nasapy Documentation

Release 0.2.3

Aaron Schlegel

Jun 16, 2020

Contents

`nasapy` is a Python wrapper for the nasa.gov API.

CHAPTER 1

Contents

1.1 API Reference

1.1.1 `Nasa` - NASA API Wrapper

class Nasa([*key=None*])

Class object containing the methods for interacting with NASA API endpoints that require an API key.

Parameters **key** – The generated API key received from the NASA API. Registering for an API key can be done on the NASA API webpage. If `None`, a 'DEMO_KEY' with a much more restricted access limit is used.

Astronomy Picture of the Day

`Nasa`.**picture_of_the_day**(*[date=None][, hd=False]*)

Returns the URL and other information for the NASA Astronomy Picture of the Day.

Parameters

- **date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date.
- **hd** – If True, returns the associated high-definition image of the Astrononmy Picture of the Day.

Return type dict. Dictionary object of the JSON data returned from the API.

```
# Initialize Nasa API Class with a demo key
n = Nasa()
# Return today's picture of the day
n.picture_of_the_day()
# Return a previous date's picture of the day with the high-definition URL␣
↪included.
n.picture_of_the_day('2019-01-01', hd=True)
```

Mars Weather Insight

Nasa.**mars_weather**()

Returns per-Sol (Martian Days) summary data for each of the last seven available Sols. Data is provided by NASA's InSight Mars lander and as such data for particular Sols may be recalculated as more data is received. For more information on the data returned, please see NASA's documentation.

Return type dict. Dictionary object representing the returned JSON data from the API.

```
# Initialize NASA API object with a demo key
n = NASA()
# Return the most recent data for the previous seven Sols (Martian Days)
n.mars_weather()
```

Near Earth Objects

All the data is from the NASA JPL Asteroid team (http://neo.jpl.nasa.gov/). The API is maintained by the SpaceRocks team

Nasa.**asteroid_feed**(*[start_date][, end_date=None]*)

Returns a list of asteroids based on their closest approach date to Earth.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to seven days after the provided `start_date`.

Return type dict. Dictionary representing the returned JSON data from the API.

```
# Initialize the NASA API with a demo key.
n = NASA()
# Get asteroids approaching Earth at the beginning of 2019.
n.asteroid_feed(start_date='2019-01-01')
```

Nasa.**get_asteroids**([*asteroid_id=None*])

Returns data from the overall asteroid data-set or specific asteroids given an ID.

Parameters **asteroid_id** – If None, the entire asteroid data set is returned. If an `asteroid_id` is provided, data on that specific asteroid is returned.

Return type dict. Dictionary object representing the returned JSON data from the NASA API.

```
n = Nasa()
# Get entire asteroid data set.
n.get_asteroids()
# Get asteroid with ID 3542519
n.get_asteroids(asteroid_id=3542519)
```

DONKI (Space Weather Database of Notifications, Knowledge, and Information)

The Space Weather Database Of Notifications, Knowledge, Information (DONKI) is a comprehensive on-line tool for space weather forecasters, scientists, and the general space science community. DONKI provides chronicles the daily interpretations of space weather observations, analysis, models, forecasts, and notifications provided by the Space Weather Research Center (SWRC), comprehensive knowledge-base search functionality to support anomaly

resolution and space science research, intelligent linkages, relationships, cause-and-effects between space weather activities and comprehensive webservice API access to information stored in DONKI.

Nasa.**coronal_mass_ejection**(*[start_date=None][, end_date=None][, accurate_only=True][, speed=0][, complete_entry=True][, half_angle=0][, catalog='ALL'][, keyword=None]*)

Returns data collected on coronal mass ejection events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.
- **accurate_only** – If True (default), only the most accurate results collected are returned.
- **complete_entry** – If True (default), only results with complete data is returned.
- **speed** – The lower limit of the speed of the CME event. Default is 0
- **half_angle** – The lower limit half angle of the CME event. Default is 0.
- **catalog** – Specifies which catalog of data to return results. Defaults to 'ALL' and must be one of {'ALL', 'SWRC_CATALOG', 'JANG_ET_AL_CATALOG'}.
- **keyword** – Filter results by a specific keyword.

Return type list. List of results representing returned JSON data. If no data is returned, an empty dictionary is returned.

```
# Initialize NASA API with a demo key
n = Nasa()
# View data from coronal mass ejection events from the last thirty days
n.coronal_mass_ejection()
# View all CME events from the beginning of 2019.
n.coronal_mass_ejection(start_date='2019-01-01', end_date=datetime.datetime.
↪today())
```

Nasa.**geomagnetic_storm**(*[start_date=None][,end_date=None]*)

Returns data collected on geomagnetic storm events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.

Return type list: List of results representing returned JSON data. If no data is returned, an empty dictionary is returned.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get geomagnetic storm events from the last thirty days.
n.geomagnetic_storm()
```

Nasa.**interplantary_shock**(*[start_date=None][, end_date=None][, location='ALL'][, catalog='ALL']*)

Returns data collected on interplantary shock events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.
- **location** – Filters returned results to specified location of the interplantary shock event. Defaults to 'ALL' and must be one of {'ALL', 'Earth', 'MESSENGER', 'STEREO A', 'STEREO B'}
- **catalog** – Filters results to a specified catalog of collected data. Defaults to 'ALL' and must be one of {'ALL', 'SWRC_CATALOG', 'WINSLOW_MESSENGER_ICME_CATALOG'}

Return type list. List of results representing returned JSON data. If no data is returned, an empty list is returned.

Nasa.**solar_flare**(*[start_date=None][, end_date=None]*)

Returns data on solar flare events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.

Return type list. If data is available in the specified date range, a list of dictionary objects representing the data from the API is returned. If no data is available, an empty dictionary is returned.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get solar flare events from May of 2019
n.solar_flare(start_date='2019-05-01', end_date='2019-05-31')
```

Nasa.**solar_energetic_particle**(*[start_date=None][, end_date=None]*)

Returns data available related to solar energetic particle events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.

