Unlocking the power of

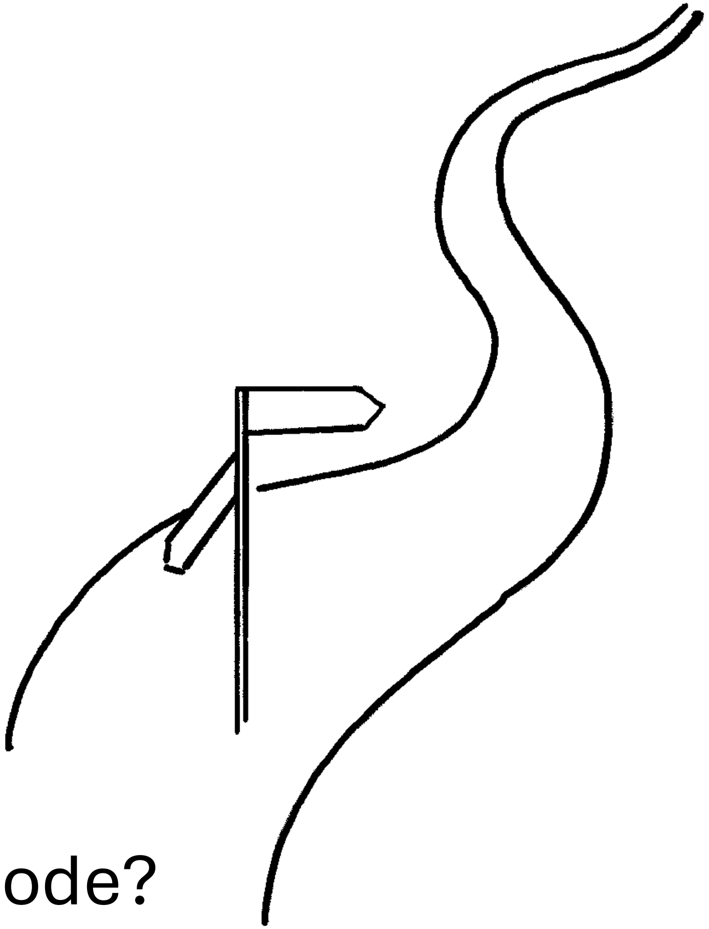
xarray

&

dask

Summary – Aim is to write faster/better code

- Why should you listen to me?
- **Why** bother?
- What does **fast/good** code mean?
- How can xarray/dask help?
  - **“Lazy”** computing
  - Thinking about your **underlying** code
  - The voodoo that is **chunking**
- But how do I **translate** my function into xarray code?

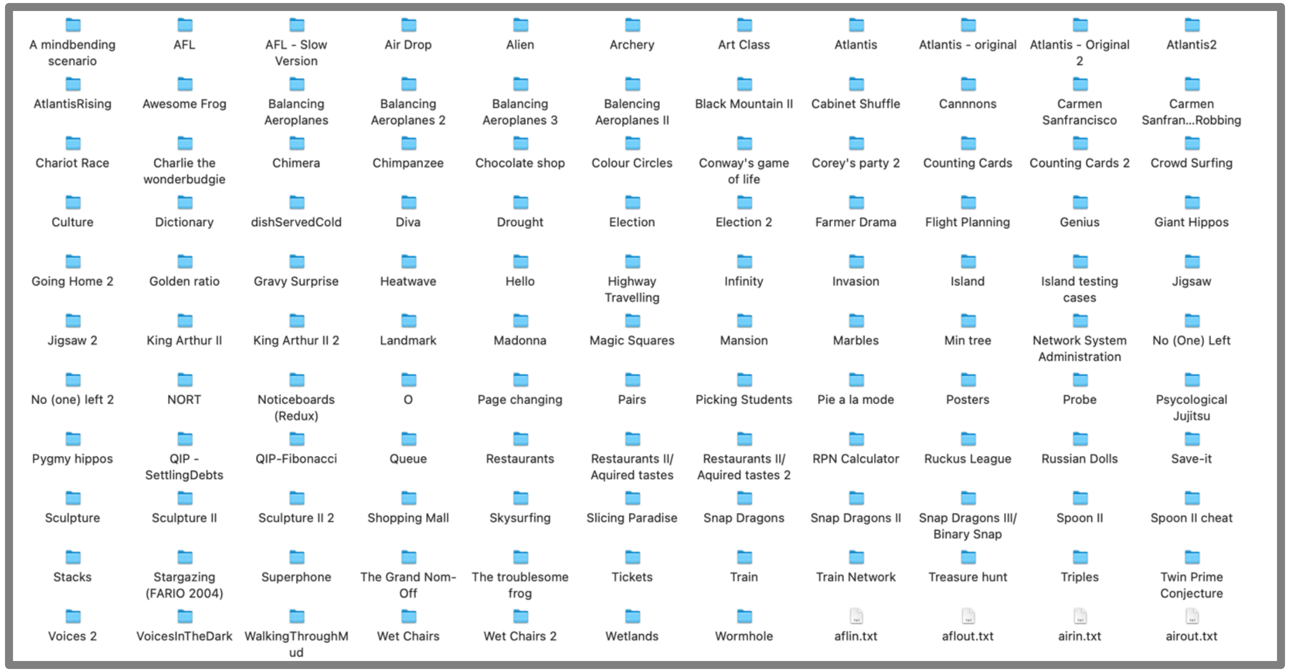


3 key moments in my journey

- Young ICT explorers (grade 6)

3 key moments in my journey

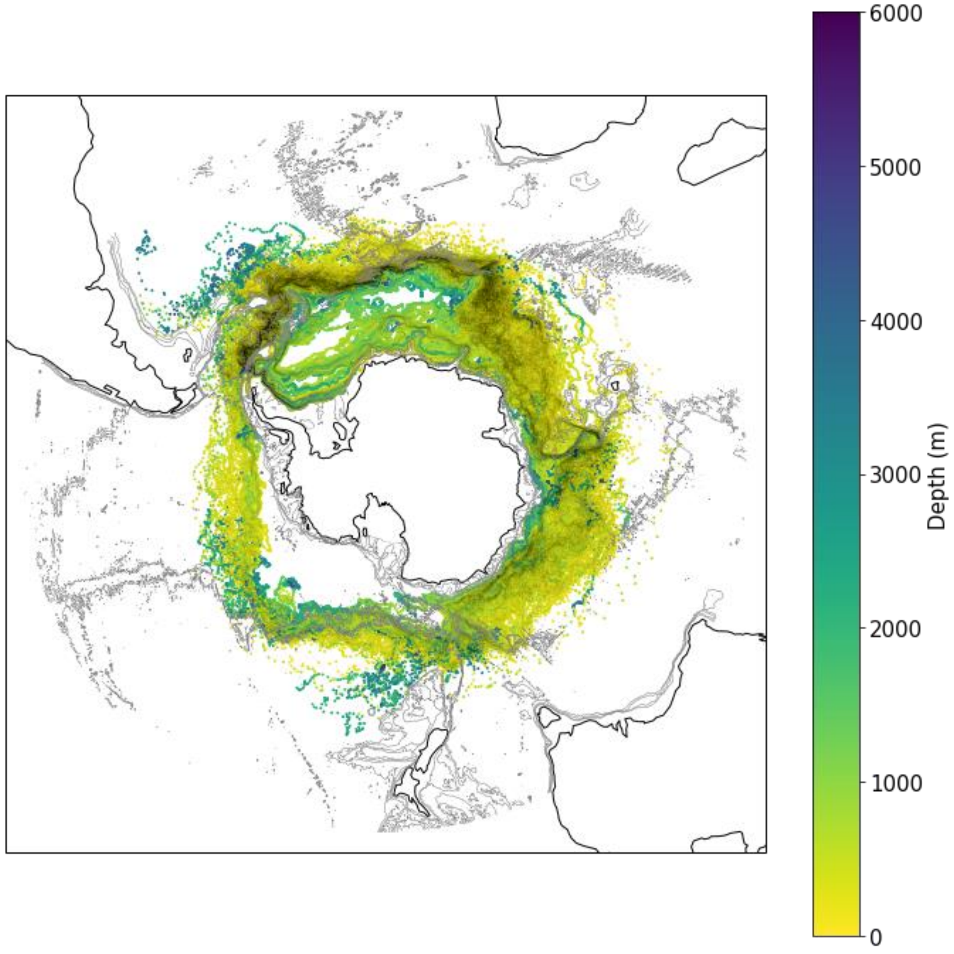
- Young ICT explorers (grade 6)
- Australian Informatics summer school (grade 11)



