pytaxize Documentation

Release 0.5.9200

Scott Chamberlain

Apr 10, 2017

Contents

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

python 2/3

I usually use Python *2.x* locally, but I've just been working on making *pytaxize* compatible with *2.x* and *3.x*. Please let me know if anything is broken on either version.

CHAPTER 2

Installation

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

python or ipython, etc.

```
import pytaxize
```

CHAPTER 3

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.

```
res = pytaxize.Ids('Poa annua', db='col')
res.get_colid()
```

```
[19275187]
```

CHAPTER 4

Vascan search

```
pytaxize.vascan_search(q = ["Helianthus annuus"])
```

```
{u'apiVersion': u'0.1',
 u'results': [{u'matches': [{u'canonicalName': u'Helianthus annuus',
     u'distribution': [{u'establishmentMeans': u'introduced',
       u'locality': u'NS',
       u'locationID': u'ISO 3166-2:CA-NS',
       u'occurrenceStatus': u'introduced'},
      {u'establishmentMeans': u'',
       u'locality': u'PE',
       u'locationID': u'ISO 3166-2:CA-PE',
       u'occurrenceStatus': u'excluded'},
      {u'establishmentMeans': u'',
       u'locality': u'NT',
       u'locationID': u'ISO 3166-2:CA-NT',
       u'occurrenceStatus': u'doubtful'},
      {u'establishmentMeans': u'introduced',
```

CHAPTER 5

Scrape taxonomic names

```
out = pytaxize.scrapenames(url = 'http://www.mapress.com/zootaxa/2012/f/z03372p265f.
↪pdf')
out['data'].head()
```

	identifiedName	offsetEnd	offsetStart	scientificName	verbatim
0	Waxiella	14	7	Waxiella	Waxiella
1	W. africana	395	385	Waxiella africana	W. africana
2	W. egbara	581	573	Waxiella egbara	W. egbara
3	W. erithraeus	771	759	Waxiella erithraeus	W. erithraeus
4	W. gwaai	951	944	Waxiella gwaai	W. gwaai

CHAPTER 6

ITIS low level functions

```
pytaxize.getacceptednamesfromtsn('208527')

'208527'
```

```
pytaxize.getcommentdetailfromtsn(tsn=180543)

                                              comment  \
0  Status: CITES - Appendix I as U. arctos (Mexic...
1  Comments: Reviewed by Erdbrink (1953), Couturi...

                    commentator commid              commtime  updatedate
0  Wilson & Reeder, eds. (2005)  18556  2007-08-20 15:06:38.0  2014-02-03
1  Wilson & Reeder, eds. (2005)  18557  2007-08-20 15:06:38.0  2014-02-03
```

```
pytaxize.gethierarchyupfromtsn(tsn = 36485)

  author  parentName parentTsn rankName taxonName    tsn
0   Raf.  Asteraceae     35420    Genus  Agoseris  36485
```

CHAPTER 7

Catalogue of Life

```
pytaxize.col_children(name=["Apis"])
```

```
[        id                name     rank
 0  6971712  Apis andreniformis  Species
 1  6971713         Apis cerana  Species
 2  6971714        Apis dorsata  Species
 3  6971715         Apis florea  Species
 4  6971716  Apis koschevnikovi  Species
 5  6845885      Apis mellifera  Species
 6  6971717    Apis nigrocincta  Species]
```

CHAPTER 8

Parse names

Parse names using GBIF's parser API

```
pytaxize.gbif_parse(scientificname=['Arrhenatherum elatius var. elatius',
                'Secale cereale subsp. cereale', 'Secale cereale ssp. cereale',
↪'Vanessa atalanta (Linnaeus, 1758)'])
```

```
  authorsParsed bracketAuthorship bracketYear                  canonicalName  \
0          True                NaN         NaN  Arrhenatherum elatius elatius
1          True                NaN         NaN         Secale cereale cereale
2          True                NaN         NaN         Secale cereale cereale
3          True           Linnaeus        1758               Vanessa atalanta

                canonicalNameComplete             canonicalNameWithMarker  \
0  Arrhenatherum elatius var. elatius  Arrhenatherum elatius var. elatius
1       Secale cereale subsp. cereale       Secale cereale subsp. cereale
2       Secale cereale subsp. cereale       Secale cereale subsp. cereale
3   Vanessa atalanta (Linnaeus, 1758)                    Vanessa atalanta

    genusOrAbove infraSpecificEpithet rankMarker  \
0  Arrhenatherum              elatius       var.
1         Secale              cereale     subsp.
2         Secale              cereale     subsp.
3        Vanessa                  NaN        NaN

                      scientificName specificEpithet        type
0  Arrhenatherum elatius var. elatius         elatius  WELLFORMED
1       Secale cereale subsp. cereale         cereale  WELLFORMED
2         Secale cereale ssp. cereale         cereale     SCINAME
3   Vanessa atalanta (Linnaeus, 1758)        atalanta  WELLFORMED
```

CHAPTER 9

Get random vector of taxon names

not working yet...

