```rust
fn main() {
    timely::execute_from_args(std::env::args(), |worker| {

        let input = worker.new_input();
        let probe = worker.dataflow(|scope| {
            input.to_stream(scope)
                 .exchange(|&x| x)
                 .inspect(|x| println!("{}", x))
                 .probe()
        });

        for round in 0..10 {
            input.send(round * round);
            input.advance_to(round + 1);
            while probe.less_than(input.time()) {
                worker.step();
            }
        }
    });
}
```

Drive the dataflow by supplying inputs and running until cleared.

"What times might `probe` still see?"
Information about distributed state.

```rust
fn main() {
    timely::execute_from_args(std::env::args(), |worker| {

        let input = worker.new_input();
        let probe = worker.dataflow(|scope| {
            input.to_stream(scope)
                 .exchange(|&x| x)
                 .inspect(|x| println!("{}", x))
                 .probe()
        });

        for round in 0..10 {
            input.send(round * round);
            input.advance_to(round + 1);
            while probe.less_than(input.time()) {
                worker.step();
            }
        }
    });
}
```