iso3166-2

Release 1.5.2

AJ McKenna

Mar 13, 2024

CONTENTS

iso3166-2 is a lightweight custom-built Python package, and accompanying API, that can be used to access all of the world's ISO 3166-2 subdivision data. Here, subdivision can be used interchangably with regions/states/provinces etc. Currently, the package and API supports data from 250 countries/territories, according to the ISO 3166-1 standard. The software uses another custom-built Python package called iso3166-updates to ensure all the subdivision data is **accurate, reliable and up-to-date**. The ISO 3166-2 was first published in 1998 and as of November 2023 there are **5,039** codes defined in it.

There are 7 main data attributes available for each subdivision within the **iso3166-2** software:

- Name - subdivsion name, as it is commonly known in English
- Local name - subdivision name in local language
- Code - ISO 3166-2 subdivision code
- Parent Code - subdivision's parent code
- Type - subdivision type, e.g. region, state, canton, parish etc
- Latitude/Longitude - subdivision coordinates
- Flag - subdivsion flag from iso3166-flag-icons repo

Note: A demo of the software and accompanying API is available here!

CHAPTER

ONE

LAST UPDATED

The ISO 3166-2 data was last updated on March 2023. A log of the latest ISO 3166-2 updates can be seen in the UPDATES.md file in the repository.

CHAPTER

TWO

LICENSE

iso3166-2 is distributed under the MIT License.

CHAPTER

THREE

CONTENTS

3.1 Usage

Below are some usage examples for the various functionalities of the **iso3166-2** software.

3.1.1 Installation

To use **iso3166-2**, firstly install via `pip`:

```
pip install iso3166-2
```

Alternatively, you can clone the repo and run `setup.py`:

```
git clone -b master https://github.com/amckenna41/iso3166-2.git
cd iso3166_2
python3 setup.py install
```

3.1.2 Accessing subdivision data per country using its ISO 3166-1 alpha codes

To access all subdivision data for a given country you should create an instance of the `ISO3166_2` class. This instance is subscriptable such that you can get the sought country subdivision data via its ISO 3166-1 **2 letter alpha-2, 3 letter alpha-3 or numeric country codes**. You can also search for a specific subdivision via its subdivision name.

For example, accessing all Canadian (CA, CAN, 124) subdivision data:

```
from iso3166_2 import *

#crete instance of ISO3166_2 class
iso = ISO3166_2()

iso["CA"]
#{'CA-AB': {'name': 'Alberta', 'localName': 'Alberta', 'type': 'Province', 'parentCode': None,...}

#CA-AB Alberta
ca_alberta = iso["CA"]["CA-AB"]
ca_alberta.name #Alberta
ca_alberta.type #Province
ca_alberta.flag_url #https://github.com/amckenna41/iso3166-flag-icons/blob/main/iso3166-
↪2-icons/CA/CA-AB.svg
```

(continues on next page)

(continued from previous page)

```
#CA-MB Manitoba
ca_manitoba = iso["CA"]["CA-MB"]
ca_manitoba.name #Manitoba
ca_manitoba.parentCode #null
ca_manitoba.localName #Manitoba

#CA-NS Nova Scotia
ca_nova_scotia = iso["CA"]["CA-NS"]
ca_nova_scotia.name #Nova Scotia
ca_nova_scotia.type #Province
ca_nova_scotia.flagUrl #https://github.com/amckenna41/iso3166-flag-icons/blob/main/
↪iso3166-2-icons/CA/CA-NS.svg
```

Accessing all Danish (DK, DNK, 208) subdivision data:

```
from iso3166_2 import *

#crete instance of ISO3166_2 class
iso = ISO3166_2()

iso["DNK"]
#{'DK-81': {'name': 'Nordjylland', 'localName': 'Nordjylland', 'type': 'Region', 'parentCode':␣
↪None,...}

#DK-81 Nordjylland
dk_nordjylland = iso["DNK"]["DK-81"]
dk_nordjylland.name #Nordjylland
dk_nordjylland.latLng #[56.831, 9.493]
dk_nordjylland.type #Region

#DK-84 Hovedstaden
dk_hovedstaden = iso["DNK"]["DK-84"]
dk_hovedstaden.name #Hovedstaden
dk_hovedstaden.flagUrl #https://github.com/amckenna41/iso3166-flag-icons/blob/main/
↪iso3166-2-icons/DK/DK-84.svg
dk_hovedstaden.parentCode #null

#DK-85 Sjælland
dk_sjalland = iso["DK"]["DNK-85"]
dk_sjalland.name #Sjælland
dk_sjalland.type #Region
dk_sjalland.flagUrl #https://github.com/amckenna41/iso3166-flag-icons/blob/main/iso3166-
↪2-icons/DK/DK-85.svg
```