CHAPTER 10

Meta

- Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms.
- License: MIT; see LICENSE file

Contents

Catalogue of Life

col.**col_children**(*name=None*, *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).

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

Usage:

```
# A basic example
import pytaxize
pytaxize.col_children(name=["Apis"])

# An example where there is no classification, results in data.frame with no rows
pytaxize.col_children(id=[15669061])

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

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

col.**col_downstream**(*name=None*, *downto=None*, *format=None*, *start=None*, *checklist=None*)

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.
- **downto** – The taxonomic level you want to go down to. See examples below. The taxonomic level IS case sensitive, and you do have to spell it correctly. See code{data(rank_ref)} for spelling.
- **checklist** – The year of the checklist to query, if you want a specific year's checklist instead of the lastest as default (numeric).
- **format** – The returned format (default = None). If NULL xml is used. Currently only xml is supported.
- **start** – The first record to return (default = None). If NULL, 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).

Returns a list of DataFrame's.

Usage:

```
# Some basic examples
pytaxize.col_downstream(name=["Apis"], downto="Species")
pytaxize.col_downstream(name=["Bryophyta"], downto="Family")

# An example that takes a bit longer
pytaxize.col_downstream(name=["Plantae","Animalia"], downto="Class")

# Using a checklist from a specific year
pytaxize.col_downstream(name=["Bryophyta"], downto=["Family"], checklist="2009")
```

col.**col_search**(*name=None*, *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:

```
# A basic example
pytaxize.col_search(name=["Apis"])
pytaxize.col_search(id=15669061) # - DOESNT WORK

# Many names
pytaxize.col_search(name=["Apis","Puma concolor"])

# Many ids - DOESNT WORK
pytaxize.col_search(id=[15669061,6862841])

# An example where there is no data
pytaxize.col_search(id=11935941)

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

Global Names Resolver

gnr.**gnr_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
import pytaxize
pytaxize.gnr_datasources()

# give me the id for EOL
out = pytaxize.gnr_datasources()
[d for d in out if d['title'] in 'EOL'][0]['id']
```

gnr.**gnr_resolve**(*names='Homo sapiens'*, *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:

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

Global Names Index

gni.**gni_parse**(*names*)

Uses the Global Names Index to parse scientific names

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

Usage:

```
import pytaxize
pytaxize.gni_parse(names = ['Cyanistes caeruleus','Helianthus annuus'])
```

gni.**gni_search**(*search_term='ani*', 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:

```
import pytaxize
pytaxize.gni_search(search_term = 'ani*')
```

gni.**gni_details**(*id=17802847, all_records=1*)

Usage:

```
import pytaxize
pytaxize.gni_details(id = 17802847)
```

Variety of functions

tax.**names_list**(*rank='genus'*, *size=10*)
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.

Usage:

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

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

tax.**gbif_parse**(*scientificname*)
Parse taxon names using the GBIF name parser.

Parameters **scientificname** – A character vector of scientific names. Returns a DataFrame containing fields extracted from parsed taxon names. Fields returned are the union of fields extracted from all species names in scientificname

Author John Baumgartner (johnbb@student.unimelb.edu.au)

References http://dev.gbif.org/wiki/display/POR/Webservice+API, http://tools.gbif.org/nameparser/api.do

Usage:

```
import pytaxize
pytaxize.gbif_parse(scientificname=['x Agropogon littoralis'])
```

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

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.

Usage:

```
# Get data from a website using its URL
out = pytaxize.scrapenames(url = 'http://en.wikipedia.org/wiki/Araneae')
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)

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

Taxonomic Identifiers Class

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

ids: A class for taxonomic identifiers

Usage:

```
from pytaxize import Ids

res = Ids('Poa annua', db='col')
res.get_colid()
```

get_colid(*ask=True*, *verbose=True*)

Get Catalogue of Life taxonomic identifiers

Usage:

```
pytaxize.get_colid(sciname=['Poa annua'])
```

ITIS

itis.**itis_ping**(***kwargs*)

