# Description

## Introduction

A graphical interface with PyQt4 has been created to build this tool. PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in, which allows to create visual interfaces.

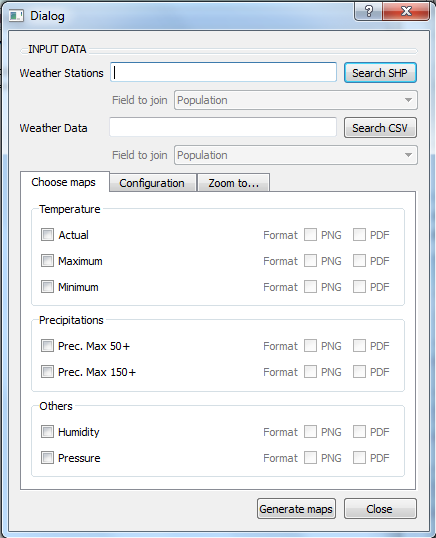
Once the interface has been built, an .ui file is generated, that can be edited with Python consoles to assign functions to the buttons.

## Steps

The following is a description of the processes followed to reach the tool:

### Building the interface

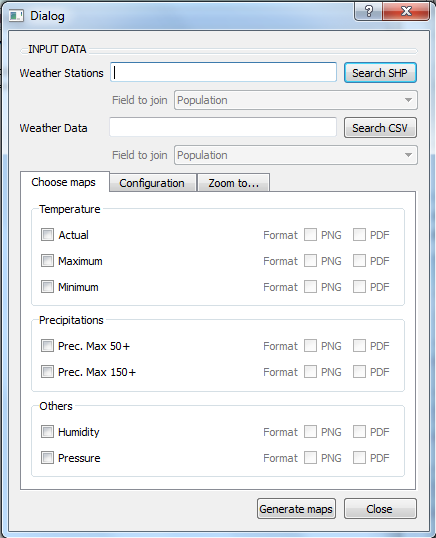
First, the interface has been created, using different kinds of widgets: layouts, spacers, buttons (Push, tool and radio buttons), check boxes… An easy-identification nomenclature of the widgets has been used to facilitate the implementation of functions with python.



Next are described the different parts of the interface:

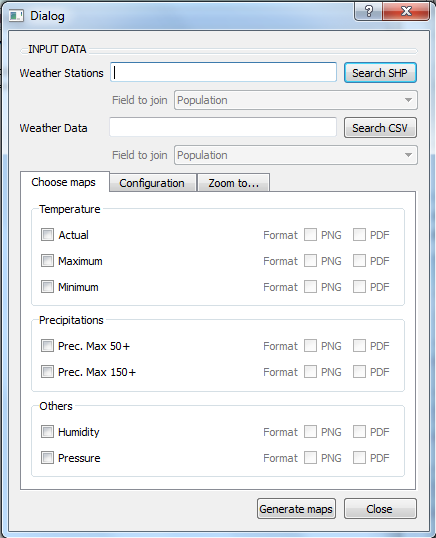
#### Input data

The program needs a shapefile with weather station location, and a csv data file of weather information of these stations. Then it’s necessary to choose the field of each input field to join them due to generate the weather maps.



#### Chose maps

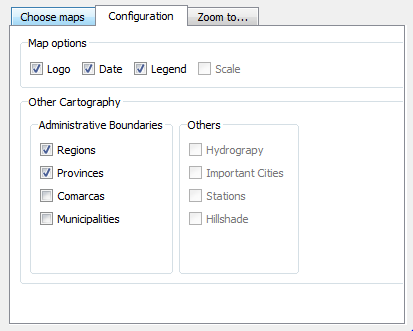
### The program allows choosing the different possible weather maps (temperature, precipitation, humidity and pressure), and the format of the output map (PDF or PNG)



#### Template configuration

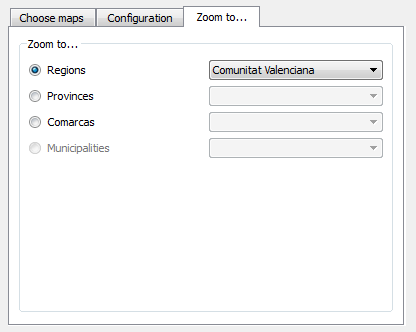
It’s possible to choose the configuration of the maps, choosing what to visualizate in the map such as legend, logo, or date of the maps generator.