Return type list. If data is available in the specified date range, a list of dictionary objects representing the data from the API is returned. If no data is available, an empty dictionary is returned.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get data from April 2017
n.solar_energetic_particle(start_date='2017-04-01', end_date='2017-04-30')
```

Nasa.**magnetopause_crossing**(*[start_date=None][, end_date=None]*)

Returns data available related to magnetopause crossing events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.

- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.

Return type list. If data is available in the specified date range, a list of dictionary objects representing the data from the API is returned. If no data is available, an empty dictionary is returned.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get data on magnetopause crossing events from 2018 to the current date.
n.magnetopause_crossing(start_date='2018-01-01')
```

Nasa.**radiation_belt_enhancement**(*[start_date=None][, end_date=None]*)

Returns data available related to radiation belt enhancement events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.

Return type list. If data is available in the specified date range, a list of dictionary objects representing the data from the API is returned. If no data is available, an empty dictionary is returned.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get data on radiation belt enhancement events from the last 30 days.
n.radiation_belt_enhancement()
```

Nasa.**hight_speed_stream**(*[start_date=None][, end_date=None]*)

Returns data available related to hight speed stream events.

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.

Return type list. If data is available in the specified date range, a list of dictionary objects representing the data from the API is returned. If no data is available, an empty dictionary is returned.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get data on hight speed stream events from the beginning of September 2019.
n.hight_speed_stream()
```

Nasa.**wsa_enlil_simulation**(*[start_date=None][, end_date=None]*)

Parameters

- **start_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to 30 days prior to the current date in UTC time.
- **end_date** – String representing a date in YYYY-MM-DD format or a datetime object. If None, defaults to the current date in UTC time.

Return type list. If data is available in the specified date range, a list of dictionary objects representing the data from the API is returned. If no data is available, an empty dictionary is returned.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get data from the first simulation performed in 2019.
wsa = n.wsa_enlil_simulation(start_date='2019-01-01')
```

EPIC (Earth Polychromatic Imaging Camera)

The EPIC API provides information on the daily imagery collected by DSCOVR's Earth Polychromatic Imaging Camera (EPIC) instrument. Uniquely positioned at the Earth-Sun Lagrange point, EPIC provides full disc imagery of the Earth and captures unique perspectives of certain astronomical events such as lunar transits using a 2048x2048 pixel CCD (Charge Coupled Device) detector coupled to a 30-cm aperture Cassegrain telescope.

`Nasa.`**`epic`**(*[color='natural'][, date=None][, available=False]*)

Parameters

- **`color`** – Specifies the type of imagery to return. Must be one of 'natural' (default) or 'enhanced'
- **`date`** – String representing a date in 'YYYY-MM-DD' format or a datetime object
- **`available`** – Alternative listing of all dates with specified color imagery

Return type list. List of dictionaries representing the returned JSON data from the EPIC API.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get EPIC data from the beginning of 2019.
e = n.epic(date='2019-01-01')
# Print the first result
e[0]
```

Exoplanets

`Nasa.`**`exoplanets`**(*[table='exoplanets'][, select=None][, count=None][, colset=None][, where=None][, order=None][, ra=None][, dec=None][, aliastable=None][, objname=None][, return_df=False]*)

Parameters

- **`table`** – Specifies which table to query. Defaults to the 'exoplanets' table.
- **`select`** – Specifies which columns within the chosen table to return. Multiple columns can be returned by comma-separating the column names and distinct values can be returned by adding 'distinct ' in front of the desired column names.
- **`count`** – Can be used to return the number of rows which fulfill the given query, including queries using where clauses or cone searches.
- **`colset`** – Returns a set of pre-defined columns that have been created by the archive. Currently, this keyword is only used by the Composite Planet Data ('compositepars') table.
- **`where`** – Takes a SQL-like query string to filter the returned results. Please see the examples section for more.
- **`order`** – Returns the data sorted by the specified column. Append ' desc' for descending or ' asc' for ascending values.
- **`ra`** – Specifies an area of the sky to search for all objects within that area.

- **dec** – Specifies an area of the sky to search for all objects within that area.
- **aliastable** – Requests a list of aliases for a particular confirmed planet.
- **objname** – When parameter `aliastable` is specified, `objname` must also be passed with the planet's name.
- **return_df** – If `True`, returns the JSON data as a pandas DataFrame.

Return type dict or pandas DataFrame.

```
# Get all exoplanets data as a pandas DataFrame.
exoplanets(return_df=True)
# Get all confirmed planets in the Kepler field.
exoplanets(where='pl_kepflag=1')
# Stars known to host exoplanets as a pandas DataFrame.
exoplanets(select='distinct pl_hostname', order='pl_hostname', return_df=True)
```

Earth Satellite Imagery

This endpoint retrieves the Landsat 8 image for the supplied location and date. The response will include the date and URL to the image that is closest to the supplied date. The requested resource may not be available for the exact date in the request.