Ping the ITIS API

Usage:

```
import pytaxize
pytaxize.itis_ping()
```

itis.**getacceptednamesfromtsn**(*tsn*, ***kwargs*)

Get accepted names from tsn

Parameters **tsn** – taxonomic serial number (TSN) (character or numeric)

Usage:

```
# TSN accepted - good name
pytaxize.getacceptednamesfromtsn('208527')
# TSN not accepted - input TSN is old name
pytaxize.getacceptednamesfromtsn('504239')
```

itis.**getanymatchcount**(*x*, ***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:

```
pytaxize.getanymatchcount(x=202385)
pytaxize.getanymatchcount(x="dolphin")
```

itis.**getcommentdetailfromtsn**(*tsn*, ***kwargs*)

Get comment detail from TSN

Parameters

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

Usage:

```
pytaxize.getcommentdetailfromtsn(tsn=180543)
```

itis.**getcommonnamesfromtsn**(*tsn*, ***kwargs*)
Get common names from tsn

Parameters

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

Usage:

```
pytaxize.getcommonnamesfromtsn(tsn=183833)
```

itis.**getcoremetadatafromtsn**(*tsn*, ***kwargs*)
Get core metadata from tsn

Usage:

```
# coverage and currrency data
pytaxize.getcoremetadatafromtsn(tsn=28727)
# no coverage or currrency data
pytaxize.getcoremetadatafromtsn(tsn=183671)
```

itis.**getcoveragefromtsn**(*tsn*, ***kwargs*)
Get coverge from tsn

Usage:

```
# coverage data
pytaxize.getcoveragefromtsn(tsn=28727)
# no coverage data
pytaxize.getcoveragefromtsn(526852)
```

itis.**getcredibilityratingfromtsn**(*tsn*, ***kwargs*)
Get credibility rating from tsn

Usage:

```
pytaxize.getcredibilityratingfromtsn(526852)
pytaxize.getcredibilityratingfromtsn(28727)
```

itis.**getcredibilityratings**(***kwargs*)
Get possible credibility ratings

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

Usage:

```
pytaxize.getcredibilityratings()
```

itis.**getcurrencyfromtsn**(*tsn*, ***kwargs*)
Get currency from tsn

Usage:

```
# currency data
pytaxize.getcurrencyfromtsn(28727)
# no currency dat
pytaxize.getcurrencyfromtsn(526852)
```

itis.**getdatedatafromtsn**(*tsn*, ***kwargs*)
Get date data from tsn

Usage:

```
pytaxize.getdatedatafromtsn(180543)
```

itis.**getexpertsfromtsn**(*tsn*, ***kwargs*)
Get expert information for the TSN.

Usage:

```
pytaxize.getexpertsfromtsn(180544)
```

itis.**gettaxonomicranknamefromtsn**(*tsn*, ***kwargs*)
Returns the kingdom and rank information for the TSN.

Parameters **tsn** – TSN for a taxonomic group (numeric)

Usage:

```
pytaxize.gettaxonomicranknamefromtsn(tsn = 202385)
```

itis.**getfullhierarchyfromtsn**(*tsn*, ***kwargs*)
Get full hierarchy from ts

Parameters **tsn** – TSN for a taxonomic group (numeric)

Usage:

```
pytaxize.getfullhierarchyfromtsn(tsn = 37906)
pytaxize.getfullhierarchyfromtsn(tsn = 100800)
```

itis.**getfullrecordfromlsid**(*lsid*, ***kwargs*)
Returns the full ITIS record for the TSN in the LSID, found by comparing the TSN in the search key to the TSN field. Returns an empty result set if there is no match or the TSN is invalid.

Parameters

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

Usage:

```
pytaxize.getfullrecordfromlsid("urn:lsid:itis.gov:itis_tsn:180543")
pytaxize.getfullrecordfromlsid("urn:lsid:itis.gov:itis_tsn:37906")
pytaxize.getfullrecordfromlsid("urn:lsid:itis.gov:itis_tsn:100800")
```

itis.**getfullrecordfromtsn**(*tsn*, ***kwargs*)
Returns the full ITIS record for a TSN

Parameters

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

Usage:

```
pytaxize.getfullrecordfromtsn("504239")
pytaxize.getfullrecordfromtsn("202385")
pytaxize.getfullrecordfromtsn("183833")
```

itis.**getgeographicdivisionsfromtsn**(*tsn*, ***kwargs*)
Get geographic divisions from tsn

Usage:

```
pytaxize.getgeographicdivisionsfromtsn(180543)
```

itis.**getgeographicvalues**(***kwargs*)
Get all possible geographic values

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

Usage:

```
pytaxize.getgeographicvalues()
```

itis.**getglobalspeciescompletenessfromtsn**(*tsn*, ***kwargs*)
Get global species completeness from tsn

Usage:

```
pytaxize.getglobalspeciescompletenessfromtsn(180541)
```

itis.**gethierarchydownfromtsn**(*tsn*, ***kwargs*)
Get hierarchy down from tsn

Parameters **tsn** – TSN for a taxonomic group (numeric)

Usage:

```
pytaxize.gethierarchydownfromtsn(tsn = 161030)
```

itis.**gethierarchyupfromtsn**(*tsn*, ***kwargs*)
Get hierarchy up from tsn

Parameters **tsn** – TSN for a taxonomic group (numeric)

Usage:

```
pytaxize.gethierarchyupfromtsn(tsn = 36485)
pytaxize.gethierarchyupfromtsn(tsn = 37906)
```

itis.**getitistermsfromcommonname**(*x*, ***kwargs*)
Get itis terms from common names

Usage:

```
pytaxize.getitistermsfromcommonname("buya")
```

itis.**getitisterms**(*x*, ***kwargs*)
Get itis terms

Usage:

```
# fails
pytaxize.getitisterms("bear")
```

itis.**getitistermsfromscientificname**(*x*, ***kwargs*)
Get itis terms from scientific names

Usage:

```
pytaxize.getitistermsfromscientificname("ursidae")
pytaxize.getitistermsfromscientificname("Ursus")
```

itis.**itis_hierarchy**(*tsn=None*, *what='full'*)
Get hierarchies from TSN values, full, upstream only, or immediate downstream only. Uses the ITIS database.

Parameters

- **tsn** – One or more TSN's (taxonomic serial number)
- **what** – One of full (full hierarchy), up (immediate upstream), or down (immediate downstream)

Details Note that *pytaxize.itis_downstream* gets taxa downstream to a particular rank, whilc this function only gets immediate names downstream.

Usage:

```
# Get full hierarchy
pytaxize.itis_hierarchy(tsn=180543)

# Get hierarchy upstream
pytaxize.itis_hierarchy(tsn=180543, "up")

# Get hierarchy downstream
pytaxize.itis_hierarchy(tsn=180543, "down")

# Many tsn's
pytaxize.itis_hierarchy(tsn=[180543,41074,36616])
```

itis.**getjurisdictionaloriginfromtsn**(*tsn*, ***kwargs*)
Get jurisdictional origin from tsn

Usage:

```
pytaxize.getjurisdictionaloriginfromtsn(180543)
```

itis.**getjurisdictionoriginvalues**(***kwargs*)
Get jurisdiction origin values

Usage:

```
pytaxize.getjurisdictionoriginvalues()
```

itis.**getjurisdictionvalues**(***kwargs*)
Get possible jurisdiction values

Usage:

```
pytaxize.getjurisdictionvalues()
```

itis.**getkingdomnamefromtsn**(*tsn*, ***kwargs*)
Get kingdom names from tsn

Usage:

```
pytaxize.getkingdomnamefromtsn(202385)
```

itis.**getkingdomnames**(***kwargs*)
Get all possible kingdom names

Usage:

```
pytaxize.getkingdomnames()
```

itis.**getlastchangedate**(***kwargs*)
Provides the date the ITIS database was last updated.

Usage:

```
pytaxize.getlastchangedate()
```

itis.**getlsidfromtsn**(*tsn*, ***kwargs*)
Gets the unique LSID for the TSN, or an empty result if there is no match.

Usage:

```
# valid TSN
pytaxize.getlsidfromtsn(155166)
# invalid TSN, returns nothing
pytaxize.getlsidfromtsn(0)
```

itis.**getothersourcesfromtsn**(*tsn*, ***kwargs*)
Returns a list of the other sources used for the TSN.

Usage:

```
pytaxize.getothersourcesfromtsn(182662)
```

itis.**getparenttsnfromtsn**(*tsn*, ***kwargs*)
Returns the parent TSN for the entered TSN.

Usage:

```
pytaxize.getparenttsnfromtsn(202385)
```

itis.**getpublicationsfromtsn**(*tsn*, ***kwargs*)
Returns a list of the pulications used for the TSN.

Usage:

```
pytaxize.getpublicationsfromtsn(70340)
```

itis.**getranknames**(***kwargs*)
Provides a list of all the unique rank names contained in the database and their kingdom and rank ID values.

