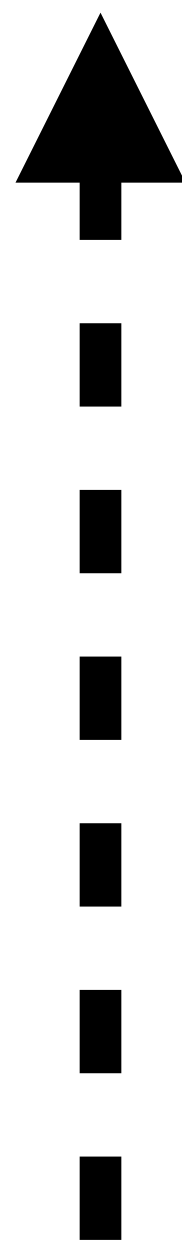
```
.iterate(|cycle| {
    cycle.filter(|&x| x > 0)
         .map(|x| x - 1)
         .exchange(|&x| x)
})
```

```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)


                .probe()
        });

        for round in 0..10 {
            input.send(round * round);
            input.advance_to(round + 1);
            while probe.less_than(input.time() - 2) {
                worker.step();
            }
        }
    });
}
```



concurrent iterations

better utilizations!

```
// convenient!
stream.map(logic);
```

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

        let input = worker.new_input();
        let probe = worker.dataflow(|scope| {
            input.to_stream(scope)
                                  <------
                .probe()
        });

        for round in 0..10 {
            input.send(round * round);
            input.advance_to(round + 1);
            while probe.less_than(input.time() - 2) {
                worker.step();
            }
        }
    });
}
```

```
.iterate(|cycle| {
    cycle.filter(|&x| x > 0)
         .map(|x| x - 1)
         .exchange(|&x| x)
})
```