Accessing all Estonian (EE, EST, 233) subdivision data:

```
from iso3166_2 import *

#crete instance of ISO3166_2 class
iso = ISO3166_2()
```

(continues on next page)

(continued from previous page)

```
iso["233"]
#{'EE-37': {'name': 'Harjumaa', 'localName': 'Harjumaa', 'type': 'County', 'parentCode': None,...}

#EE-39 Hiiumaa
ee_hiiumaa = iso["233"]["EE-39"]
ee_hiiumaa.name #Hiiumaa
ee_hiiumaa.localName #Hiiumaa
ee_hiiumaa.latLng #[58.924, 22.592]

#EE-130 Alutaguse
ee_alutaguse = iso["233"]["EE-130"]
ee_alutaguse.name #Alutaguse
ee_alutaguse.parentCode #EE-45
ee_alutaguse.flagUrl #https://github.com/amckenna41/iso3166-flag-icons/blob/main/iso3166-
↪2-icons/EE/EE-130.svg

#EE-338 Kose
ee_kose = iso["233"]["EE-338"]
ee_kose.name #Kose
ee_kose.type #Rural municipality
ee_kose.parentCode #EE-37
```

3.1.3 Accessing subdivision data for all countries

To access ALL subdivision data for ALL available countries, you need to access the `all` attribute within the object instance of the `ISO3166_2` class. You can then access an individual country's subdivision data by passing in the sought ISO 3166-1 **2 letter alpha-2, 3 letter alpha-3 or numeric country code**.

```
import iso3166_2 as iso

all_data = iso.country.all

all_data["LU"] #all subdivision data for Luxembourg
all_data["PW"] #all subdivision data for Palau
all_data["TUV"] #all subdivision data for Tuvalu
all_data["UKR"] #all subdivision data for Ukraine
all_data["876"] #all subdivision data for Wallis & Futuna
all_data["716"] #all subdivision data for Zimbabwe
```

3.1.4 Adding custom subdivisions

Add or delete a custom subdivision to an existing country on the main **iso3166-2.json** object. The purpose of this functionality is similar to that of the user-assigned code elements of the ISO 3166-1 standard. Custom subdivisions and subdivision codes can be used for in-house/bespoke applications that are using the **iso3166-2** software but require additional custom subdivisions to be represented. If the input custom subdivision code already exists then an error will be raised, otherwise it will be appended to the object.

If the added subdivision is required to be deleted from the object, then you can call the same function with the alpha-2 and subdivision codes' parameters but also setting the `delete` parameter to 1/True.

```
from iso3166_2 import *

#adding custom Belfast province to Ireland (IE)
iso.custom_subdivision("IE", "IE-BF", name="Belfast", local_name="Béal Feirste", type=
↪"province", lat_lng=[54.596, -5.931], parent_code=None, flag_url=None)

#adding custom Mariehamn province to Aland Islands (AX)
iso.custom_subdivision("AX", "AX-M", name="Mariehamn", local_name="Maarianhamina", type=
↪"province", lat_lng=[60.0969, 19.934], parent_code=None, flag_url=None)

#deleting above custom subdivisions from object
iso.custom_subdivision("IE", "IE-BF", delete=1)
iso.custom_subdivision("AX", "AX-M", delete=1)
```

Warning: When adding a custom subdivision the software will be out of sync with the official ISO 3166-2 dataset, therefore its important to keep track of the custom subdivisions you add to the object.

To return to the original dataset you can delete the added custom subdivision, as described above, or you could reinstall the software.

