

flags_iso3166svg

Rolf presents:



Fun with Flags - A homage to The Big Bang Theory.

flags_iso3166svg.py is a series of base64 encoded svg images of Country's Flags and Regional Flags, with a small set of country details, capital, languages etc and standard UTC timezone information, designed to work with wxPython. Emoji flag unicode values are included.

Developed and tested on Linux, wxPython 4.2.1 gtk3 wxWidgets 3.2.2.1

They are accessed via their ISO 3166 country codes

See: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes

The SVG images were a proverbial pain in the arse and there was me thinking svg images were the next big thing and standardised.

The SVG images of Flags out there, are a complete mishmash of structure and size. Sizes go from 100+ bytes for simple coloured flags to 2+ MegaBytes for regional flags, which seem to have a penchant for adding coats of arms, shields, castles, animals, you name it.

The structure of the files runs from pure SVG xml to Inkscape type structures that embed PNG images in base64.

Basically, svg files that exceeded 50Kb, have been excluded and replaced with Inkscape versions, simply to keep the size of flags_iso3166svg.py manageable.

flags_iso3166svg.py attempts to use Cairo via 'cairosvg' to convert the svg images and I recommend you install it, as wx.svg cannot handle some of the files and especially the Inkscape created ones.

Each flags program has a Use_Cairo (True/False) option, as cairosvg can be slow when loading a full database of flags, then it can be turned On when the loading has finished for detailed bigger flags.

Even if you turn it off initially, if Inkscape files are detected it is automatically turned on, try to cope with them.

pip3 install --upgrade cairosvg

Cairosvg on Windows machines

For those using the Windows operating system, for cairosvg to work, you will need to give python access to the cairo DLL's on your machine. These are installed by both Gimp and Inkscape, so look for them in their directories or install them in the current directory.

There several methods of providing access, which are discussed here:

<https://stackoverflow.com/questions/46265677/get-cairosvg-working-in-windows>

- install cairosvg (python -m pip install cairosvg)
- run import cairosvg in a script.
- if it works, you're set. otherwise (OSError: no library called "cairo" was found):
 - get a copy of libcairo-2.dll
 - I did this by installing from uniconvertor-2.0rc4-win64_headless.msi <https://sk1project.net/uc2/download/>
 - then look for where libcairo-2.dll was installed to.
 - say the path is C:\path\cairo\dlls\libcairo-2.dll
 - in your script add to the top (before import cairosvg)

```
import os
os.environ['path'] += r';C:\path\cairo\dlls'
```

Summing up:

In the file "flags_iso3166svg.py" add:

```
from base64 import b64decode
import wx.svg
import wx
```

```
import os
os.environ['path'] += r';C:\path\cairo\dlls'
```

```
try:
    import cairosvg
```

```
.....
```

The other option is to use:

```
os.add_dll_directory(path)
```

The DLL's reported to be required are these:












 libcairo-2.dll	1 327 Ko
 libfontconfig-1.dll	370 Ko
 libfreetype-6.dll	716 Ko
 libgcc_s_sjlj-1.dll	86 Ko
 libiconv-2.dll	1 000 Ko
 liblzma-5.dll	213 Ko
 libpixmap-1-0.dll	892 Ko
 libpng16-16.dll	320 Ko
 libwinpthread-1.dll	57 Ko
 libxml2-2.dll	1 674 Ko
 zlib1.dll	105 Ko

Table of Contents

Cairosvg on Windows machines.....	2
Dictionary structures.....	7
The catalog dictionary.....	8
The country dictionary.....	8
The lookup dictionary.....	8
The regional dictionary.....	8
The details dictionary.....	9
Regional Flags.....	10
Accessing the flags.....	10
Looking up the codes and country name.....	11
Accessing Regional flags.....	11
Accessing Country Details.....	12
Accessing Language Details.....	12
Accessing timezone names.....	12
Using timezone name.....	13
Accessing emoji flags.....	13
Accessing what you're after.....	14
Additional Helper Functions.....	15
Demonstration Programs.....	16
Demonstration images.....	17

These images are aimed at being used in Menus, BitmapComboBox, ListCtrl, TreeCtrl etc. wherever a bitmap can be inserted, to add some clarity and Va-Va-Voom, to your code.