`Nasa.`**`earth_imagery`**(*lat, lon[, dim=0.025][, date=None][, cloud_score=False]*)

Retrieves the URL and other information from the Landsat 8 image database for the specified lat/lon location and date.

Parameters

- **lat** – Latitude of the desired imagery location
- **lon** – Longitude of the desired imagery location
- **dim** – Width and height of the image in degrees.
- **date** – Date the image was taken. If specified, must be a string representing a date in 'YYYY-MM-DD' format or a datetime object. If None, the most recent image available from the current date is returned.
- **cloud_score** – Calculate the percentage of the image covered by clouds.

Return type dict. Dictionary object representing the returned JSON data from the API.

```
# Initialize API connection with a Demo Key
n = Nasa()
# Get imagery at latitude 1.5, longitude 100.75 and include the computed cloud␣
↪score calculation.
n.earth_imagery(lon=100.75, lat=1.5, cloud_score=True)
```

`Nasa.`**`earth_assets`**(*lat, lon[, dim=0.025][, begin_date=None][, end_date=None]*)

Retrieves the datetimes and asset names of available imagery for a specified lat-lon location over a given date range. The satellite that takes the images passes over each point approximately once every sixteen days.

Parameters

- **lat** – Latitude of the desired imagery location
- **lon** – Longitude of the desired imagery location
- **begin_date** – Beginning of date range in which to search for available assets. Must be a string representing a date in 'YYYY-MM-DD' format or a datetime object

- **end_date** – End of date range in which to search for available assets. If not specified, defaults to the current date. If specified, Must be a string representing a date in 'YYYY-MM-DD' format or a datetime object

Return type dict. Dictionary object representing the returned JSON data from the API.

```
# Get assets available beginning from 2014-02-01 at lat-lon 100.75, 1.5
n.earth_assets(lat=100.75, lon=1.5, begin_date='2014-02-01')
```

Mars Rover Photos

Nasa.**mars_rover**(*[sol=None][, earth_date=None][, camera='all'][, rover='curiosity'][, page=1]*)

Retrieves image data collected by the Mars rovers Curiosity, Discovery and Spirit.

Parameters

- **sol** – The sol (Martian rotation or day) on which the images were collected. Either this parameter or `earth_date` must be provided. The parameter `earth_date` is an alternative parameter for searching for a specific date. The sol values count up from the rover's landing date, for example, the Curiosity's 100th sol would be the 100th Martian rotation since the rover landed.
- **earth_date** – Alternative search parameter for finding data on a specific date. Must be a string representing a date in 'YYYY-MM-DD' format or a datetime object. Either `earth_date` or `sol` must be specified.
- **camera** – Filter results to a specific camera on the Mars Curiosity, Opportunity or Spirit rovers. Defaults to 'all', which includes all cameras and must be one of {'all', FHAZ', 'RHAZ', 'MAST', 'CHEMCAM', 'MAHLI', 'MARDI', 'NAVCAM', 'PANCAM', 'MINITES'}
- **rover** – Specifies the Mars rover to return data. Defaults to the Curiosity rover which has more available cameras. Must be one of {'curiosity', 'opportunity', 'spirit'}
- **page** – Page number of results to return. 25 results per page are returned.

Return type list. List of dictionaries representing the returned JSON data from the Mars Rover API.

GeneLab Search

Nasa.**genelab_search**(*term=None*, *database='cgene'*, *page=0*, *size=25*, *sort=None*, *order='desc'*, *ffield=None*, *fvalue=None*)

Retrieves available data from the GeneLab and other bioinformatics databases such as the National Institutes of Health (NIH) / National Center for Biotechnology Information (NCBI), Gene Expression Omnibus (GEO), the European Bioinformatics Institute's (EBI) Proteomics Identification (PRIDE), and the Argonne National Laboratory's (ANL) Metagenomics Rapid Annotations using Subsystems Technology (MG-RAST).

Parameters

- **term** – Search by specific keyword(s). Case-insensitive boolean operators (AND, OR, NOT) can be used as well to include and filter specific keywords.
- **database** – Determines the database(s) to query. Defaults to the 'cgene' (GeneLab) database, but other available databases include 'nih_geo_gse' (NIH GEO), 'ebi_pride' (EBI PRIDE), or 'mg_rast' (MG-RAST). Multiple databases can be queried by separating values with commas. For example, 'cgene,nih_geo_gse,ebi_pride,mg_rast' will query all available databases.
- **page** – Specifies the page of results to return. Defaults to the first page (0).

- **size** – Specifies the number of results to return per page. Default is 25 results per page.
- **sort** – Sorts by a specific field name in the returned JSON data.
- **order** – Determines the sorting order. Must be one of 'desc' (descending) or 'asc' (ascending).
- **ffield** – Filters the returned data based on the defined field. Should be paired with the `fvalue` parameter. Only the 'cgene' (GeneLab) database can be filtered.
- **fvalue** – Filters the returned data based on value or values in the specified `ffield` parameter field. Only the 'cgene' (GeneLab) database can be filtered.

Return type dict. Dictionary object representing the returned JSON data.

```
# Find Gene studies in the cgene database related to 'mouse liver'
n.genelab_search(term='mouse liver')
```

Techport

The NASA TechPort system provides an API to make technology project data available in a machine-readable format. This API can be used to export TechPort data into either an XML or a JSON format.

`Nasa.`**techport**(*[project_id=None][, last_updated=None][, return_format='json']*)

Retrieves available NASA project data.

