pytaxize

Release 0.7.1

Jul 10, 2020

Modules

This is a port of the R package taxize. There is a lot going on in the R version of this library, so it will take a while to get all the same functionality over here.

Why? A significant advantage of a Python version of *taxize* will be for those that are pythonistas at heart. Also, you could use *pytaxize* in a web app, whereas you could with *taxize* (e.g., in a Shiny app), but it wouldn't scale, be very fast, etc.

CHAPTER 1

Installation

Stable from pypi

```
pip install pytaxize
```

Development version

```
sudo pip install git+git://github.com/sckott/pytaxize.git#egg=pytaxize
```

CHAPTER 2

Taxonomic Ids

I've started working on a class interface for taxonomic IDs, which will have a bunch of extension methods to do various things with taxon ids. What's available right now is just getting COL ids.

```
from pytaxize import Ids
res = Ids('Poa annua')
res.ncbi()
res.ids
```

```
{'Poa annua': [{'id': '93036',
   'name': 'Poa annua',
   'rank': 'species',
   'uri': 'https://www.ncbi.nlm.nih.gov/taxonomy/93036'}]}
```

CHAPTER 3

ITIS

```
from pytaxize import itis
itis.accepted_names(504239)

{'acceptedName': 'Dasiphora fruticosa',
   'acceptedTsn': '836659',
   'author': '(L.) Rydb.'}
```

```
itis.hierarchy_up(tsn = 36485)

{'author': 'Raf.',
 'parentName': 'Asteraceae',
 'parentTsn': '35420',
 'rankName': 'Genus',
 'taxonName': 'Agoseris',
 'tsn': '36485'}
```

CHAPTER 4

Catalogue of Life

```
from pytaxize import col
x = col.children(name=["Apis"])
x[0][0:2]
```

```
[{'id': '7a4a38c5095963949d6d6ec917d471de',
  'name': 'Apis andreniformis',
  'rank': 'Species'},
 {'id': '39610a4ceff7e5244e334a3fbc5e47e5',
  'name': 'Apis cerana',
  'rank': 'Species'}]
```

CHAPTER 5

Modules

5.1 pytaxize modules

pytaxize is split up into modules:

- Taxonomic IDs - Class for getting taxonomic ids from different data sources
- Scientific names to common names - convert among scientific and common names
- NCBI - methods for NCBI
- Integrated Taxonomic Information System - methods for ITIS
- Global Names Resolver - resolve names with the Global Names Resolver
- Global Names Index - various methods for the GNI data source
- Global Biodiversity Information Facility - Taxonomic names related methods for GBIF
- Catalogue of Life - methods for COL - will be transitioned to COL+ soon
- Other methods - variety of other methods

You can import the entire library, or each module individually as needed.

5.2 Taxonomic Identifiers Class

class pytaxize.**Ids**(*name*)

ids: A class for taxonomic identifiers

Usage:

```
from pytaxize import Ids

x = Ids('Poa annua')
x
```

(continues on next page)

(continued from previous page)

```
x.name
x.ncbi()
x.ids
x.db_ids

# more than one result
x = Ids(name="Echinacea")
x.ncbi()
x.ids
x.ids["Echinacea"]

# more than one name supplied
x = Ids(name=['Helianthus annuus', 'Poa annua', 'Echinacea'])
x
x.ncbi()
x
x.ids
x.ids["Helianthus annuus"]
x.ids["Poa annua"]
x.ids["Echinacea"]

# extract just ids
out = x.extract_ids()
out["Echinacea"]

# ITIS
x = Ids("Helianthus annuus")
x.itis(type="scientific")
x.extract_ids()
```

5.3 Children

class pytaxize.**Children**(*ids*)

Children: Retrieve taxonomic children

Usage:

```
from pytaxize import Children

# ITIS
## one id
x = Children(179913)
x
x.ids
x.itis()

## many ids - with one invalid id
x = Children([179913, 174321, 9999999])
x
x.ids
res = x.itis()
res[179913]
res[174321]
res[9999999]
```