Usage:

```
pytaxize.getranknames()
```

itis.**getrecordfromlsid**(*lsid*, ***kwargs*)

Gets the partial ITIS record for the TSN in the LSID, found by comparing the TSN in the search key to the TSN field. Returns an empty result set if there is no match or the TSN is invalid.

Usage:

```
pytaxize.getrecordfromlsid("urn:lsid:itis.gov:itis_tsn:180543")
```

itis.**getreviewyearfromtsn**(*tsn*, ***kwargs*)

Returns the review year for the TSN.

Usage:

```
pytaxize.getreviewyearfromtsn(180541)
```

itis.**getscientificnamefromtsn**(*tsn*, ***kwargs*)

Returns the scientific name for the TSN. Also returns the component parts (names and indicators) of the scientific name.

Usage:

```
pytaxize.getscientificnamefromtsn(531894)
```

itis.**gettaxonauthorshipfromtsn**(*tsn*, ***kwargs*)

Returns the author information for the TSN.

Usage:

```
pytaxize.gettaxonauthorshipfromtsn(183671)
```

itis.**gettaxonomicranknamefromtsn**(*tsn*, ***kwargs*)

Returns the kingdom and rank information for the TSN.

Parameters **tsn** – TSN for a taxonomic group (numeric)

Usage:

```
pytaxize.gettaxonomicranknamefromtsn(tsn = 202385)
```

itis.**gettaxonomicusagefromtsn**(*tsn*, ***kwargs*)

Returns the usage information for the TSN.

Usage:

```
pytaxize.gettaxonomicusagefromtsn(526852)
```

itis.**gettsnbyvernacularlanguage**(*language*, ***kwargs*)

Get tsn by vernacular language not the international language code (character)

Usage:

```
pytaxize.gettsnbyvernacularlanguage("french")
```

itis.**gettsnfromlsid**(*lsid*, ***kwargs*)

Gets the TSN corresponding to the LSID, or an empty result if there is no match.

Usage:

```
pytaxize.gettsnfromlsid(lsid="urn:lsid:itis.gov:itis_tsn:28726")
pytaxize.gettsnfromlsid("urn:lsid:itis.gov:itis_tsn:0")
```

itis.**getunacceptabilityreasonfromtsn**(*tsn*, ***kwargs*)
Returns the unacceptability reason, if any, for the TSN.

Usage:

```
pytaxize.getunacceptabilityreasonfromtsn(183671)
```

itis.**getvernacularlanguages**(***kwargs*)
Provides a list of the unique languages used in the vernacular table.

Usage:

```
pytaxize.getvernacularlanguages()
```

itis.**searchbycommonname**(*x*, ***kwargs*)
Search for tsn by common name

Usage:

```
pytaxize.searchbycommonname(x="american bullfrog")
pytaxize.searchbycommonname("ferret-badger")
pytaxize.searchbycommonname("polar bear")
```

itis.**searchbycommonnamebeginswith**(*x*, ***kwargs*)
Search for tsn by common name beginning with

Usage:

```
pytaxize.searchbycommonnamebeginswith("inch")
```

itis.**searchbycommonnameendswith**(*x*, ***kwargs*)
Search for tsn by common name ending with

Usage:

```
pytaxize.searchbycommonnameendswith("snake")
```

itis.**itis_searchcommon**(*x*, *which='begin'*, ***kwargs*)
Searches common name and acts as thin wrapper around *pytaxize.searchbycommonnamebeginswith* and *pytaxize.searchbycommonnameendswith*

Usage:

```
pytaxize.itis_searchcommon("inch")
pytaxize.itis_searchcommon("inch", which = "end")
```

itis.**searchbyscientificname**(*x*, ***kwargs*)
Search by scientific name

Usage:

```
pytaxize.searchbyscientificname(x="Tardigrada")
pytaxize.searchbyscientificname("Quercus douglasii")
```

itis.**searchforanymatch**(*x*, ***kwargs*)
Search for any match

Usage:

```
pytaxize.searchforanymatch(x=202385)
pytaxize.searchforanymatch(x="dolphin")
```

itis.**searchforanymatchpaged**(*x*, *pagesize*, *pagenum*, *ascend*, ***kwargs*)

Search for any matched page for descending (logical)

Usage:

```
pytaxize.searchforanymatchpaged(x=202385, pagesize=100, pagenum=1, ascend=False)
pytaxize.searchforanymatchpaged("Zy", pagesize=100, pagenum=1, ascend=False)
```

Changelog

0.5.0 (2016-02-16)

- xxxx

License

MIT

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

V