Parameters

- **project_id** – The ID of the project record. If not specified, all available projects will be returned.
- **last_updated** – Returns projects only updated after the specified date. Must be a string representing a date in 'YYYY-MM-DD' format or a datetime object.
- **return_format** – Specifies the return format of the data. Defaults to 'json', but 'xml' formatted data is also available.

Return type dict or str. If `return_format` is 'json', a dictionary representing the JSON formatted data is returned. Otherwise, a string formatted for XML is returned.

TLE (Two-Line Element Set Data)

The TLE API provides up to date two line element set records, the data is updated daily from CelesTrak and served in JSON format. A two-line element set (TLE) is a data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time.

tle(*[search_satellite=None][, satellite_number=None]*)

Returns two-line element set records provided by CelesTrak.

Parameters

- **search_satellite** – Searches satellites by name designation.
- **satellite_number** – Specfic satellite ID number.

Return type dict. Specfic satellite ID number.

```
tle(satellite_number=43553)
```

NASA Image and Video Library

media_search(*[query=None][, center=None][, description=None][, keywords=None][, location=None][, media_type=None][, nasa_id=None][, page=1][, photographer=None][, secondary_creator=None][, title=None][, year_start=None][, year_end=None]*)

Performs a general search for images from the images.nasa.gov API based on parameters and criteria specified. At least one parameter must be provided.

Parameters

- **query** – Query terms to search.
- **center** – NASA center that published the results.
- **description** – Search for specific terms in the 'description' field of the resulting data.
- **keywords** – Search and filter for specific terms in the 'keywords' field of the resulting data. Multiple values should be comma-separated.
- **location** – Filter terms in the 'locations' field of the resulting data.
- **media_type** – Filter results to specific media types. Options include 'image', 'audio', 'image,audio', 'audio,image'. The default `None` includes all media types.
- **nasa_id** – The media asset's NASA ID.
- **page** – Page number of results to return. Starts at 1.
- **photographer** – The primary photographer's name.
- **secondary_creator** – A secondary photographer/videographer's name.
- **title** – Search terms in the 'title' field of the resulting data.
- **year_start** – The start year for results. If provided, must be a string representing a year in YYYY format or a datetime object.
- **year_end** – The end year for results. If provided, must be a string representing a year in YYYY format or a datetime object.

Return type dict. Dictionary containing matching search results.

```
# Search for media related to 'apollo 11' with 'moon landing' in the description␣
↪of the items.
r = media_search(query='apollo 11', description='moon landing')
# Output the first returned media item from the resulting collection.
r['items'][0]
```

media_asset_manifest(*nasa_id*)

Returns the media asset's manifest, which contains the available versions of the asset and it's metadata location.

Parameters **nasa_id** – The ID of the media asset.

Return type list. List of dictionaries containing the media asset's manifest.

```
# Get the manifest for the NASA media asset 'as11-40-5874'
media_asset_manifest(nasa_id='as11-40-5874')
```

media_asset_metadata(*nasa_id*)

Retrieves the specified media asset's metadata.

Parameters **nasa_id** – The ID of the media asset.

Return type dict. Dictionary containing the metadata of the provided media asset ID.

media_asset_captions(*nasa_id*)

Retrieves the captions and location of the captions .srt file for a media asset from the NASA image API.

Parameters **nasa_id** – The ID of the media asset.

Return type dict. Dictionary object containing the resulting data from the API given the media asset ID. The dictionary will contain two keys, `location` and `captions`. The `location` key can be used to download the .srt file directly while the `captions` key can be used in conjunction with a library such as srt for parsing media asset captions.

Solar System Dynamics (SSD) and Center for Near-Earth Object Studies (CNEOS)

The following functions provide a Pythonic interface for NASA's Solar System Dynamics and Center for Near-Earth Object Studies APIs.

close_approach(*[date_min='now'][, date_max='+60'][, dist_min=None][, dist_max='0.05'][, h_min=None][, h_max=None][, v_inf_min=None][, v_inf_max=None][, v_rel_min=None][, v_rel_max=None][, orbit_class=None][, pha=False][, nea=False][, comet=False][, nea_comet=False][, neo=False][, kind=None][, spk=None][, des=None][, body='Earth'][, sort='date'][, limit=None][, fullname=False]*)

Provides data for currently known close-approach data for all asteroids and comets in NASA's Jet Propulsion Laboratory's (JPL) Small-Body Database.

Parameters

- **date_min** – Excludes data earlier than the given date. Defaults to 'now', representing the current date, but can also be a string representing a date in 'YYYY-MM-DD' format or 'YYYY-MM-DDThh:mm:ss' format or a datetime object.
- **date_max** – Excludes data later than the given date. Defaults to '+60', representing 60 days after the `date_min` parameter. Accepts a string of '+D' where D represents the number of days or a string representing a date in 'YYYY-MM-DD' format or 'YYYY-MM-DDThh:mm:ss' format or a datetime object. 'now' is also an acceptable value and will exclude date later than the current date.
- **dist_min** – Excludes data with an approach distance less than the given value (if provided). The default unit is AU (astronomical units), and LD (lunar distance) is also available. For example, '0.05' or 0.05 would return AU units whereas '0.05LD' would return LD units.
- **dist_max** – Excludes data with an approach distance greater than the given value (if specified). The default unit is AU (astronomical units), and LD (lunar distance) is also available. For example, '0.05' would return AU units whereas '0.05LD' would return LD units.
- **h_min** – Exclude data from objects with H-values less than the given value.
- **h_max** – Exclude data from objects with H-values greater than the given value.
- **v_inf_min** – Exclude data with V-infinity less than this positive value in km/s
- **v_inf_max** – Exclude data with V-infinity greater than this positive value in km/s
- **v_rel_min** – Exclude data with V-relative less than this positive value in km/s
- **v_rel_max** – Exclude data with V-relative greater than this positive value in km/s
- **orbit_class** – Limits data to specified orbit-class
- **pha** – If True, limits the resulting data to only PHA objects
- **nea** – If True, limits the returned data to only NEA objects