There are currently 249 Country flags and 315 Regional flags (at last count) in flags_iso3166svg.py.

Regional flags are currently for Australia, Canada, Germany, Spain, France, Italy, Great Britain, Japan, Russia and the USA

You simply need to import flags_iso3166svg and you will be able to access the flag images.

For those that do not want to access, or to load all the flags, there are versions by continent, that only contain countries within that continental area, namely:

```
flags_iso3166svg_Africa.py
flags_iso3166svg_Americas.py
flags_iso3166svg_Asia.py
flags_iso3166svg_Europe.py
flags_iso3166svg_Oceania.py
flags_iso3166svg_NoRegions.py (is world flags but omits the regional flags)
```

The decision on which country belongs in which continent is based on the United Nations geoscheme:

https://en.wikipedia.org/wiki/United_Nations_geoscheme

Russia straddling 2 continents, is divided, roughly, by the Ural mountain range
<= UTC+04:00 is Europe whilst > UTC+04:00 is Asia

I'll admit now, the majority of the images were originally pinched from wikipedia.org, so cheers Wiki.

It is based on the ISO 3166-1 country codes.

The images are stored under 'b64_' and the ISO 3166-1 alpha-2 code and that code

is linked to it, so:

```
b64_ES = EmbeddedSvgImage(
    b'PD94bWwgdMVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz4KPHN2ZyB4bWxucz0iaHR0'
    b'cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSI5MDAiIGhlaWdodD0iNjAwIj4KPHJl'
    b'Y3Qgd2lkdGg9IjkwMCIGA GVpZ2h0PSI2MDAiIGZpbGw9IiNjNjBiMWUiLz4KPHJlY3Qgd2lk'
    b'dGg9IjkwMCIGA GVpZ2h0PSIzMdAiIHk9IjE1MCIGZmlsbD0iI2ZmYzQwMCIVPgo8L3N2Zz4=',
    'ES')
```

would be the entry for Spain's national flag.

Where the first item is the SVG in base64 and the 2nd item, is the iso2 code used internally to build a cache of a small sized version of the bitmap, for speed purposes.

If the small bitmap has already been generated for a menu for example, the cached bitmap is available, should you subsequently need to add it to a combobox or other widget.

Access to the Flag of Spain would be via `flags_iso3166svg.ES` which would provide the following properties:

- `Bitmap`,
- `ScaledBitmap`,
- `Data`,
- `Icon`,
- `Image`,
- `Png`,
- `MediumScaledBitmap`
- and `LargeScaledBitmap`

and methods:

- `GetBitmap(tx=0.0, ty=0.0, scale=1.0, width=-1, height=-1, stride=-1)`
this utilises `cairosvg.svg2png(width, height)`
or `wx.svg.ConvertToBitmap(tx=0.0, ty=0.0, scale=1.0, width=-1, height=-1, stride=-1)`
- `GetScaledBitmap(size=small_flag_size, window=None)`
this utilises `cairosvg.svg2png(size[0], size[1])`
or `wx.svg.ConvertToScaledBitmap(wx.Size(size), window)`
- `GetData()`
- `GetIcon()`
- `GetImage()`
- and `GetPng()`

```
small_flag_size = (24, 16)
medium_flag_size = (32, 24)
large_flag_size = (48, 32)
```


flags_iso3166svg also contains a list of entries called:

index

and 6 dictionaries:

- catalog,
- country,
- lookup,
- regional,
- details,
- languages,
- timezone

The index allows a simple check if an entry exists e.g.:

```
>>> if "USA" in flags_iso3166svg.index: True
True
>>> if "United States of America" in flags_iso3166svg.index: True
True
```

