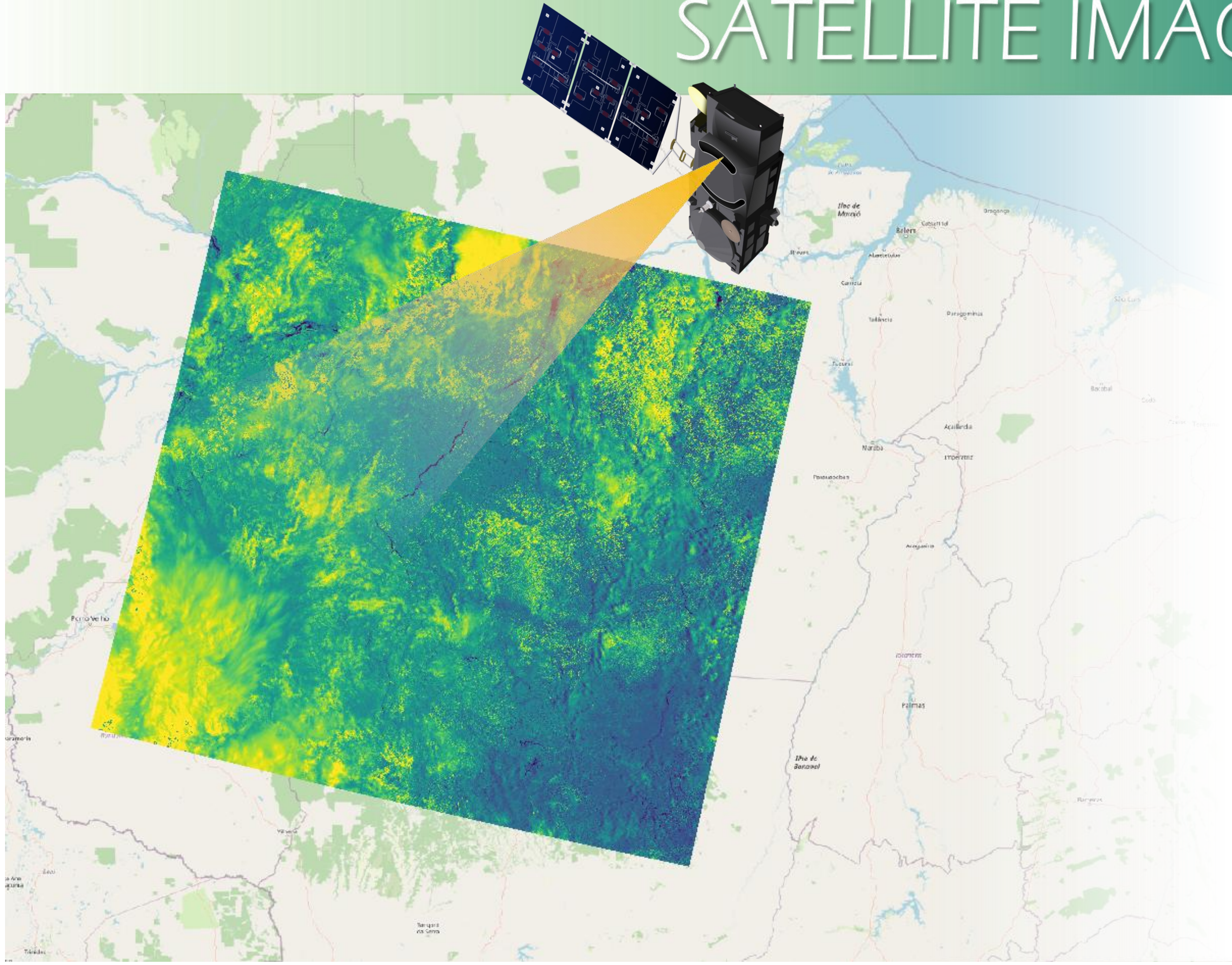
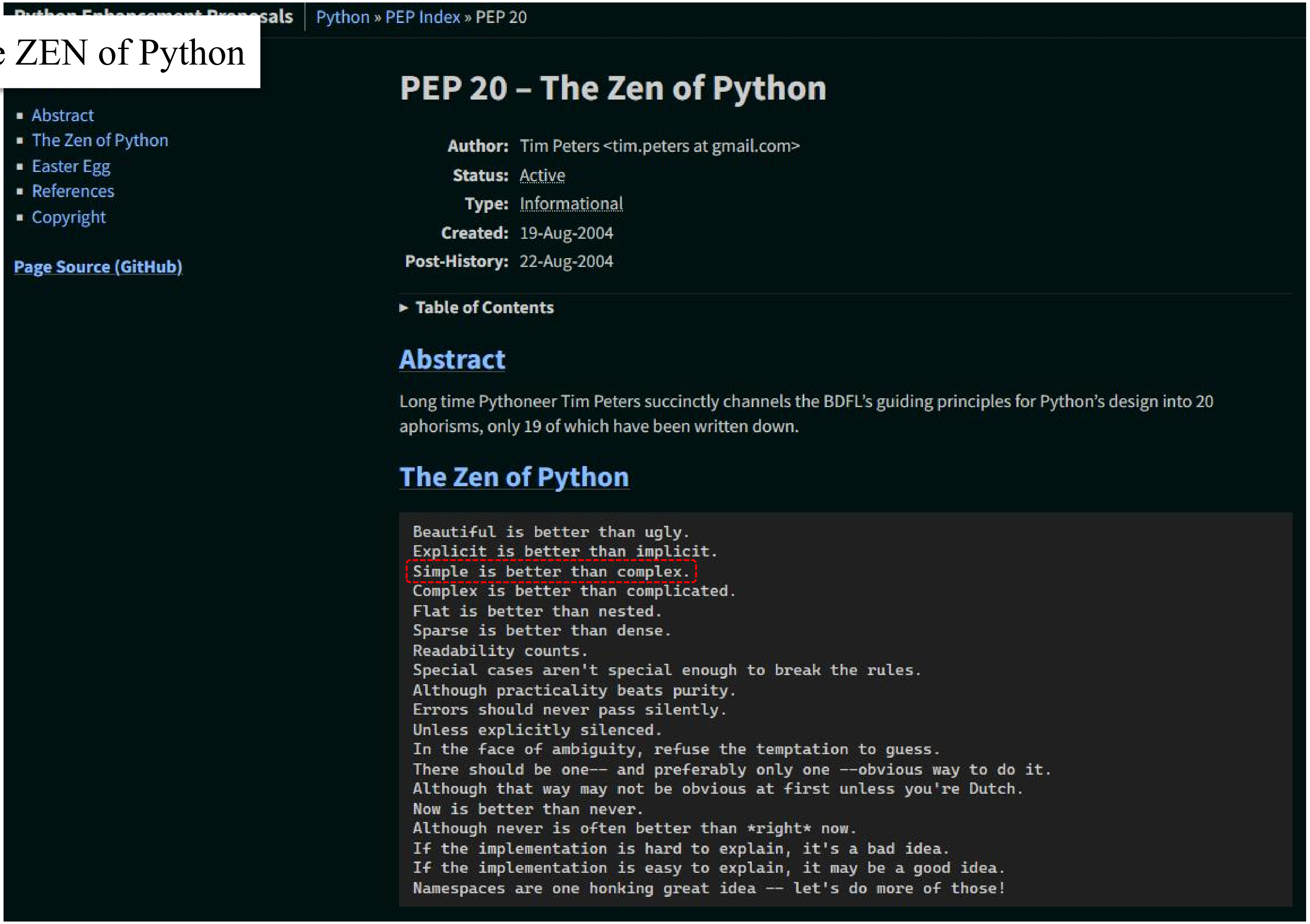
01-12-2022 : Atelier numérique de l'OMP - La performance dans le monde Python.

USER-FRIENDLY PARALLEL DATA EXTRACTION IN SATELLITE IMAGES WITH concurrent.futures



David Guimarães
david.guimaraes@ird.fr

PEP20 : The ZEN of Python



Python Enhancement Proposals | Python » PEP Index » PEP 20

- Abstract
- The Zen of Python
- Easter Egg
- References
- Copyright

Page Source (GitHub)

PEP 20 – The Zen of Python

Author:	Tim Peters <tim.peters at gmail.com>
Status:	Active
Type:	Informational
Created:	19-Aug-2004
Post-History:	22-Aug-2004

▸ **Table of Contents**

Abstract

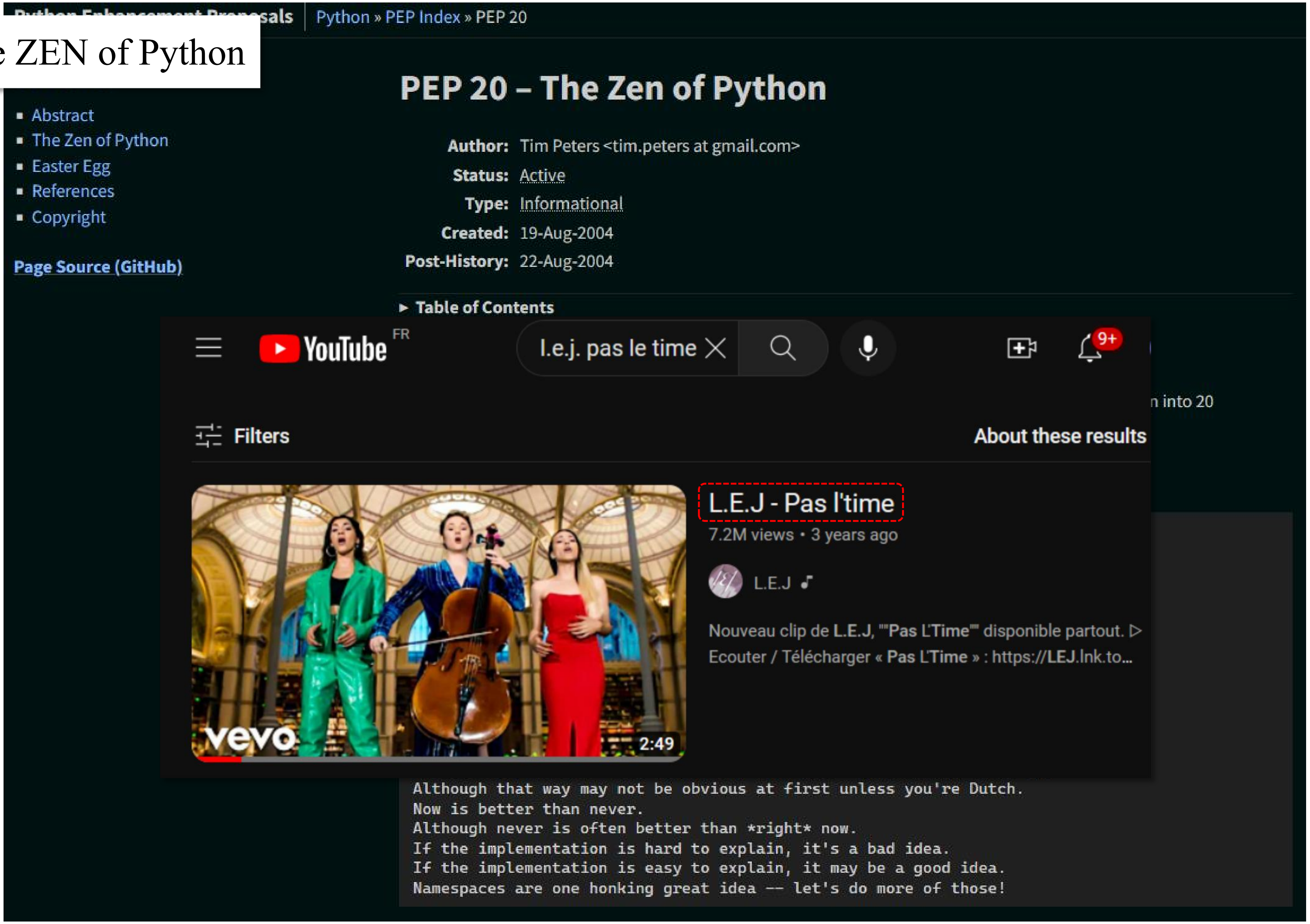
Long time Pythoneer Tim Peters succinctly channels the BDFL's guiding principles for Python's design into 20 aphorisms, only 19 of which have been written down.

The Zen of Python

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

source: https://peps.python.org/pep-0020/

PEP20 : The ZEN of Python



Python Enhancement Proposals | Python » PEP Index » PEP 20

PEP 20 – The Zen of Python

Page Source (GitHub)

Author:	Tim Peters <tim.peters at gmail.com>
Status:	Active
Type:	Informational
Created:	19-Aug-2004
Post-History:	22-Aug-2004

▸ Table of Contents

n into 20

L.E.J - Pas l'time

7.2M views • 3 years ago

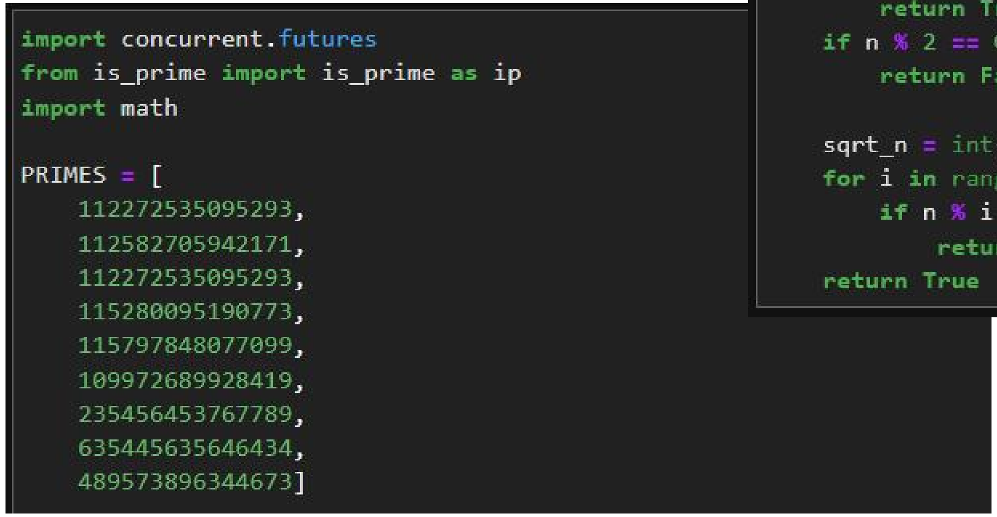
L.E.J

Nouveau clip de L.E.J, ""Pas L'Time"" disponible partout. ▷ Ecouter / Télécharger « Pas L'Time » : https://LEJ.lnk.to...

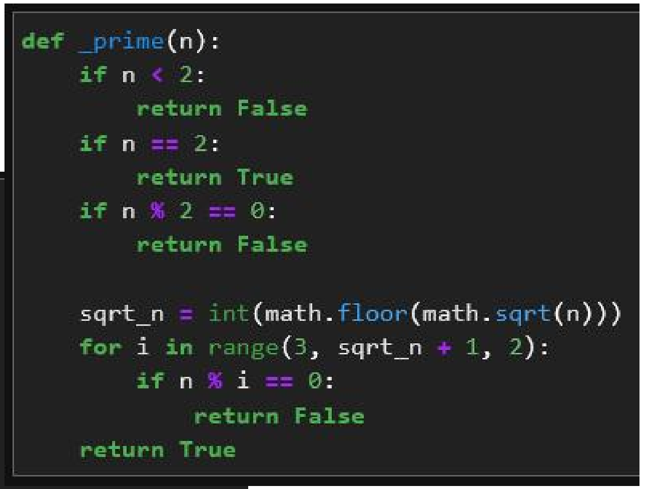
```
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

source: https://peps.python.org/pep-0020/

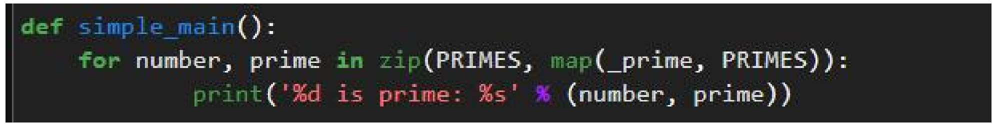
Part 1 : Context



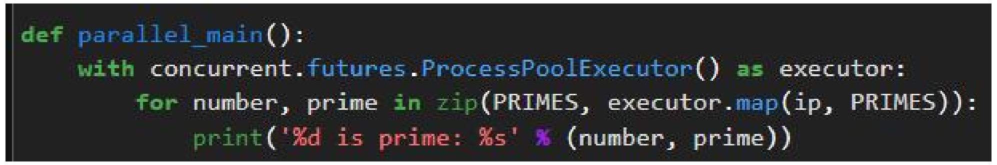
```python
import concurrent.futures
from is_prime import is_prime as ip
import math

PRIMES = [
    112272535095293,
    112582705942171,
    112272535095293,
    115280095190773,
    115797848077099,
    109972689928419,
    235456453767789,
    635445635646434,
    489573896344673]
```



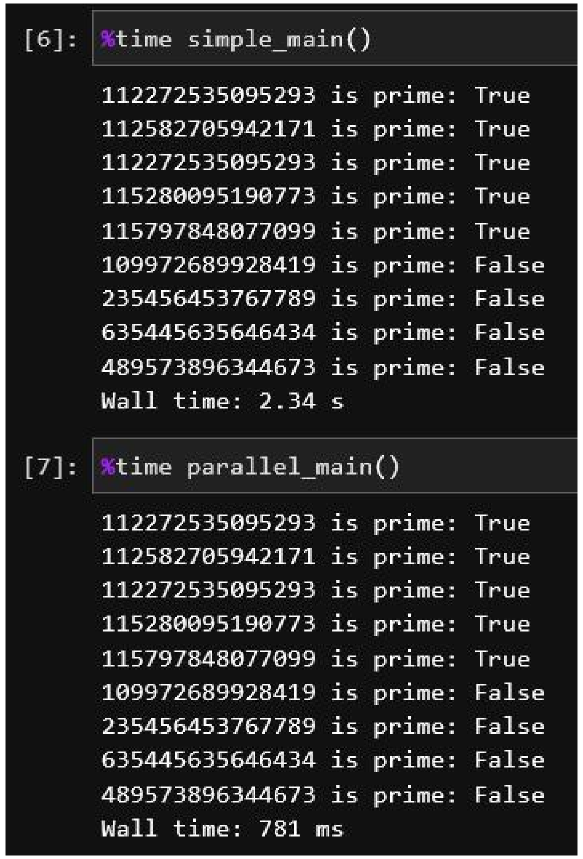
```python
def _prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False

    sqrt_n = int(math.floor(math.sqrt(n)))
    for i in range(3, sqrt_n + 1, 2):
        if n % i == 0:
            return False
    return True
```