- **comet** – If True, limits the returned data to comet objects only
- **nea_comet** – If True, limits the returned data to NEA comet objects only
- **neo** – If True, limits the returned data to only NEO objects
- **kind** – Filters returned data to specified type of object. Available options include 'a'=asteroid, 'an'=numbered-asteroids, 'au'=unnumbered-asteroids, 'c'=comets, 'cn'=numbered-comets, 'cu'=unnumbered-comets, 'n'=numbered-objects, and 'u'=unnumbered-objects
- **spk** – Return data only for the matching SPK-ID.
- **des** – Filters data to objects matching the given destination.
- **body** – Filters data to close-approaches of the specified body. 'ALL' or '*' returns all close-approaches to the available bodies.
- **sort** – Sorts the returned data by the specified field. Defaults to 'date' ascending. To sort by descending, add a '-' in front of the sort value, for example, '-date'.
- **limit** – Limit data to the first number of results specified by the parameter. Must be greater than 0.
- **fullname** – Includes the full-format object name/designation

Return type dict

```
# Get all close-approach object data in the year 2019 with a maximum approach␣
↪distance of 0.01AU.
nasapy.close_approach(date_min='2019-01-01', date_max='2019-12-31', dist_max=0.01)
# Get close-approach data for asteroid 433 Eros within 0.2AU from the years 1900␣
↪to 2100.
nasapy.close_approach(des='433', date_min='1900-01-01', date_max='2100-01-01',␣
↪dist_max=0.2)
# Return close-approach data from the beginning of 2000 to the beginning of 2020␣
↪as a pandas DataFrame.
nasapy.close_approach(date_min='2000-01-01', date_max='2020-01-01', return_
↪df=True)
```

Each close-approach record is a list containing the following fields in the corresponding order:

- des - primary designation of the asteroid or comet (e.g., 443, 2000 SG344)
- orbit_id - orbit ID
- jd - time of close-approach (JD Ephemeris Time)
- cd - time of close-approeach (formatted calendar date/time)
- dist - nominal approach distance (au)
- dist_min - minimum (3-sigma) approach distance (au)
- dist_max - maximum (3-sigma) approach distance (au)
- v_rel - velocity relative to the approach body at close approach (km/s)
- v_inf - velocity relative to a massless body (km/s)
- t_sigma_f - 3-sigma uncertainty in the time of close-approach (formatted in days, hours, and minutes; days are not included if zero; example "13:02" is 13 hours 2 minutes; example "2_09:08" is 2 days 9 hours 8 minutes)
- body - name of the close-approach body (e.g., Earth)

  - only output if the body query parameters is set to ALL
- h - absolute magnitude H (mag)
- fullname - formatted full-name/designation of the asteroid or comet
  - optional - only output if requested with the appropriate query flag
  - formatted with leading spaces for column alignment in monospaced font tables

fireballs(*[date_min=None][, date_max=None][, energy_min=None][, energy_max=None][, impact_e_min=None][, impact_e_max=None][, vel_min=None][, vel_max=None][, alt_min=None][, alt_max=None][, req_loc=False][, req_alt=False][, req_vel=False][, req_vel_comp=False][, vel_comp=False][, sort='date'][, limit=None]*)

Returns available data on fireballs (objects that burn up in the upper atmosphere of Earth).

Parameters

- **date_min** – Excludes data earlier than the given date. Can be a string representing a date in 'YYYY-MM-DD' format or 'YYYY-MM-DDThh:mm:ss' format or a datetime object.
- **date_max** – Excludes data later than the given date. Can be a string representing a date in 'YYYY-MM-DD' format or 'YYYY-MM-DDThh:mm:ss' format or a datetime object.
- **energy_min** – Excludes data with total-radiated-energy less than the positive value of the specified value in joules $\times 10^{10}$.
- **energy_max** – Excludes data with total-radiated-energy greater than the positive value of the specified value in joules $\times 10^{10}$.
- **impact_e_min** – Excludes data with estimated impact energy less than the positive value of the specified value in kilotons (kt)
- **impact_e_max** – Excludes data with estimated impact energy greater than the positive value of the specified value in kilotons (kt)
- **vel_min** – Excludes data with velocity-at-peak-brightness less than the positive value of the specified value in km/s
- **vel_max** – Excludes data with velocity-at-peak-brightness greater than the positive value of the specified value in km/s
- **alt_min** – Excludes data from objects with an altitude less than the specified value
- **alt_max** – Excludes data from objects with an altitude greater than the specified value
- **req_loc** – If True, latitude and longitude required for object to be included in results.
- **req_alt** – If True, objects without an altitude are excluded.
- **req_vel** – If True, objects without a velocity are not included in results.
- **req_vel_comp** – If True, excludes objects without velocity components.
- **vel_comp** – If True, include velocity components
- **sort** – Sorts data on specified field. Default sort order is ascending, for descending, prepend a '-'. For example, for date descending, the sort value would be '-date'.
- **limit** – Limits data to the first number of results specified. Must be greater than 0 if passed.

