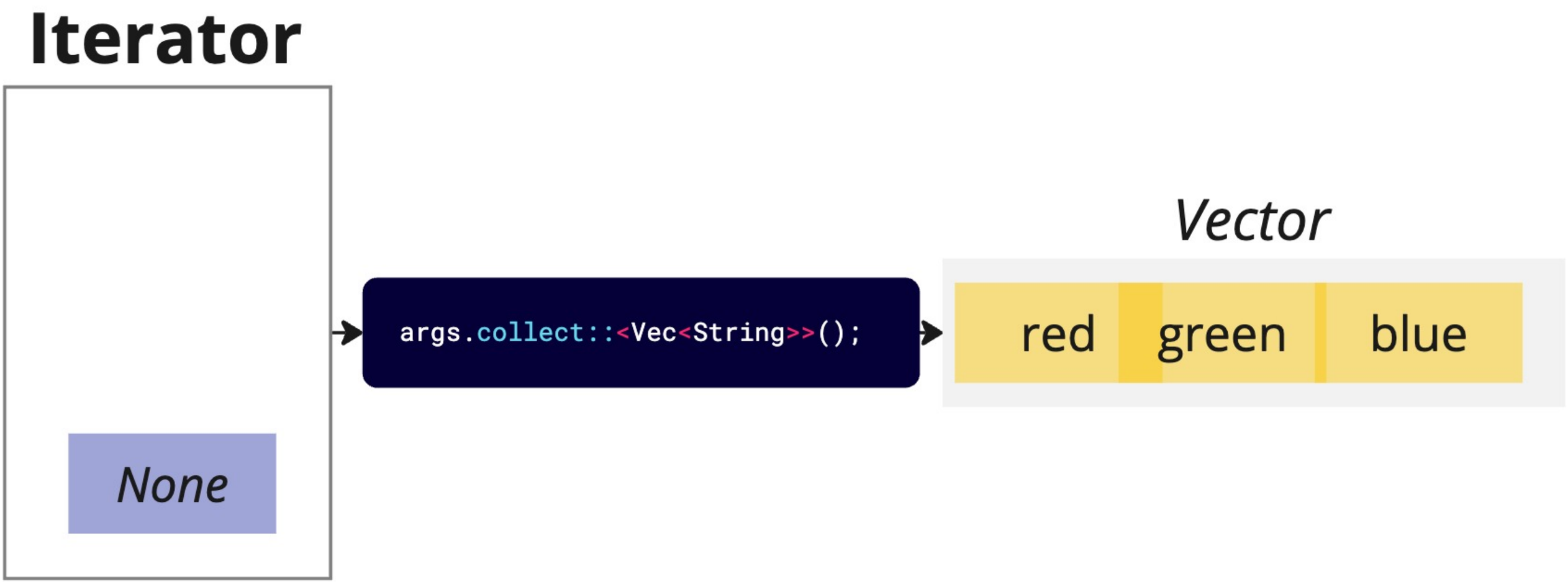


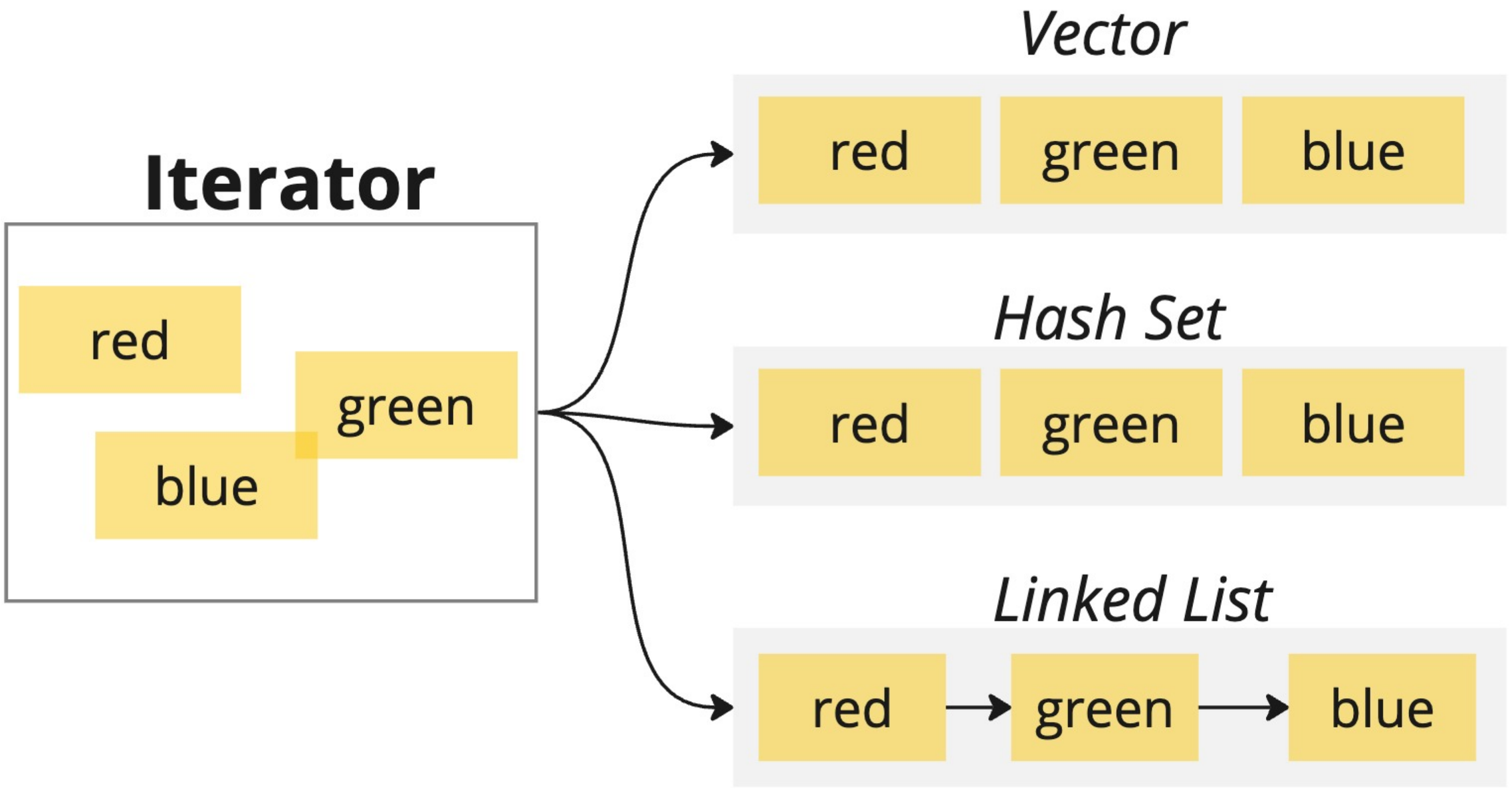
Rust doesn't have 'nil' or 'null' like other languages

It uses 'Some' and 'None' instead. More on this later.


Iterator
red
green
blue
I want to get these elements into a more useful data structure


Iterator
next()
next()
next()
next()
None
Vector
red
green
blue


Iterator
None
args.collect::<Vec<String>>();
Vector
red
green
blue


Iterator
red
green
blue
Vector
red
green
blue
Hash Set
red
green
blue
Linked List
red
green
blue

```
args.collect::<Vec<String>>();
```

The type you put right here is going to ***change how your code works***

Very different from Typescript or (typed) Python!

```rust
let args = env::args().collect::<Vec<String>>();

let mut iter = args.iter();
```