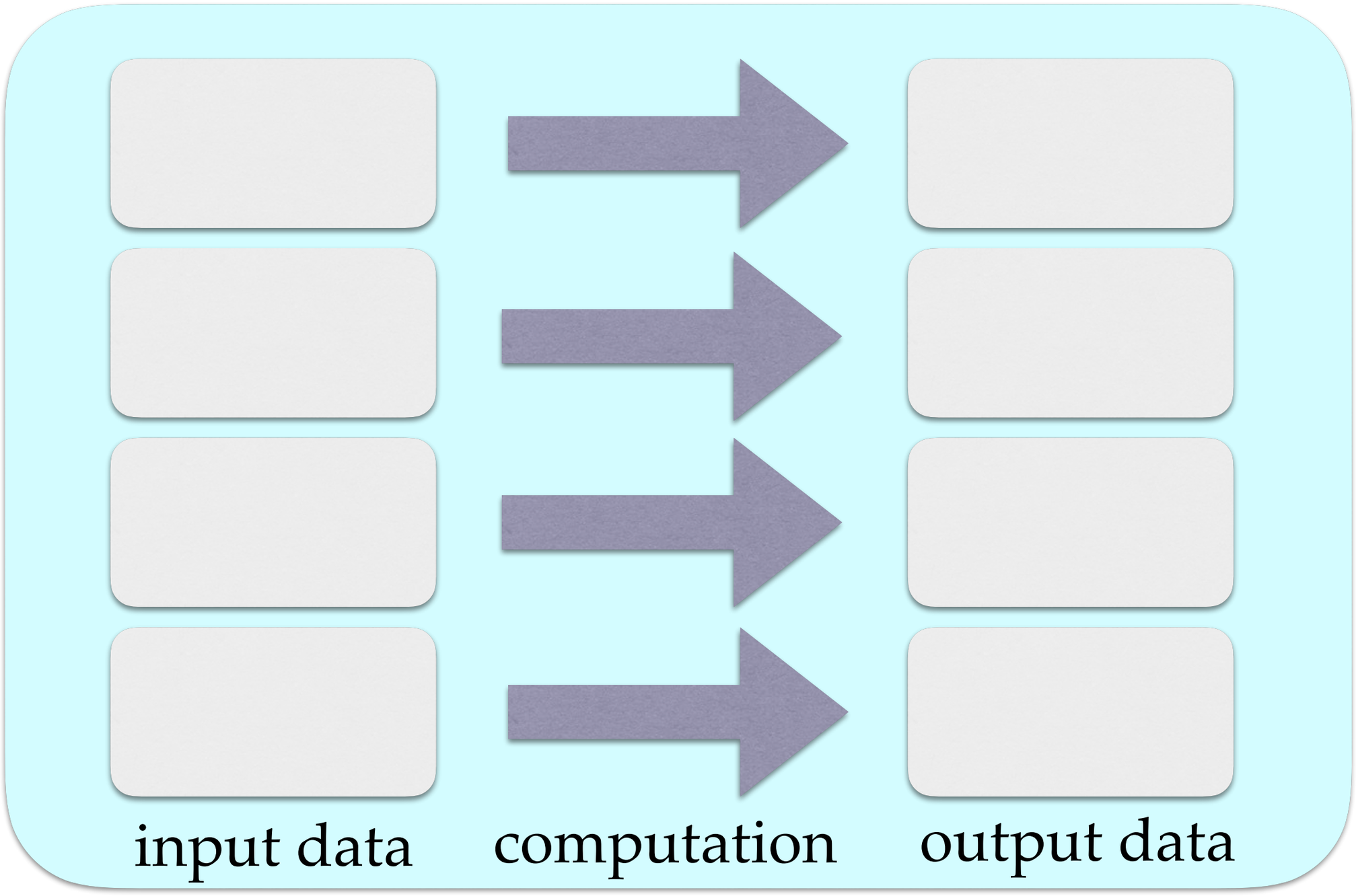
Parallelism in Python

Simon Brown

Stochastic Solutions Limited

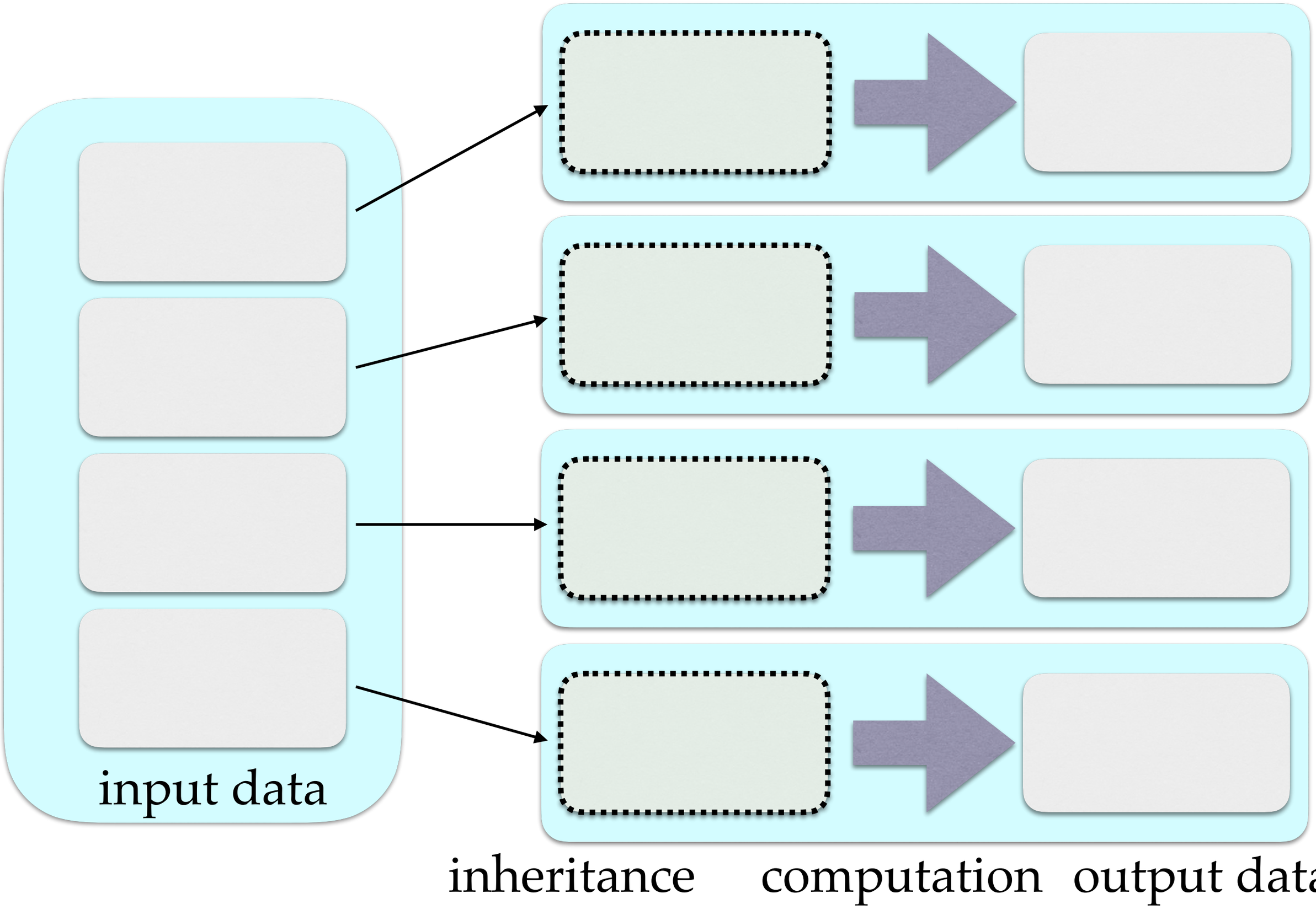
Threads



Threads

```
import threading

inputs = build_input_data()
outputs = [None] * N

def f(i):
    outputs[i] = compute_something(inputs[i])

threads = []
for i in range(N):
    thread = threading.Thread(target=f, args=(i,))
    thread.start()
    threads.append(thread)

for i in range(N):
    threads[i].join()
```