```python
def simple_main():
    for number, prime in zip(PRIMES, map(_prime, PRIMES)):
            print('%d is prime: %s' % (number, prime))
```



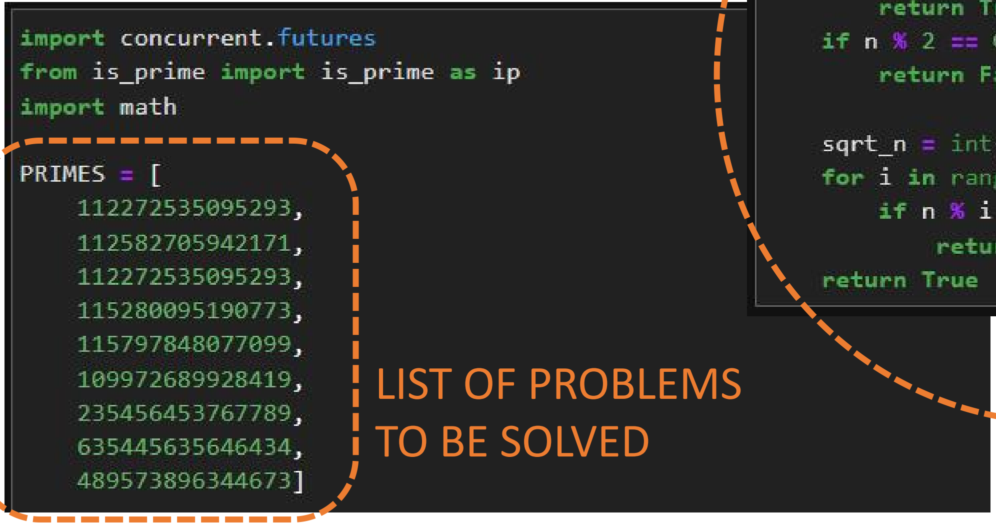
```python
def parallel_main():
    with concurrent.futures.ProcessPoolExecutor() as executor:
        for number, prime in zip(PRIMES, executor.map(ip, PRIMES)):
            print('%d is prime: %s' % (number, prime))
```



```
[6]: %time simple_main()

     112272535095293 is prime: True
     112582705942171 is prime: True
     112272535095293 is prime: True
     115280095190773 is prime: True
     115797848077099 is prime: True
     109972689928419 is prime: False
     235456453767789 is prime: False
     635445635646434 is prime: False
     489573896344673 is prime: False
     Wall time: 2.34 s

[7]: %time parallel_main()

     112272535095293 is prime: True
     112582705942171 is prime: True
     112272535095293 is prime: True
     115280095190773 is prime: True
     115797848077099 is prime: True
     109972689928419 is prime: False
     235456453767789 is prime: False
     635445635646434 is prime: False
     489573896344673 is prime: False
     Wall time: 781 ms
```

https://docs.python.org/3/library/concurrent.futures.html#processpoolexecutor-example

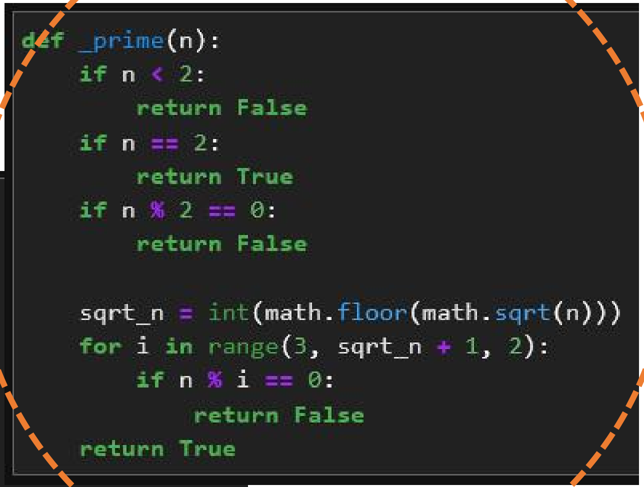
Anatomy of a program



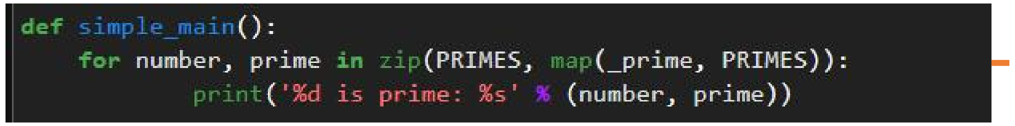
```python
import concurrent.futures
from is_prime import is_prime as ip
import math

PRIMES = [
    112272535095293,
    112582705942171,
    112272535095293,
    115280095190773,
    115797848077099,
    109972689928419,
    235456453767789,
    635445635646434,
    489573896344673]
```

LIST OF PROBLEMS TO BE SOLVED

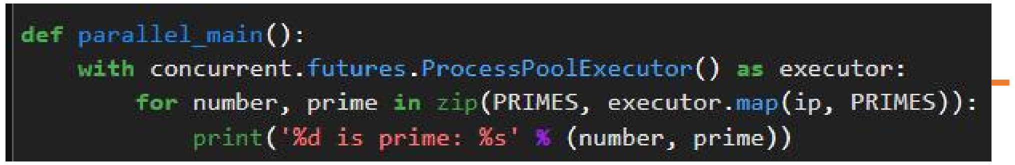
FUNCTION TO "SOLVE" THE PROBLEM



```python
def _prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False

    sqrt_n = int(math.floor(math.sqrt(n)))
    for i in range(3, sqrt_n + 1, 2):
        if n % i == 0:
            return False
    return True
```

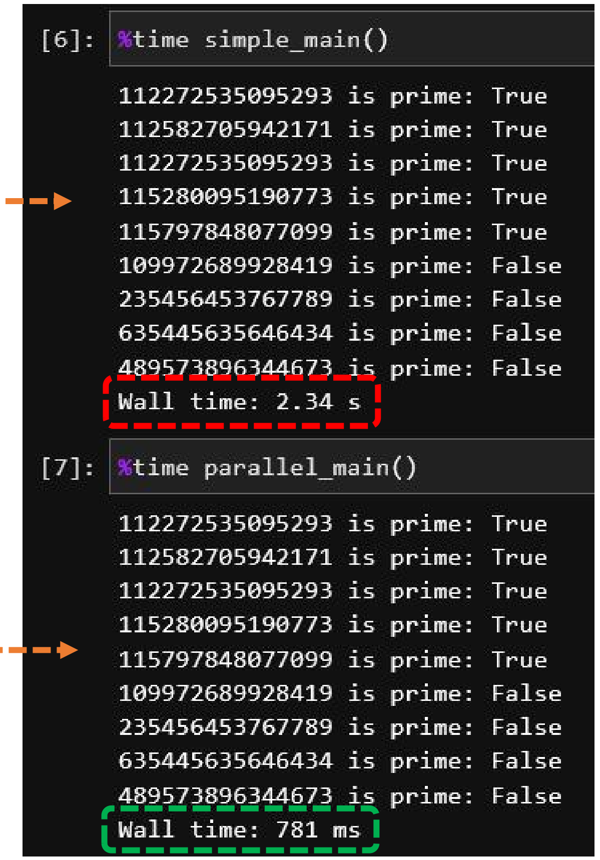


```python
def simple_main():
    for number, prime in zip(PRIMES, map(_prime, PRIMES)):
        print('%d is prime: %s' % (number, prime))
```

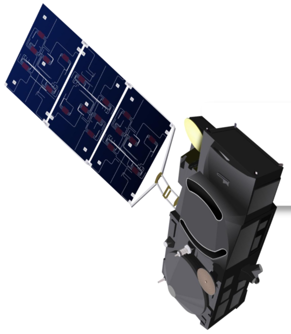


```python
def parallel_main():
    with concurrent.futures.ProcessPoolExecutor() as executor:
        for number, prime in zip(PRIMES, executor.map(ip, PRIMES)):
            print('%d is prime: %s' % (number, prime))
```

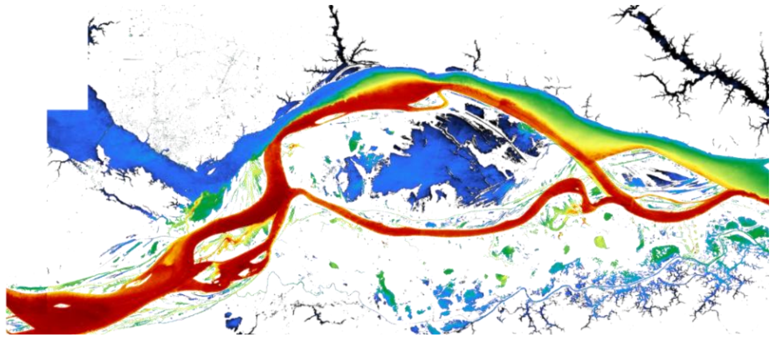
POSSIBLE SOLUTIONS

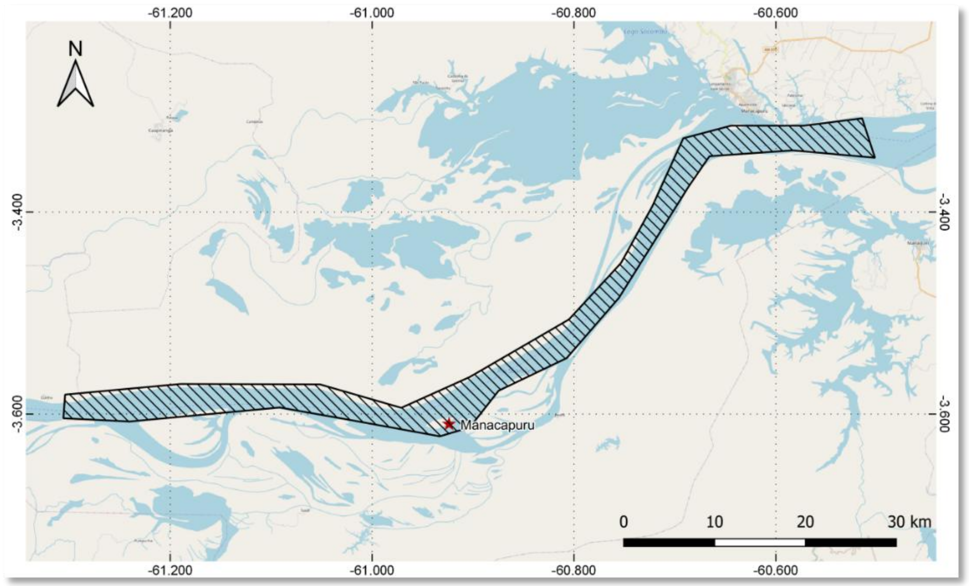


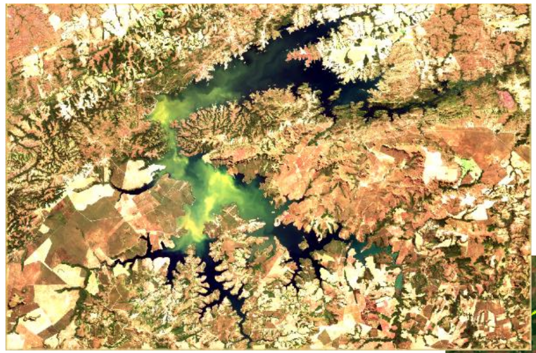
```
[6]: %time simple_main()