Return type dict.

```
# Get all available data in reverse chronological order
nasapy.fireballs()
# Return the earlieset record
nasapy.fireballs(limit=1)
# Get data from the beginning of 2019
nasapy.fireballs(date_min='2019-01-01')
# Return fireball data from the beginning of the millennium to the beginning of␣
↪2020 as a pandas DataFrame.
nasapy.fireballs(date_min='2000-01-01', date_max='2020-01-01', return_df=True)
```

mission_design(*[des=None][, spk=None][, sstr=None][, orbit_class=False][, mjd0=None][, span=None][, tof_min=None][, tof_max=None][, step=None]*)

Provides access to the Jet Propulsion Laboratory/Solar System Dynamics small body mission design suite API.

Parameters

- **des** – The designation (provisional or IAU-number) of the desired object to search.
- **spk** – The SPK-ID of the desired object to search.
- **sstr** – Object search string.
- **orbit_class** – If True, returns the orbit class in human readable format instead of the default three-letter code.
- **mjd0** – First launch date in Modified Julian Date. Must be between [33282, 73459].
- **span** – Duration of the launch-date period to be explored in days. Must be between [10, 9200].
- **tof_min** – Minimum time of flight in days. Must be between [10, 9200].
- **tof_max** – Maximum time of flight in days. Must be between [10, 9200].
- **step** – Time step used to advance the launch date and the time of flight. Size of transfer map is limited to 1,500,000 points.

Return type dict. Dictionary object representing the returned JSON data from the API.

```
# Search for mission design data for SPK-ID 2000433
r = nasapy.mission_design(spk=2000433)
# Print the object data from the returned dictionary object.
r['object']
# Get Missions to 1 Ceres
r = nasapy.mission_design(des=1, mjd0=59000, span=1800, tof_min=120, tof_max=1500,
↪ step=5)
r['object']
```

nhats(*[spk=None][, des=None][, delta_v=12][, duration=450][, stay=8][, launch='2020-2045'][, magnitude=None][, orbit_condition_code=None][, plot=False]*)

Returns data available from the Near-Earth Object Human Space Flight Accessible Targets Study (NHATS) in the Small Bodies Database

Parameters

- **spk** – Returns data for a specific object by its SPK-ID.
- **des** – Returns data for a specific object by its designation.
- **delta_v** – Minimum total delta-v in km/s. Must be one of {12, 4, 5, 6, 7, 8, 9, 10, 11}
- **duration** – Minimum total distribution in number of days. Must be one of {450, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 420}

- **stay** – Minimum stay in days. Must be one of {8, 16, 24, 32}
- **launch** – The proposed launch window as a year range. Must be one of {'2020-2045', '2020-2025', '2025-2030', '2030-2035', '2035-2040', '2040-2045'}
- **magnitude** – The object's maximum absolute magnitude, also denoted as H. Must be one of {16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30}
- **orbit_condition_code** – The object's maximum orbit condition code (OCC). Must be one of {0, 1, 2, 3, 4, 5, 6, 7, 8}
- **plot** – If True, include base-64 encoded plot image file content. Will include a new output field 'plot_base64' in the returned results if True.

Return type dict. Dictionary object representing the returned JSON data from the API.

```
# Get all available summary data for NHATS objects.
n = nhats()
# Get the results from a 'standard' search on the NHATS webpage.
nhats(delta_v=6, duration=360, stay=8, magnitude=26, launch='2020-2045', orbit_
↪condition_code=7)
# Return data for a specific object by its designation
nhats(des=99942)
```

scout (*[tdes=None][, plot=None][, data_files=None][, orbits=None][, n_orbits=None][, eph_start=None][, eph_stop=None][, eph_step=None][, obs_code=None][, fov_diam=None][, fov_ra=None][, fov_dec=None][, fov_vmag=None]*)

Provides access and data available from NASA's Center for Near-Earth Object Studies (CNEOS) Scout system.

Parameters

- **tdes** – Filter results by an object's temporary designation.
- **plot** – Includes the plot files for the specified object of the select type. Options include 'el' (elements), 'ca' (close approach) and *sr* (systematic-ranging) or any combination delimited by ':'. For example, 'ca:el:sr' would include plot files of each available type.
- **data_files** – Returns available data files or the requested data file for the specified object. Currently only 'mpc' is available.
- **orbits** – If True, returns the sampled orbits data for a specified object.
- **n_orbits** – Limits the number of sampled orbits to this value. Must be in range [1, 1000].
- **eph_start** – Get the ephemeris for the specified object at the specified time in UTC.
- **eph_stop** – Sets the ephemeris stop-time. Also requires `eph_start` if specified.
- **eph_step** – Sets the ephemeris step size. Requires both `eph_start` and `eph_stop` to be specified.
- **obs_code** – Gets the ephemeris for the specified object relative to the specified MPC observatory code.
- **fov_diam** – Specifies the size (diameter) of the field-of-view in arc-minutes.
- **fov_ra** – Specifies the field-of-view center (R.A component). Requires parameters `fov_diam` and `fov_dec` to be set as well. Invalid if `eph_stop` is passed.
- **fov_dec** – Specifies the field-of-view center (Dec. component). Requires `fov_diam` and `fov_ra` to be passed as well. Invalid if `eph_stop` is set.
- **fov_vmag** – Filters ephemeris results to those with V-magnitude of this value or brighter. Requires `fov_diam` to also be specified.