*** Warning: you made need to brush up on list comprehension over dictionaries

There is built in redundancy, to provide multiple ways to access the data

Dictionary structures

flags_iso3166svg.catalog	iso2 = image catalog["ES"] = ES iso3 = image catalog["ESP"] = ES (Not regional entries) name = image catalog["Spain"] = ES
flags_iso3166svg.country	name = (iso2, iso3, regional(True/False) country["Spain"] = ('ES', 'ESP', 0)
flags_iso3166svg.lookup	iso2 = (iso2, iso3, name, regional(True/False) lookup["ES"] = ('ES', 'ESP', 'Spain', 0) iso3 = (iso2, iso3, name, regional(True/False) lookup["ESP"] = ('ES', 'ESP', 'Spain', 0) name = (iso2, iso3, name, regional(True/False) lookup["Spain"] = ('ES', 'ESP', 'Spain', 0) capital = (iso2, iso3, name, regional(True/False) lookup["Madrid Spain"] = ('ES', 'ESP', 'Spain', 0) and rarely altname = (iso2, iso3, name, regional(True/False)

For regional entries in the above, the iso2 entry is a concatenation of country, underscore "_" and region code

e.g. US_AL = United State of America Alabama and there is no iso3 code, it is blank

```
flag_iso3166svg.regional regional_iso = (region_iso, region_name,  
regional(True))
```

```

flags_iso3166svg.regional["US_AL"] =
    ('US_AL', 'United States of America Alabama', 1)

flags_iso3166svg.regional['United States of America Alabama'] =
    ('US_AL', 'United States of America Alabama', 1)

flag_iso3166svg.details      iso2 = {"continent": continent name,
                                     "capital":   city name,
                                     "currency":  currency name",
                                     "web":       country's Top-Level domain,
                                     "lang":      list of language codes
                                     "tz":        timezone using UTC standard
time offset,
                                     "altname":   Alternate name, mainly for
Russian oblasts e.g. Severnaya Osetiya Respublika = North Ossetia
                                     }

```

See: https://en.wikipedia.org/wiki/Country_code_top-level_domain ("web")
https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes ("lang")

Timezone may differ if it is a regional code or a main entry
Where a main entry has multiple time zones, this value may be a range e.g.
"-08:00 | -03:30" for Canada from British Columbia in the West to Newfoundland
in the East the regional entry should contain just the timezone appropriate for
the region.

Where a region has more than one timezone, the main one is listed e.g.
most of British Columbia is in -08:00 whilst it's South East is in -07:00

```

flags_iso3166svg.languages  language code = language description
                          There are 489 language entries

```

```

flags_iso3166svg.timezone   iso2 = iana timezone name
                          There are 542 timezone entries

```

```

flags_iso3166.emoji         iso2 = emoji code
                          Stored as a bytes object requires decode() for display

```

The catalog dictionary

links the country's alpha-2 code, the alpha-3 code and the country's name to
the EmbeddedImage(..)

The country dictionary

links the country name with the alpha-2 code, the alpha-3 and a final
country/region parameter False(0) if this is a main country or True (1) if this
is a regional entry

The lookup dictionary

links the alpha-2 code, the alpha-3 code and the country's name and contains a
final country/region parameter False (0) if this is a main country or True (1)
if this is a regional entry

It includes entries for the capital city and rarely any alternative name.
(These are included to extend search facilities.)

The regional dictionary

links the iso3166-2 regional code with the country region name.
The dictionary key is the iso3166-1 alpha-2 code and the additional regional
code, which may be 2 or 3 chars

For example the code for Andalucia, Spain is ES-AN ES for Spain and AN for Andalucia, this is stored as ES_AN in the catalog.

The code for England, Great Britain is GB-ENG GB for Great Britain and ENG for England, this would be stored as GB_ENG

The details dictionary

links the country's alpha-2 code with another dictionary of country details, namely:

- which continent it is on;
- the name of the capital;
- the name of the currency;
- the Top_level domain code;
- timezone
- alternate name
- and the codes of languages used in the country.

The language codes conform ISO 639-1 and ISO 639-3

See: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
https://en.wikipedia.org/wiki/List_of_ISO_639-3_codes

The timezones are the iana timezone names

See: <https://www.iana.org/time-zones>

Links the iso country/regional code to the iana timezone name, or my best attempt :)

A full entry for Spain would look like this:

```
b64_ES = EmbeddedSvgImage(
    b'PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz4KPHN2ZyB4bWxucz0iaHR0
    b'cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSI5MDAiIGhlaWdodD0iNjAwIj4KPHJl
    b'Y3Qgd2lkdGg9IjkwMCIgaGVpZ2h0PSI2MDAiIGZpbGw9IiNjNjBiMWUiLz4KPHJlY3Qgd2lk
    b'dGg9IjkwMCIgaGVpZ2h0PSIzMDAiIHk9IjE1MCIgZmlsbD0iI2ZmYzQwMCivPgo8L3N2Zz4=
    'ES')

ES = b64_ES
index.append("ES")
index.append("ESP")
index.append("Spain")
catalog["ES"] = ES
catalog["ESP"] = ES
catalog["Spain"] = ES
country["Spain"] = ('ES', 'ESP', 0)
lookup["ES"] = ('ES', 'ESP', 'Spain', 0)
lookup["ESP"] = ('ES', 'ESP', 'Spain', 0)
lookup["Spain"] = ('ES', 'ESP', 'Spain', 0)
details["ES"] =
{"continent": "Europe", "capital": "Madrid", "currency": "Euro", "web": ".es", "lang": "e
s-ES | ca | gl | eu | oc", "tz": "+01:00", "altname": ""}
```

Regional Flags

Regions are a limited effort to include the flags of the major regions of key countries included for example would be the flags of US states and the regions of Spain.

The images are stored under the ISO 3166-1 alpha-2 code and the ISO 3166-2 code, so `US_AL = EmbeddedImage(..)`

There is no alpha-3 code so that is left blank and in both the country dictionary and missing in the regional dictionary.

Both contain a 1 (True) as the final parameter, denoting this is a regional flag

The full entry for the flag of Alabama, USA for example is this:

```
b64_US_AL = EmbeddedSvgImage(
    b'PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSI2MDAiIGhl'
    b'aWdodD0iNDAwIj48cGF0aCBmaWxsPSIjZmZmIiBkPSJNMCaWdYwMHY0MDBIMHoiLz48cGF0'
    b'aCBkPSJNMCwwIDYwMCw0MDBNMCw0MDAgNjAwLDAiIHNo9rZT0iI2IxMDAyMSIgc3Ryb2t'
    b'LXdpZHRoPSI2OCIVPjwvc3ZnPg==', 'US_AL')
```

```
US_AL = b64_US_AL
index.append("US_AL")
index.append("United States of America Alabama")
catalog["US_AL"] = US_AL
catalog["United States of America Alabama"] = US_AL
country["United States of America Alabama"] = ('US_AL', '', 1)
lookup["US_AL"] = ('US_AL', '', 'United States of America Alabama', 1)
lookup["United States of America Alabama"] = ('US_AL', '', 'United States of
America Alabama', 1)
details["US_AL"] =
{"continent": "Americas", "capital": "", "currency": "", "web": "", "lang": "", "tz": "-
06:00", "altname": ""}
regional["US_AL"] = ('US_AL', 'United States of America Alabama', 1)
regional["United States of America Alabama"] = ('US_AL', 'United States of
America Alabama', 1)
```

Accessing the flags

Other than:

```
>>> flags_iso3166svg.ES
<wx.lib.embeddedimage.EmbeddedSvgImage object at 0x7fd7f4a95de0>
```

Access via properties using the catalog via the iso alpha-2 code, iso alpha-3 code or the country's name:

```
>>> flags_iso3166svg.catalog['ES'].Bitmap
<wx._core.Bitmap object at 0x7fd7f4a2d090>
>>> flags_iso3166svg.catalog['ESP'].Bitmap
<wx._core.Bitmap object at 0x7fd7f4a2cdc0>
>>> flags_iso3166svg.catalog['Spain'].Bitmap
<wx._core.Bitmap object at 0x7fd7f4a2cee0>
```

ditto for Image:

```
flags_iso3166svg.catalog['Spain'].Image
<wx._core.Image object at 0x7fd7f4a2d090>
```

or Icon:

```
>>> flags_iso3166svg.catalog['Spain'].Icon
<wx._core.Icon object at 0x7fd7f4a2cee0>
```

or Data:

```
>>> flags_iso3166svg.catalog['Spain'].Data
b'<?xml version="1.0" encoding="utf-8"?>\n<svg
xmlns="http://www.w3.org/2000/svg" width="900" height="600">\n<rect width="900"
height="600" fill="#c60b1e"/>\n<rect width="900" height="300" y="150"
fill="#ffc400"/>\n</svg>'
```

The same is true of access via the methods:

```
>>> flags_iso3166svg.catalog['Spain'].GetBitmap()
<wx._core.Bitmap object at 0x7fd7f4a2d090>
>>> flags_iso3166svg.catalog['ES'].GetImage()
<wx._core.Image object at 0x7fd7f4a2cee0>
```

Looking up the codes and country name

Use the lookup dictionary to get iso2, iso3, country name and if it's a main entry or a regional map e.g. ES ESP Spain 0

```
>>> flags_iso3166svg.lookup.get("ES")
('ES', 'ESP', 'Spain', 0)
>>> flags_iso3166svg.lookup.get("ESP")
('ES', 'ESP', 'Spain', 0)
>>> flags_iso3166svg.lookup.get("Spain")
('ES', 'ESP', 'Spain', 0)
```

Accessing Regional flags

This also works for Regional entries (although you will notice the missing iso3 code and the 1 at the end, indicating a regional map):

```
>>> flags_iso3166svg.lookup.get("ES_AN")
('ES_AN', '', 'Spain Andalucia', 1)
>>> flags_iso3166svg.lookup.get("Spain Andalucia")
('ES_AN', '', 'Spain Andalucia', 1)
```

Access is the same:

```
>>> flags_iso3166svg.regional.get("ES_AN")
('ES_AN', 'Spain Andalucia', 1)
>>> flags_iso3166svg.regional.get("Spain Andalucia")
('ES_AN', 'Spain Andalucia', 1)

>>> flags_iso3166svg.ES_AN
< wx.lib.embeddedimage.EmbeddedSvgImage object at 0x7f7cb21ada80>
>>> flags_iso3166svg.catalog['ES_AN']
<wx.lib.embeddedimage.EmbeddedSvgImage object at 0x7f7cb21ada80>
>>> flags_iso3166svg.catalog['ES_AN'].GetBitmap()
<wx._core.Bitmap object at 0x7f7cb167c280>
>>> flags_iso3166svg.catalog['Spain Andalucia'].GetBitmap()
<wx._core.Bitmap object at 0x7f7cb167c310>
>>> flags_iso3166svg.catalog['Spain Andalucia'].GetImage()
<wx._core.Image object at 0x7f7cb167c280>
```

Accessing Country Details

```
>>> flags_iso3166svg.details["ES"]
{'continent': 'Europe', 'capital': 'Madrid', 'currency': 'Euro', 'web':
'.es', 'lang': 'es-ES | ca | gl | eu | oc', 'tz': '+01:00', 'altname': ''}
>>> flags_iso3166svg.details["ES"].get("currency")
'Euro'
>>> flags_iso3166svg.details["ES"].get("lang")
'es-ES | ca | gl | eu | oc'
>>> flags_iso3166svg.details["ES"].get("tz")
'+01:00'
```

Accessing Language Details

```
>>> flags_iso3166svg.language.get('es')
['Spanish - Castilian']
>>> flags_iso3166svg.language.get('wa')
['Walloon']
>>> flags_iso3166svg.language.get('wol')
['Wolof']
>>> flags_iso3166svg.language.get('oc')
['Occitan (post 1500)']
```

Accessing timezone names

The zoneinfo module was introduced with python3.9

<https://docs.python.org/3/library/zoneinfo.html>

```
>>> flags_iso3166.timezone.get('AF')
['Asia/Kabul']
>>> flags_iso3166.timezone.get('US_IL')
['America/Chicago']
>>> flags_iso3166.timezone.get('AU_NSW')
['Australia/Sydney']
```

The timezone dictionary is a clumsy attempt to tie the iso3166-1 and iso3166-2 country and region codes to the iana timezone names.

Where there is no direct link, I attempt to tie the iso3166-2 code to the nearest or most appropriate name.

I fully expect that somewhere in there, I will have made an error.

For countries with multiple timezones, that I didn't get around to subdividing, I've selected the commonest or the capital region.

Country entries that are unused, for whatever reason, have been retained but given a number as a suffix to keep them out of the way but still available for future use.

Using timezone name

```
from datetime import datetime, timezone
from zoneinfo import ZoneInfo
zone = flags_iso3166.timezone.get('AU_NSW')
datetime.now(ZoneInfo(zone[0]))

datetime.datetime(2023, 9, 11, 2, 48, 19, 153133,
tzinfo=zoneinfo.ZoneInfo(key='Australia/Sydney'))
```

Note:

The zoneinfo module does not directly provide time zone data, and instead pulls time zone information from the system time zone database or the first-party PyPI package [tzdata](https://pypi.org/project/tzdata/), if available. Some systems, including notably Windows systems, do not have an IANA database available, and so for projects targeting cross-platform compatibility that require time zone data, it is recommended to declare a dependency on tzdata.

<https://pypi.org/project/tzdata/>

For those having trouble with Zoneinfo i.e. those on Windows due to a difference between IANA timezones and the fact that Windows does not use IANA timezones according to this:

<https://stackoverflow.com/questions/62330675/get-local-time-zone-name-on-windows-python-3-9-zoneinfo>

Try using pytz instead i.e.

```
from datetime import datetime, timezone
import pytz
datetime.now(pytz.timezone('America/Los_Angeles')).strftime('%H:%M:%S - %d %b %Y')
```

Where:

```
pytz.timezone('America/Los_Angeles')
would be replaced with:
pytz.timezone(zone[0])
```

See above

Accessing emoji flags

```
emoji = flags_iso3166.emoji.get(iso2).decode()
```

emoji can then simply be used as a string like any other string

All countries have an **emoji** flag, very few sub.regions do except the UK, Canada and the USA.

This may change and the database has the proposed emoji unicode string for each one.

This is unicode "waving black flag" + the letter codes + the terminator 'cancel tag'.

Hopefully, if they get around to it, they will start working.

Accessing what you're after

For clarity with many of the examples below, the import statement for `flags_iso3166svg` will have been:

```
import flags_iso3166svg as flags
```

A list of countries excluding regions (the last entry is `False (0)` indicating it's not a regional entry)

```
countries = [item for item in flags.country.items() if not item[1][-1]]

('Andorra', ('AD', 'AND', 0))
('United Arab Emirates', ('AE', 'ARE', 0))
('Afghanistan', ('AF', 'AFG', 0))
('Antigua and Barbuda', ('AG', 'ATG', 0))
('Anguilla', ('AI', 'AIA', 0))
...
```

A list of regions for a country: e.g. for the US

```
us_regions = [item for item in flags.regional.items() if
item[0].startswith('US')]

('US_AK', ('US_AK', 'United States of America Alaska', 1))
('US_AL', ('US_AL', 'United States of America Alabama', 1))
('US_AR', ('US_AR', 'United States of America Arkansas', 1))
...
```

A list of regions for a country: e.g. for the US excluding the individual iso3166-2 entries like USAK

```
us_regions = [item for item in flags.regional.items() if
item[0].startswith('United States ')]

('United States of America Alaska', ('US_AK', 'United States of America Alaska',
1))
('United States of America Alabama', ('US_AL', 'United States of America
Alabama', 1))
('United States of America Arkansas', ('US_AR', 'United States of America
Arkansas', 1))
...
```

Alternatively you could use the country dictionary: (the last entry if `True` indicating it's a regional entry)

```
us_regions = [item for item in flags.country.items() if
item[0].startswith("United States") and item[1][-1]]
for i in us_regions: print(i)

('United States of America Alaska', ('US_AK', '', 1))
('United States of America Alabama', ('US_AL', '', 1))
('United States of America Arkansas', ('US_AR', '', 1))
...
```

Additional Helper Functions

`flags_iso3166svg.Regional_Entries`

returns a list of all entries in the country dictionary that are regional entries

So `len(flags_iso3166svg.Regional_Entries)` would tell you how many Regional entries there are

`flags_iso3166svg.Country_Entries`

returns a list of all entries in the country dictionary that are NOT regional entries

So `len(flags_iso3166svg.Country_Entries)` would tell you how many Countries there are

function `lookupiso2(value)` will return the iso2 code for any value

```
>>> flags.lookupiso2('Japan')
'JP'
>>> flags.lookupiso2('JPN')
'JP'
>>> flags.lookupiso2('JP')
'JP'
```

function `lookupiso3(value)` will return the iso2 code for any value

function `lookupname(value)` will return the country or region name for any value

function `isregional(value)` will return True/False for any value

function `lookupbasecountry(value)` will return the iso2 code for any value (if `isregional` returns True for example)

```
>>> import flags_iso3166 as flags
>>> flags.lookupbasecountry('United States of America Alabama')
('US', 'USA', 'United States of America', 0)
>>> flags.lookupbasecountry('US_AL')
('US', 'USA', 'United States of America', 0)
>>> flags.lookupbasecountry('JP_22')
('JP', 'JPN', 'Japan', 0)
>>> if flags.isregional('JP_17'):
    flags.lookupname('JP_17')
    flags.lookupbasecountry('JP_17')

'Japan Ishikawa'
('JP', 'JPN', 'Japan', 0)
```

Demonstration Programs

The demonstration programs (flagsdemosvg.py, flagsdemosvg2.py, flagsdemosvg3.py and flagsdemo4.py) attempt to give an example of many of the methods and properties, so even if much of the above, is gobbledigook, hopefully you will see how it can be used.

Flag images are demonstrated for Menus, Comboboxes, a Button, StaticBitmaps, a ListCtrl, a TreeCtrl and SuperToolTips.

Have fun with Flags!

Dear dear, will Sheldon ever forgive me? :)

Regards,
Rolf

rolfofsaxony@gmx.com

Anyone with insights into any misuses or misinterpretations I've made concerning svg files or their manipulation, feel free to point them out.
Also if you have access to better sources for svg flags, let me know.

As always, errors, bugs and insults, on a postcard please.

Demonstration images

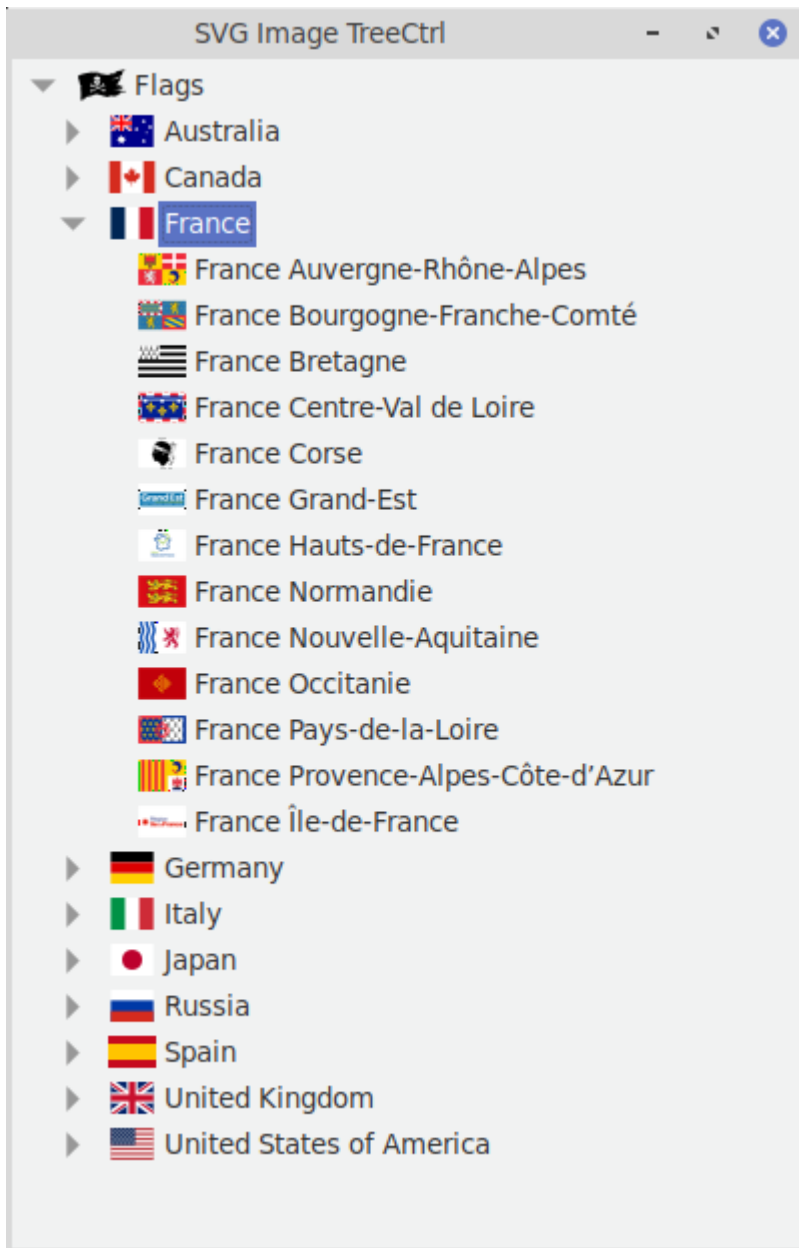
Demo 1



Demo 2



Demo 3



























Demo 4

Search flags_iso3166svg

Search

mal

Results

Country	Region	Code	Local Time	Capital	Aka	
 Falkland Islands (Malvinas)	 Falkland Islands (Malvinas)	FK	05:10:05 - 27 Jul 2024	Stanley		
 Equatorial Guinea	 Equatorial Guinea	GQ	09:10:05 - 27 Jul 2024	Malabo		
 Guatemala	 Guatemala	GT	02:10:05 - 27 Jul 2024	Guatemala City		
 Mali	 Mali	ML	08:10:05 - 27 Jul 2024	Bamako		
 Malta	 Malta	MT	10:10:05 - 27 Jul 2024	Valletta		
 Maldives	 Maldives	MV	13:10:05 - 27 Jul 2024	Male		
 Malawi	 Malawi	MW	10:10:05 - 27 Jul 2024	Lilongwe		
 Malaysia	 Malaysia	MY	16:10:05 - 27 Jul 2024	Kuala Lumpur		
 Russia	 Russia Yamalo-Nenetskiy avtonomnyy...	RU_YAN	13:10:05 - 27 Jul 2024	Moscow	Yamalo....	
 Somalia	 Somalia	SO	11:10:05 - 27 Jul 2024	Mogadishu		
 United States of America	 United States of America Virgin Islands	US_VI	04:10:05 - 27 Jul 2024	Charlotte Amalie		
 Virgin Islands US	 Virgin Islands US	VI	04:10:05 - 27 Jul 2024	Charlotte Amalie		