3.1.5 Searching for a subdivision

The `search()` function allows you to search for a specific subdivision via its subdivision name. The search functionality uses a fuzzy search algorithm via "thefuzz" package, searching for subdivisions with an exact name match or those with an approximate name match, according to a score via the "likeness" input parameter.

```
from iso3166_2 import *

#searching for the Monaghan county in Ireland (IE-MN) - returning exact matching␣
↪subdivision
iso.search("Monaghan")

#searching for any subdivisions that have "Southern" in their name, using a likeness␣
↪score of 0.7
iso.search("Southern", likeness=0.7)
```

Note: A demo of the software and API is available here.

3.2 API

The main API endpoint is: https://iso3166-2-api.vercel.app/api. This endpoint displays the API documentation and forms the base URL for the 5 other endpoints.

The other endpoints available in the API are:

- https://iso3166-2-api.vercel.app/api/all
- https://iso3166-2-api.vercel.app/api/alpha/<input_alpha>
- https://iso3166-2-api.vercel.app/api/subdivision/<input_subdivision>

- https://iso3166-2-api.vercel.app/api/country_name/<input_country_name>
- https://iso3166-2-api.vercel.app/api/name/<input_name>

Five paths/endpoints are available in the API - */api/all*, */api/alpha*, */api/subdivision*, */api/country_name* and */api/name*.

- The */api/all* path/endpoint returns all of the ISO 3166 subdivision data for all countries.
- The */api/alpha* endpoint accepts the 2 letter alpha-2, 3 letter alpha-3 or numeric country codes appended to the path/endpoint e.g. */api/alpha/JP*. A single alpha code or list of them can be passed to the API e.g. */api/alpha2/FR,DEU,HUN,360,504*. If an invalid alpha code is input then an error will be returned.
- The */api/subdivision* endpoint accepts the ISO 3166-2 subdivision codes, e.g */api/subd/GB-ABD*. You can also input a list of subdivision code and the data for each will be returned e.g */api/subd/IE-MO,FI-17,RO-AG*. If the input subdivision code is not in the correct format then an error will be raised. Similarly if an invalid subdivision code that doesn't exist is input then an error will be raised.
- The */api/country_name* endpoint accepts the country/territory name as it is most commonly known in english, according to the ISO 3166-1 e.g */api/country_name/Denmark*. A single country name or list of them can be passed into the API e.g. */api/country_name/France,Moldova,Benin*. A closeness function is utilised so the most approximate name from the input will be used e.g. Sweden will be used if input is */api/country_name/Swede*. If no country is found from the closeness function or an invalid name is input then an error will be returned.
- The */api/name* endpoint accepts the ISO 3166-2 subdivision names, e.g */api/name/Derry*. You can also input a list of subdivision name from the same or different countries and the data for each will be returned e.g */api/name/Paris,Frankfurt,Rimini*. A closeness function is utilised to find the matching subdivision name, if no exact name match found then the most approximate subdivisions will be returned. Some subdivisions may have the same name, in this case each subdivision and its data will be returned e.g */api/name/Saint George,Sucre*. This endpoint also has the likeness score (*?likeness=*) query string parameter that can be appended to the URL. This can be set between 1 - 100, representing a % of likeness to the input name the return subdivisions should be, e.g: a likeness score of 90 will return fewer potential matches whose name only match to a high degree compared to a score of 10 which will create a larger search space, thus returning more potential subdivision matches. A default likeness of 100 (exact match) is used, if no match found then this is reduced to 90. If an invalid subdivision name that doesn't match any is input then an error will be raised.
- The main API endpoint (*/* or */api*) will return the homepage and API documentation.

3.2.1 Accessing subdivision data for all countries

Python Requests:

```
import requests

base_url = "https://iso3166-2-api.vercel.app/api/"
all_data = requests.get(base_url + "all").json()

all_data["AD"] #subdivision data for Andorra
all_data["IE"] #subdivision data for Ireland
all_data["PW"] #subdivision data for Palau
```

curl:

```
$ curl -i https://iso3166-2-api.vercel.app/api/all
```

3.2.2 Accessing subdivision data per country, using its ISO 3166-1 alpha codes

For example, accessing all subdivision data for France, Germany or Gambia (FR, DE, GM), via their **alpha-2 country code**:

Python Requests:

```
import requests

base_url = "https://iso3166-2-api.vercel.app/api/"
input_alpha2 = "FR" #DE, GM
all_data = requests.get(base_url + f'/alpha/{input_alpha}').json()

all_data["FR"] #subdivision data for France
```

curl:

```
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/FR
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/DE
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/GM
```

For example, accessing all subdivision data for Greece, Mexico or Montenegro (GRC, MEX, MNE), via their **alpha-3 country code**:

Python Requests:

```
import requests

base_url = "https://iso3166-2-api.vercel.app/api/"
input_alpha2 = "GRC" #MEX, MNE
all_data = requests.get(base_url + f'/alpha/{input_alpha}').json()

all_data["GR"] #subdivision data for Greece
```

curl:

```
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/GRC
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/MEX
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/MNE
```

For example, accessing all subdivision data for Nicaragua, Papa New Guinea or Qatar (558, 598, 634), via their **alpha numeric code**:

Python Requests:

```
import requests

base_url = "https://iso3166-2-api.vercel.app/api/"
input_alpha2 = "558" #598, 634 (NI, PG, QA)
all_data = requests.get(base_url + f'/alpha/{input_alpha}').json()

all_data["NI"] #subdivision data for Nicaragua
```

curl:

```
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/558
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/598
$ curl -i https://iso3166-2-api.vercel.app/api/alpha/634
```

3.2.3 Accessing all subdivision data for a specific subdivision, using its subdivision code

For example, accessing all subdivision data for Alūksnes novads, Colón and Northern Cape (LV-007, PA-3, ZA-NC):

Python Requests:

```
import requests

base_url = "https://iso3166-2-api.vercel.app/api/"
input_subdivision = "LV-007" #PA-3, ZA-NC
all_data = requests.get(base_url + f'/subdivision/{input_subdivision}').json()

all_data["LV-007"] #data for LV-007 subdivision
```

curl:

```
$ curl -i https://iso3166-2-api.vercel.app/api/subdivision/LV-007
$ curl -i https://iso3166-2-api.vercel.app/api/subdivision/PA-3
$ curl -i https://iso3166-2-api.vercel.app/api/subdivision/ZA-NC
```

3.2.4 Accessing all subdivision data for a specific country, using its name

For example, accessing all subdivision data for Tajikistan, Seychelles, Uganda:

Python Requests:

```
import requests

base_url = "https://iso3166-2-api.vercel.app/api/"
input_country_name = "Tajikistan" #Seychelles, Uganda
all_data = requests.get(base_url + f'/country_name/{input_country_name}').json()

all_data["TJ"] #subdivision data for Tajikistan
all_data["SC"] #subdivision data for Seychelles
all_data["UG"] #subdivision data for Uganda
```

curl:

```
$ curl -i https://iso3166-2-api.vercel.app/api/country_name/Tajikistan
$ curl -i https://iso3166-2-api.vercel.app/api/country_name/Seychelles
$ curl -i https://iso3166-2-api.vercel.app/api/country_name/Uganda
```

3.2.5 Accessing all subdivision data for a specific subdivision, using its subdivision name

For example, accessing all subdivision data for Saarland, Brokopondo, Delaware:

Python Requests:

```
import requests

base_url = "https://iso3166-2-api.vercel.app/api/"
input_name = "Saarland" #Brokopondo, Delaware (DE-SL, SR-BR, US-DE)
all_data = requests.get(base_url + f'/name/{input_name}').json()

all_data["DE-SL"] #subdivision data for Saarland
all_data["SR-BR"] #subdivision data for Brokopondo
all_data["US-DE"] #subdivision data for Delaware
```

curl:

```
$ curl -i https://iso3166-2-api.vercel.app/api/name/Saarland
$ curl -i https://iso3166-2-api.vercel.app/api/name/Brokopondo
$ curl -i https://iso3166-2-api.vercel.app/api/name/Delaware
```

Note: A demo of the software and API is available here.

3.3 Contributing

If you have found a bug or an issue in the software or API then please raise an issue on the repository's Issues tab.

If you would like to contribute any functional/feature changes to the software, please make a pull request on the repository.

Any other queries or issues, please contact me via email: amckenna41@qub.ac.uk