pyNEID

Release 0.1.0

Mihseh Kong

Jul 09, 2021

CONTENTS:

This package is designed to be an API interface to the NEID Data Archive.

With pyNEID you can:

- log in using your NEID Archive credentials, or use without login for access to public data only
- query the archive for tables of L0, L1, and L2 metadata
- download FITS files of any level
- more features to come…

The package may be installed via pip:

```
$ pip install git+https://github.com/Caltech-IPAC/pyNEID.git
```

Or by downloading directly from the GitHub repository.

The *Archive class* contains many useful methods for working with the NIED archive.

This package is under active development by the NASA Exoplanet Science Institute.

Please report any bugs, issues, or feature requests to the NEID Help Desk.

CHAPTER

ONE

API

1.1 Archive Class

class pyneid.neid.**Archive**(***kwargs*)

'Archive' class provides NEID archive access functions for searching NEID data via TAP interface.

The user's NEID credential given at login as a cookie file is used to search the proprietary data. In addition in the same python session as 'login', a token string will be saved in memory for searching the proprietary data without entering the cookie file.

In the following examples, "Neid" will represent an instance of the Archive class:

Examples

```
>>> import os
>>> import sys
```

```
>>> from pyneid.neid import Neid
```

```
>>> Neid.query_datetime ('l0',
>>>                       '2020-01-01 00:00:00/2020-12-31 23:59:59',
>>>                       outpath='./meta.xml')
```

download(*metapath*, *datalevel*, *format*, *outdir*, ***kwargs*)

The download method allows users to download FITS files shown in the retrieved metadata file. The column 'filepath' must be included in the metadata file columns in order to download files.

Required Arguments:

metapath (string): a full path metadata table obtained from running query methods

datalevel (string): l0, l1, l2, eng, solarl0, solarl1, solarl2, or solareng

format (string): metasata table's format: ipac, votable, csv, or tsv.

outdir (string): the directory for depositing the returned files

Optional Keyword Arguments:

cookiepath (string): cookie file path for downloading the proprietary NEID data.

start_row (integer): starting row

end_row (integer): ending row

Exampled:

```
>>> Neid.download ('./criteria.tbl',
>>>     'l0',
>>>     'ipac',
>>>     './dnload_dir',
>>>     cookiepath='mycookie',
>>>     start_row=0,
>>>     end_row=10)
```

login(***kwargs*)

login method validates a user has a valid NEID account; it takes two 'keyword' arguments: userid and password. If the inputs are not provided in the keyword, the auth method prompts for inputs.

Required Keyword Arguments:

```
>>> userid     (string): a valid user id assigned by NEID;
>>> password   (string): a valid password in the NEID's user table;
```

Optional Keyword Arguments:

```
>>> cookiepath (string): a file path provided by the user to save
>>>     returned cookie (in login method) or to serve as input
>>>     argument for the subsequent Neid query and download methods.
```

Calling synopsis:

```
>>> Neid.login (userid='xxxx',
>>>             password='xxxxxx',
>>>             cookiepath='mycookie'), or
```

```
>>> Neid.login (cookiepath='mycookie'): and the program will
>>>     prompt for userid and password
```

login method returns both cookie header and a token string in the returned message jason structure.

If cookiepath is provided, the cookie will be saved to the cookiepath.

The token string will be saved in the variable "token" in memory to be used for other Neid methods in the same python session.

query_adql(*query*, ***kwargs*)

'query_adql' method receives a qualified ADQL query string from user input.

Required Arguments:

query (string): a ADQL query

Optional Keyword Arguments:

outpath (string): a full output filepath of the returned metadata table

cookiepath (string): a full cookie file path saved from login for querying the proprietary NEID data.

format (string): votable, ipac, csv, tsv (default: votable)

maxrec (integer): (optional) maximum records to be returned default: -1 or not specified will return all requested records

Example

```
>>> query = "select * from neidl0"
```

```
>>> Neid.query_adql (query,
>>>     cookiepath='mycookie',
>>>     outpath='./adql..tbl')
```

query_criteria(*param*, ***kwargs*)

'query_criteria' method allows the search of NEID data by multiple the parameters specified in a dictionary (param).