     112272535095293 is prime: True
     112582705942171 is prime: True
     112272535095293 is prime: True
     115280095190773 is prime: True
     115797848077099 is prime: True
     109972689928419 is prime: False
     235456453767789 is prime: False
     635445635646434 is prime: False
     489573896344673 is prime: False
     Wall time: 2.34 s

[7]: %time parallel_main()

     112272535095293 is prime: True
     112582705942171 is prime: True
     112272535095293 is prime: True
     115280095190773 is prime: True
     115797848077099 is prime: True
     109972689928419 is prime: False
     235456453767789 is prime: False
     635445635646434 is prime: False
     489573896344673 is prime: False
     Wall time: 781 ms
```

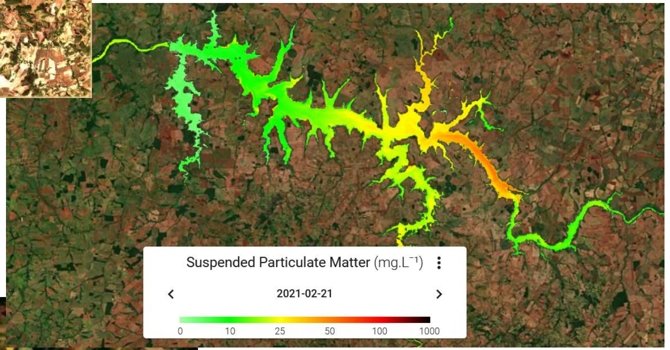


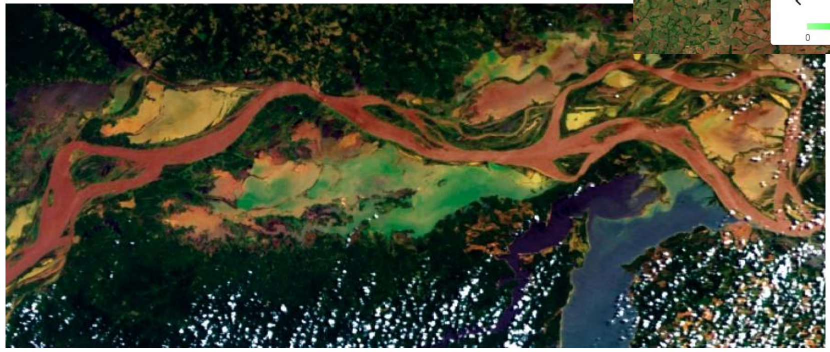
Part 2 : Real scenario









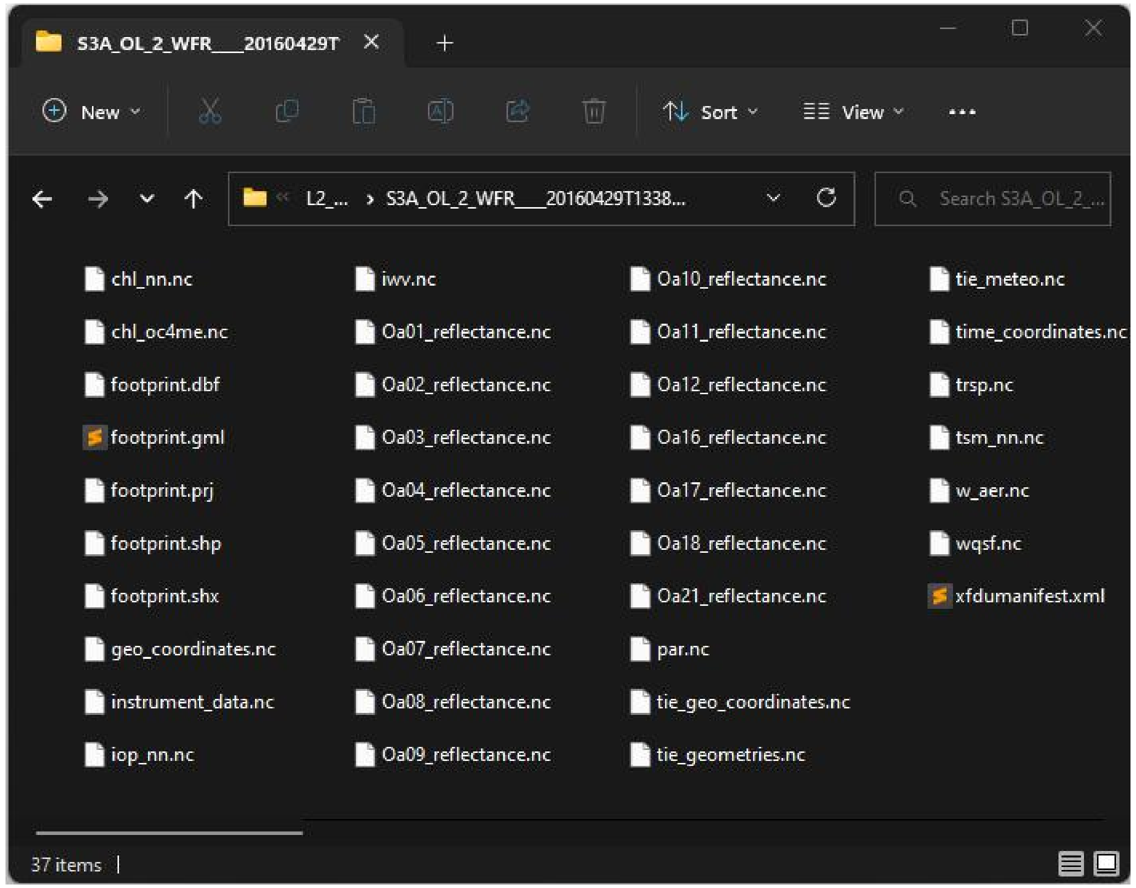


Sentinel-3

```
['chl_nn.nc',
 'chl_oc4me.nc',
 'geo_coordinates.nc',
 'instrument_data.nc',
 'iop_nn.nc',
 'iwv.nc',
 'Oa01_reflectance.nc',
 'Oa02_reflectance.nc',
 'Oa03_reflectance.nc',
 'Oa04_reflectance.nc',
 'Oa05_reflectance.nc',
 'Oa06_reflectance.nc',
 'Oa07_reflectance.nc',
 'Oa08_reflectance.nc',
 'Oa09_reflectance.nc',
 'Oa10_reflectance.nc',
 'Oa11_reflectance.nc',
 'Oa12_reflectance.nc',
 'Oa16_reflectance.nc',
 'Oa17_reflectance.nc',
 'Oa18_reflectance.nc',
 'Oa21_reflectance.nc',
 'par.nc',
 'tie_geometries.nc',
 'tie_geo_coordinates.nc',
 'tie_meteo.nc',
 'time_coordinates.nc',
 'trsp.nc',
 'tsm_nn.nc',
 'wqsf.nc',
 'w_aer.nc']
```