Return type dict. Dictionary object representing the returned JSON data from the API.

```
# Get all available summary data.
scout()
# Return all summary data as a pandas DataFrame.
scout(return_df=True)
# Return data and plot files for a specific object by its temporary designation.␣
↪Note the object may no longer
# exist in the current database
scout(tdes='P20UvyK')
# Get ephemeris data for a specific object at the current time with a Field of␣
↪View diameter of 5 arc-minutes
# with a limiting V-magnitude of 23.1.
scout(tdes='P20UvyK', fov_diam=5, fov_vmag=23.1)
```

sentry(*[spk=None][, des=None][, h_max=None][, ps_min=None][, ip_min=None][, last_obs_days=None][, complete_data=False][, removed=False]*)

Provides data available from the Center for Near Earth Object Studies (CNEOS) Sentry system.

Parameters

- **spk** – Returns data available for the object matching the specified SPK-ID.
- **des** – Selects data for the matching designation.
- **h_max** – Limits data to those with an absolute magnitude, less than or equal to the specified value. Must be in the range [-10:100].
- **ps_min** – Limits results to those with a Palermo scale (PS) greater than or equal to the specified value. Must be in the range [-20:20].
- **ip_min** – Filters data to that which has an impact probability (IP) greater than or equal to the specified value.
- **last_obs_days** – Number of days since last observation. If negative, filters data to those which have not been observed within the specified number of days. If passed, must have an absolute value greater than 6.
- **complete_data** – If True, requests the full dataset to be returned.
- **removed** – If True, requests the list of removed objects to be returned.

Return type dict. Dictionary object representing the returned JSON data.

```
# Get summary data for available sentry objects.
sentry()
# Get summary data as a pandas DataFrame
sentry(return_df=True)
# Get data for a specific Sentry object by its designation.
sentry(des=99942)
# Get data for objects removed from the Sentry system.
sentry(removed=1)
```

Other Methods

julian_date(*[dt=None][, year=None][, month=1][, day=1][, hour=0][, minute=0][, second=0][, modified=True]*)

Calculates the Julian date or modified Julian date (if specified).

Parameters

- **dt** – Datetime object to convert into a Julian date. Note if a datetime object is supplied to `dt` the other parameters will not be evaluated. If None, returns the current datetime converted into a Julian date.
- **year** – Four digit year, such as 2019 or '2019'.
- **month** – Month number. For example 1 = January, 12 = December.
- **day** – Day of the month.
- **hour** – Hour of the day.
- **minute** – Minute of the day.
- **second** – Second of the day.
- **modified** – If True, returns the modified Julian date, which is the computed Julian Date - 24000000.5.

Return type float. The computed Julian or Modified Julian date.

```
# Return the modified Julian Date for the current time.
julian_date()
# Return the non-modified Julian Date for the current time.
julian_date(modified=False)
# Get the modified Julian Date for 2019-01-01 at midnight.
julian_date(year=2019)
```

The equation for calculating the Julian date is defined as:

$$J = 367(Year) - /large[(\frac{7(Year + \frac{Month+9}{12})}{4}).] + \frac{275(Month)}{9}. + Day + 1721013.5 + \frac{(\frac{\frac{Second}{60} + Minute}{60}) + Hour}{24}$$

1.2 Version History

1.2.1 Version 0.2.7

- Calling the `techport()` method without a project ID now returns data as expected. Thank you to user Burzlurker for pointing this out and providing a fix!
- Implemented a fix for when the `X-RateLimit-Remaining` header object was not available in the returned API data and thus caused an error.

1.2.2 Version 0.2.6

- `sentry` function now returns a summary object when `return_df=True` and a `des` or `spk` parameter are not specified.

1.2.3 Version 0.2.5

- `sentry` function now returns results as expected when not returning a pandas DataFrame.

1.2.4 Version 0.2.4

- Adds `exoplanet` function for providing access to NASA's Exoplanet Archive.

1.2.5 Version 0.2.3

- Fixes bug in `nhats` function when `return_df` parameter is set to `True`.

1.2.6 Version 0.2.2

- An optional `return_df` parameter has been implemented in the listed functions below. When set as `True`, the resulting JSON data will be coerced into a pandas DataFrame to allow easier and more straightforward data analysis for those interested. Please see the individual function documentation for more information and examples.
  - `fireballs`
  - `close_approach`
  - `nhats`
  - `sentry`
  - `scout`
- General bug fixes * The `sentry` function should now operate correctly when passing a `des` or `spk` parameter.

1.2.7 Version 0.2.1

- Added *sentry* function that wraps the CNEOS Sentry System API for providing Near-Earth Object impact risk assessment data.

1.2.8 Version 0.2.0

Initial release.

CHAPTER 2

Installation

`nasapy` is most easily installed using `pip`.

```
pip install nasapy
```

The library can also be cloned or downloaded into a location of your choosing and then installed using the *setup.py* file per the following:

```
git clone git@github.com:aschleg/nasapy.git
cd nasapy
python setup.py install
```

CHAPTER 3

Requirements

- Python 3.4+
- `requests>=2.18`
- `pandas>=0.22.0`
- Although not strictly required to use `nasapy`, the pandas library is needed for returning results as a DataFrame.

CHAPTER 4

Tutorials and In-Depth Examples

The following are articles that explore a facet of the *nasapy* library in more depth.