A mindbending scenario	AFL	AFL - Slow Version	Air Drop	Alien	Archery	Art Class	Atlantis	Atlantis - original	Atlantis - Original 2	Atlantis2
AtlantisRising	Awesome Frog	Balancing Aeroplanes	Balancing Aeroplanes 2	Balancing Aeroplanes 3	Balencing Aeroplanes II	Black Mountain II	Cabinet Shuffle	Cannnons	Carmen Sanfrancisco	Carmen Sanfran...Robbing
Chariot Race	Charlie the wonderbudgie	Chimera	Chimpanzee	Chocolate shop	Colour Circles	Conway's game of life	Corey's party 2	Counting Cards	Counting Cards 2	Crowd Surfing
Culture	Dictionary	dishServedCold	Diva	Drought	Election	Election 2	Farmer Drama	Flight Planning	Genius	Giant Hippos
Going Home 2	Golden ratio	Gravy Surprise	Heatwave	Hello	Highway Travelling	Infinity	Invasion	Island	Island testing cases	Jigsaw
Jigsaw 2	King Arthur II	King Arthur II 2	Landmark	Madonna	Magic Squares	Mansion	Marbles	Min tree	Network System Administration	No (One) Left
No (one) left 2	NORT	Noticeboards (Redux)	O	Page changing	Pairs	Picking Students	Pie a la mode	Posters	Probe	Psycological Jujitsu
Pygmy hippos	QIP - SettlingDebts	QIP-Fibonacci	Queue	Restaurants	Restaurants II/ Aquired tastes	Restaurants II/ Aquired tastes 2	RPN Calculator	Ruckus League	Russian Dolls	Save-it
Sculpture	Sculpture II	Sculpture II 2	Shopping Mall	Skysurfing	Slicing Paradise	Snap Dragons	Snap Dragons II	Snap Dragons III/ Binary Snap	Spoon II	Spoon II cheat
Stacks	Stargazing (FARIO 2004)	Superphone	The Grand Nom-Off	The troublesome frog	Tickets	Train	Train Network	Treasure hunt	Triples	Twin Prime Conjecture
Voices 2	VoicesInTheDark	WalkingThroughMud	Wet Chairs	Wet Chairs 2	Wetlands	Wormhole	aflin.txt	aflout.txt	airin.txt	airout.txt

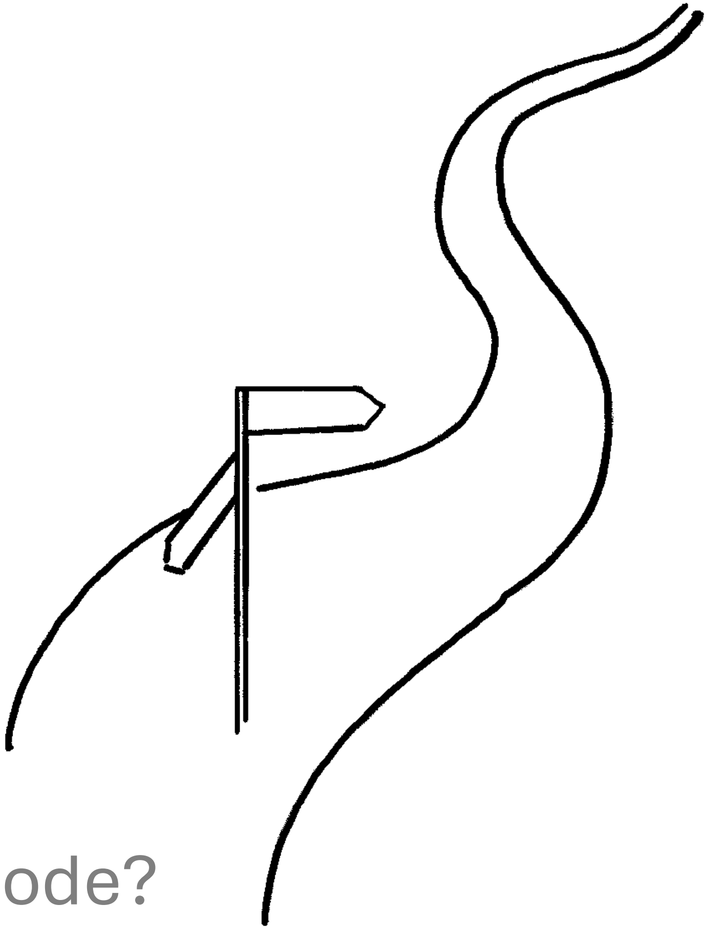
3 key moments in my journey

- Young ICT explorers (grade 6)
- Australian Informatics summer school (grade 11)
- AABW project in undergraduate (2nd year)



Summary – Aim is to write faster/better code

- Why should you listen to me?
- **Why** bother?
- What does **fast/good** code mean?
- How can xarray/dask help?
  - **“Lazy”** computing
  - Thinking about your **underlying** code
  - The voodoo that is **chunking**
- But how do I **translate** my function into xarray code?



Why bother writing good code?

