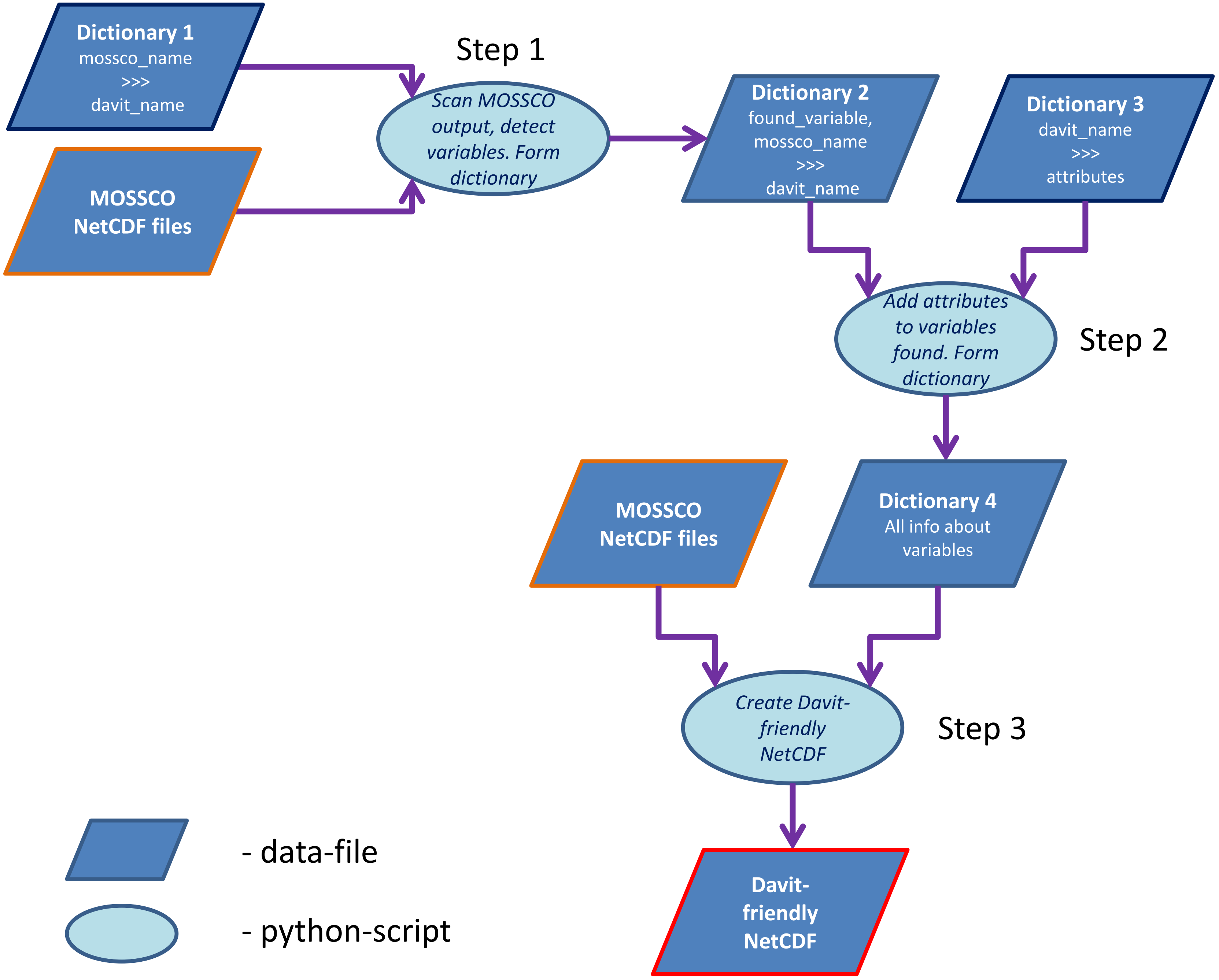

Dictionary 1
mossco_name
>>>
davit_name
MOSSCO NetCDF files
Step 1
Scan MOSSCO output, detect variables. Form dictionary
Dictionary 2
found_variable,
mossco_name
>>>
davit_name
Dictionary 3
davit_name
>>>
attributes
Add attributes to variables found. Form dictionary
Step 2
MOSSCO NetCDF files
Dictionary 4
All info about variables
Create Davit-friendly NetCDF
Step 3
Davit-friendly NetCDF
- data-file
- python-script

Dictionary 1. MOSSCO_NAME >>> DAVIT_NAME

Dictionary 1
mossco_name
>>>
davit_name

1. ASCII text file
2. Binds **variable name** from MOSSCO-output with the **correct name**
3. Created by user

Example:

```
// This file is a dictionary describing default connections between the variable names
// in BAW-format and mossco-format. The names are given in '' or in "". First stays
// the BAW name and after >>> follows the MOSSCO variable name. Comments may follow
// after //, spaces, newlines and tabs may be used freely for readibility
// -----------------------------------------------------------------------------------------------------------------

'Mesh2_face_Stroemungsgeschwindigkeit_x_2d'   >>> 'depth_averaged_x_velocity_in_water'
'Mesh2_face_Stroemungsgeschwindigkeit_y_2d'   >>> 'depth_averaged_y_velocity_in_water'
'Mesh2_face_Temperatur_3d'                    >>> 'temperature_in_water'
'Mesh2_face_depth_2d'                         >>> 'bathymetry'
'Mesh2_face_Wasserstand_2d'                   >>> 'water_depth_at_soil_surface'
'Mesh2_face_WaveHeight_2d'                    >>> 'wave_height'
```

Dictionary 2. VARIABLES TO INCLUDE

Dictionary 2
found_variable,
mossco_name
>>>
davit_name

1. ASCII text file
2. Lists variables that can be converted into new format
3. If variable name **was found** in dictionary 1 - shows new name after conversion (Davit-friendly name)
4. If variable name **was not found** in dictionary 1 (NOT_INCLUDED) variable **will not be added** to new file
5. Created by script
6. Can be modified by user

Example:

```
// format: "filename", "mossco variable name" >>> "corresponding baw format variable name"
// format: "filename", "mossco variable name" >>> NOT_INCLUDED
// spaces and tabs may be used freely for readability. Comments may follow after "//“
// ---------------------------------------------------------------------------------------------------------------

"\\Widar\home\netcdf_reference_3d.nc", "time"                                   >>> "nMesh2_data_time"
"\\Widar\home\netcdf_reference_3d.nc", "depth_averaged_x_velocity_in_water"     >>> "Mesh2_face_Stroemungsgeschwindigkeit_x_2d"
"\\Widar\home\netcdf_reference_3d.nc", "depth_averaged_y_velocity_in_water"     >>> "Mesh2_face_Stroemungsgeschwindigkeit_y_2d"
"\\Widar\home\netcdf_reference_3d.nc", "detritus-P_upward_flux_at_soil_surface"  >>>  NOT_INCLUDED
"\\Widar\home\netcdf_reference_3d.nc", "fast_detritus_C_upward_flux_at_soil_surface" >>>  NOT_INCLUDED
"\\Widar\home\netcdf_reference_3d.nc", "layer_height_at_soil_surface"           >>>  NOT_INCLUDED
"\\Widar\home\topo.nc",                "bathymetry"                             >>> "Mesh2_face_depth"
```

Dictionary 3. DESCRIPTION OF BAW VARIABLES

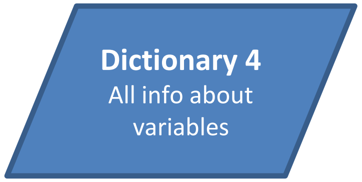
Dictionary 3
davit_name
>>>
attributes

1. ASCII text file
2. **CDL format**
3. **Describes** dtype, dimensions, attributes of Davit-friendly variables.
4. Created by user

Example:

```
double Mesh2_face_depth(nMesh2_time, nMesh2_face) ;
   Mesh2_face_depth:long_name = "Topographie" ;
   Mesh2_face_depth:units = "m" ;
   Mesh2_face_depth:name_id = 17 ;
   Mesh2_face_depth:_FillValue = 1.e+31 ;
   Mesh2_face_depth:cell_measures = "area: Mesh2_face_area" ;
   Mesh2_face_depth:cell_methods = "nMesh2_time: mean area: mean" ;
   Mesh2_face_depth:coordinates = "Mesh2_face_x Mesh2_face_y Mesh2_face_lon Mesh2_face_lat" ;
   Mesh2_face_depth:grid_mapping = "Mesh2_crs" ;
   Mesh2_face_depth:standard_name = "sea_floor_depth_below_geoid" ;
   Mesh2_face_depth:mesh = "Mesh2" ;
   Mesh2_face_depth:location = "face" ;
```

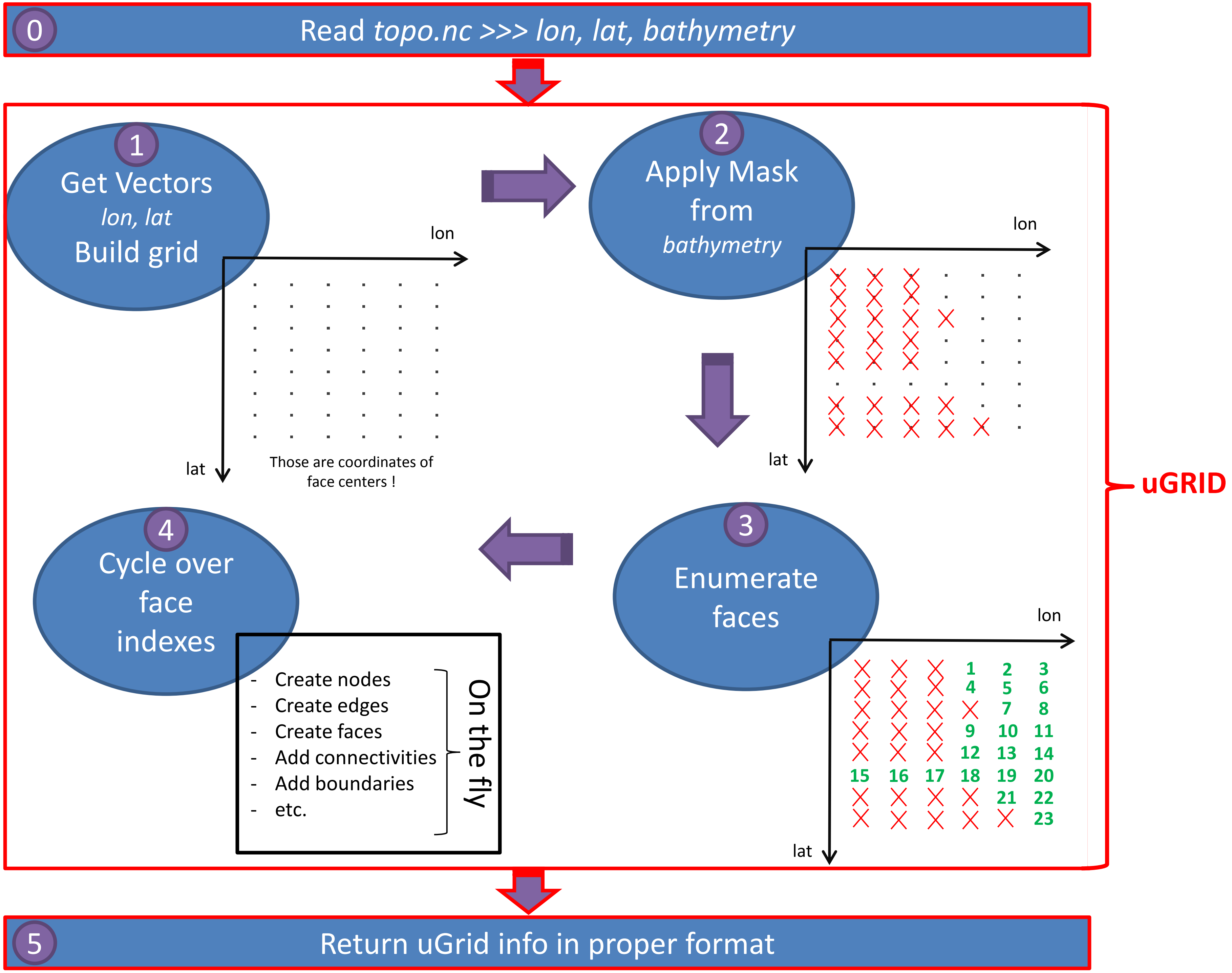
Dictionary 4. ALL INFO ABOUT CONVERTED VARIABLES



1. ASCII text file
2. **CDL format**
3. Combines info from Dictionary 2 and 3
4. **Describes** dtype, dimensions, attributes of Davit-friendly variables.
5. **Points** to exact data location for these variables (additional attributes)
6. Created by script
7. Can be modified by user

Example:

```
double Mesh2_face_depth (nMesh2_time, nMesh2_face) ;
    Mesh2_face_depth:long_name = "Topographie" ;
    Mesh2_face_depth:units = "m" ;
    Mesh2_face_depth:name_id = 17 ;
    Mesh2_face_depth:_FillValue = 1.e+31 ;
    Mesh2_face_depth:cell_measures = "area: Mesh2_face_area" ;
    Mesh2_face_depth:cell_methods = "nMesh2_time: mean area: mean" ;
    Mesh2_face_depth:coordinates = "Mesh2_face_x Mesh2_face_y Mesh2_face_lon Mesh2_face_lat" ;
    Mesh2_face_depth:grid_mapping = "Mesh2_crs" ;
    Mesh2_face_depth:standard_name = "sea_floor_depth_below_geoid" ;
    Mesh2_face_depth:mesh = "Mesh2" ;
    Mesh2_face_depth:location = "face" ;
    Mesh2_face_depth: _mossco_filename = "\\Widar\home\topo.nc" ;
    Mesh2_face_depth: _mossco_varname = "bathymetry" ;
```


0
Read *topo.nc >>> lon, lat, bathymetry*
1
Get Vectors
lon, lat
Build grid
lon
lat
Those are coordinates of face centers !
2
Apply Mask from
bathymetry
lon
lat
3
Enumerate faces
lon
lat
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
4
Cycle over face indexes
- Create nodes
- Create edges
- Create faces
- Add connectivities
- Add boundaries
- etc.
On the fly
uGRID
5
Return uGrid info in proper format

NODES

Init-Setters

Node.set_index()
Node.set_x()
Node.set_y()

Getters

Node.get_node_index()
Node.get_node_x()
Node.get_node_y()

EDGES

Init-Setters

Edge.set_index()
Edge.set_node1() <<< object *Node*
Edge.set_node2() <<< object *Node*

Getters

Edge.get_index()
Edge.get_edge_x()
Edge.get_edge_y()
Edge.get_edge_length()
Edge.get_node1() >>> object *Node*
Edge.get_node2() >>> object *Node*

FACES

Init-Setters

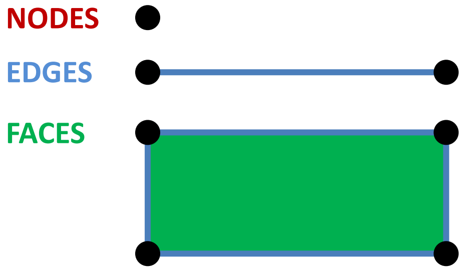
Face.set_index()
Face.set_nodes() <<< list of objects *Node* (3 or 4)

Getters

Face.get_index()
Face.get_face_x()
Face.get_face_y()
Face.get_face_center_x()
Face.get_face_center_y()
Face.get_area()
Face.get_nNodes() >>> int, number of nodes
Face.get_nodes() >>> list of objects *Node* (3 or 4)
Face.get_edges() >>> list of objects *Edge* (3 or 4)

uGRID

container with…



Init-Setters

Grid2D.set_x_vector()
Grid2D.set_y_vector()
Grid2D.set_mask()

Getters

Grid2D.get_nMaxMesh2_face_nodes()

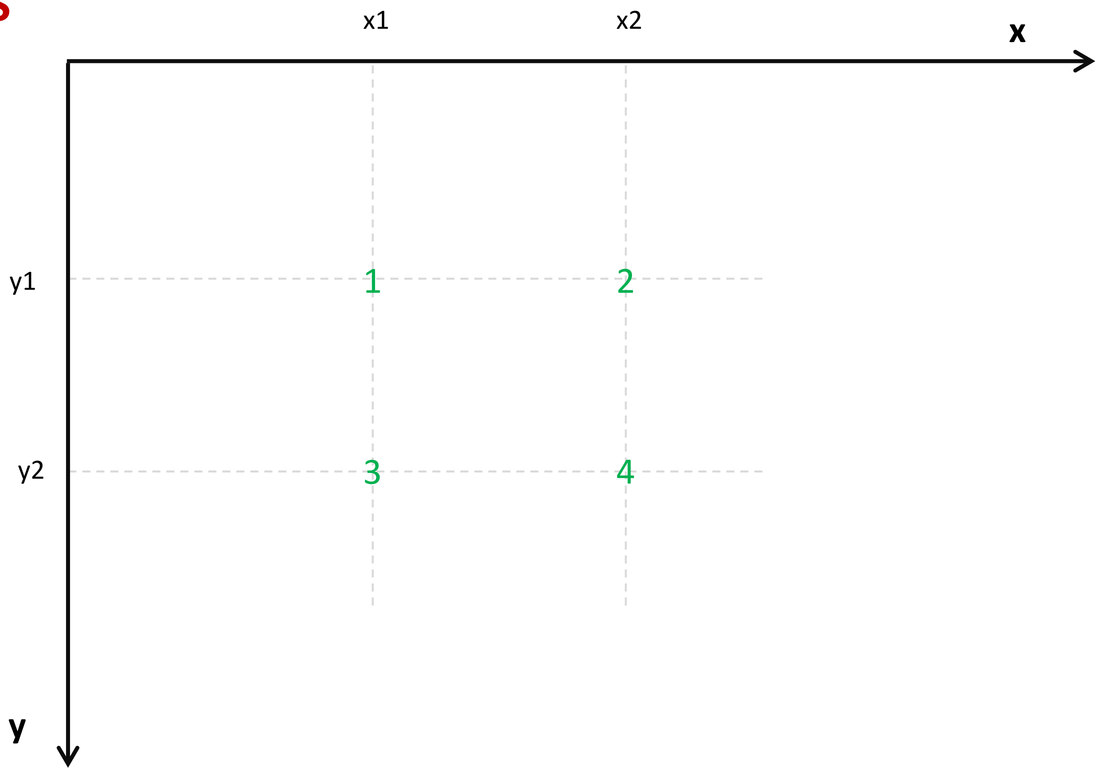
Grid2D.get_Mesh2_node_x()
Grid2D.get_Mesh2_node_y()
Grid2D.get_Mesh2_edge_x()
Grid2D.get_Mesh2_edge_y()
Grid2D.get_Mesh2_face_x()
Grid2D.get_Mesh2_face_y()
Grid2D.get_Mesh2_face_center_x()
Grid2D.get_Mesh2_face_center_y()

Grid2D.get_Mesh2_edge_nodes()
Grid2D.get_Mesh2_edge_faces()
Grid2D.get_Mesh2_face_nodes()
Grid2D.get_Mesh2_face_edges()
Grid2D.get_Mesh2_face_edges()

Grid2D.get_Mesh2_edge_bnd_x()
Grid2D.get_Mesh2_edge_bnd_y()
Grid2D.get_Mesh2_face_bnd_x()
Grid2D.get_Mesh2_face_bnd_y()

Example of block 4: „Cycle over face indexes“ (step 1)

NODES
EDGES
FACES

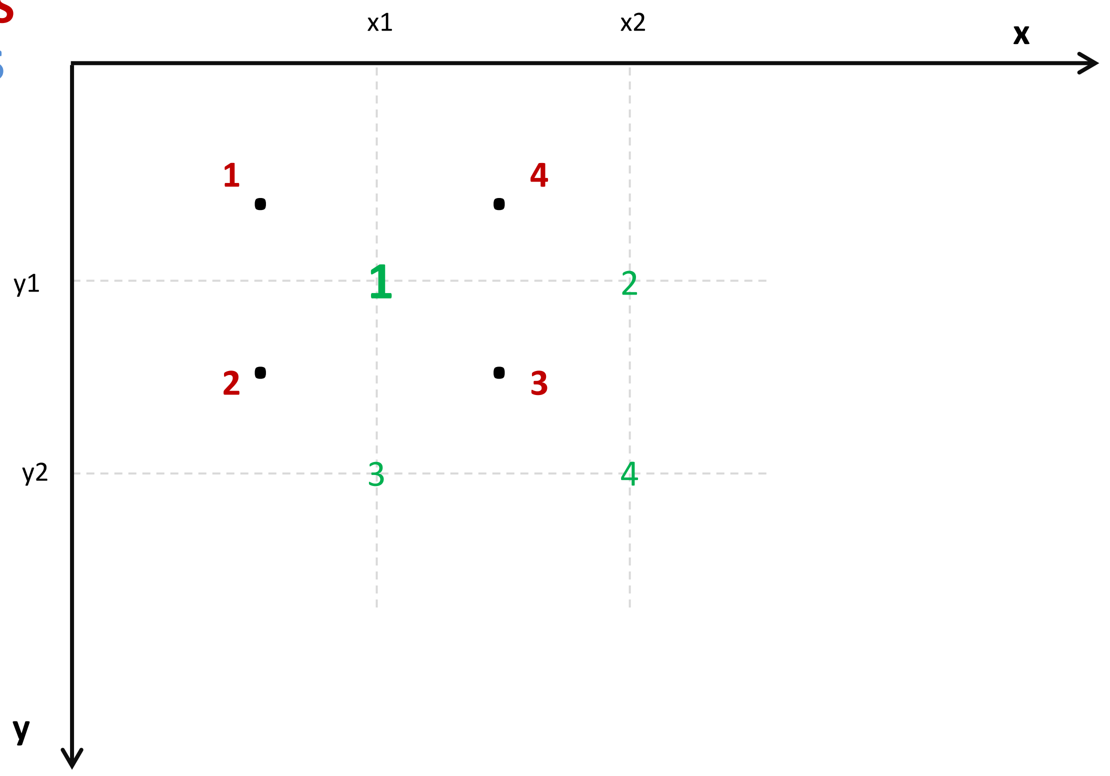


Example of block 4: „Cycle over face indexes“ (step 2)

NODES

EDGES

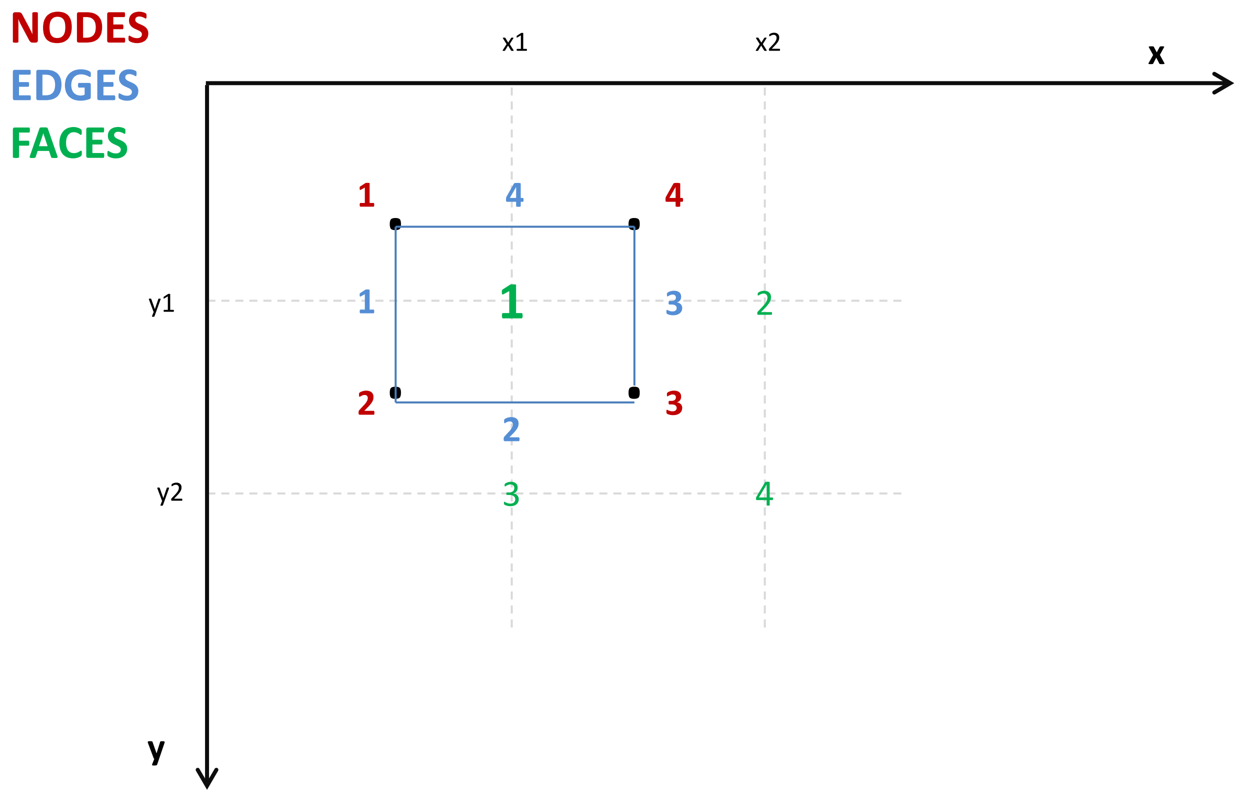
FACES



Connectivities

Mesh2_face_nodes(1) = [1,2,3,4]

Example of block 4: „Cycle over face indexes“ (step 3)



Connectivities

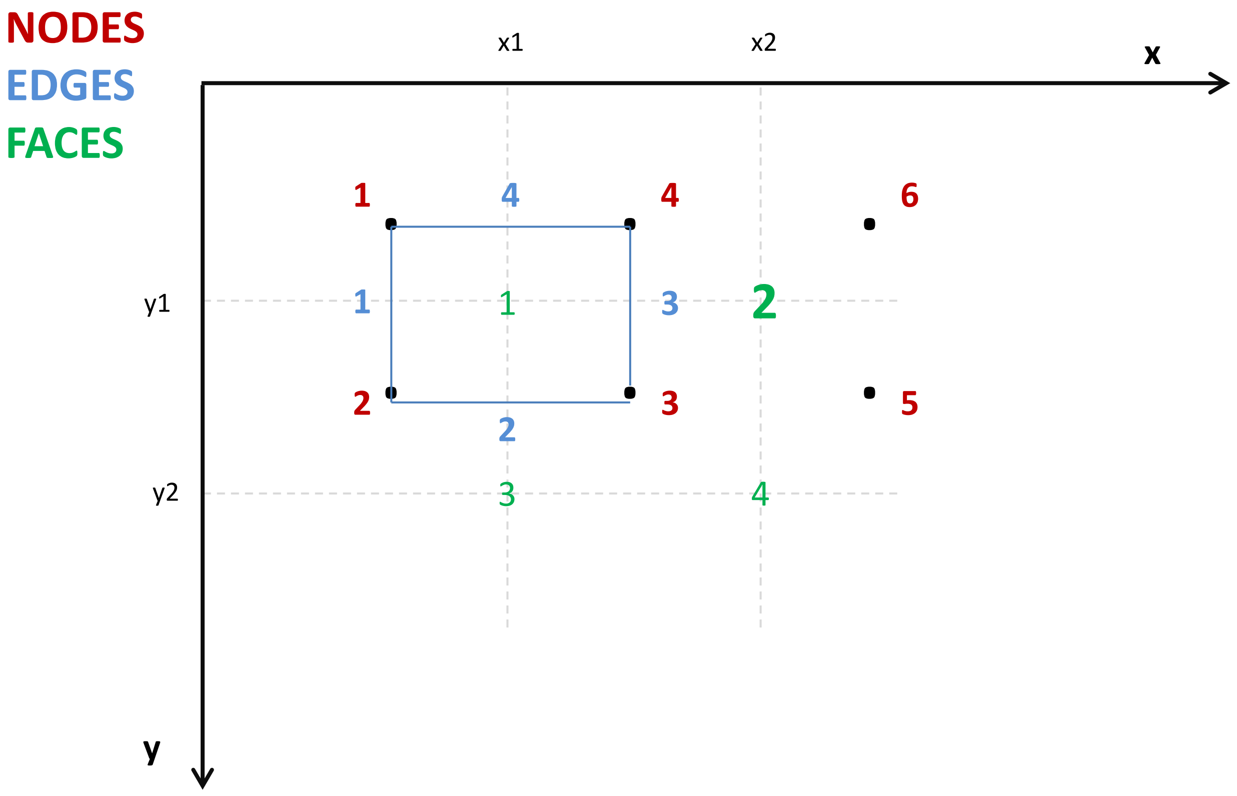
Mesh2_face_nodes(1) = [1,2,3,4]

Mesh2_face_edges(1) = [1,2,3,4]

Mesh2_edge_nodes(1) = [1,2]
Mesh2_edge_nodes(2) = [2,3]
Mesh2_edge_nodes(3) = [3,4]
Mesh2_edge_nodes(4) = [4,1]

Mesh2_edge_faces(1) = [1,-999]
Mesh2_edge_faces(2) = [1,3]
Mesh2_edge_faces(3) = [1,2]
Mesh2_edge_faces(4) = [1,-999]

Example of block 4: „Cycle over face indexes“ (step 4)



Connectivities

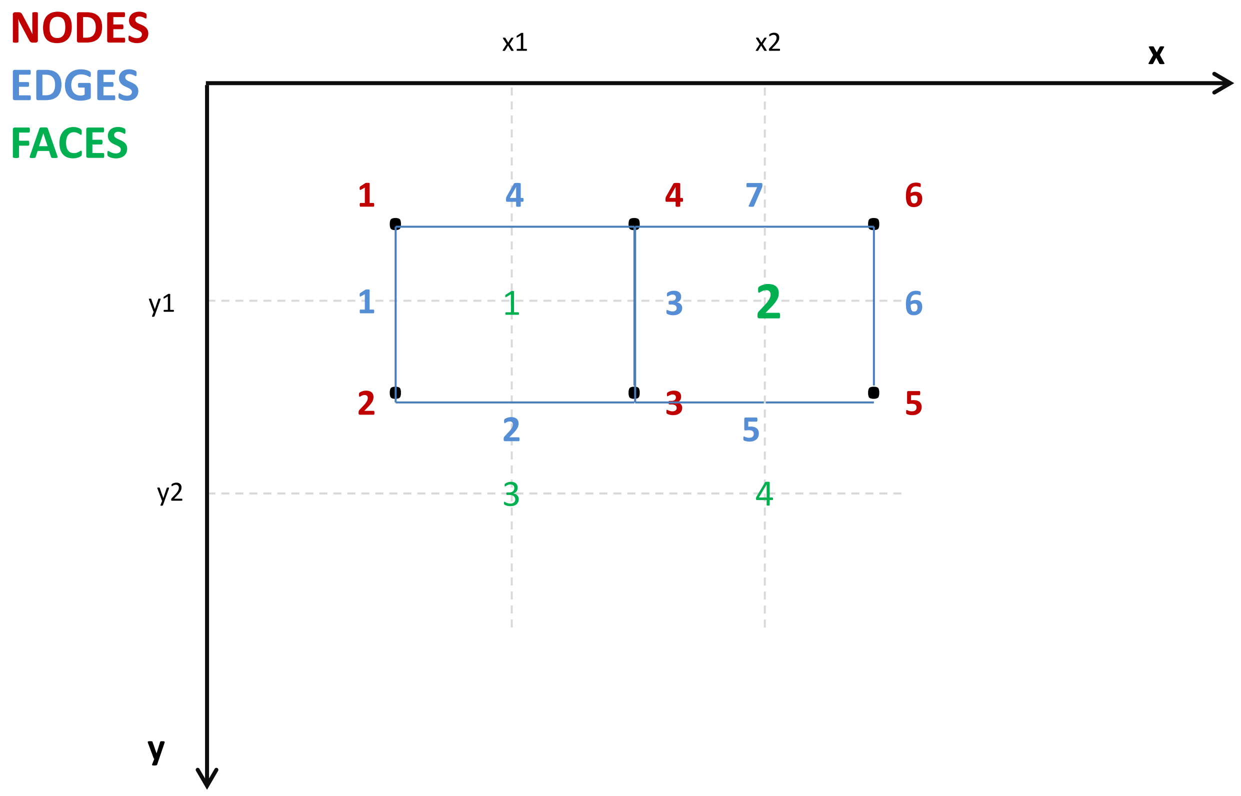
Mesh2_face_nodes(1) = [1,2,3,4]
Mesh2_face_nodes(2) = [4,3,5,6]

Mesh2_face_edges(1) = [1,2,3,4]

Mesh2_edge_nodes(1) = [1,2]
Mesh2_edge_nodes(2) = [2,3]
Mesh2_edge_nodes(3) = [3,4]
Mesh2_edge_nodes(4) = [4,1]

Mesh2_edge_faces(1) = [1,-999]
Mesh2_edge_faces(2) = [1,3]
Mesh2_edge_faces(3) = [1,2]
Mesh2_edge_faces(4) = [1,-999]

Example of block 4: „Cycle over face indexes“ (step 5)



Connectivities

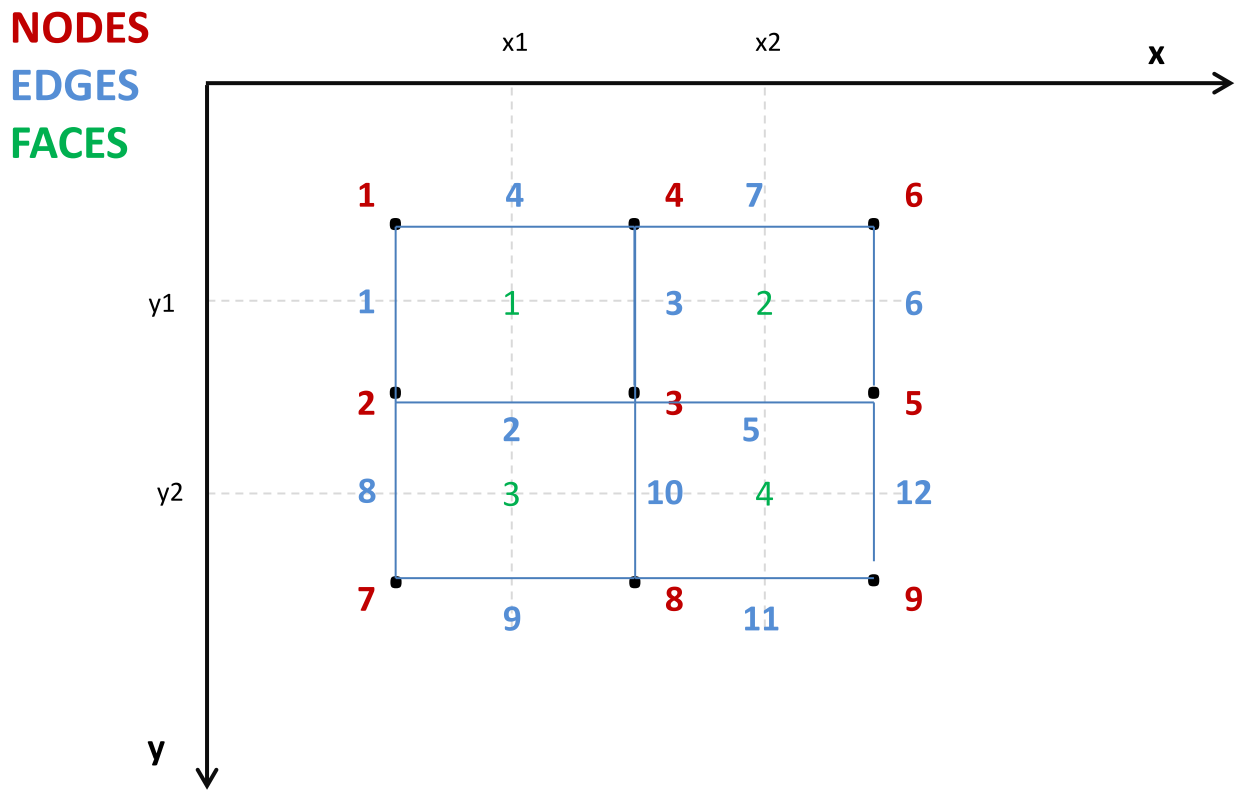
Mesh2_face_nodes(1) = [1,2,3,4]
Mesh2_face_nodes(2) = [4,3,5,6]