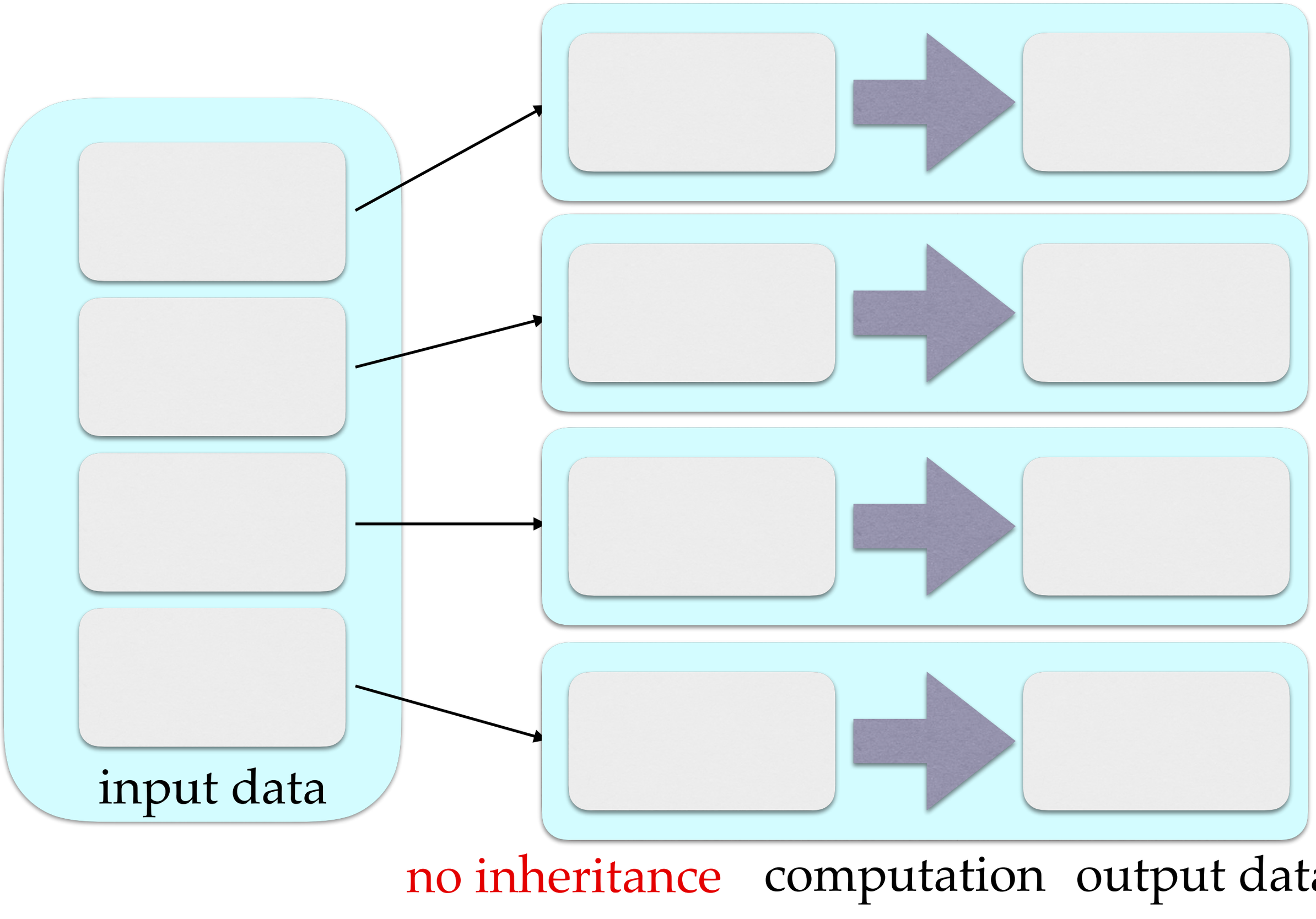
Subprocesses (Linux & MacOS)



Subprocesses (Linux & MacOS)

```
import multiprocessing

inputs = get_input_data()
outputs = [None] * N

def f(i):
    outputs[i] = compute_something(inputs[i])

proc = []
for i in range(N):
    proc = multiprocessing.Process(target=f, args=(i,))
    proc.start()
    procs.append(proc)

for i in range(N):
    outputs[i] = fetch_result_from_subprocess(i)
    procs[i].join()
```

Subprocesses (Windows)



Other libraries and modules

Symmetric Multiprocessing

- dispy
- forkfun
- pprocess
- processing
- PyCSP
- PyMP
- Ray

Cluster Computing

- disco
- DistributedPython
- job_stream
- jug
- mpi4py
- pp
- superpy