- Plot Earth Fireball Impacts with nasapy, pandas and folium
- Analyzing the Next Decade of Earth Close-Approaching Objects with nasapy
- Get All NASA Astronomy Pictures of the Day from 2019

CHAPTER 5

Examples and Usage

Although not strictly required to begin interacting with the NASA API, it is recommended to sign up to receive an API access key that has a significantly higher usage limit available compared to the demo key option. Many methods do not require an API key, but for those that do, it is typically a good option to use a provided API key rather than the demo key. Using a received API key allows for 1,000 requests per hour, while the demo key has 30 requests limit per hour and 50 requests per day.

5.1 Authentication

Assuming an API key was received after signing up, authentication to the NASA API happens when initializing the `Nasa` class.

```
nasa = Nasa(key=key)
```

If using a demo key, the initialization does not require any parameters to be passed.

```
nasa = Nasa()
```

5.2 Remaining Requests Available

The `limit_remaining` attribute of the initialized `Nasa` class allows one to see the number of available requests remaining.

```
nasa.limit_remaining
```

5.3 Examples

The following are some quick examples to get started.

5.3.1 Astronomy Picture of the Day

```
# Return today's picture of the day
nasa.picture_of_the_day()
# Return a previous date's picture of the day with the high-definition URL included.
nasa.picture_of_the_day('2019-01-01', hd=True)
```

5.3.2 Mars Weather

```
# Return the most recent data for the previous seven Sols (Martian Days)
nasa.mars_weather()
```

5.3.3 Asteroid Feed

```
# Get asteroids approaching Earth at the beginning of 2019.
nasa.asteroid_feed(start_date='2019-01-01')
```

5.3.4 Get Asteroid Data

```
# Get entire asteroid data set.
nasa.get_asteroids()
# Get asteroid with ID 3542519
nasa.get_asteroids(asteroid_id=3542519)
```

5.3.5 DONKI (Space Weather Database of Notifications, Knowledge and Information)

```
# Coronal Mass Ejection Event Data

# View data from coronal mass ejection events from the last thirty days
nasa.coronal_mass_ejection()
# View all CME events from the beginning of 2019.
nasa.coronal_mass_ejection(start_date='2019-01-01', end_date=datetime.datetime.
↪today())

# Geomagnetic Storm Event Data

# Get geomagnetic storm events from the last thirty days.
nasa.geomagnetic_storm()

# Solar Flare Event Data

# Get solar flare events from May of 2019
nasa.solar_flare(start_date='2019-05-01', end_date='2019-05-31')

# Solar Energetic Particle Data

# Get data from April 2017
nasa.solar_energetic_particle(start_date='2017-04-01', end_date='2017-04-30')
```

(continues on next page)

(continued from previous page)

```
# Magnetopause Crossing Data

# Get data on magnetopause crossing events from 2018 to the current date.
nasa.magnetopause_crossing(start_date='2018-01-01')

# Radiation Belt Enhancement Data

# Get data on radiation belt enhancement events from the last 30 days.
nasa.radiation_belt_enhancement()

# Hight Speed Stream Data

# Get data on hight speed stream events from the beginning of September 2019.
nasa.hight_speed_stream()

# WSA Enlil-Simulation Data

# Get data from the first simulation performed in 2019.
wsa = n.wsa_enlil_simulation(start_date='2019-01-01')
wsa[0]
```

5.3.6 EPIC (DSCOVR's Earth Polychromatic Imaging Camera)

```
# Get EPIC data from the beginning of 2019.
e = nasa.epic(date='2019-01-01')
# Print the first result
e[0]
```

5.3.7 Exoplanets

```
# Get all exoplanets data as a pandas DataFrame.
exoplanets(return_df=True)
# Get all confirmed planets in the Kepler field.
exoplanets(where='pl_kepflag=1')
# Stars known to host exoplanets as a pandas DataFrame.
exoplanets(select='distinct pl_hostname', order='pl_hostname', return_df=True)
```

5.3.8 Landsat Images for a given Latitude-Longitude

```
# Get imagery at latitude 1.5, longitude 100.75 and include the computed cloud score␣
↪calculation.
nasa.earth_imagery(lon=100.75, lat=1.5, cloud_score=True)

# Get assets available beginning from 2014-02-01 at lat-lon 100.75, 1.5
nasa.earth_assets(lat=100.75, lon=1.5, begin_date='2014-02-01')
```

5.3.9 Available Image data collected by the Mars rovers Curiosity, Discovery and Spirit.

```
# Return image data collected on Curiosity's 1000th sol.
nasa.mars_rover(sol=1000)
```

5.3.10 Access GeneLab and Other Bioinformatics Databases

```
# Find Gene studies in the cgene database related to 'mouse liver'
n.genelab_search(term='mouse liver')
```

5.3.11 CelesTrak Two-Line Element Set Records

```
# Retrieve available data for a specific satellite ID.
tle(satellite_number=43553)
```

5.3.12 Search for Available Imagery and Audio from the images.nasa.gov API

```
# Search for media related to 'apollo 11' with 'moon landing' in the description of␣
↪the items.
r = media_search(query='apollo 11', description='moon landing')
# Print the first returned media item from the resulting collection.
r['items'][0]
```

5.3.13 Getting the Julian and Modified Julian Date

```
# Return the modified Julian Date for the current time.
julian_date()
# Return the non-modified Julian Date for the current time.
julian_date(modified=False)
# Get the modified Julian Date for 2019-01-01 at midnight.
julian_date(year=2019)
```

Index