*yweather* is a Python module that provides an interface to the *Yahoo! Weather RSS feed*. 
1.1 Introduction

This is the documentation for yweather. yweather is a Python module that provides an interface to the Yahoo! Weather RSS feed.

1.1.1 Prerequisites

yweather requires Python 2.6, 2.7, or 3 to run.

1.1.2 Installation

There are multiple ways to install yweather. If you are unsure about which method to use, try pip.

pip (recommended)

pip is a tool for installing and managing Python packages. To install yweather, run:

```
$ pip install yweather
```

This will download yweather from the Python Package Index and install it in your Python’s site-packages directory.

Tarball Release

1. Download the most recent release from yweather’s PyPi page.
2. Unpack the tarball.
3. From inside the yweather-0.X directory, run python setup.py install

This will install yweather in your Python’s site-packages directory.

Install the Development Version

yweather’s code is hosted at GitHub. To install the development version, do the following:

1. Make sure Git is installed. Test if it’s installed by running git --version
2. `git clone git://github.com/tsroten/yweather.git`

3. `pip install -e yweather`

This will link the `yweather` directory into your `site-packages` directory. You can find out where your `site-packages` directory is by running:

```
python -c "from distutils.sysconfig import get_python_lib; print(get_python_lib())"
```

### 1.1.3 Basic Usage

```python
>>> import yweather
>>> client = yweather.Client()
>>> client.fetch_woeid("Oslo, Norway")
'862592'
>>> oslo_weather = client.fetch_weather("862592")
>>> oslo_weather["atmosphere"]['pressure']
'30.24'
>>> oslo_weather["condition"]['text']
'Mostly Cloudy'
```

This code creates a `yweather.Client` instance that allows you to fetch a location’s `WOEID` and weather. The weather data is returned as a `dict`.

### 1.2 Usage

Let’s learn how to use `yweather`.

#### 1.2.1 Create a Client Object

`yweather` consists of a single class, `Client`.

```python
>>> import yweather
>>> client = yweather.Client()
```

By creating an instance of `Client`, you’ve created an object that you can use to fetch location identifiers and weather data from Yahoo! Weather.

#### 1.2.2 Fetch a Location’s WOEID

Yahoo! Weather gives every location a unique `WOEID`. In order to fetch weather data from Yahoo!, you must first know the location’s `WOEID`.

```python
>>> client.fetch_woeid("Beijing, China")
'2151330'
>>> client.fetch_woeid("96734")
'12798281'
>>> client.fetch_woeid("10 South Main Street, Harrisonburg, VA")
'12767058'
```

You can retrieve a `WOEID` by passing a general or specific address. The above example used city and country, ZIP code, and complete address.
1.2.3 Fetch a Location’s Weather

Once you have a location’s WOEID, you can use it to fetch the location’s weather. Weather data is returned as a dict. Its structure is detailed in fetch_weather()’s API documentation.

```python
>>> beijing_weather = client.fetch_weather("2151330")
>>> beijing_weather["guid"]
'CHXX0008_2013_01_06_7_00_CST'
>>> beijing_weather["description"]
'Yahoo! Weather for Beijing, CN'
>>> beijing_weather["condition"]["temp"]
'28'
```

The returned dict contains metadata along with the weather data itself. By default, United States customary units are used, but by changing the metric argument, you can receive data according to the metric system.

```python
>>> kailua_weather = client.fetch_weather("12798281", metric=True)
>>> kailua_weather["forecast"][0]["high"]
'25'
>>> kailua_weather["units"]["forecast"]["high"]
'°C'
```

The units used for each data value are accessible with the units key.

1.2.4 Using a Location’s LID

Because Yahoo! Weather’s data comes from The Weather Channel, weather data is also accessible via a The Weather Channel LID. This provides access to a 5-day forecast versus the 2-day forecast available with a location’s WOEID.

```python
>>> client.fetch_lid("2151330")
'CHXX0008'
>>> beijing_weather = client.fetch_weather("CHXX0008")
>>> len(beijing_weather["forecast"])  
5
```

The fetch_lid() method takes a WOEID and returns a LID. You can pass the LID to the fetch_weather() method.

1.3 API

yweather.WOEID_LOOKUP_URL
   The URL used to fetch a location’s corresponding WOEID.

yweather.WEATHER_URL
   The URL used to fetch a WOEID’s weather.

yweather.LID_LOOKUP_URL
   The URL used to fetch a location’s corresponding LID.

yweather.LID_WEATHER_URL
   The URL used to fetch a LID’s weather.

yweather.WEATHER_NS
   The XML namespace used in the weather RSS feed.

yweather.GEO_NS
   The XML namespace used for the lat/long coordinates in the RSS feed.
yweather CONDITION_IMAGE_URL
The URL of an image depicting the current conditions.

yweather UNITS
A dict that maps data names to units.

class yweather Client
Interface with the Yahoo! Weather RSS feed. Provides methods to search for location data and fetch weather data.

fetch_lid (woeid)
Fetch a location’s corresponding LID.

Parameters woeid (string) – the location’s WOEID.

Returns a string containing the requested LID or None if the LID could not be found.

Raises
• urllib.error.URLError – urllib.request could not open the URL (Python 3).
• urllib2.URLError – urllib2 could not open the URL (Python 2).

fetch_weather (id [metric=False])
Fetch a location’s weather.

id can be either a WOEID or LID. The weather data returned for each is identical except that the WOEID returns a 2-day forecast and the LID returns a 5-day forecast. The LID uses an undocumented API, so use it at your own risk.

The returned data is a dict with the requested weather data. It loosely follows the Yahoo! Weather RSS feed response structure, but has some noticeable differences. The following table outlines the data structure.

<table>
<thead>
<tr>
<th>Keys</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>The title of the feed, which includes the location city. For example “Yahoo! Weather - Sunnyvale, CA”.</td>
</tr>
<tr>
<td>link</td>
<td>The URL of the forecast for this location.</td>
</tr>
<tr>
<td>language</td>
<td>The language of the weather forecast, for example, en-us for US English.</td>
</tr>
<tr>
<td>description</td>
<td>The overall description of the feed including the location, for example “Yahoo! Weather for Sunnyvale, CA”.</td>
</tr>
<tr>
<td>lastBuildDate</td>
<td>The last time the feed was updated. For example, Fri, 04 Jan 2013 6:56 am PST.</td>
</tr>
<tr>
<td>ttl</td>
<td>Time to Live; how long in minutes this feed should be cached.</td>
</tr>
<tr>
<td>logo</td>
<td>The URL for the Yahoo! Weather logo associated with this feed.</td>
</tr>
<tr>
<td>guid</td>
<td>Unique identifier for the forecast, made up of the location ID, the date, and the time.</td>
</tr>
<tr>
<td>location city</td>
<td>city name</td>
</tr>
<tr>
<td>location region</td>
<td>state, territory, or region, if given.</td>
</tr>
<tr>
<td>location country</td>
<td>two-character country code</td>
</tr>
<tr>
<td>geo lat</td>
<td>The latitude of the location.</td>
</tr>
<tr>
<td>geo long</td>
<td>The longitude of the location.</td>
</tr>
<tr>
<td>units wind chill</td>
<td>°F or °C</td>
</tr>
<tr>
<td>units wind direction</td>
<td>°</td>
</tr>
<tr>
<td>units wind speed</td>
<td>mph or km/h</td>
</tr>
<tr>
<td>units atmosphere humidity</td>
<td>%</td>
</tr>
<tr>
<td>units atmosphere visibility</td>
<td>mi or km</td>
</tr>
<tr>
<td>units atmosphere pressure</td>
<td>psi or hPa</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.1 – continued from previous page

<table>
<thead>
<tr>
<th>Keys</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>units</td>
<td>condition temp</td>
</tr>
<tr>
<td>units</td>
<td>forecast low</td>
</tr>
<tr>
<td>units</td>
<td>forecast high</td>
</tr>
<tr>
<td>wind</td>
<td>chill</td>
</tr>
<tr>
<td>wind</td>
<td>direction</td>
</tr>
<tr>
<td>wind</td>
<td>compass</td>
</tr>
<tr>
<td>wind</td>
<td>speed</td>
</tr>
<tr>
<td>atmosphere</td>
<td>humidity</td>
</tr>
<tr>
<td>atmosphere</td>
<td>visibility</td>
</tr>
<tr>
<td>atmosphere</td>
<td>pressure</td>
</tr>
<tr>
<td>atmosphere</td>
<td>rising</td>
</tr>
<tr>
<td>atmosphere</td>
<td>state</td>
</tr>
<tr>
<td>astronomy</td>
<td>sunrise</td>
</tr>
<tr>
<td>astronomy</td>
<td>sunset</td>
</tr>
<tr>
<td>condition</td>
<td>text</td>
</tr>
<tr>
<td>condition</td>
<td>code</td>
</tr>
<tr>
<td>condition</td>
<td>image</td>
</tr>
<tr>
<td>condition</td>
<td>temp</td>
</tr>
<tr>
<td>condition</td>
<td>date</td>
</tr>
<tr>
<td>forecast</td>
<td>contains a <code>list</code> where each item is a <code>dict</code> that contains the weather forecast for a specific day.</td>
</tr>
<tr>
<td>– day</td>
<td>day</td>
</tr>
<tr>
<td>– date</td>
<td>the date to which this forecast applies. The date is in “dd Mmm yyyy” format, for example “3 Nov 2005”.</td>
</tr>
<tr>
<td>– low</td>
<td>low</td>
</tr>
<tr>
<td>– high</td>
<td>high</td>
</tr>
<tr>
<td>– text</td>
<td>text</td>
</tr>
<tr>
<td>– code</td>
<td>code</td>
</tr>
</tbody>
</table>

The differences between this data structure and Yahoo! Weather’s are:

- `units` breaks down the data units further and uses more helpful key names.
- `logo` represents the RSS feed’s `<image>` tag.
- `guid` was moved to the top level.
- `condition` has the `image` key, which provides easy access to a URL of an image depicting the current sky conditions.
- `atmosphere` has the `state` key, which provides a textual description of the barometric pressure state.
- `geo` is now a `dict` with `lat` and `long` keys.
- `wind` includes the `compass` key, which provides wind direction according to a compass (e.g. NNW, SE, or W).

Example usage of the returned `dict`:

```python
>>> print result["wind"]["compass"]
NNW
>>> print result["atmosphere"]["pressure"], result["units"]["atmosphere"]["pressure"]
29.95 psi
>>> print len(result["forecast"])
2
>>> print result["forecast"][0]["text"]
Partly Cloudy
```
Parameters

- **id** *(string)* – the location’s **WOEID** or **LID**.
- **metric** *(bool)* – return metric data; defaults to **False**.

Returns a **dict** containing the location’s weather data or **None** if the weather data couldn’t be fetched.

Raises

- **urllib.error.URLError** – **urllib.request** could not open the URL (Python 3).
- **urllib2.URLError** – **urllib2** could not open the URL (Python 2).

`fetch_woeid(location)`

Fetch a location’s corresponding **WOEID**.

Parameters **location** *(string)* – a location (e.g. 23454 or Berlin, Germany).

Returns a **string** containing the requested **WOEID** or **None** if the **WOEID** could not be found.

Raises

- **urllib.error.URLError** – **urllib.request** could not open the URL (Python 3).
- **urllib2.URLError** – **urllib2** could not open the URL (Python 2).

1.4 Change Log

1.4.1 v0.1.1 (2016-03-31)

- Updates the Yahoo! Weather URL (thanks tsipinio).
- Add Python 3.5 classifier.
- Fix PEP8 style issues.

1.4.2 v0.1 (2013-01-05)

- Initial release.

1.5 Glossary

**LID** Location ID. The Weather Channel uses these to identify locations around the world. Because Yahoo! Weather gets their weather data from The Weather Channel, the Location ID can be used on Yahoo’s RSS Feed to gain access to a 5-day weather forecast. Note: this usage is undocumented.
metric system  The metric system is a system of measurement that is used around the world (in all but three countries). It is often considered to be synonymous with the International System of Units. Some example units include grams and meters. See Wikipedia’s article for more information.

United States customary units  A system of measurements commonly used across the United States. Some example units include miles, pounds, and inches. See Wikipedia’s article for more information.

WOEID  Where On Earth IDentifier. Yahoo! GeoPlanet uses these 32-bit identifiers to reference spatial entities around the world. See Yahoo! GeoPlanet’s Key Concepts for more information.
Indices and tables

- genindex
- modindex
- search
y
yweather.5