```
<class 'netCDF4._netCDF4.Dimension'>:
            name = 'columns', size = 4865,
            name = 'rows', size = 4091
```

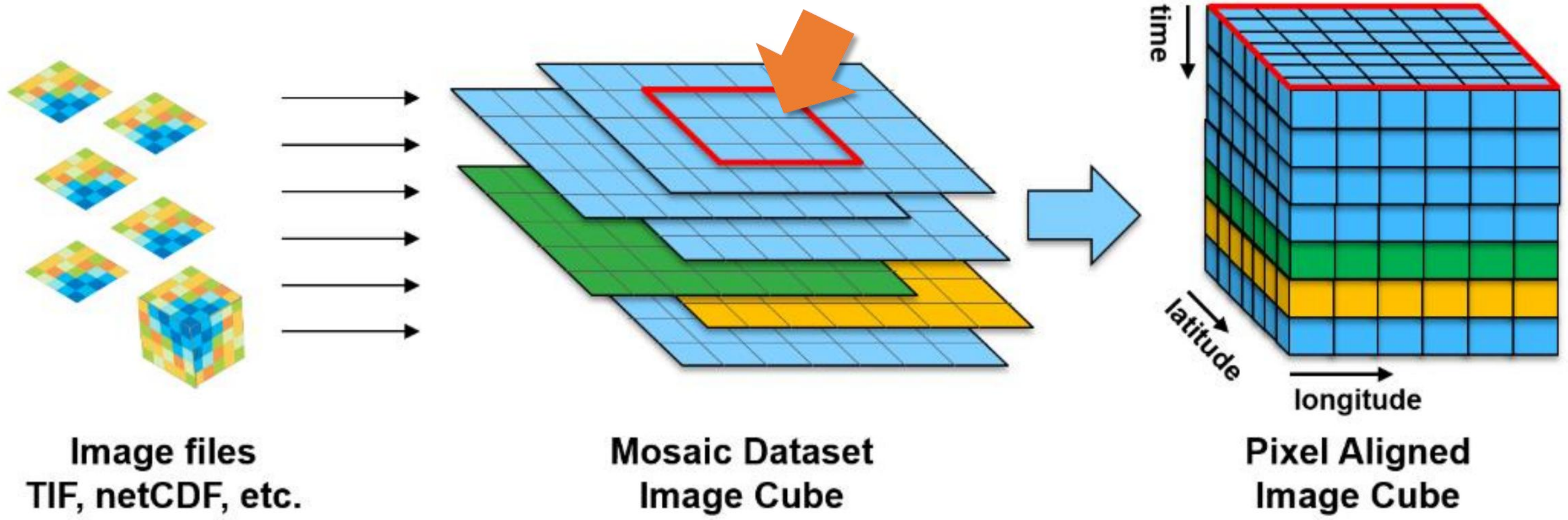
```
<class 'netCDF4._netCDF4.Dimension'>:
            name = 'tie_columns', size = 77,
            name = 'tie_rows', size = 4091
```



```
chl_nn.nc            iwv.nc                  Oa10_reflectance.nc      tie_meteo.nc
chl_oc4me.nc         Oa01_reflectance.nc     Oa11_reflectance.nc      time_coordinates.nc
footprint.dbf        Oa02_reflectance.nc     Oa12_reflectance.nc      trsp.nc
footprint.gml        Oa03_reflectance.nc     Oa16_reflectance.nc      tsm_nn.nc
footprint.prj        Oa04_reflectance.nc     Oa17_reflectance.nc      w_aer.nc
footprint.shp        Oa05_reflectance.nc     Oa18_reflectance.nc      wqsf.nc
footprint.shx        Oa06_reflectance.nc     Oa21_reflectance.nc      xfdumanifest.xml
geo_coordinates.nc   Oa07_reflectance.nc     par.nc
instrument_data.nc   Oa08_reflectance.nc     tie_geo_coordinates.nc
iop_nn.nc            Oa09_reflectance.nc     tie_geometries.nc
37 items
```

- OAA : Observation (Viewing) Azimuth Angle,
- OZA : Observation (Viewing) Zenith Angle,
- SAA : Sun Azimuth Angle,
- SZA : Sun Zenith Angle,

What is a Sat. IMG?



source: https://eox.at/2021/01/earth-observation-data-cubes-as-a-service/

Mapping the "anatomy" of a vector (.shp, .geojson, etc.)

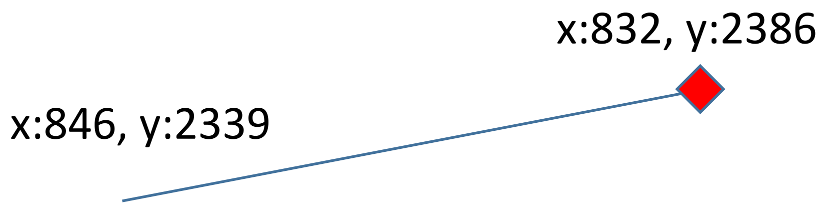
x:846, y:2339

LON / LAT

0

```
[ [ -61.304292937753878, -3.580746157823998, 0.0 ],
[ -61.189738482291098, -3.570230611586103, 0.0 ],
[ -61.051796038163282, -3.570623959031377, 0.0 ],
[ -60.971175020538652, -3.593840244684938, 0.0 ],
[ -60.9034262153111, -3.563360680269377, 0.0 ],
[ -60.805020782919101, -3.506299406583337, 0.0 ],

.
.
.

[ -61.305985273744547, -3.604193025913813, 0.0 ],
[ -61.304292937753878, -3.580746157823998, 0.0 ] ]
```

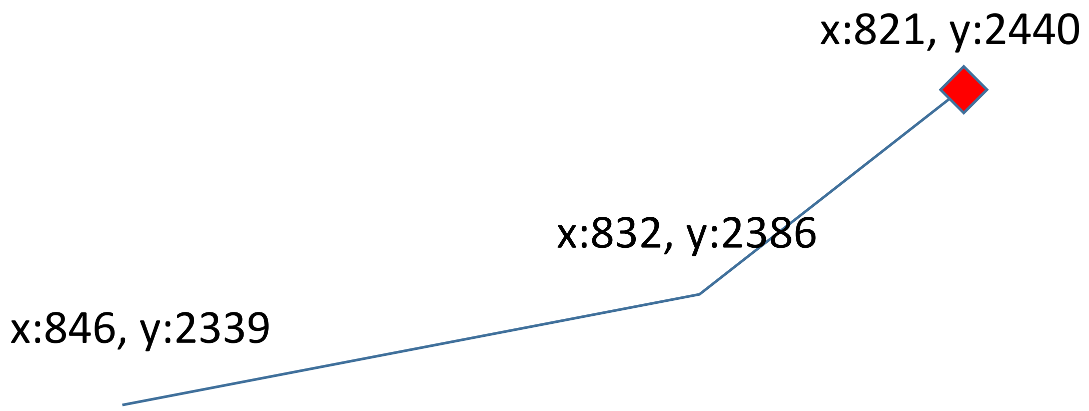
Mapping the “anatomy” of a vector (.shp, .geojson, etc.)



LON / LAT

```
[ [ -61.304292937753878, -3.580746157823998, 0.0 ],
1 [ -61.189738482291098, -3.570230611586103, 0.0 ],
[ -61.051796038163282, -3.570623959031377, 0.0 ],
[ -60.971175020538652, -3.593840244684938, 0.0 ],
[ -60.9034262153111, -3.563360680269377, 0.0 ],
[ -60.805020782919101, -3.506299406583337, 0.0 ],

.
.
.

[ -61.305985273744547, -3.604193025913813, 0.0 ],
[ -61.304292937753878, -3.580746157823998, 0.0 ] ]
```

Mapping the “anatomy” of a vector (.shp, .geojson, etc.)

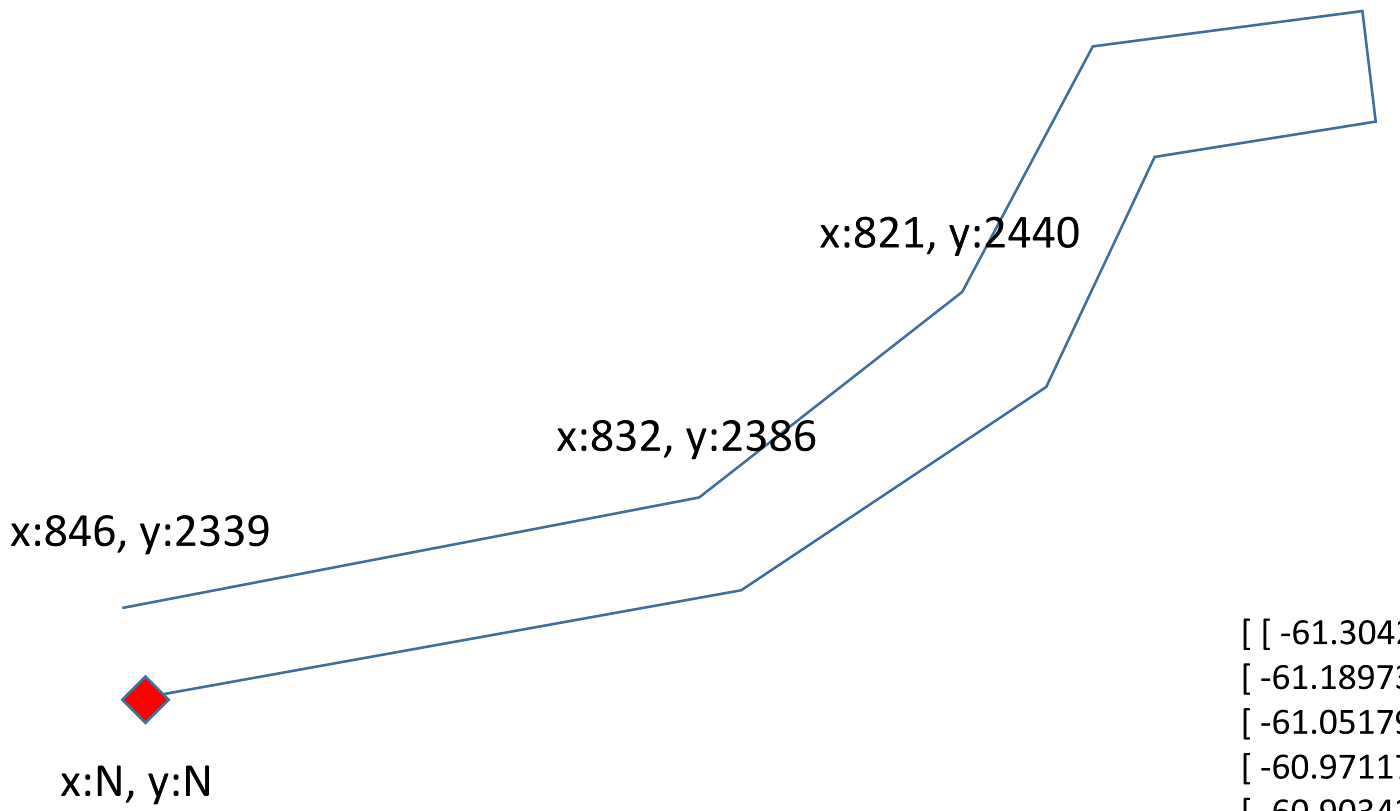


LON / LAT

```
[ [ -61.304292937753878, -3.580746157823998, 0.0 ],
[ -61.189738482291098, -3.570230611586103, 0.0 ],
[ -61.051796038163282, -3.570623959031377, 0.0 ],
[ -60.971175020538652, -3.593840244684938, 0.0 ],
[ -60.9034262153111, -3.563360680269377, 0.0 ],
[ -60.805020782919101, -3.506299406583337, 0.0 ],

.
.
.