Moreover, selecting the base map and other cartography is allowed. It’s possible to draw administrative boundaries layers (regions, provinces, comarcas and municipalities) and other interesting layers such as Weather Stations or hydrography.



#### Zooming options

It’s possible to zoom the map to a determinate zone. As default, maps will be generated in regional scale. However, it’s possible to zoom the map to a specific province or comarca.



#### Generate maps

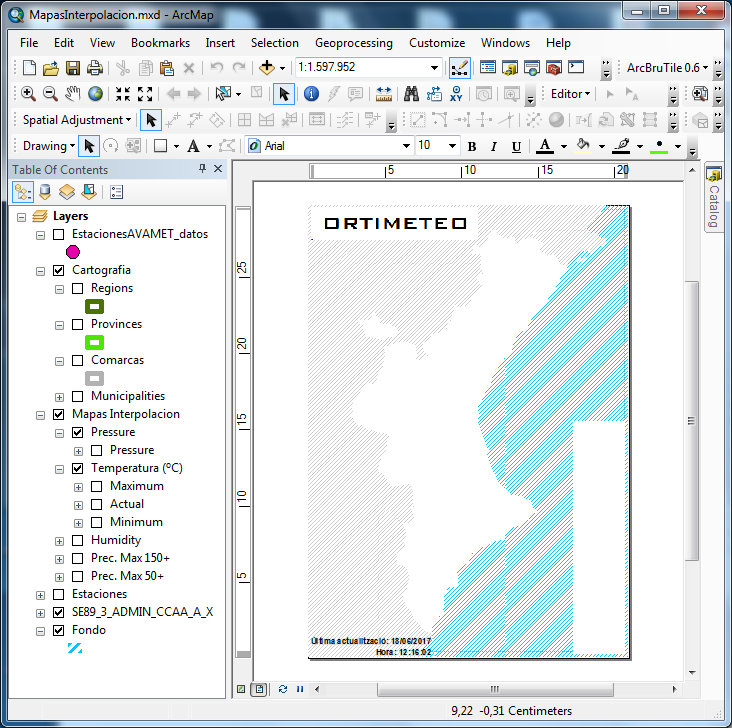
Once maps to generate, configuration and zooming options are defined, the program will generate the weater maps according to the user’s specifications.



### Creating map template

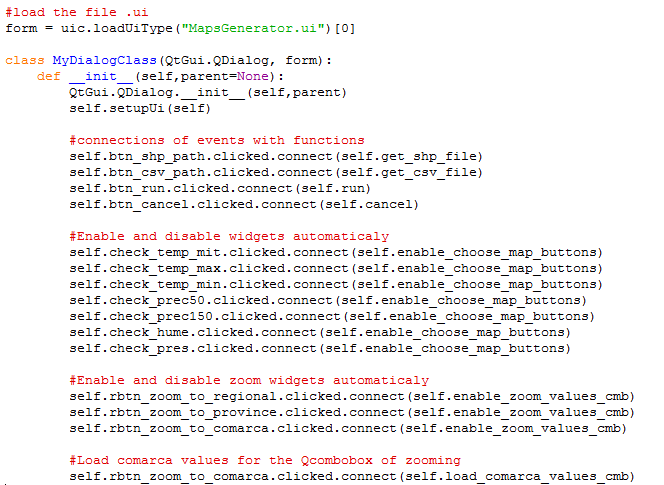
Once the interface is created, a print template map has been built, attending to the user specifications defined on the interface.

An easy-identification nomenclature of the map objects (logo, legend, date, layers) has been used, as in the software interface, to facilitate the implementation of functions with python.



### Functionality of the interface. Application of python functions.

First, a connection between button clicks and functions has been introduced.