5.4 Classification

class pytaxize.**Classification**(*ids*)

Classification: Retrieve taxonomic hierarchy for taxonomic IDs

Usage:

```
from pytaxize import Classification

# ITIS
## one id
x = Classification(99208)
x
x.ids
res = x.itis()
res[99208]

## many ids - with one invalid id
x = Classification([99208, 129313, 9999999])
x
x.ids
res = x.itis()
res[99208]
res[129313]
res[9999999]

# NCBI
x = Classification(9606)
x
x.ids
x.ncbi()
```

5.5 Scientific and common names

scicomm.**sci2comm**()

Multiply dispatched method: sci2comm

Get common names from scientific names or ids

param x (str|list(str)|Ids) One or more scientific names or partial names, or an *Ids* object

param db (str) Data source, default: "ncbi". NCBI only supported right now, other sources to come.

param **kwargs Curl options passed on to *requests.get*

return dict, keys are supplied scientific names, and values are common names

note Remember to set your Entrez API key as *ENTREZ_KEY*

Usage:

```
from pytaxize import scicomm

# from names (str or list of str's)
scicomm.sci2comm('Helianthus annuus')
scicomm.sci2comm('Puma concolor')
```

(continues on next page)

(continued from previous page)

```
scicomm.sci2comm(['Helianthus annuus', 'Poa annua'])
scicomm.sci2comm('Gadus morhua')
scicomm.sci2comm('Pomatomus saltatrix')
scicomm.sci2comm('Loxodonta africana')

scicomm.sci2comm('Lycaon pictus', db="itis")

## no results
### not a real name
scicomm.sci2comm('foo bar')
### good name, many id results, but no common names
scicomm.sci2comm("Echinacea")

# from an Ids object
from pytaxize import Ids
x = Ids('Helianthus annuus')
x.ncbi()
scicomm.sci2comm(x)
x.itis()
scicomm.sci2comm(x)
x = Ids('Lycaon pictus')
x.itis()
x.ids
scicomm.sci2comm(x)
```

Get common names from scientific names or ids

param x (str|list(str)|Ids) One or more scientific names or partial names, or an *Ids* object

param db (str) Data source, default: "ncbi". NCBI only supported right now, other sources to come.

param **kwargs Curl options passed on to *requests.get*

return dict, keys are supplied scientific names, and values are common names

note Remember to set your Entrez API key as *ENTREZ_KEY*

Usage:

```
from pytaxize import scicomm

# from names (str or list of str's)
scicomm.sci2comm('Helianthus annuus')
scicomm.sci2comm('Puma concolor')
scicomm.sci2comm(['Helianthus annuus', 'Poa annua'])
scicomm.sci2comm('Gadus morhua')
scicomm.sci2comm('Pomatomus saltatrix')
scicomm.sci2comm('Loxodonta africana')

scicomm.sci2comm('Lycaon pictus', db="itis")

## no results
### not a real name
scicomm.sci2comm('foo bar')
### good name, many id results, but no common names
scicomm.sci2comm("Echinacea")

# from an Ids object
```

(continues on next page)

(continued from previous page)

```
from pytaxize import Ids
x = Ids('Helianthus annuus')
x.ncbi()
scicomm.sci2comm(x)
x.itis()
scicomm.sci2comm(x)
x = Ids('Lycaon pictus')
x.itis()
x.ids
scicomm.sci2comm(x)
```

Other signatures: Ids

5.6 NCBI

ncbi.**search**(*modifier=None*, *rank_query=None*)

Search NCBI's taxonomic data - get NCBI taxonomic IDs

Parameters

- **sci_com** – list of common or scientific names
- **modifier** – A modifier to the *sci_com* given. Options include: Organism, Scientific Name, Common Name, All Names, Division, Filter, Lineage, GC, MGC, Name Tokens, Next Level, PGC, Properties, Rank, Subtree, Synonym, Text Word. These are not checked, so make sure they are entered correctly, as is.
- **rank_query** – A taxonomic rank name to modify the query sent to NCBI. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional.