Required Arguments:

param (dict): a dictionary containing the allowable Neid search parameters – datalevel, datetime, position, target, program; it is a way of combining multiple constraints in a search.

Optional Keyword Arguments:

outpath (string): a full output filepath for the returned metadata table

cookiepath (string): a full cookie file path saved from login for querying the proprietary NEID data.

format (string): votable, ipac, csv, tsv (default: votable)

maxrec (integer): maximum records to be returned default: -1 or not specified will return all requested records

Example

```
>>> param = dict()
>>> param['datalevel'] = 'l0'
>>> param['datetime'] = '2021-01-14 00:00:00/2021-01-14 23:59:59'
>>> param['object'] = 'HD 95735'
```

```
>>> Neid.query_criteria (param,
>>>     cookiepath='./neidadmincookie.txt',
>>>     format='ipac',
>>>     outpath='./criteria.tbl')
```

query_datetime(*datalevel*, *datetime*, ***kwargs*)

'query_datetime' method search NEID data by 'datetime' range

Required Arguments:

datalevel (string): l0, l1, l2, eng, solarl0, solarl1, solarl2, solareng datetime (string): a datetime string in the format of

datetime1/datetime2 where '/' separates the two datetime values of format 'yyyy-mm-dd hh:mm:ss'

the following inputs are acceptable:

datetime1/: will search data with datetime later than (>=) datetime1

/datetime2: will search data with datetime earlier than (<=) datetime2

datetime1: will search data with datetime equal to (=) datetime1, (this is not recommended)

Optional Keyword Arguments:

outpath (string): a full output filepath of the returned metadata table; without this input the returned data will be sa table in the memory.

cookiepath (string): (optional) a full cookie file path saved from login for querying the proprietary NEID data.

format (string): (optional) Output format: votable, ipac, csv, tsv (default: votable)

maxrec (integer): (optional) maximum records to be returned default: -1 or not specified will return all requested records

Examples

```
>>> Neid.query_datetime ('l0',
>>>                      '2020-11-16 06:10:55/2020-11-18 00:00:00',
>>>                      outpath=outpath)
```

```
>>> Neid.query_datetime ('l1',
>>>                      '2020-11-16 06:10:55/',
>>>                      cookiepath='mycookie',
>>>                      outpath=outpath)
```

```
>>> Neid.query_datetime ('l2',
>>>                      '/2020-112-18 00:00:00',
>>>                      outpath=outpath)
```

query_object(*datalevel*, *object*, ***kwargs*)

'query_object' method search NEID data by 'object name'. This method resolves the object name into coordiates to be used as the center of the circle position search with default radius of 0.5 deg.

Required Arguments:

datalevel: l0, l1, l2, eng, solarl0, solarl1, solarl2, or solareng

object (string): an object name resolvable by SIMBAD, NED, and ExoPlanet's name_resolve;

Optional Keyword Arguments:

outpath (string): a full output filepath of the returned metadata table

cookiepath (string): a full cookie file path saved from login for querying the proprietary NEID data.

format (string): votable, ipac, csv, tsv (default: votable)

maxrec (integer): maximum records to be returned default: -1 or not specified will return all requested records

Example

```
>>> Neid ('l1',
>>>       'WD 1145+017',
>>>       cookiepath= 'mycookie',
>>>       outpath=outpath)
```

query_piname(*datalevel*, *piname*, ***kwargs*)
'query_piname' method search NEID data by PI name

Required Arguments:

datalevel (string): l0, l1, l2, eng, solarl0, solarl1, solarl2, or solareng

piname (string): PI name as formated in the project's catalog

Optional Keyword Arguments:

outpath (string): a full output filepath of the returned metadata table

cookiepath (string): a full cookie file path saved from login for querying the proprietary NEID data.

format (string): votable, ipac, csv, tsv (default: votable)

maxrec (integer): maximum records to be returned default: -1 or not specified will return all requested records

query_position(*datalevel*, *position*, ***kwargs*)
'query_position' method search NEID data by 'position'

Required Arguments:

datalevel (string): l0, l1, l2, eng, solarl0, solarl1, solarl2, solareng position (string): a position string in the format of