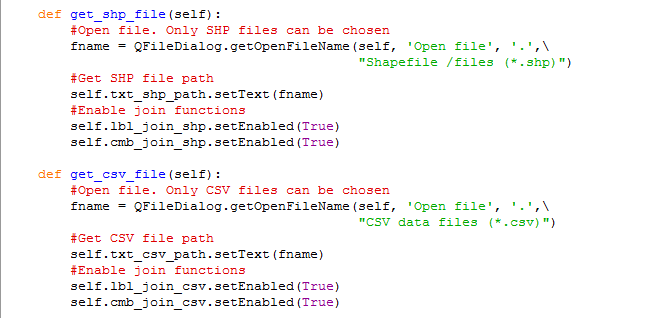


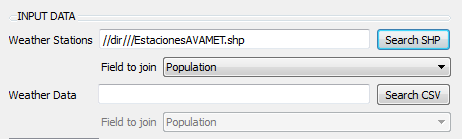
#### Button functions

Next, functions have been defined. There four functions. Two for loading the input data files, one for the “Generate maps” button, and other for the “Close” one.

##### Loading input data files

Those functions uses the QFileDialog class, that provides a dialog that allow users to select files or directories. Once input data files are selected, the program shows the path of these files, and allows you to introduce the join parameters.





##### Close

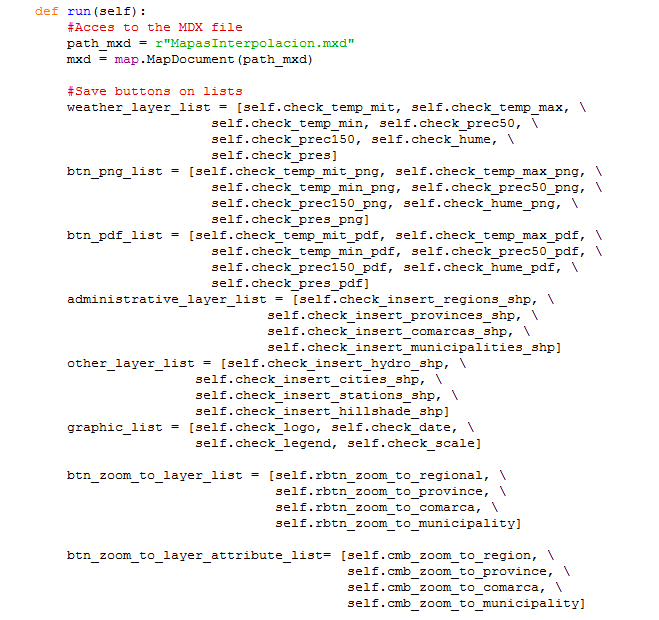
When clicking *Close*, the tool ends working.



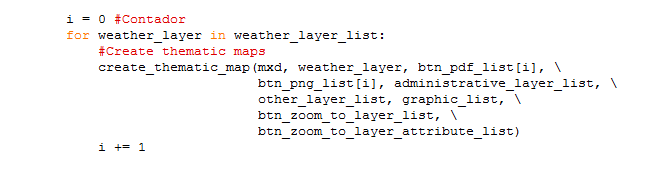
##### Generate maps

When clicking *Generate maps*, the function generates the map according to the data specifications.

First it’s necessary to access to the MXD file. Then, all widgets of the interface have been saved in different lists, according on their uses, in order to use them easily after.



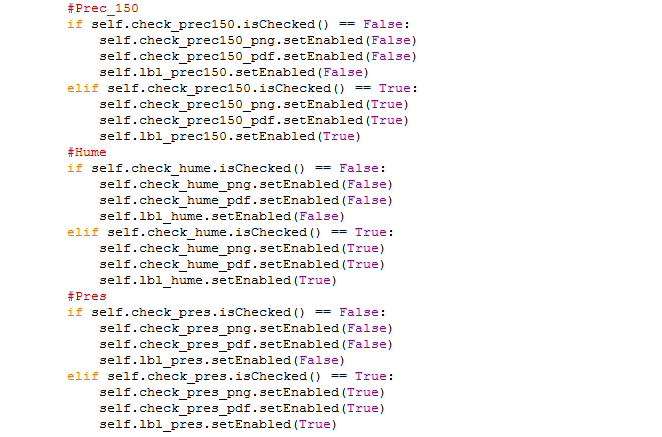
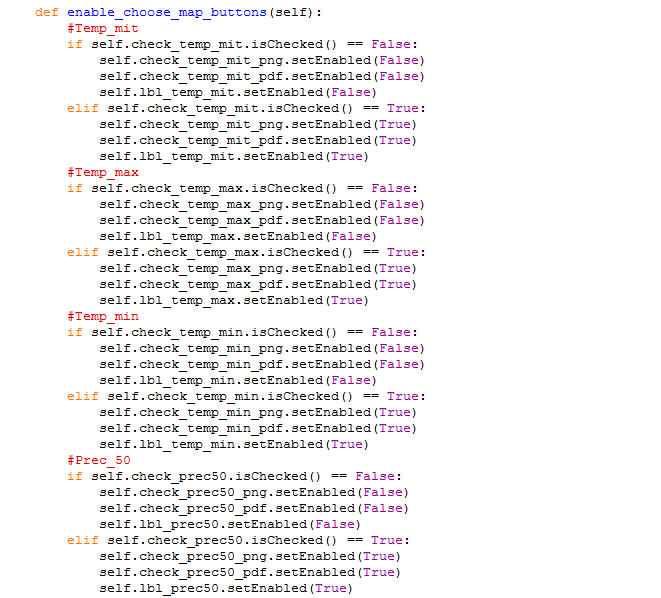
Then, creating thematic maps function will be applied to all possible maps, savend in the weather layer list. Create\_thematic\_map function is described after.