Note Remember to set your Entrez API key as *ENTREZ_KEY*

Returns dict, named with values given to *sci_com*, where each value in the dict is a list of NCBI taxonomic identifiers

Usage:

```
from pytaxize import ncbi

ncbi.search(sci_com = "Apis")

# Many names
ncbi.search(sci_com=["Apis", "Puma concolor", "Pinus"])

# Example with more than 1 result
ncbi.search(sci_com='Satyrium')
ncbi.search(sci_com=['Satyrium', 'Pinus'])

# common names
ncbi.search(sci_com = 'bear')
```

ncbi.**hierarchy**()

Get a full taxonomic hierarchy from NCBI

Parameters **ids** – one or more NCBI taxonomy ids

Note Remember to set your Entrez API key as *ENTREZ_KEY*

Returns dict, named with ids given to *ids*, where each value in the dict is a list of taxa, each a dict with the fields `ScientificName`, `Rank`, and `TaxId`

Usage:

```
from pytaxize import ncbi
ncbi.hierarchy(ids=9606)
ncbi.hierarchy(ids=[9606,55062,4231])
```

5.7 ITIS

itis.**accepted_names**(***kwargs*)

Get accepted names from tsn

Parameters

- **tsn** – taxonomic serial number (TSN) (character or numeric)
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
# TSN accepted - good name
itis.accepted_names(tsn=208527)
# TSN not accepted - input TSN is old name
itis.accepted_names(tsn=504239)
```

itis.**any_match_count**(***kwargs*)

Get any match count.

Parameters

- **x** – text or taxonomic serial number (TSN) (character or numeric)
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.any_match_count(x=202385)
itis.any_match_count(x="dolphin")
```

itis.**comment_detail**(*as_dataframe=False*, ***kwargs*)

Get comment detail from TSN

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.comment_detail(tsn=180543)
```

itis.**common_names**(*as_dataframe=False*, ***kwargs*)

Get common names from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.common_names(tsn=183833)
# no common names
itis.common_names(tsn=726872)
```

itis.**core_metadata**(*as_dataframe=False*, ***kwargs*)
Get core metadata from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
# coverage and currrency data
itis.core_metadata(tsn=28727)
# no coverage or currrency data
itis.core_metadata(tsn=183671)
```

itis.**coverage**(*as_dataframe=False*, ***kwargs*)
Get coverge from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
# coverage data
itis.coverage(tsn=28727)
# no coverage data
itis.coverage(526852)
# as data_frame
itis.coverage(526852, as_dataframe=True)
```

itis.**credibility_rating**(*as_dataframe=False*, ***kwargs*)
Get credibility rating from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.credibility_rating(tsn=526852)
itis.credibility_rating(28727)
```

itis.**credibility_ratings**()
Get possible credibility ratings

Parameters ****kwargs** – Curl options passed on to *requests.get*

Returns a dict

Usage:

```
from pytaxize import itis
itis.credibility_ratings()
```

itis.**currency**(*as_dataframe=False*, ***kwargs*)
Get currency from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
# currency data
itis.currency(28727)
# no currency dat
itis.currency(526852)
# as data_frame
itis.currency(526852, as_dataframe=True)
```

itis.**date_data**(*as_dataframe=False*, ***kwargs*)
Get date data from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.date_data(tsn=180543)
```

itis.**experts**(*as_dataframe=False*, ***kwargs*)
Get expert information for the TSN.

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available

- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.experts(tsn=180544)
```

itis.**rank_name**(*as_dataframe=False*, ***kwargs*)

Returns the kingdom and rank information for the TSN.