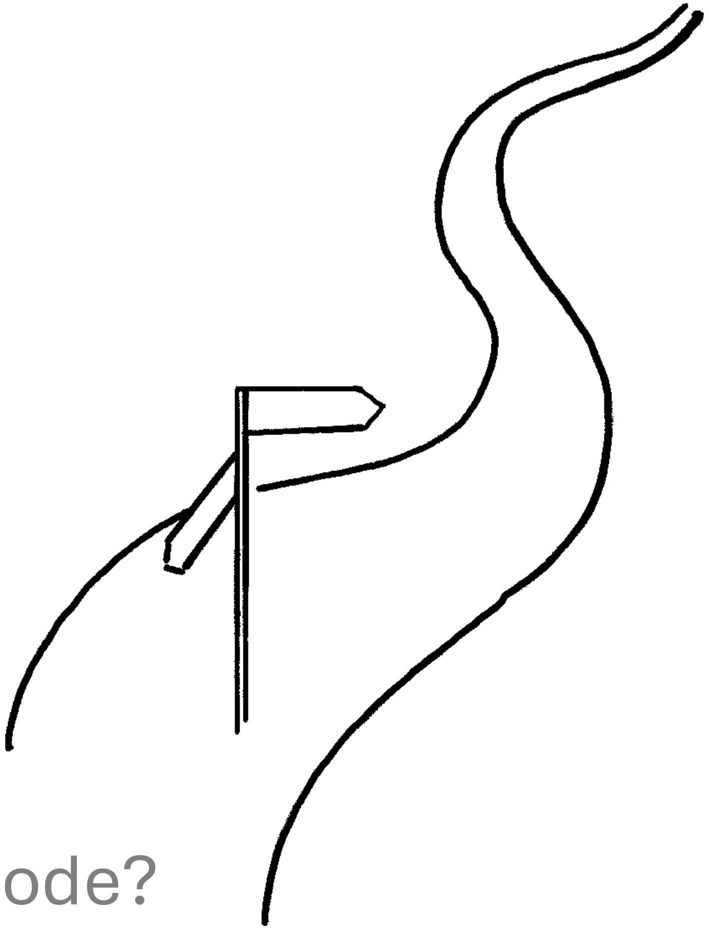
Save time

Do more

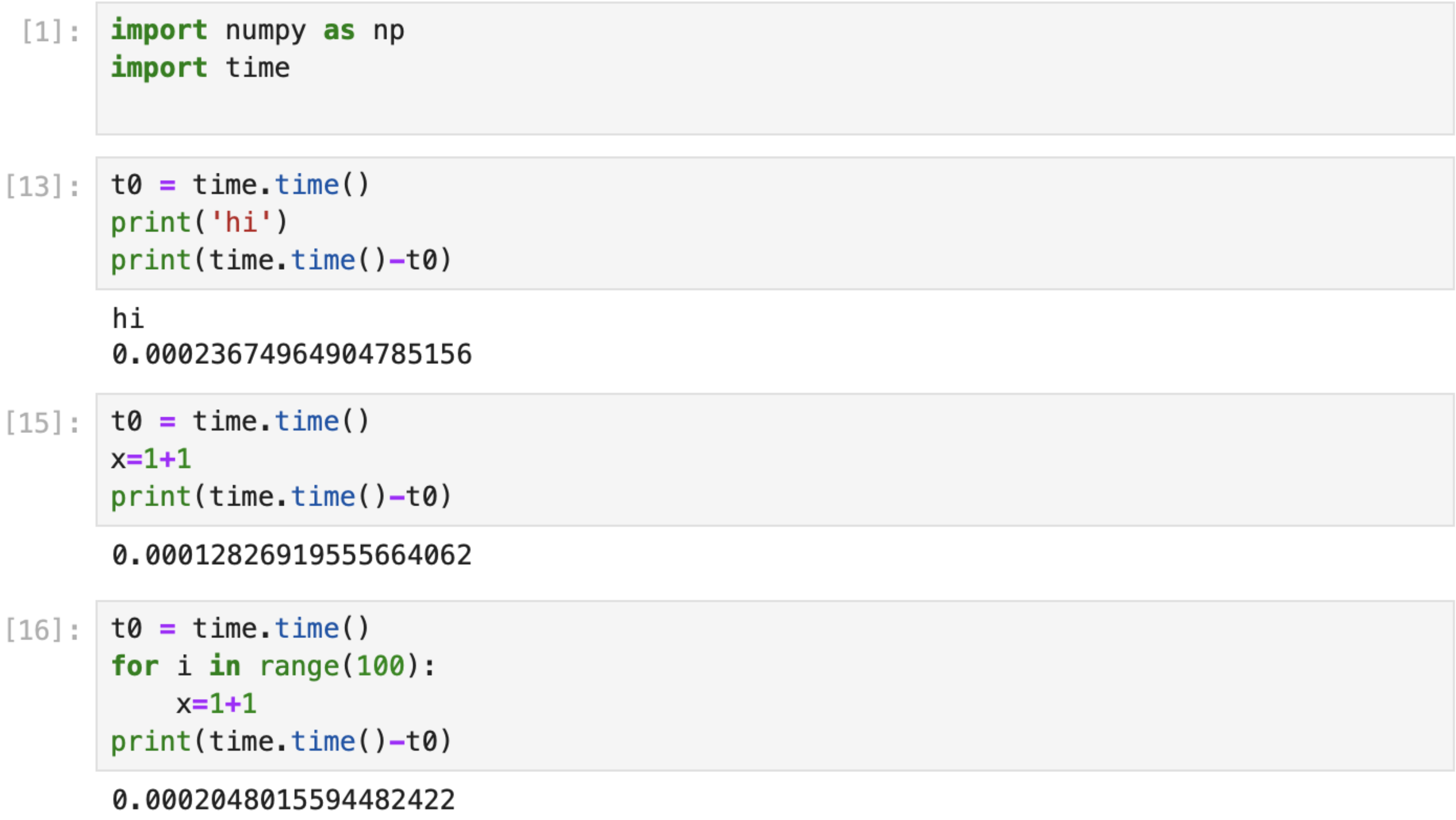
Make something work at all

Summary – Aim is to write faster/better code

- Why should you listen to me?
- **Why** bother?
- What does **fast/good** code mean?
- How can xarray/dask help?
  - **“Lazy”** computing
  - Thinking about your **underlying** code
  - The voodoo that is **chunking**
- But how do I **translate** my function into xarray code?



Timing examples [switch to notebook]



```
[1]: import numpy as np
     import time
```

```
[13]: t0 = time.time()
      print('hi')
      print(time.time()-t0)
```

```
hi
0.00023674964904785156
```

```
[15]: t0 = time.time()
      x=1+1
      print(time.time()-t0)
```

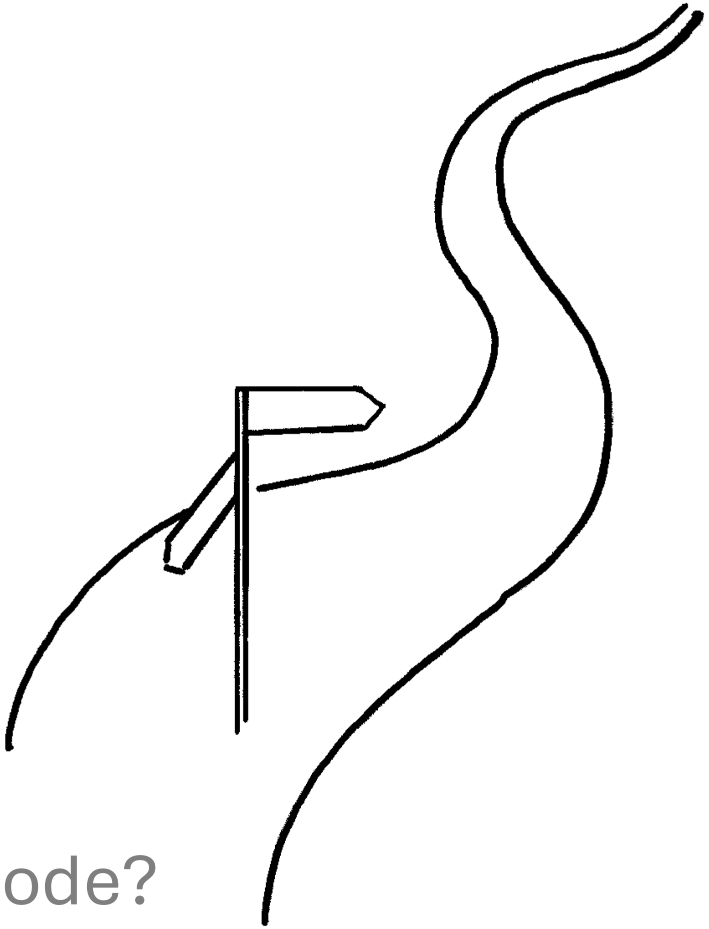
```
0.00012826919555664062
```

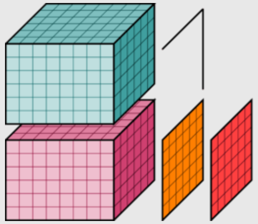
```
[16]: t0 = time.time()
      for i in range(100):
          x=1+1
      print(time.time()-t0)
```

```
0.0002048015594482422
```