1. circle ra dec radius;
2. polygon ra1 dec1 ra2 dec2 ra3 dec3 ra4 dec4;
3. box ra dec width height;

All ra dec should be specified in decimal degree J2000 coordinates.

e.g. instrument = 'l1', pos = 'circle 230.0 45.0 0.5'

Optional Keyword Arguments:

outpath (string): a full output filepath for the returned metadata table

cookiepath (string): a full cookie file path saved from login for querying the proprietary NEID data.

format (string): votable, ipac, csv, tsv (default: votable)

maxrec (integer): maximum records to be returned default: -1 or not specified will return all requested records

Examples

```
>>> Neid.query_position ('l1',
>>>                      'circle 230.0 45.0 0.5',
>>>                      cookiepath='mycookie',
>>>                      outpath=outpath)
```

query_program(*datalevel*, *program*, ***kwargs*)

'query_program' method search NEID data by 'program'

Required Arguments:

datalevel (string): l0, l1, l2, eng, solarl0, solarl1, solarl2, solareng

program (string): program ID in the project's catalog

Optional Keyword Arguments:

outpath (string): a full output filepath of the returned metadata table

cookiepath (string): a full cookie file path saved from login for querying the proprietary NEID data.

format (string): votable, ipac, csv, tsv (default: ipac)

maxrec (integer): maximum records to be returned default: -1 or not specified will return all requested records

Examples

```
>>> Neid.query_position ('l1',
>>>                      'circle 230.0 45.0 0.5',
>>>                      cookiepath='mycookie',
>>>                      outpath=outpath)
```

query_qobject(*datalevel*, *qobject*, ***kwargs*)

'query_qobject' method search NEID data for 'qobject' column value. This method resolves the object name into coordiates to be used as the center of the circle position search with default radius of 0.5 deg.

Required Arguments:

datalevel: l0, l1, l2, eng, solarl0, solarl1, solarl2, solareng

qobject (string): an object name as specified in the QOBJECT column. This is usually the Gaia DR2 ID

Optional Keyword Arguments:

outpath (string): a full output filepath of the returned metadata table

cookiepath (string): a full cookie file path saved from login for querying the proprietary NEID data.

format (string): votable, ipac, csv, tsv (default: votable)

maxrec (integer): maximum records to be returned default: -1 or not specified will return all requested records

1.2 NeidTap Class

class pyneid.neid.**NeidTap**(*url*, ***kwargs*)

NeidTap class provides client access to NEID's TAP service. Public data doesn't not require user login, optional NEID login via NeidLogin class are used to search a user's proprietary data.

Parameters

- **query** (*string*) – a SQL statement in specified query language request (string): (optional) default 'doQuery' lang (string): (optional) default 'ADQL' phase (string): (optional) default 'RUN' format (string): (optional) default 'votable' maxrec (int): (optional) default '2000'
- **cookiefile** (*string*) – a full path cookie file containing user info
- **debug** (*bool*) – default False

Examples

```
>>> service = NeidTap(url, cookiefile=cookiepath)
# or
>>> service = NeidTap(url)
# or
>>> job = service.send_async (query, format='votable', request='doQuery', ...)
# or
>>> job = service.send_sync (query, format='votable', request='doQuery', ...)
```

get_data(*resultpath*)
loop until job is complete, then download the data to the given resultpath

msg
tapjob contains async job's status; resulttbl is the result of sync saved an astropy table

1.3 TapJob Class

class pyneid.neid.**TapJob**(*statusurl*, ***kwargs*)
TapJob class is used internally by TapClient class to store a Tap job's parameters and returned job status and result urls.

1.4 Object Lookup Class

class pyneid.neid.**objLookup**(*object*, ***kwargs*)

objLookup wraps ExoPlanet's web name resolver into a python class; the exoLookup checks the exoplanet archive database and if that fails it checks with the Sesame web service at CDS. Sesame checks the CDS database and if that fails it checks NED. So this class covers SIMBAD, NED, and ExoPlanet search.

Parameters **object** (*char*) – object name to be resolved

debug = 0
{ objLookup.init

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX