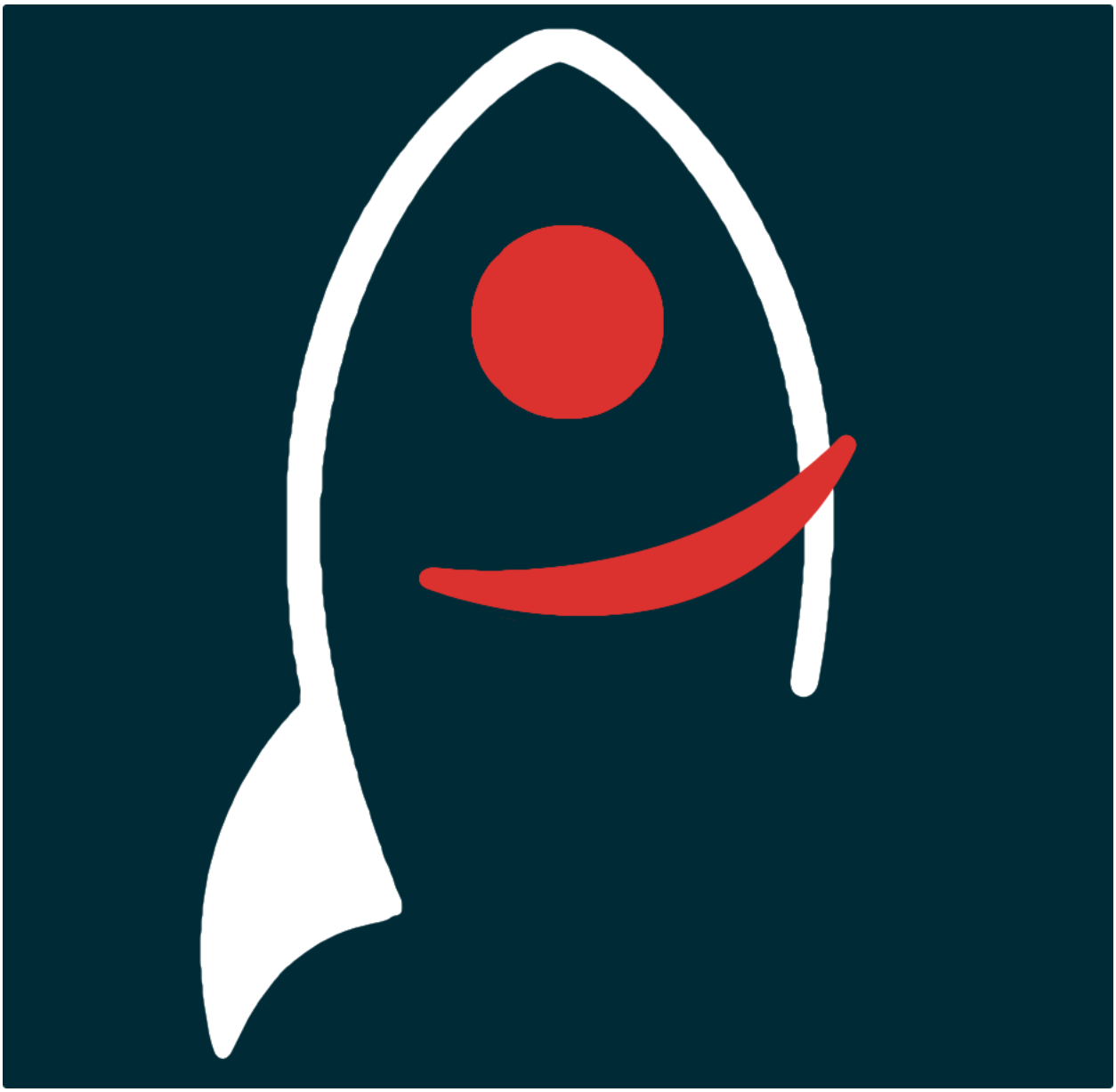

neddy Documentation

Release v0.3.0

Dave Young

2021

TABLE OF CONTENTS

query the Nasa Extra-Galactic (NED) database via the command-line and programmatically.

Documentation for neddy is hosted by Read the Docs (development version and master version). The code lives on github. Please report any issues you find here.

CHAPTER

ONE

FEATURES

-

1.1 Installation

The easiest way to install neddy is to use `pip` (here we show the install inside of a conda environment):

```
conda create -n neddy python=3.7 pip
conda activate neddy
pip install neddy
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/neddy.git
cd neddy
python setup.py install
```

To upgrade to the latest version of neddy use the command:

```
pip install neddy --upgrade
```

To check installation was successful run `neddy -v`. This should return the version number of the install.

1.1.1 Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/neddy.git
cd neddy
python setup.py develop
```

Pull requests are welcomed!

1.2 Initialisation

Before using neddy you need to use the `init` command to generate a user settings file. Running the following creates a yaml settings file in your home folder under `~/.config/neddy/neddy.yaml`:

```
neddy init
```

The file is initially populated with neddy's default settings which can be adjusted to your preference.

If at any point the user settings file becomes corrupted or you just want to start afresh, simply trash the `neddy.yaml` file and rerun `neddy init`.

1.2.1 Modifying the Settings

Once created, open the settings file in any text editor and make any modifications needed.

1.2.2 Basic Python Setup

If you plan to use `neddy` in your own scripts you will first need to parse your settings file and set up logging etc. One quick way to do this is to use the `fundamentals` package to give you a logger, a settings dictionary and a database connection (if connection details given in settings file):

```
## SOME BASIC SETUP FOR LOGGING, SETTINGS ETC
from fundamentals import tools
from os.path import expanduser
home = expanduser("~")
settingsFile  = home + "/.config/neddy/neddy.yaml"
su = tools(
    arguments={"settingsFile": settingsFile},
    docString=__doc__,
)
arguments, settings, log, dbConn = su.setup()
```

1.3 Todo List

Todo:

- nice!

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/neddy/checkouts/develop/docs/source/_template_.m
line 1.)

1.4 Release Notes

v0.3.0 - May 19, 2020

- Now compatible with Python 3.*

CHAPTER

TWO

API REFERENCE

2.1 Modules

neddy.commonutils	*common tools used throughout package*
neddy.utKit	*Unit testing tools*

2.1.1 commonutils *(module)*

common tools used throughout package

Sub-modules

getpackagepath	*Get common file and folder paths for the host package*

2.1.2 utKit *(module)*

Unit testing tools

Classes

utKit(moduleDirectory[, dbConn])	*Override dryx utKit*

Sub-modules

utKit(moduleDirectory[, dbConn])	*Override dryx utKit*

class utKit(*moduleDirectory*, *dbConn=False*)

Bases: fundamentals.utKit.utKit

Override dryx utKit

get_project_root()

Get the root of the ``python`` package - useful for getting files in the root directory of a project

Return

- `rootPath` – the root path of a project

`refresh_database`()
Refresh the unit test database

`setupModule`()
The setupModule method

Return

- `log` – a logger
- `dbConn` – a database connection to a test database (details from yaml settings file)
- `pathToInputDir` – path to modules own test input directory
- `pathToOutputDir` – path to modules own test output directory

`tearDownModule`()
The tearDownModule method

2.2 Classes

`neddy.conesearch`	*The worker class for the conesearch module*
`neddy.namesearch`	*The worker class for the namesearch module*

2.2.1 conesearch *(class)*

`class conesearch`(*log*, *ra=False*, *dec=False*, *radiusArcsec=False*, *nearestOnly=False*, *unclassified=False*, *quiet=False*, *listOfCoordinates=False*, *outputFilePath=False*, *verbose=False*, *redshift=False*)
Bases: `neddy._basesearch._basesearch`

The worker class for the conesearch module

Key Arguments

- `log` – logger
- `ra` – ra
- `dec` – dec
- `radiusArcsec` – radiusArcsec
- `nearestOnly` – return only the nearest object from NED
- `unclassified` – include the unclassified sources in the return results
- `quiet` – don't print to stdout
- `listOfCoordinates` – list of coordinates ra dec radiusArcsec
- `outputFilePath` – path to output file
- `verbose` – return more metadata for matches
  - `redshift` – redshift constraint

```
- @review: when complete, clean conesearch class
- @review: when complete add logging
- @review: when complete, decide whether to abstract class to another module
```

Methods

`get`()	*get the conesearch object*
`get_crossmatch_names`([listOfCoordinates, ...])	*get corssmatch names*

get()
get the conesearch object

Return

- `conesearch`

```
- @review: when complete, clean get method
- @review: when complete add logging
```

get_crossmatch_names(*listOfCoordinates=False*, *radiusArcsec=False*)
get corssmatch names

Key Arguments

- `listOfCoordinates` – list of the coordinates to conesearch
- `radiusArcsec` – the search radius

Return

- None

```
- @review: when complete, clean get_crossmatch_names method
- @review: when complete add logging
```

2.2.2 namesearch *(class)*

class namesearch(*log*, *names*, *quiet=False*, *verbose=False*, *searchParams=False*, *outputFilePath=False*)
Bases: `neddy._basesearch._basesearch`

The worker class for the namesearch module

Key Arguments

- `log` – logger
- `name` – name
- `quiet` – don't print to stdout
- `verbose` – return more metadata for matches
- `searchParams` – list of dictionaries to prepend to results
- `outputFilePath` – path to file to output results to

Methods

get()	*get the namesearch object*

get()
get the namesearch object

Return

- results

2.3 A-Z Index

Modules

neddy.commonutils	*common tools used throughout package*
neddy.utKit	*Unit testing tools*

Classes

neddy.conesearch	*The worker class for the conesearch module*
neddy.namesearch	*The worker class for the namesearch module*

Functions

CHAPTER

THREE

RELEASE NOTES

v0.3.0 - May 19, 2020

- Now compatible with Python 3.*

PYTHON MODULE INDEX

INDEX