Summary – Aim is to write faster/better code

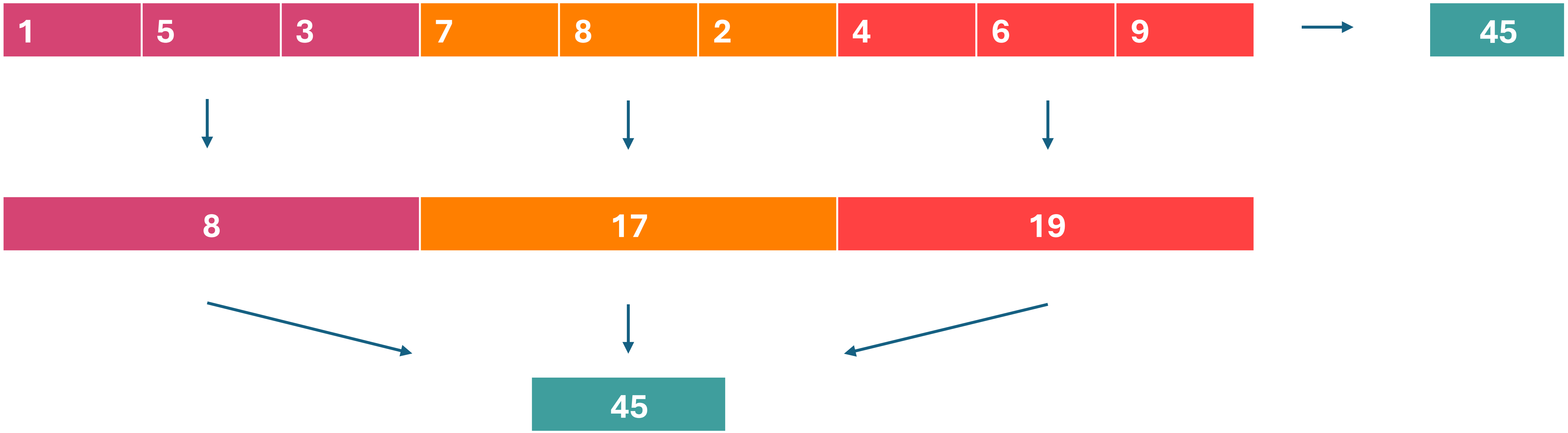
- Why should you listen to me?
- **Why** bother?
- What does **fast/good** code mean?
- How can **xarray/dask** help?
  - **“Lazy”** computing
  - Thinking about your **underlying** code
  - The voodoo that is **chunking**
- But how do I **translate** my function into xarray code?





Lazy computing

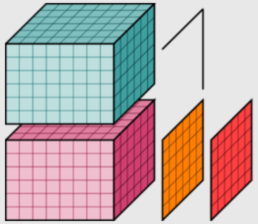
- Lazy ≈ “do it later”
- Smaller pieces
- Example: **summing** in different chunks



1 5 3 7 8 2 4 6 9 → 45

8 17 19

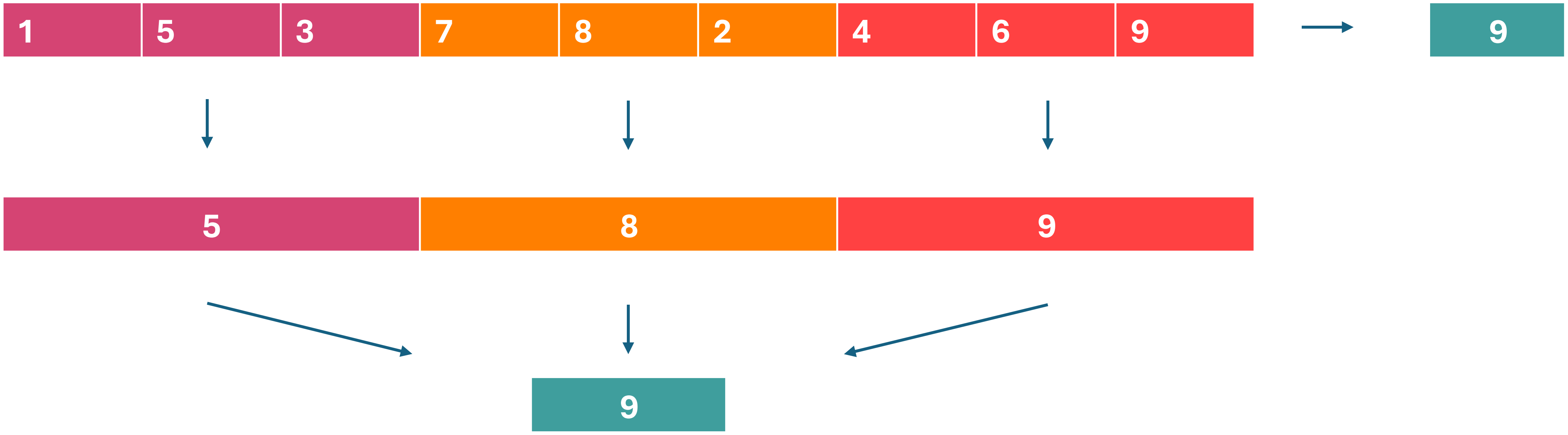
45



Lazy computing

- Lazy ≈ “do it later”
- Smaller pieces

- Example: **finding maximum** in different chunks

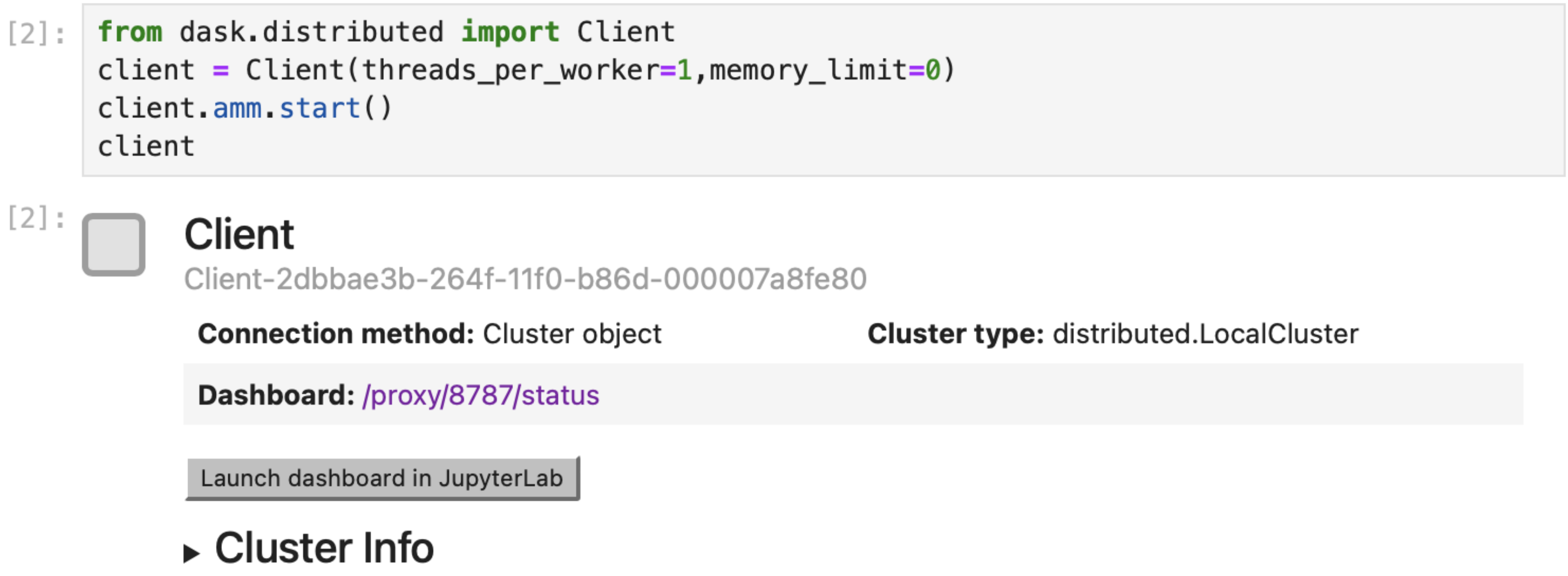


1 5 3 7 8 2 4 6 9 → 9

5 8 9

9

[switch to notebook]



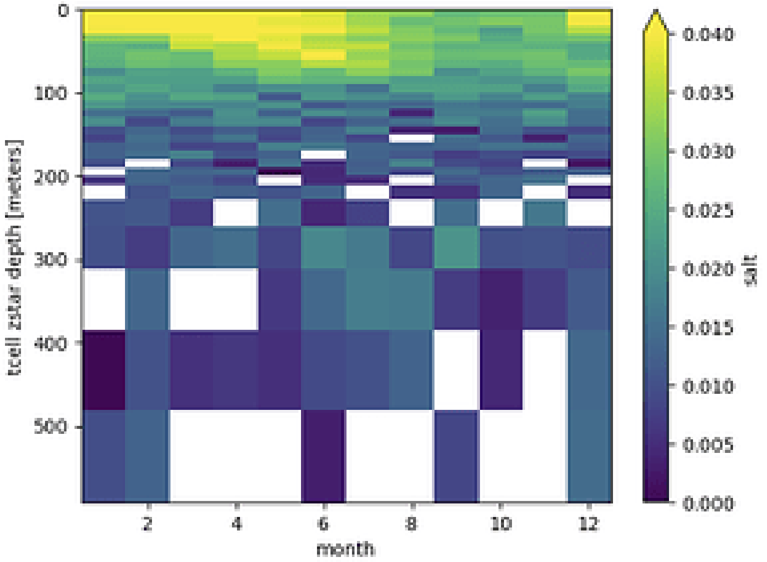
```
[2]: from dask.distributed import Client
     client = Client(threads_per_worker=1,memory_limit=0)
     client.amm.start()
     client
```

```
[2]:
Client
Client-2dbbae3b-264f-11f0-b86d-000007a8fe80

Connection method: Cluster object          Cluster type: distributed.LocalCluster
Dashboard: /proxy/8787/status
```

Underlying code: an example

Salinity standard deviation



“the xarray code is wrong” – an interpretation I heard many times

Variance:

$$\sigma^2 = \frac{1}{n}\sum_i (x_i - \bar{x})^2$$

or

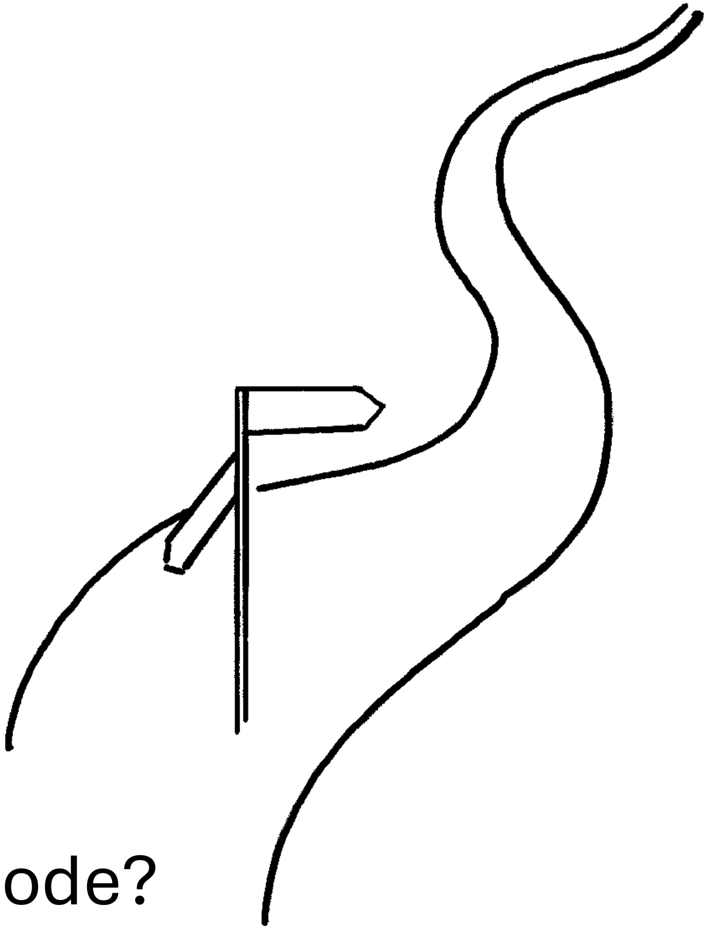
$$\sigma^2 = \frac{1}{n}\sum_i (x_i^2) - \frac{1}{n}\sum_i (\bar{x}^2)$$

The black magic of chunking

- Match the chunks on file
- Either chunk *when opening* or *after loading*
- Double check your assumptions
  - Test in small batches
  - Keep the task stream open
- Dask uses ~2x the RAM of a dataset
- Sometimes dask overhead is too much to help
- Sometimes it’s faster to just shove small pieces into a for loop

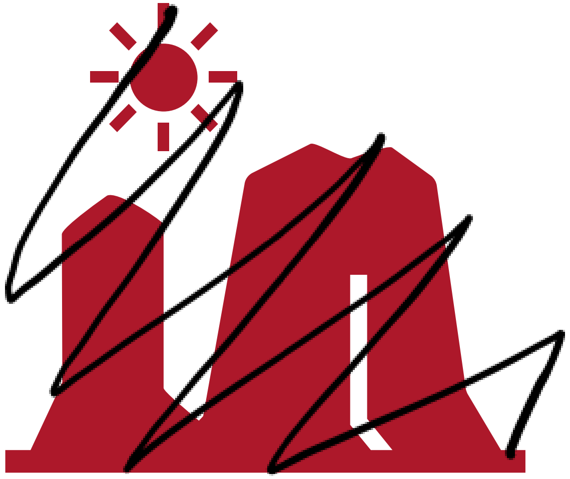
Summary – Aim is to write faster/better code

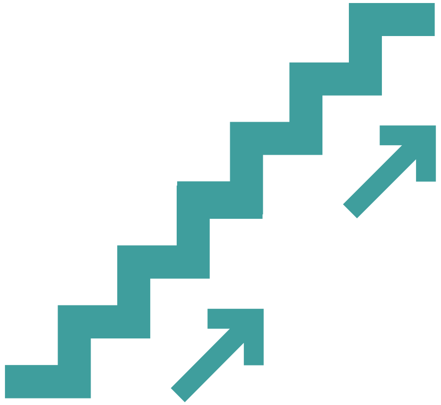
- Why should you listen to me?
- **Why** bother?
- What does **fast/good** code mean?
- How can xarray/dask help?
  - **“Lazy”** computing
  - Thinking about your **underlying** code
  - The voodoo that is **chunking**
- But how do I **translate** my function into xarray code?



Writing code to work well with xarray/dask

A completely different way of thinking





Rethinking code: your tools

Can you break your problem into these pieces?

- Any maths
  `(multiply, add, np.log(), ds.cumsum(), …)`
- Dimension reduction by maths
  `(ds.sum(), ds.std(), …)`
- Dimension reduction by picking one
  `(ds.max(), ds.argmin(),ds.sel(), …)`
- Moving stuff
  `(ds.shift(), ds.coarsen(), …)`
- Grabbing strange pieces
  `(ds.where(), ds.groupby(), …)`

Full list here: https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html

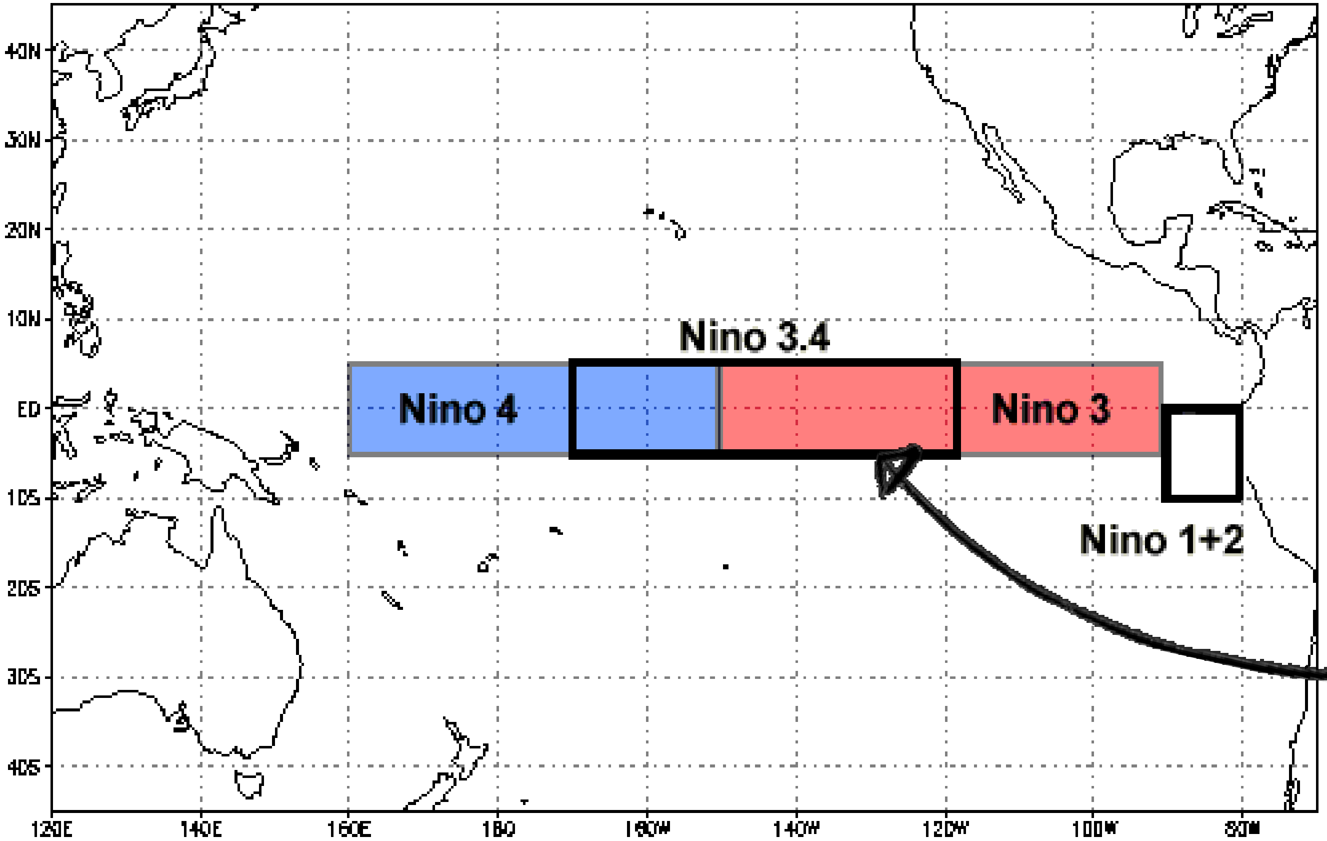
Rethinking code: examples

1. NINO34 index
2. Weighted rolling average
3. 20°C isotherm
4. Model analogue forecasts

?

Example 1: NINO34 index

- I have sea surface temperature (SST) from a model



Average that box

Example 1: NINO34 index

- I have sea surface temperature (SST) from a model
- I want to calculate the average SST within the NINO34 region (5°S-5°N, 190°-240°E) for every time

Example 1: NINO34 index

- I have sea surface temperature (SST) from a model
- I want to calculate the average SST within the NINO34 region (5°S-5°N, 190°-240°E) for every time

1 Get data within the NINO34 region

```
nino34_sst = sst.sel(lat=slice(-5,5),lon=slice(190,240))
nino34_sst = sst.where((sst.lat <-5)&(sst.lat <-5)
       &(sst.lon >190)&(sst.lon<240))
```

2 Average in latitude and longitude space

```
nino34_sst.mean((”lat”,”lon”))
```

Example 2: Rolling weighted average through time

$$\text{Weighted rolling average} = \frac{rolling\ sum\ of\ (weights \times data)}{rolling\ sum\ of\ weights}$$

Example 2: Rolling weighted average through time

$$\text{Weighted rolling average} = \frac{rolling\ sum\ of\ (weights \times data)}{rolling\ sum\ of\ weights}$$

1 Multiply data by weights

2 Find rolling sum

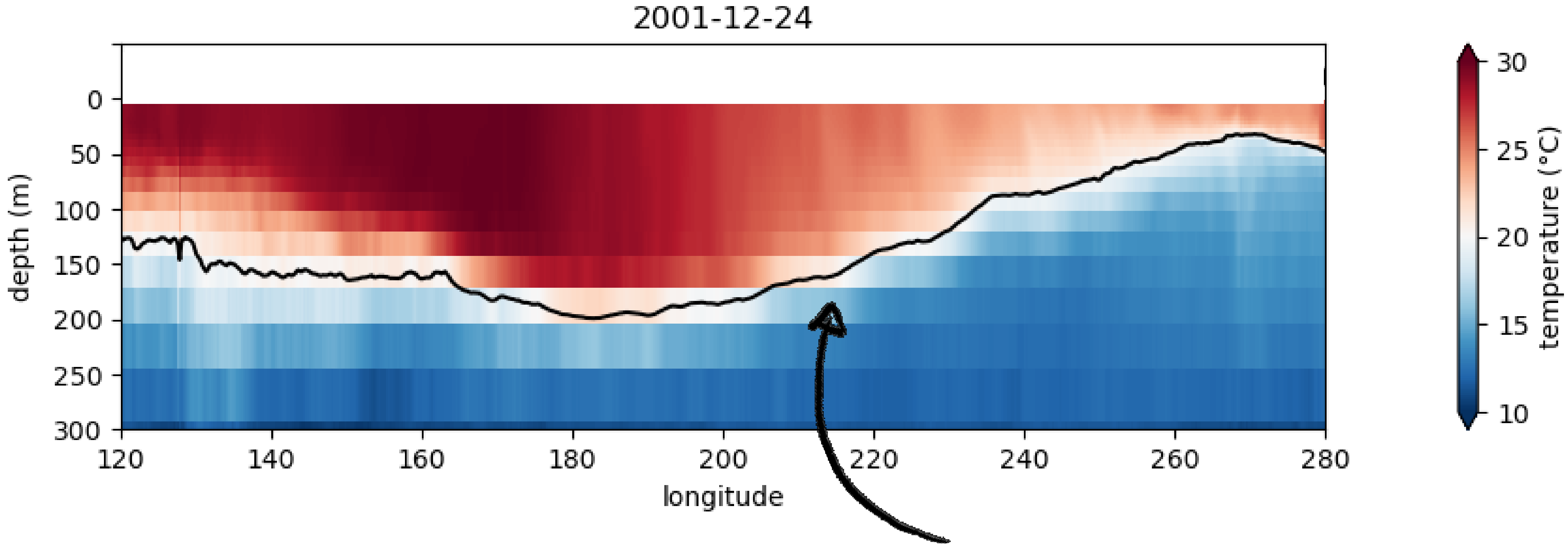
```
numerator = (weights*data).rolling({‘time’:3}).sum()
```