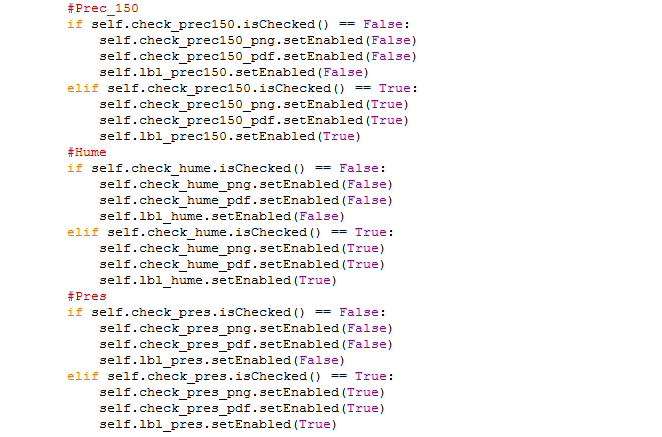


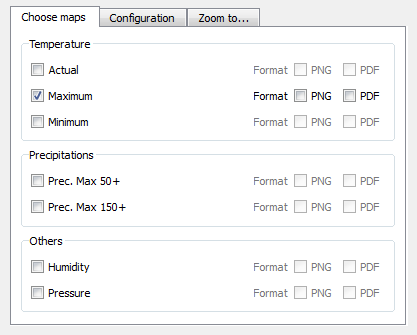
#### Other functions

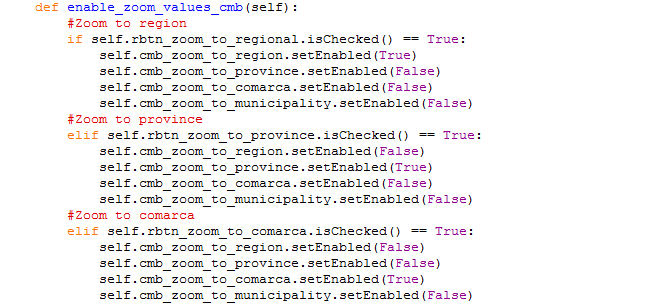
##### Enabling and disabling widgets

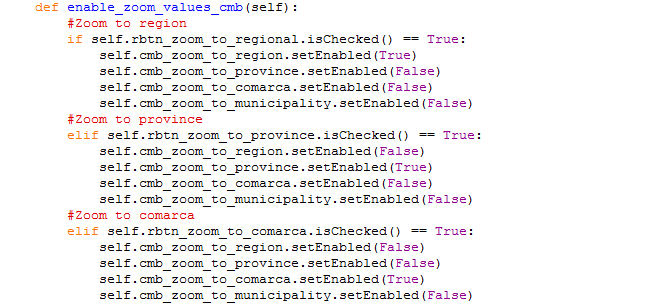
In orther to facilitate the order of the parameters to check in the program, two enabling and disabling widgets functions have been created. One for the format output map widgets, and other for the zooming ones.