[ -61.305985273744547, -3.604193025913813, 0.0 ],
[ -61.304292937753878, -3.580746157823998, 0.0 ] ]
```

2

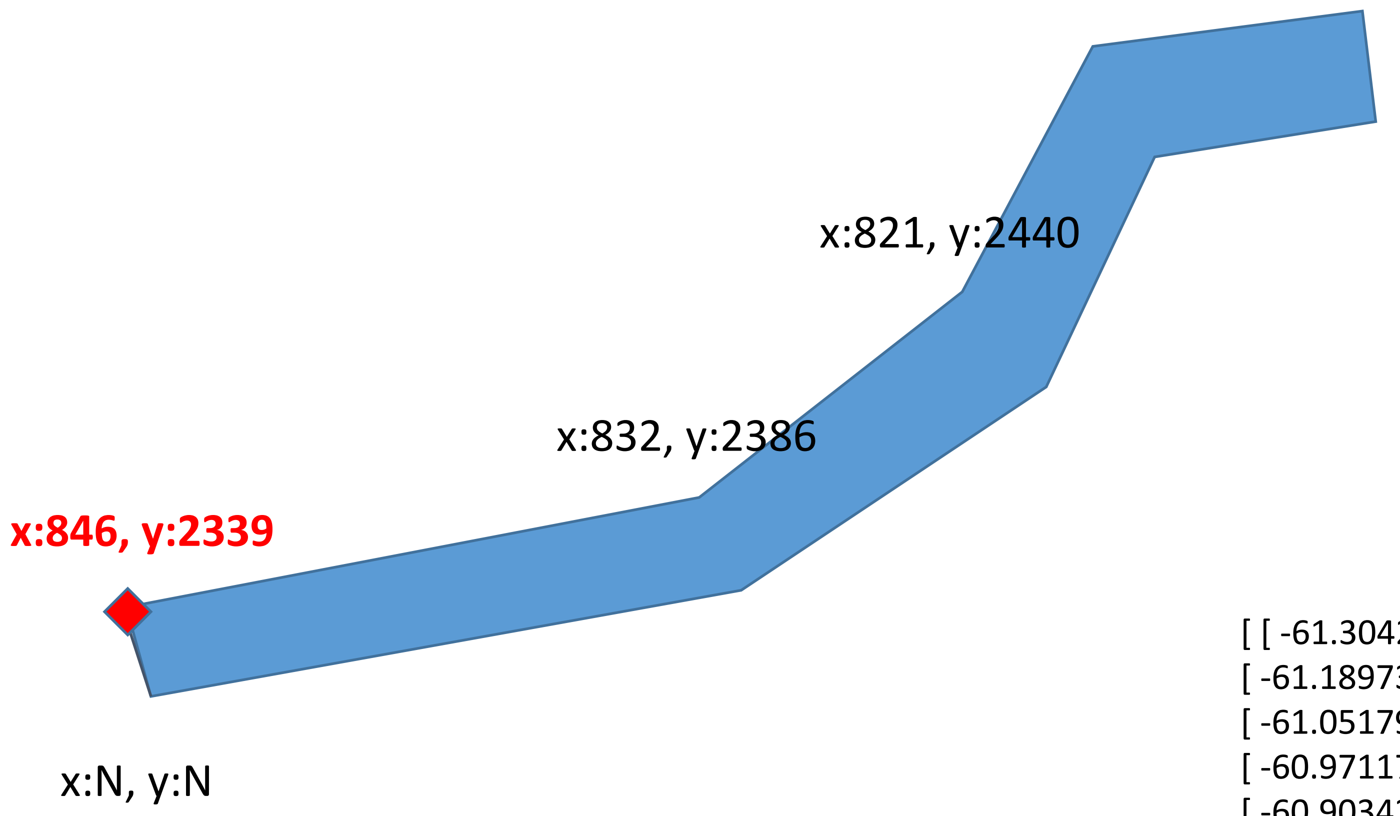
Mapping the “anatomy” of a vector (.shp, .geojson, etc.)



LON / LAT

```
[ [ -61.304292937753878, -3.580746157823998, 0.0 ],
[ -61.189738482291098, -3.570230611586103, 0.0 ],
[ -61.051796038163282, -3.570623959031377, 0.0 ],
[ -60.971175020538652, -3.593840244684938, 0.0 ],
[ -60.9034262153111, -3.563360680269377, 0.0 ],
[ -60.805020782919101, -3.506299406583337, 0.0 ],

.
.
.

N [ -61.305985273744547, -3.604193025913813, 0.0 ],
[ -61.304292937753878, -3.580746157823998, 0.0 ] ]
```

Mapping the “anatomy” of a vector (.shp, .geojson, etc.)



LON / LAT

```
[ [ -61.304292937753878, -3.580746157823998, 0.0 ],
[ -61.189738482291098, -3.570230611586103, 0.0 ],
[ -61.051796038163282, -3.570623959031377, 0.0 ],
[ -60.971175020538652, -3.593840244684938, 0.0 ],
[ -60.9034262153111, -3.563360680269377, 0.0 ],
[ -60.805020782919101, -3.506299406583337, 0.0 ],

.
.
.

[ -61.305985273744547, -3.604193025913813, 0.0 ],
end [ -61.304292937753878, -3.580746157823998, 0.0 ] ]
```

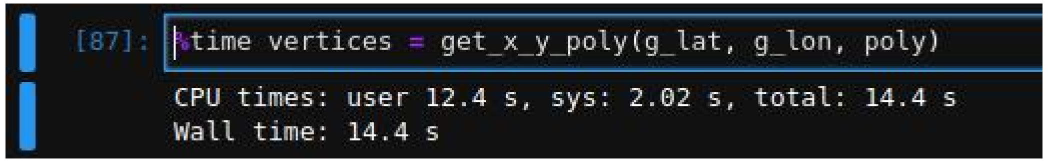
RIGHT ?

nope.

```
import concurrent.futures
```

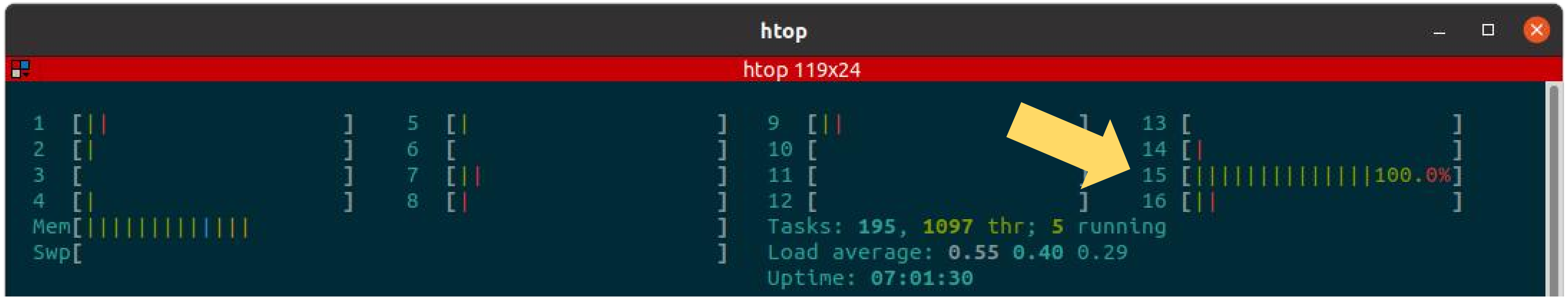
avoid OS crash (-:

```
with concurrent.futures.ProcessPoolExecutor(max_workers=os.cpu_count() - 1) as executor:
        try:
            result = list(executor.map(self.vect_dist_subtraction, coord_vect_pairs))

        except concurrent.futures.process.BrokenProcessPool as ex:
            logging.error(f"{ex} This might be caused by limited system resources. "
                          f"Try increasing system memory or disable concurrent processing. ")
```

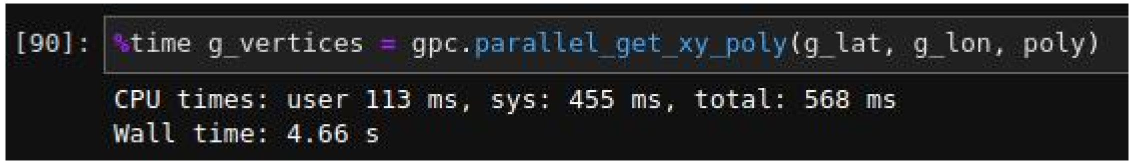


```
[87]: %time vertices = get_x_y_poly(g_lat, g_lon, poly)