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.rank_name(tsn = 202385)
```

itis.**hierarchy_full**(*as_dataframe=False*, ***kwargs*)

Get full hierarchy from ts

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.hierarchy_full(tsn = 37906)
itis.hierarchy_full(tsn = 100800)
# as data_frame
itis.hierarchy_full(tsn = 100800, as_dataframe=True)
```

itis.**full_record**(*lsid=None*, ***kwargs*)

Returns the full ITIS record for a TSN or LSID

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **lsid** – lsid for a taxonomic group (character)
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.full_record(tsn="504239")
itis.full_record(tsn="202385")
itis.full_record(tsn="183833")
itis.full_record(lsid="urn:lsid:itis.gov:itis_tsn:180543")
itis.full_record(lsid="urn:lsid:itis.gov:itis_tsn:37906")
itis.full_record(lsid="urn:lsid:itis.gov:itis_tsn:100800")
```

itis.**geographic_divisions**(*as_dataframe=False*, ***kwargs*)
Get geographic divisions from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.geographic_divisions(tsn=180543)
```

itis.**geographic_values**()
Get all possible geographic values

Parameters ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.geographic_values()
```

itis.**hierarchy_down**(*as_dataframe=False*, ***kwargs*)
Get hierarchy down from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.hierarchy_down(tsn = 179913)
itis.hierarchy_down(tsn = 161994)
itis.hierarchy_down(tsn = 9999999)
```

itis.**hierarchy_up**(*as_dataframe=False*, ***kwargs*)
Get hierarchy up from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.hierarchy_up(tsn = 36485)
itis.hierarchy_up(tsn = 37906)
```

itis.**terms**(*what='both'*, *as_dataframe=False*, ***kwargs*)
Get itis terms

Parameters

- **x** – query term
- **what** – One of both (search common and scientific names), common (search just common names), or scientific (search just scientific names)
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.terms("bear")
itis.terms("bear", what="common")
itis.terms("Poa", what="scientific")
itis.terms("bear", as_dataframe=True)
```

itis.**global_species_completeness**(*as_dataframe=False*, ***kwargs*)

Get global species completeness from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.global_species_completeness(180541)
```

itis.**jurisdictional_origin**(*as_dataframe=False*, ***kwargs*)

Get jurisdictional origin from tsn

Parameters

- **tsn** – (int) TSN for a taxonomic group
- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.jurisdictional_origin(180543)
itis.jurisdictional_origin(180543, True)
```

itis.**jurisdiction_origin_values**(***kwargs*)

Get jurisdiction origin values

Parameters

- **as_dataframe** – (bool) specify return type, if pandas is available
- ****kwargs** – Curl options passed on to *requests.get*

Usage:

```
from pytaxize import itis
itis.jurisdiction_origin_values()
```

itis.**jurisdiction_values**()
Get possible jurisdiction values

Parameters ****kwargs** – Curl options passed on to *requests.get*

Returns list

Usage:

```
from pytaxize import itis
itis.jurisdiction_values()
```

5.8 Global Names Resolver

gn.**datasources**()
Get data sources for the Global Names Resolver.

Retrieve data sources used in Global Names Index, see http://gni.globalnames.org/ for information.

Usage:

```
# all data sources
from pytaxize import gn
gn.gnr.datasources()
```

gn.**resolve**(*source=None*, *format='json'*, *resolve_once='false'*, *with_context='false'*, *best_match_only='false'*, *header_only='false'*, *preferred_data_sources='false'*, *http='get'*)
Uses the Global Names Resolver to resolve scientific names

Parameters

- **names** – List of taxonomic names
- **source** – Source to pull from, one of x, y, z
- **format** – One of json or xml
- **resolve_once** – Logical, true or false
- **with_context** – Return context with taxonomic names
- **best_match_only** – Logical, if true (default) return the best match only
- **header_only** – Return header only, logical
- **preferred_data_sources** – Return only preferred data sources.
- **http** – The HTTP method to use, one of "get" or "post". Default="get"

Usage:

```
from pytaxize import gn
gn.resolve('Helianthus annus')
gn.resolve(['Helianthus annus','Poa annua'])
```

5.9 Global Names Index

gn.**parse**()
Uses the Global Names Index to parse scientific names

Parameters **names** – List of scientific names.

Usage:

```
from pytaxize import gn
gn.gni.parse(names = ['Cyanistes caeruleus','Helianthus annuus'])
```

gn.**search**(*per_page=30*, *page=1*)

Search for names against the Global names index

Parameters

- **search_term** – Search term
- **per_page** – Items to return per page
- **page** – Page to return

Usage:

```
from pytaxize import gn
gn.gni.search(search_term = 'ani*')
```

gn.**details**(*all_records=1*)

Usage:

```
from pytaxize import gn
gn.gni.details(id = 17802847)
```

5.10 Global Biodiversity Information Facility

5.11 Catalogue of Life

col.**children**(*id=None*, *format=None*, *start=None*, *checklist=None*)

Search Catalogue of Life for for direct children of a particular taxon.

Parameters

- **name** – The string to search for. Only exact matches found the name given will be returned, unless one or wildcards are included in the search string. An * (asterisk) character denotes a wildcard; a % (percentage) character may also be used. The name must be at least 3 characters long, not counting wildcard characters.
- **id** – The record ID of the specific record to return (only for scientific names of species or infraspecific taxa)
- **format** – format of the results returned. Valid values are format=xml and format=php; if the format parameter is omitted, the results are returned in the default XML format. If format=php then results are returned as a PHP array in serialized string format, which can be converted back to an array in PHP using the unserialize command
- **start** – The first record to return. If omitted, the results are returned from the first record (start=0). This is useful if the total number of results is larger than the maximum number of results returned by a single Web service query (currently the maximum number of results returned by a single query is 500 for terse queries and 50 for full queries).

- **checklist** – The year of the checklist to query, if you want a specific year's checklist instead of the lastest as default (numeric). Valid years are 2010 through the previous year from the current date. If none given, the "lastest" checklist is used

You must provide one of name or id. The other parameters (format and start) are optional. Returns A list of data.frame's.

Usage:

```
from pytaxize import col
col.children(name=["Apis"])

# An example where there is no classification, results in data.frame with no rows
col.children(id=["4fdb38d6220462049eab9e3f285144e0"])

# Use a specific year's checklist
col.children(name=["Apis"], checklist="2012")
col.children(name=["Apis"], checklist="2009")

# Pass in many names or many id's
out = col.children(name=["Buteo","Apis","Accipiter"], checklist="2012")
# get just one element in list of output
out[0]
```

col.**search**(*id=None*, *start=None*, *checklist=None*)

Search Catalogue of Life for taxonomic IDs

Parameters

- **name** – The string to search for. Only exact matches found the name given will be returned, unless one or wildcards are included in the search string. An * (asterisk) character denotes a wildcard; a % (percentage) character may also be used. The name must be at least 3 characters long, not counting wildcard characters.
- **id** – The record ID of the specific record to return (only for scientific names of species or infraspecific taxa)
- **start** – The first record to return. If omitted, the results are returned from the first record (start=0). This is useful if the total number of results is larger than the maximum number of results returned by a single Web service query (currently the maximum number of results returned by a single query is 500 for terse queries and 50 for full queries).
- **checklist** – The year of the checklist to query, if you want a specific year's checklist instead of the lastest as default (numeric).

You must provide one of name or id. The other parameters (format and start) are optional.

Usage:

```
from pytaxize import col

col.search(name=["Apis"])
col.search(id=15669061)

# Many names
col.search(name=["Apis","Puma concolor"])

# Many ids - DOESNT WORK
col.search(id=[15669061,6862841])
```

(continues on next page)

(continued from previous page)

```
# An example where there is no data
col.search(id=11935941)

# Example with more than 1 result
col.search(name=['Poa'])
```

5.12 Other methods

`tax.`**`names_list`**(*size=10*, *as_dataframe=False*)
Get a random vector of species names.

Parameters

- **`rank`** – Taxonomic rank, one of species, genus (default), family, order.
- **`size`** – Number of names to get. Maximum depends on the rank.
- **`as_dataframe`** – (optional) Type: boolean. Return as pandas data frame? default: False

Usage:

```
import pytaxize
pytaxize.names_list(size=10)
pytaxize.names_list('species', size=10)
pytaxize.names_list('family', size=10)
pytaxize.names_list('order', size=10)
pytaxize.names_list('order', 2)
pytaxize.names_list('order', 15)
```

`tax.`**`vascan_search`**(*format='json'*, *raw=False*)
Search the CANADENSYS Vascan API.

Parameters

- **`q`** – Taxonomic rank, one of species, genus (default), family, order.
- **`format`** – Number of names to get. Maximum depends on the rank.
- **`raw`** – Raw data or not (default)
- **`callopts`** – Further args passed to request

Usage:

```
import pytaxize
pytaxize.vascan_search(q = ["Helianthus annuus"])
pytaxize.vascan_search(q = ["Helianthus annuus"], raw=True)
pytaxize.vascan_search(q = ["Helianthus annuus", "Crataegus dodgei"], raw=True)

# format type
## json
pytaxize.vascan_search(q = ["Helianthus annuus"], format="json", raw=True)

## xml
pytaxize.vascan_search(q = ["Helianthus annuus"], format="xml", raw=True)

# lots of names, in this case 50
```

(continues on next page)

(continued from previous page)

```
splist = pytaxize.names_list(rank='species', size=50)
pytaxize.vascan_search(q = splist)
```

tax.**scrapenames**(*file=None*, *text=None*, *engine=None*, *unique=None*, *verbatim=None*, *detect_language=None*, *all_data_sources=None*, *data_source_ids=None*, *as_dataframe=False*)

Resolve names using Global Names Recognition and Discovery.

Uses the Global Names Recognition and Discovery service, see http://gnrd.globalnames.org/.

Parameters

- **url** – An encoded URL for a web page, PDF, Microsoft Office document, or image file, see examples
- **file** – When using multipart/form-data as the content-type, a file may be sent. This should be a path to your file on your machine.
- **text** – Type: string. Text content; best used with a POST request, see examples
- **engine** – (optional) Type: integer, Default: 0. Either 1 for TaxonFinder, 2 for NetiNeti, or 0 for both. If absent, both engines are used.
- **unique** – (optional) Type: boolean. If True (default), response has unique names without offsets.
- **verbatim** – (optional) Type: boolean, If True (default to False), response excludes verbatim strings.
- **detect_language** – (optional) Type: boolean, When True (default), NetiNeti is not used if the language of incoming text is determined not to be English. When 'false', NetiNeti will be used if requested.
- **all_data_sources** – (optional) Type: bolean. Resolve found names against all available Data Sources.
- **data_source_ids** – (optional) Type: string. Pipe separated list of data source ids to resolve found names against. See list of Data Sources.
- **as_dataframe** – (optional) Type: boolean. Return as pandas data frame? default: False

Usage:

```
import pytaxize

# Get data from a website using its URL
out = pytaxize.scrapenames(url = 'https://en.wikipedia.org/wiki/Spider')
out['data'].head() # data
out['meta'] # metadata

# Scrape names from a pdf at a URL
out = pytaxize.scrapenames(url = 'http://www.mapress.com/zootaxa/2012/f/
↪z03372p265f.pdf')
out['data'].head() # data
out['meta'] # metadata

# With arguments
pytaxize.scrapenames(url = 'http://www.mapress.com/zootaxa/2012/f/z03372p265f.pdf
↪', unique=True)
pytaxize.scrapenames(url = 'http://www.mapress.com/zootaxa/2012/f/z03372p265f.pdf
↪', all_data_sources=True)
```

(continues on next page)

(continued from previous page)

```
# Get data from text string as an R object
pytaxize.scrapenames(text='A spider named Pardosa moesta Banks, 1892')
```

taxo.**taxo_datasources**()

Get data sources for Taxosaurus. Retrieve data sources used in Global Names Index, see http://taxosaurus.org/ for information.

Usage:

```
# all data sources
import pytaxize
pytaxize.taxo_datasources()

# Output a dict
pytaxize.taxo_datasources(False)
```

taxo.**taxo_resolve**(*source=None*, *code=None*, *http='get'*)

Uses Taxosaurus to resolve scientific names

Parameters

- **query** – List of taxonomic names
- **source** – (optional) Source to pull from
- **code** – (optional) the abbreviation for one of the nomenclature codes (ICZN: International Code of Zoological Nomenclature; ICN: International Code of Nomenclature for algae, fungi, and plants; ICNB: International Code of Nomenclature of Bacteria)
- **http** – (optional) The HTTP method to use, one of "get" or "post". Default="get"

Usage:

```
import pytaxize
pytaxize.taxo_resolve(query='Helianthus annus')
pytaxize.gnr_resolve(['Helianthus annus','Poa annua'])
```

pytaxize modules Introduction to pygbif modules.

Taxonomic Identifiers Class The Ids class

Children The Children class

Classification The Classification class

Scientific and common names Scientific to common names, and vice versa

NCBI The ncbi module: NCBI methods

ITIS The itis module: ITIS methods

Global Names Resolver The gn module: methods for Global Names Index and Resolver

Global Biodiversity Information Facility The gbif module: GBIF methods

Catalogue of Life The col module: Catalogue of Life methods

Other methods Variety of other methods

CHAPTER 6

All the rest

6.1 Changelog

6.1.1 0.7.0 (2020-06-15)

- first release to pypi

6.2 Contributors

- Scott Chamberlain
- Colin Talbert
- akshayah3
- panks
- Yanghao Li
- Ben Morris
- Bishakh Ghosh
- Yoav Ram

6.3 Contributing

6.3.1 Bug reports

Please report bug reports on our issue tracker.

6.3.2 Feature requests

Please put feature requests on our issue tracker.

6.3.3 Pull requests

When you submit a PR you'll see a template that pops up - it's reproduced here.

- Provide a general summary of your changes in the Title
- Describe your changes in detail
- If the PR closes an issue make sure include e.g., *fix #4* or similar, or if just relates to an issue make sure to mention it like *#4*
- If introducing a new feature or changing behavior of existing methods/functions, include an example if possible to do in brief form
- Did you remember to include tests? Unless you're changing docs/grammar, please include new tests for your change

6.3.4 Writing tests

We're using *pytest* for testing. See the **'pytest docs'_** for help on contributing to or writing tests.

Before running tests for the first time, you'll need install pytaxize dependencies, but also pytest and a couple other packages:

```
$ pip install -e .
$ pip install pytest vcrpy coverage lxml requests multipledispatch enum
```

The Makefile has a task for testing under Python 3:

```
$ make test
```

6.3.5 Code formatting

We're using the Black formatter, so make sure you use that before submitting code - there's lots of text editor integrations, a command line tool, etc.

6.4 Contributor Code of Conduct

As contributors and maintainers of this project, we pledge to respect all people who contribute through reporting issues, posting feature requests, updating documentation, submitting pull requests or patches, and other activities.

We are committed to making participation in this project a harassment-free experience for everyone, regardless of level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, or religion.

Examples of unacceptable behavior by participants include the use of sexual language or imagery, derogatory comments or personal attacks, trolling, public or private harassment, insults, or other unprofessional conduct.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct. Project maintainers who do not follow the Code of Conduct may be removed from the project team.

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by opening an issue or contacting one or more of the project maintainers.

This Code of Conduct is adapted from the Contributor Covenant (https://contributor-covenant.org), version 1.0.0, available at https://contributor-covenant.org/version/1/0/0/

6.5 LICENSE

Changelog See what has changed in recent pygbif versions.

Contributors pygbif contributors.

Contributing Learn how to contribute to the pygbif project.

Contributor Code of Conduct Expected behavior in this community. By participating in this project you agree to abide by its terms.

LICENSE The pygbif license.

6.6 Indices and tables

- genindex

Python Module Index

p

Index

V