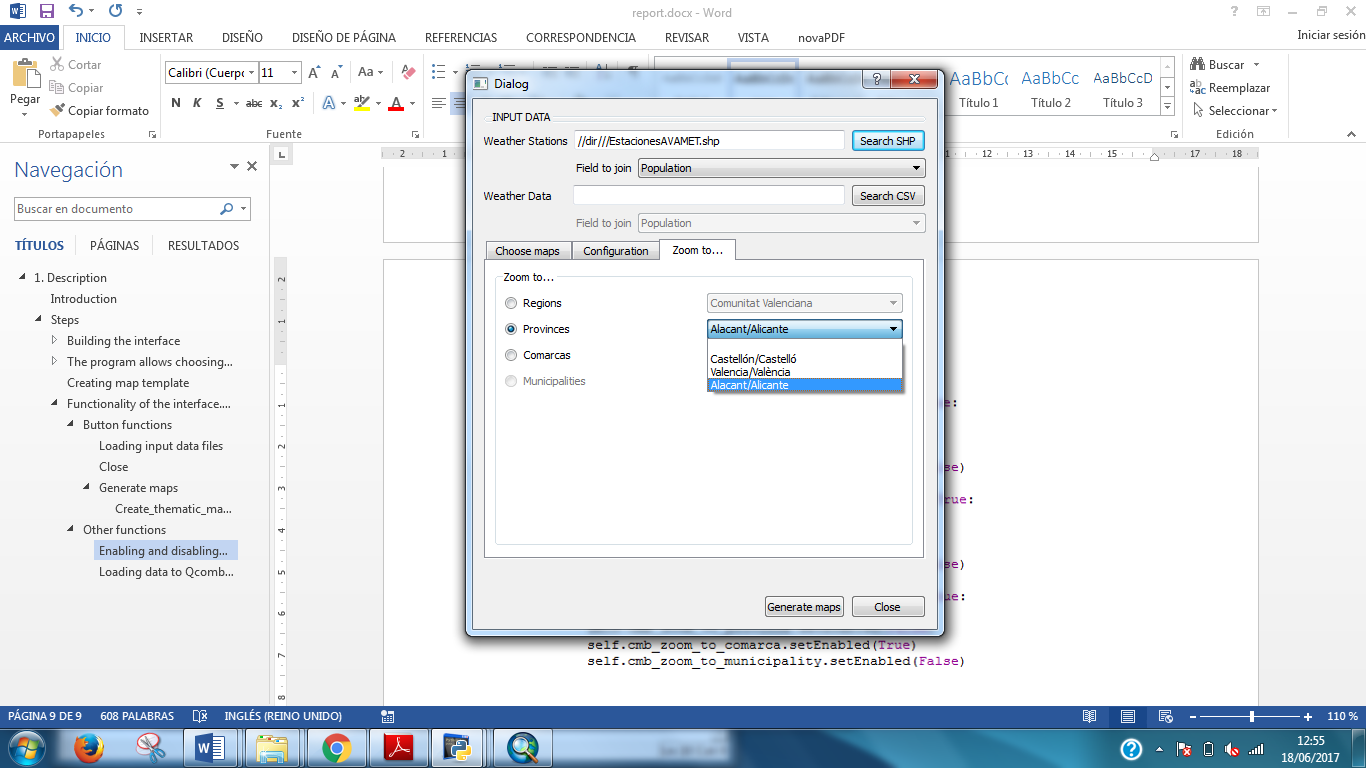






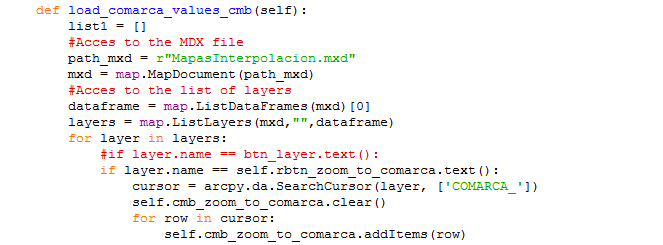


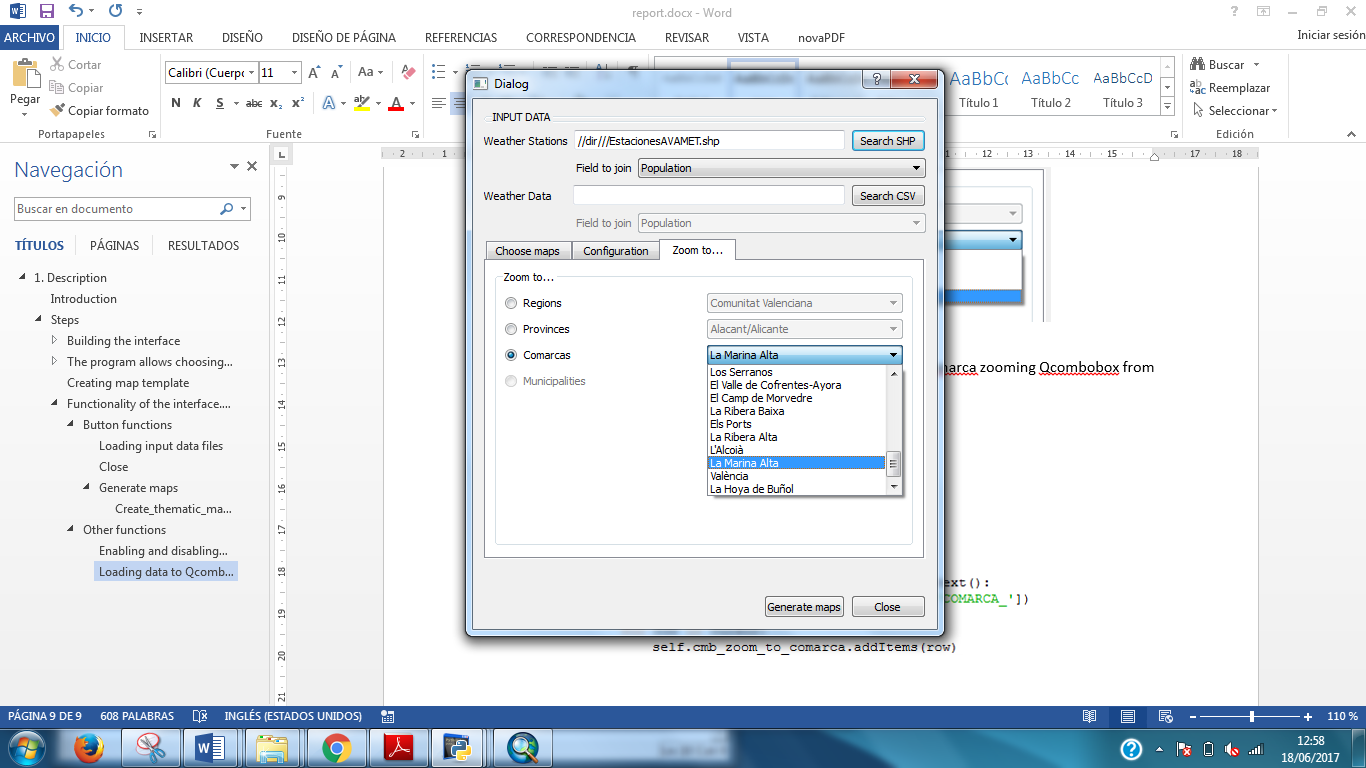




##### Loading data to Qcombobox automatically

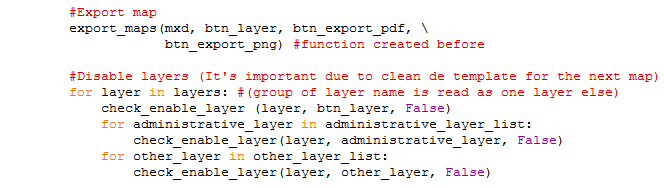
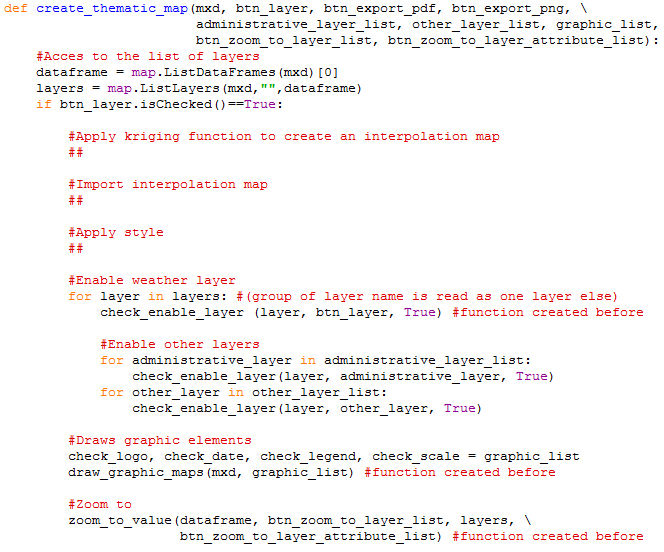
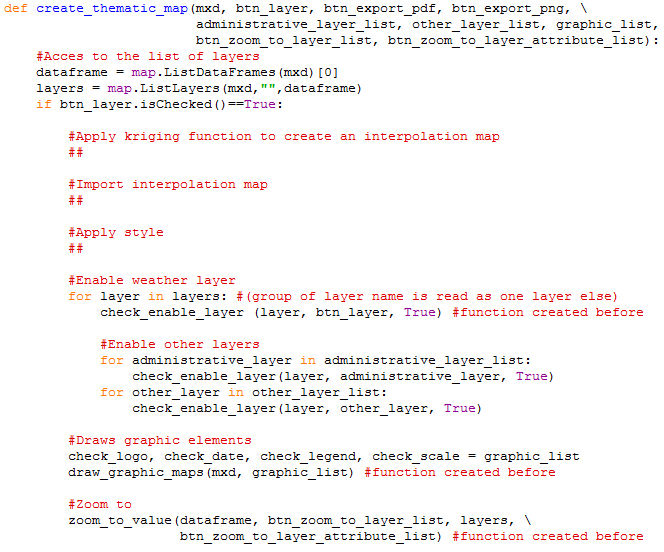
At last, a function to load automatically comarca names for the comarca zooming Qcombobox from the layer data has been created.





#### Create\_thematic\_map function

This functions is the one that has the code for generating the weather maps. This, in turn, uses other functions that are described below. The workflow for generating these maps has been the following:



##### Apply kriging function to create an interpolation map

Fdfsdfsdf

##### Import interpolation map

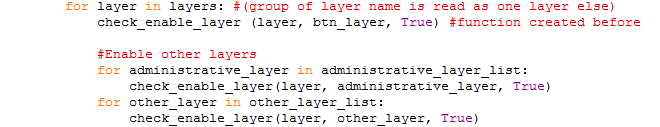
Djsfsdfsdf

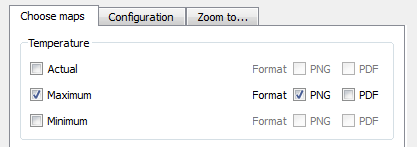
##### Apply style

Sdfsdfsdf

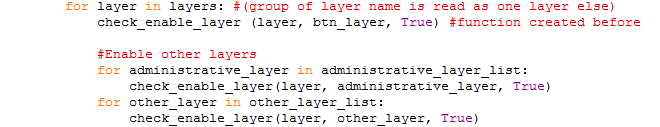
##### Enable weather layer

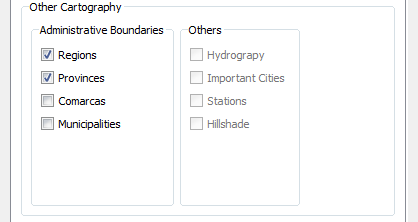
The program enables the weather layer chose by the user.



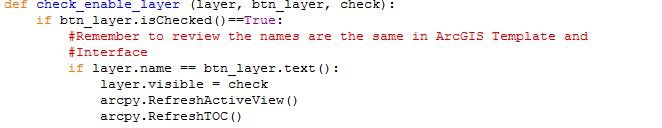


Moreover, enables other layers chose by the user in Configuration Window.



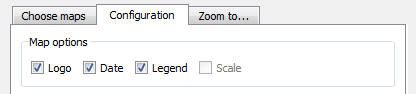


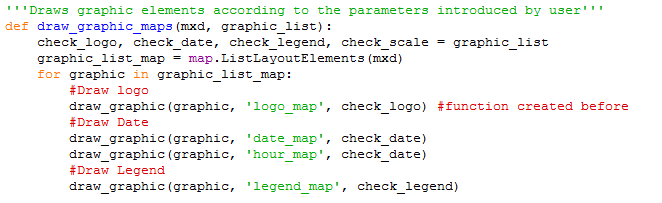
In order to do this, program uses *check\_enable\_layer* function. By default, no layer is enabled. This function enables the layer introduced by user on the interface.



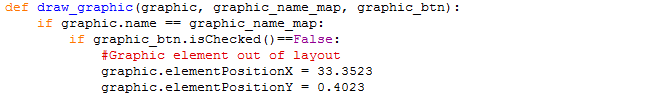
##### Draw graphic elements

Next Step is to draw graphic elements (logo, date, legend) chose by user in Configuration layer.



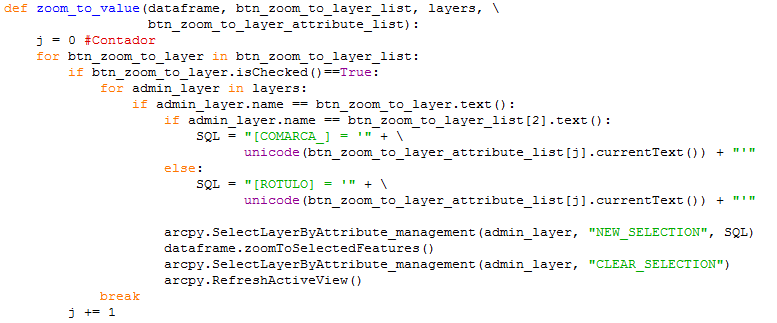


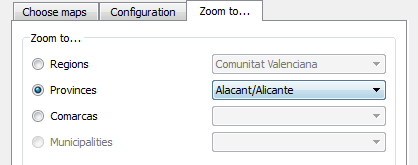
This function uses another function called *draw\_graphic* that draws the graphic element out of layout if user haven’t chosen it.



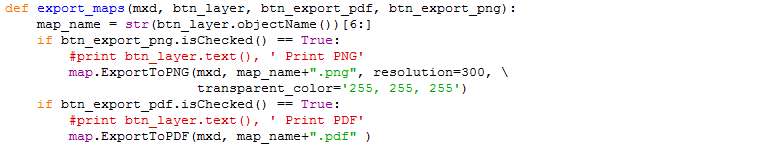
##### Zooming layout

When layer maps are generated and added to the template and graphic objects are enabled or disabled, program zooms the layout to a specific area, according to the user’s specifications.





Once output maps are all configured according to the parameters introduced in the interface, maps are exported in the formats selected.



##### Disable layers

And last, as programs allows you generating more than one map it’s important to restart the layer’s template configuration. That's why all layers are disabled again.