CPU times: user 12.4 s, sys: 2.02 s, total: 14.4 s
Wall time: 14.4 s
```

14.4s

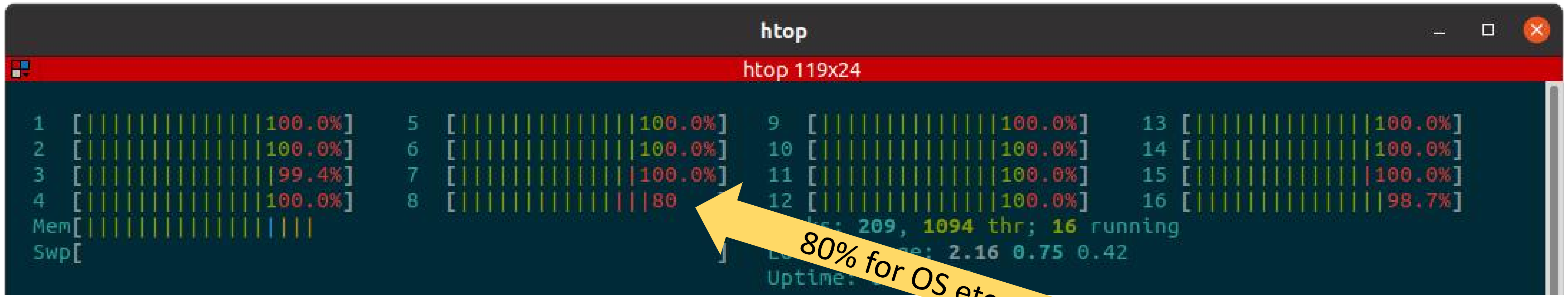


```
htop 119x24

1  [||                    ]   5  [|                     ]   9  [||                    ]   13 [                     ]
2  [|                     ]   6  [                      ]   10 [                      ]   14 [|                    ]
3  [                      ]   7  [||                    ]   11 [                      ]   15 [|||||||||||||100.0%]
4  [|                     ]   8  [|                     ]   12 [                      ]   16 [||                   ]
Mem[|||||||||||||                                     ]   Tasks: 195, 1097 thr; 5 running
Swp[                                                  ]   Load average: 0.55 0.40 0.29
                                                          Uptime: 07:01:30
```



```
[90]: %time g_vertices = gpc.parallel_get_xy_poly(g_lat, g_lon, poly)

      CPU times: user 113 ms, sys: 455 ms, total: 568 ms
      Wall time: 4.66 s
```

4.6 s



```
htop
htop 119x24

  1  [|||||||||||||100.0%]   5  [|||||||||||||100.0%]  9  [|||||||||||||100.0%]   13 [|||||||||||||100.0%]
  2  [|||||||||||||100.0%]   6  [|||||||||||||100.0%]  10 [|||||||||||||100.0%]   14 [|||||||||||||100.0%]
  3  [|||||||||||||99.4%]    7  [|||||||||||||100.0%]  11 [|||||||||||||100.0%]   15 [|||||||||||||100.0%]
  4  [|||||||||||||100.0%]   8  [|||||||||||||||80          12 [|||||||||||||100.0%]   16 [|||||||||||||98.7%]
  Mem[|||||||||||||||||||                                   ks: 209, 1094 thr; 16 running
  Swp[                                                ]  Lo       ge: 2.16 0.75 0.42
                                                         Uptime:
```

80% for OS etc.

```
array([[-61.30429294,  -3.58074616],
       [-61.18973848,  -3.57023061],
       [-61.05179604,  -3.57062396],
       [-60.97117502,  -3.59384024],
       [-60.90342622,  -3.56336068],
       [-60.80502078,  -3.50629941],
       [-60.75351909,  -3.45050586],
       [-60.72245698,  -3.39425948],
       [-60.69176635,  -3.3269743 ],
       [-60.64421466,  -3.31443418],
       [-60.57262505,  -3.31452306],
       [-60.51454132,  -3.30721351],
       [-60.50184937,  -3.34625831],
       [-60.58341863,  -3.3389403 ],
       [-60.66584003,  -3.34476071],
       [-60.68655868,  -3.37488964],
       [-60.75565447,  -3.48512231],
       [-60.80730728,  -3.54463971],
       [-60.87472835,  -3.57689186],
       [-60.90334283,  -3.61322746],
       [-60.93298959,  -3.6220597 ],
       [-61.02141493,  -3.60596636],
       [-61.09150085,  -3.59352661],
       [-61.14960697,  -3.59895818],
       [-61.24046167,  -3.60740423],
       [-61.30598527,  -3.60419303],
       [-61.30429294,  -3.58074616]])
```

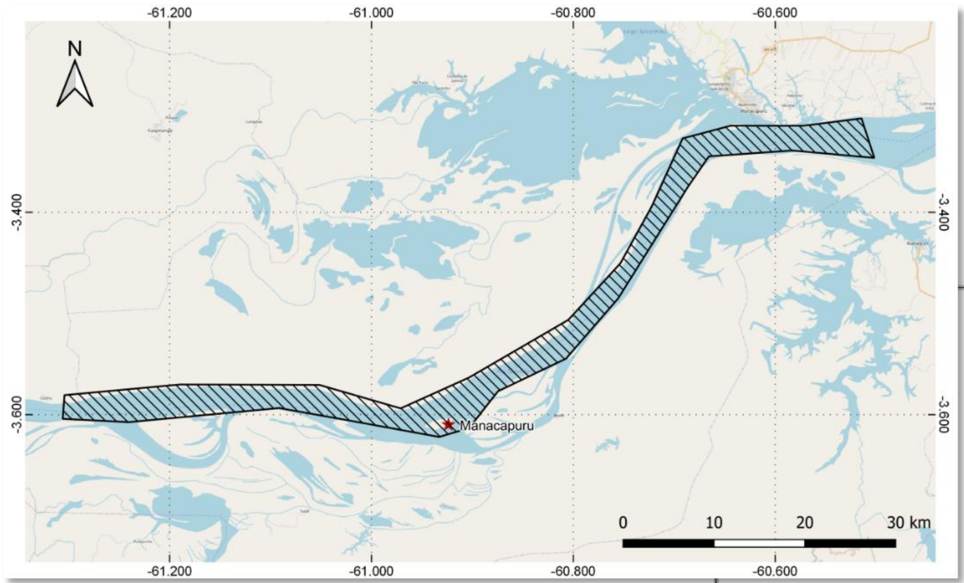
```
array([[-61.30429294,  -3.58074616],
       [-61.18973848,  -3.57023061],
       [-61.05179604,  -3.57062396],
       [-60.97117502,  -3.59384024],
       [-60.90342622,  -3.56336068],
       [-60.80502078,  -3.50629941],
       [-60.75351909,  -3.45050586],
       [-60.72245698,  -3.39425948],
       [-60.69176635,  -3.3269743 ],
       [-60.64421466,  -3.31443418],
       [-60.57262505,  -3.31452306],
       [-60.51454132,  -3.30721351],
       [-60.50184937,  -3.34625831],
       [-60.58341863,  -3.3389403 ],
       [-60.66584003,  -3.34476071],
       [-60.68655868,  -3.37488964],
       [-60.75565447,  -3.48512231],
       [-60.80730728,  -3.54463971],
       [-60.87472835,  -3.57689186],
       [-60.90334283,  -3.61322746],
       [-60.93298959,  -3.6220597 ],
       [-61.02141493,  -3.60596636],
       [-61.09150085,  -3.59352661],
       [-61.14960697,  -3.59895818],
       [-61.24046167,  -3.60740423],
       [-61.30598527,  -3.60419303],
       [-61.30429294,  -3.58074616]])
```

```
array([[ 846, 2339],
       [ 832, 2386],
       [ 821, 2440],
       [ 823, 2475],
       [ 806, 2499],
       [ 777, 2534],
       [ 754, 2549],
       [ 731, 2556],
       [ 703, 2563],
       [ 695, 2581],
       [ 689, 2609],
       [ 681, 2632],
       [ 695, 2641],
       [ 699, 2607],
       [ 708, 2574],
       [ 721, 2569],
       [ 767, 2552],
       [ 791, 2536],
       [ 808, 2512],
       [ 824, 2504],
       [ 830, 2493],
       [ 831, 2456],
       [ 833, 2426],
       [ 839, 2404],
       [ 850, 2368],
       [ 854, 2342],
       [ 846, 2339]])
```

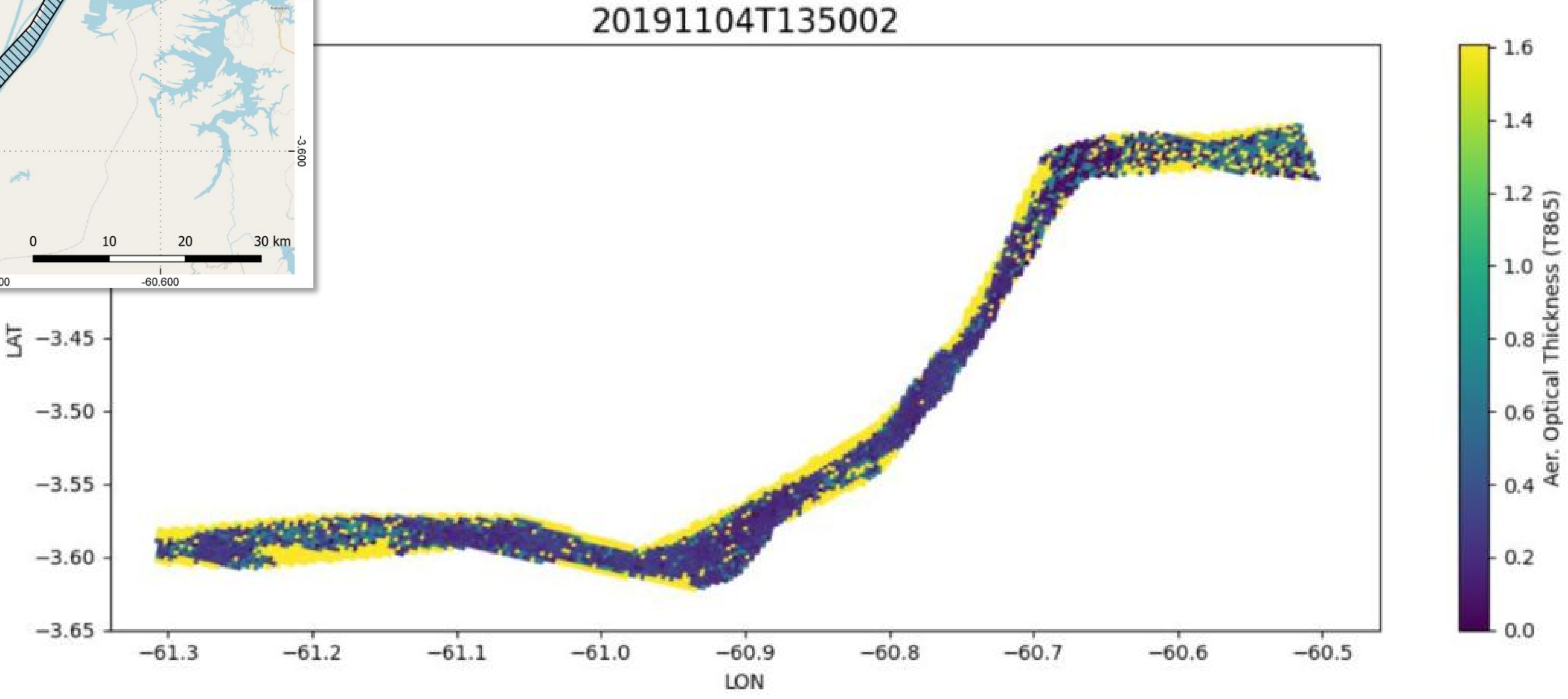
```
array([[-61.30429294,  -3.58074616],
       [-61.18973848,  -3.57023061],
       [-61.05179604,  -3.57062396],
       [-60.97117502,  -3.59384024],
       [-60.90342622,  -3.56336068],
       [-60.80502078,  -3.50629941],
       [-60.75351909,  -3.45050586],
       [-60.72245698,  -3.39425948],
       [-60.69176635,  -3.3269[illegible]43 ],
       [-60.64421466,  -3.314434[illegible]
       [-60.57262505,  -3.31452300[illegible]
       [-60.51454132,  -3.30721351[illegible]
       [-60.50184937,  -3.34625[illegible]
       [-60.58341863,  -3.3389[illegible]
       [-60.66584003,  -3.34476[illegible]
       [-60.68655868,  -3.3748896[illegible]
       [-60.75565447,  -3.4851223[illegible]
       [-60.80730728,  -3.544639[illegible]
       [-60.87472835,  -3.57[illegible]186],
       [-60.90334283,  -3.61322746],
       [-60.93298959,  -3.6220597 ],
       [-61.02141493,  -3.60596636],
       [-61.09150085,  -3.59352661],
       [-61.14960697,  -3.59895818],
       [-61.24046167,  -3.60740423],
       [-61.30598527,  -3.60419303],
       [-61.30429294,  -3.58074616]])
```

```
array([[ 846, 2339],
       [ 832, 2386],
       [ 821, 2440],
       [ 823, 2475],
       [ 806, 2499],
       [ 777, 2534],
       [ 754, 2549],
       [ 731, 2556],
       [ 703, 2563],
       [ 695, 2581],
       [illegible]609],
       [illegible]32],
       [illegible]641],
       [illegible]2607],
       [illegible] 2574],
       [illegible]1, 2569],
       [ 767, 2552],
       [ 791, 2536],
       [ 808, 2512],
       [ 824, 2504],
       [ 830, 2493],
       [ 831, 2456],
       [ 833, 2426],
       [ 839, 2404],
       [ 850, 2368],
       [ 854, 2342],
       [ 846, 2339]])
```

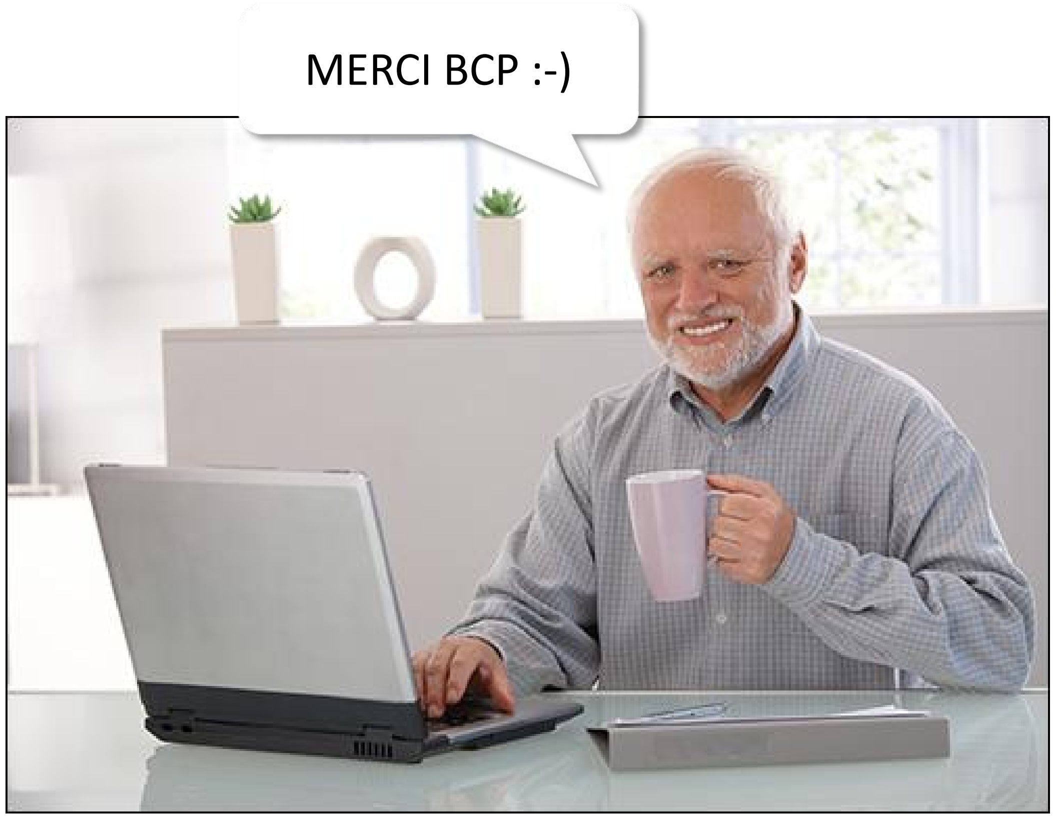
RESULT



-61.200
-61.000
-60.800
-60.600
-3.400
-3.600
N
Manacapuru
0
10
20
30 km



20191104T135002
LAT
−3.45
−3.50
−3.55
−3.60
−3.65
−61.3
−61.2
−61.1
−61.0
−60.9
−60.8
−60.7
−60.6
−60.5
LON
Aer. Optical Thickness (T865)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6



David Guimarães : dvdgmf@gmail.com
https://github.com/daviguima
https://github.com/hybam-dev/sen3r

https://hub.docker.com/r/hybam/hybam-dev

https://github.com/daviguima
https://github.com/hybam-dev/sen3r

https://pypi.org/project/sen3r/

https://github.com/hybam-dev/sen3r/blob/master/sen3r/nc_engine.py

GitHub

```python
class ParallelCoord:

    @staticmethod
    def vect_dist_subtraction(coord_pair, grid):
        subtraction = coord_pair - grid
        dist = np.linalg.norm(subtraction, axis=2)
        result = np.where(dist == dist.min())
        target_x_y = [result[0][0], result[1][0]]
        return target_x_y

    def parallel_get_xy_poly(self, lat_arr, lon_arr, polyline):
        # Stack LAT and LON in the Z axis
        grid = np.concatenate([lat_arr[..., None], lon_arr[..., None]], axis=2)

        # Polyline is a GeoJSON coordinate array
        polyline = polyline.squeeze()  # squeeze removes one of the dimensions of the array
        # https://numpy.org/doc/stable/reference/generated/numpy.squeeze.html

        # Generate a list containing the lat, lon coordinates for each point of the input poly
        coord_vect_pairs = []
        for i in range(polyline.shape[0]):
            coord_vect_pairs.append(np.array([polyline[i, 1], polyline[i, 0]]).reshape(1, 1, -1))

        # for future reference
        # https://stackoverflow.com/questions/6832554/multiprocessing-how-do-i-share-a-dict-among-multiple-processes
        cores = utils.get_available_cores()
        with concurrent.futures.ProcessPoolExecutor(max_workers=cores) as executor:
            try:
                result = list(executor.map(self.vect_dist_subtraction, coord_vect_pairs, [grid]*len(coord_vect_pairs)))

            except concurrent.futures.process.BrokenProcessPool as ex:
                print(f"{ex} This might be caused by limited system resources. "
                      f"Try increasing system memory or disable concurrent processing. ")

        return np.array(result)
```