3 Find rolling sum of weights

4 Divide

```
numerator = (weights).rolling({‘time’:3}).sum()
```

Example 3: 20° isotherm



Find that depth

Example 3: 20° isotherm

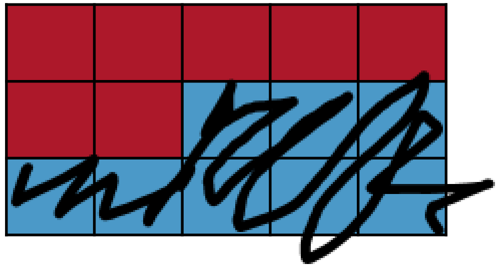
- Find 20° C isotherm

= find depth that is 20° C

≈ find deepest depth that is warmer than 20° C

1 Depth warmer than 20°C

```
warm_depth = depth.where(ds>20)
```



2 Deepest depth

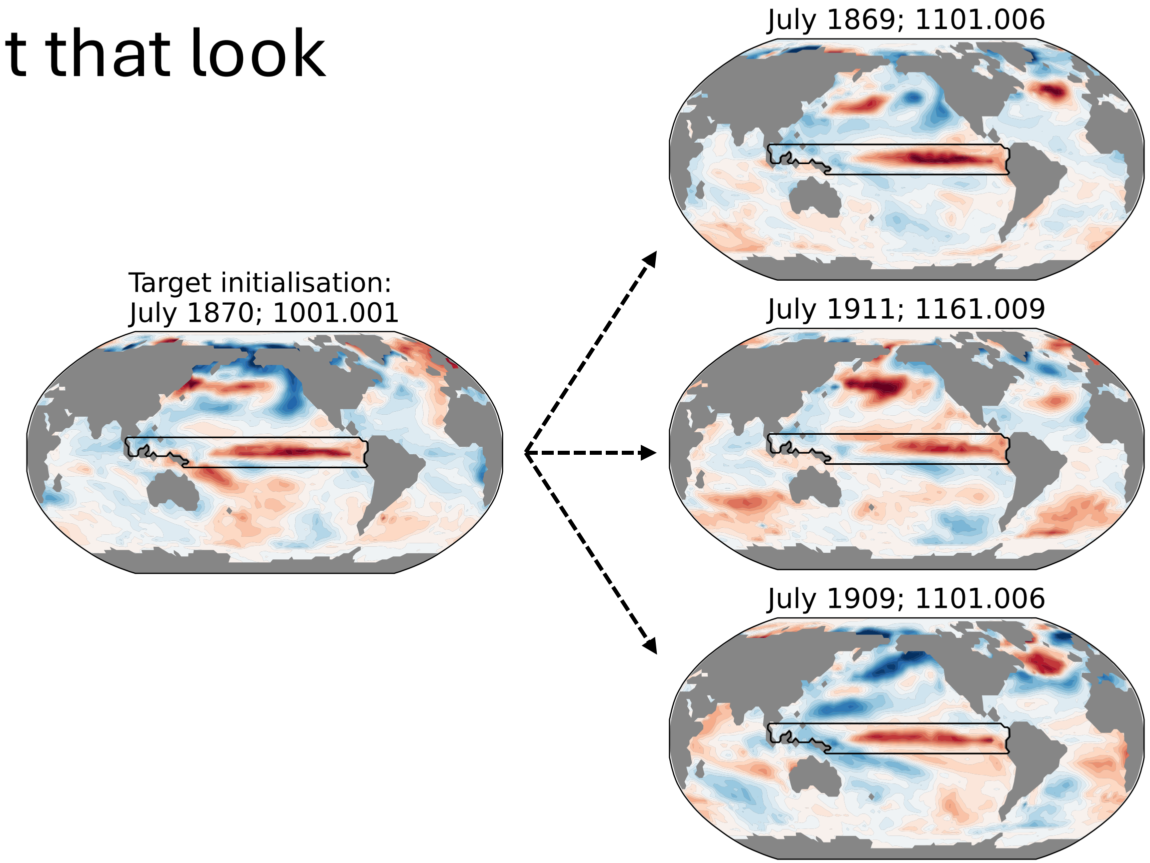
```
warm_depth.argmax("depth")
```

or

```
warm_depth.idxmax("depth")
```

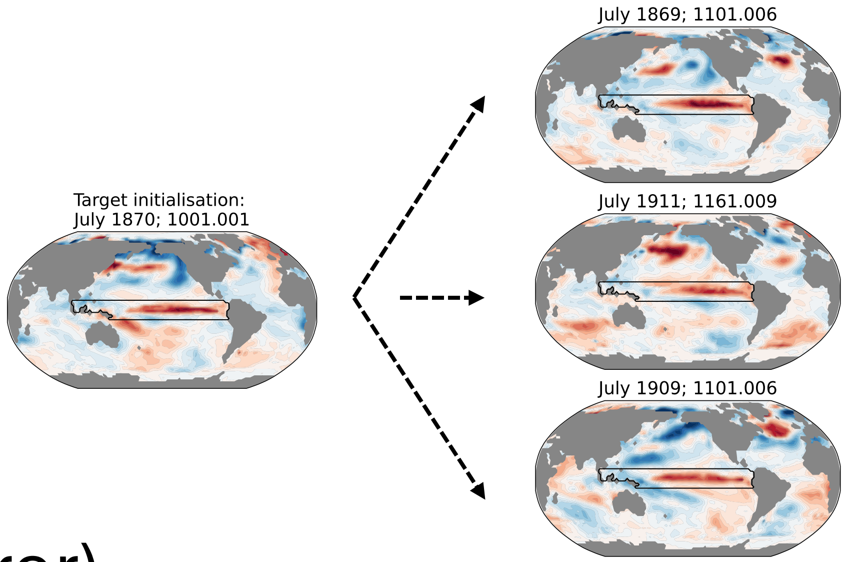
Example 4: analogue forecasts

- Find the months of model output that look most similar to my target month



Example 4: analogue forecasts

- Find the months of model output that look most similar to my target month



1 Calculate similarity (ie, mean squared error)

```
mse = ((all_sst-target_sst)**2).mean((‘lat’,’lon’))
```

2 Find the indices of lowest mean squared error

```
mse.idxmin(‘time’)
```

The last resort: xr.apply_ufunc

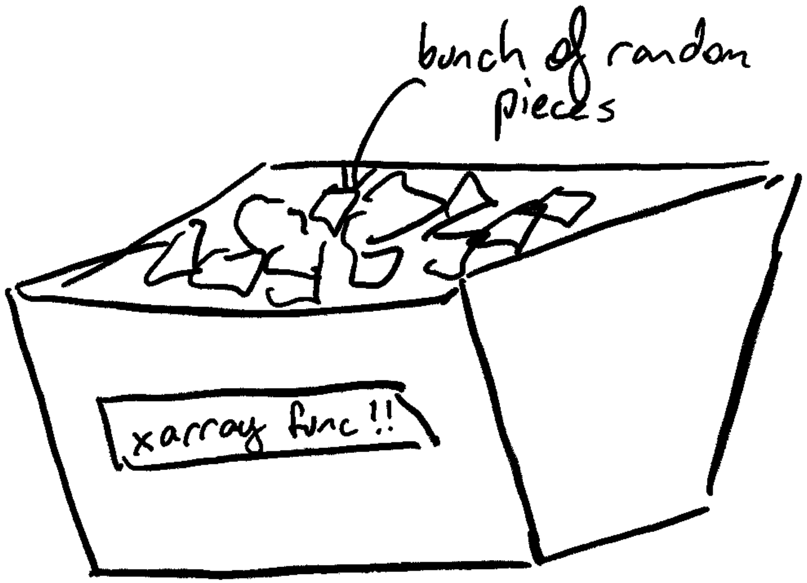


Fig 1. My nice contained function

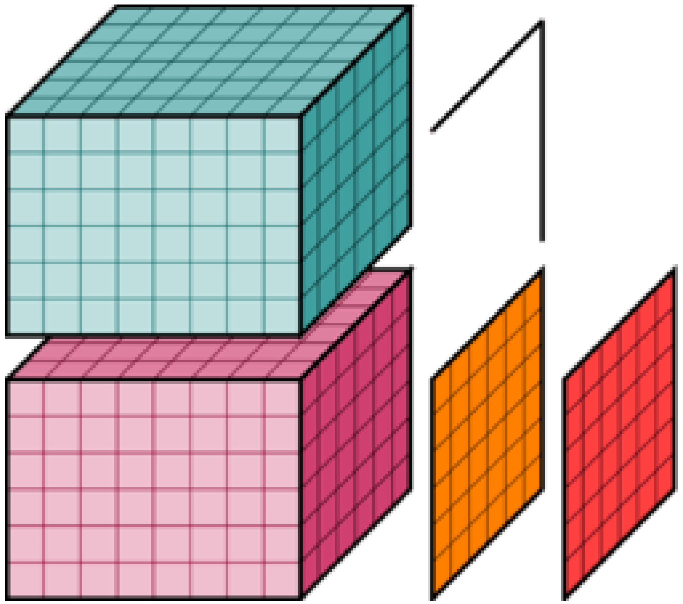
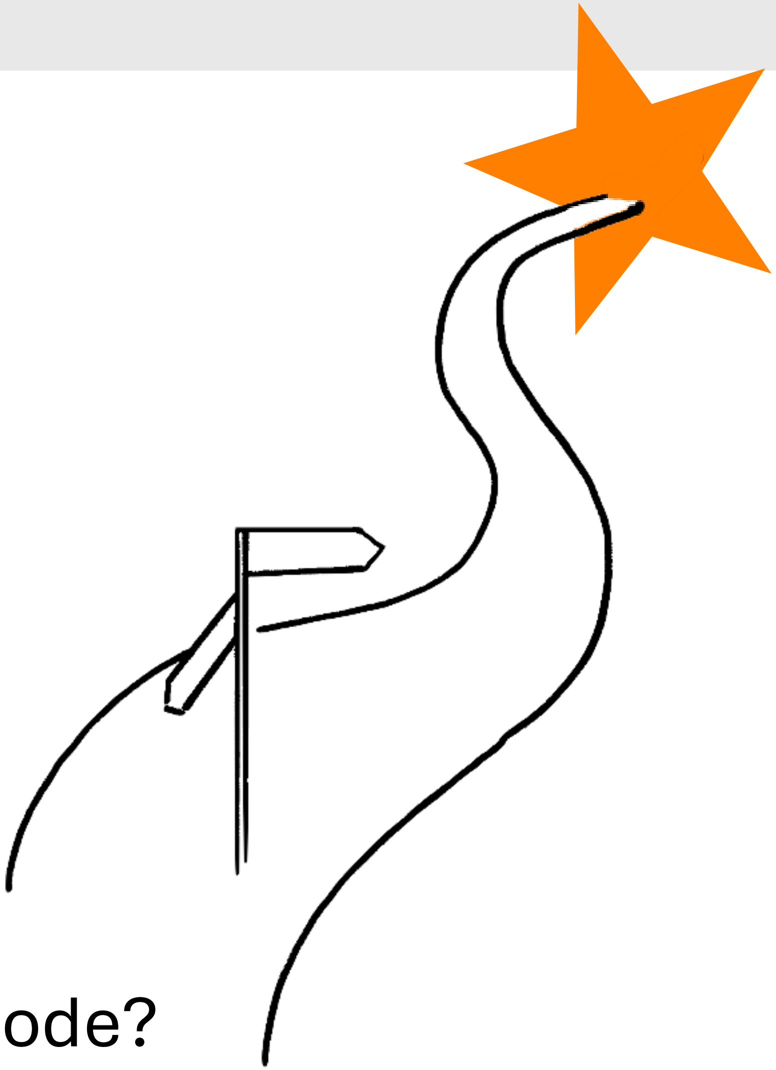


Fig 2. xarray

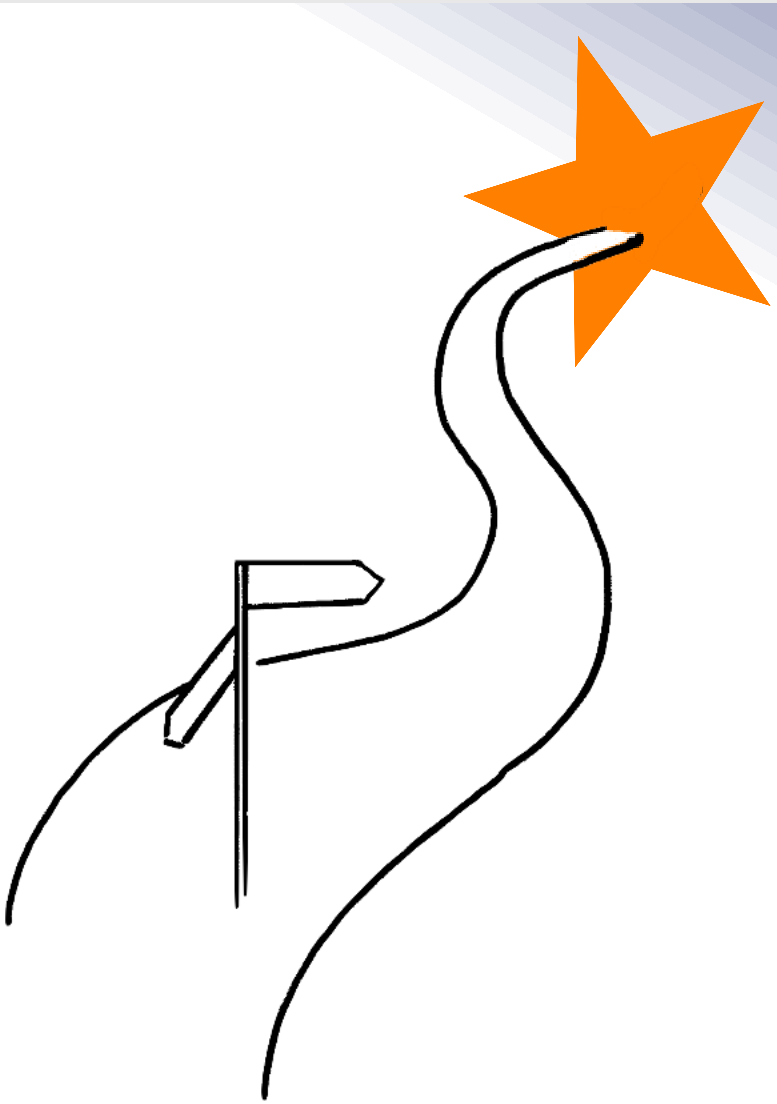
cosima-recipes/Tutorials/Apply_function_to_every_gridpoint.ipynb

Summary – Aim is to write faster/better code

- Why shoul n to me?
- **Why** b
- What **st/goo** mean?
- How xarray/
  - **“La”** comp
  - Thinki **lerlying** code
  - The vo **unking**
- But how c **te** my function into xarray code?



Good luck



jemma.jeffree@anu.edu.au
@jemmajeffree on hive/github