Mesh2_face_edges(1) = [1,2,3,4]
Mesh2_face_edges(2) = [3,5,6,7]

Mesh2_edge_nodes(1) = [1,2]
Mesh2_edge_nodes(2) = [2,3]
Mesh2_edge_nodes(3) = [3,4]
Mesh2_edge_nodes(4) = [4,1]
Mesh2_edge_nodes(5) = [3,5]
…

Mesh2_edge_faces(1) = [1,-999]
Mesh2_edge_faces(2) = [1,3]
Mesh2_edge_faces(3) = [1,2]
Mesh2_edge_faces(4) = [1,-999]
Mesh2_edge_faces(5) = [2,4]
…

Example of block 4: „Cycle over face indexes“ (step 6)



Connectivities

Mesh2_face_nodes(1) = [1,2,3,4]
Mesh2_face_nodes(2) = [4,3,5,6]
Mesh2_face_nodes(3) = [2,7,8,3]
Mesh2_face_nodes(4) = [3,8,9,4]

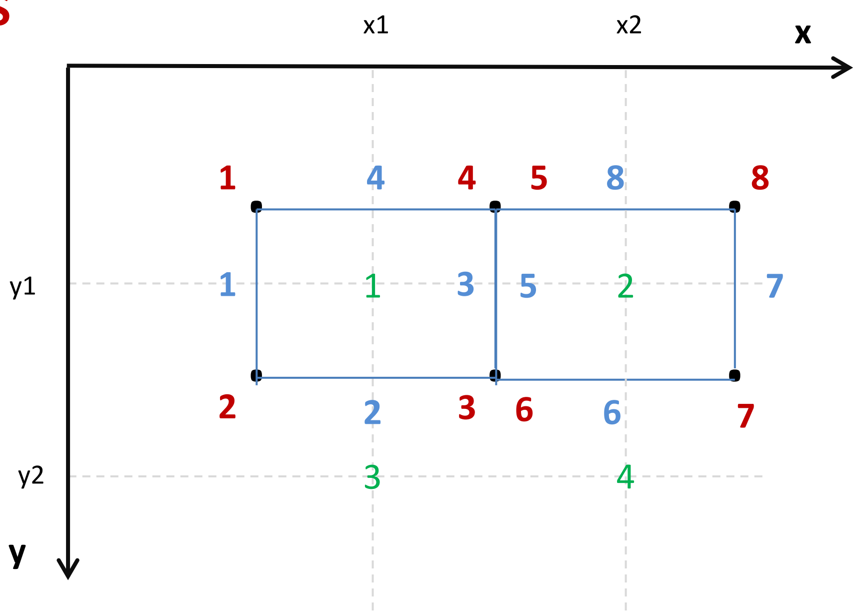
Mesh2_face_edges(1) = [1,2,3,4]
Mesh2_face_edges(2) = [3,5,6,7]
Mesh2_face_edges(3) = [8,9,10,2]
Mesh2_face_edges(4) = [10,11,12,5]

Mesh2_edge_nodes(1) = [1,2]
Mesh2_edge_nodes(2) = [2,3]
Mesh2_edge_nodes(3) = [3,4]
Mesh2_edge_nodes(4) = [4,1]
Mesh2_edge_nodes(5) = [3,5]
...
Mesh2_edge_nodes(12) = [9,5]

Mesh2_edge_faces(1) = [1,-999]
Mesh2_edge_faces(2) = [1,3]
Mesh2_edge_faces(3) = [1,2]
Mesh2_edge_faces(4) = [1,-999]
Mesh2_edge_faces(5) = [2,4]
...
Mesh2_edge_faces(12) = [4,-999]

Example of „double“ nodes/edges

NODES
EDGES
FACES



Determination of Zone, for finding neighbours

🟥 1	🟨 2	🟨 ---	🟨 ---	🟨 ---	🟨 ---	🟨 ---	🟨 3	🟨 4
🟪 5	🟩 6	🟩 7	🟩 ---	🟩 ---	🟩 ---	🟩 ---	🟩 8	🟦 9
🟪 10	🟩 11	🟩 12	🟩 ---	🟩 ---	🟩 13	🟩 14	🟩 15	🟦 16
🟪 17	🟩 18	🟩 19	🟩 20	🟩 21	🟩 22	🟩 23	🟩 ---	🟦 24
🟪 25	🟩 26	🟩 27	🟩 28	🟩 29	🟩 30	🟩 31	🟩 32	🟦 33

⬜ ---	Masked	🟪	Zone 3
🟥	Zone 1	🟩	Zone 4
🟨	Zone 2	🟦	Zone 5