Astral v3.2

Contents

Astral is a python package for calculating the times of various aspects of the sun and moon.

It can calculate the following

Dawn

The time in the morning when the sun is a specific number of degrees below the horizon.

Sunrise

The time in the morning when the top of the sun breaks the horizon (assuming a location with no obscuring features.)

Noon

The time when the sun is at its highest point directly above the observer.

Midnight

The time when the sun is at its lowest point.

Sunset

The time in the evening when the sun is about to disappear below the horizon (asuming a location with no obscuring features.)

Dusk

The time in the evening when the sun is a specific number of degrees below the horizon.

Daylight

The time when the sun is up i.e. between sunrise and sunset

Night

The time between astronomical dusk of one day and astronomical dawn of the next

Twilight

The time between dawn and sunrise or between sunset and dusk

The Golden Hour

The time when the sun is between 4 degrees below the horizon and 6 degrees above.

The Blue Hour

The time when the sun is between 6 and 4 degrees below the horizon.

Time At Elevation

the time when the sun is at a specific elevation for either a rising or a setting sun.

Solar Azimuth

The number of degrees clockwise from North at which the sun can be seen

Solar Zenith

The angle of the sun down from directly above the observer

Solar Elevation

The number of degrees up from the horizon at which the sun can be seen

Rahukaalam

"Rahukaalam or the period of Rahu is a certain amount of time every day that is considered inauspicious for any new venture according to Indian Vedic astrology".

Moonrise and Moonset

Like the Sun but for the moon

Moon Azimuth and Zenith

Also like the Sun but for the moon

Moon Phase

The phase of the moon for a specified date.

Astral also comes with a geocoder containing a local database that allows you to look up information for a small set of locations (new locations can be added).

The following examples demonstrates some of the functionality available in the module

Sun

```
from astral import LocationInfo
city = LocationInfo("London", "England", "Europe/London", 51.5, -0.116)
print((
    f"Information for {city.name}/{city.region}\n"
    f"Timezone: {city.timezone}\n"
    f"Latitude: {city.latitude:.02f}; Longitude: {city.longitude:.02f}\n"
))
```

```
Information for London/England
Timezone: Europe/London
Latitude: 51.50; Longitude: -0.12
```

```
import datetime
from astral.sun import sun
s = sun(city.observer, date=datetime.date(2009, 4, 22))
print((
    f'Dawn:    {s["dawn"]}\n'
    f'Sunrise: {s["sunrise"]}\n'
    f'Noon:    {s["noon"]}\n'
    f'Sunset:  {s["sunset"]}\n'
    f'Dusk:    {s["dusk"]}\n'
))
```

```
Dawn:    2009-04-22 04:13:04.997608+00:00
Sunrise: 2009-04-22 04:50:17.127004+00:00
Noon:    2009-04-22 11:59:02+00:00
Sunset:  2009-04-22 19:08:41.711407+00:00
Dusk:    2009-04-22 19:46:06.423846+00:00
```

Note

The example above calculates the times of the sun in the UTC timezone. If you want to return times in a different timezone you can pass the *tzinfo* parameter to the function.

```
>>> city = LocationInfo("London", "England", "Europe/London",
51.5, -0.116)
>>> london = Location(city)
>>> s = sun(city.observer, date=datetime.date(2009, 4, 22),
tzinfo=london.timezone)
```

or

```
>>> timezone = zoneinfo.ZoneInfo("Europe/London")
>>> s = sun(city.observer, date=datetime.date(2009, 4, 22),
tzinfo=timezone)
```

Moon

The moon rise/set times can be obtained like the sun's functions

```
from astral import LocationInfo
city = LocationInfo("London", "England", "Europe/London", 51.5, -0.116)

import datetime
from astral.moon import moonrise
from astral.location import Location
dt = datetime.date(2021, 10, 28)
london = Location(city)
rise = moonrise(city.observer, dt)  # returns a UTC time
print(rise)
```

```
2021-10-28 22:02:00+00:00
```

And for a local time

```
rise = moonrise(city.observer, dt, city.tzinfo)
```

Phase

```
import datetime
from astral import moon
phase = moon.phase(datetime.date(2018, 1, 1))
print(phase)
```

```
13.255666666666668
```

The moon phase method returns an number describing the phase, where the value is between 0 and 27.99. The following lists the mapping of various values to the description of the phase of the moon.

0 .. 6.99	New moon
7 .. 13.99	First quarter
14 .. 20.99	Full moon
21 .. 27.99	Last quarter

If for example the number returned was 27.99 then the moon would be almost at the New Moon phase, and if it was 24.00 it would be half way between the Last Quarter and a New Moon.

> **Note**
>
> The moon phase does not depend on your location. However what the moon actually looks like to you does depend on your location. If you're in the southern hemisphere it looks different than if you were in the northern hemisphere.
>
> See http://moongazer.x10.mx/website/astronomy/moon-phases/ for an example.
>
> For an example of using this library to generate moon phases including the names in various languages and the correct Unicode glyphs see the project by PanderMusubi on Github.

Geocoder

```
>>> from astral.geocoder import database, lookup
>>> lookup("London", database())
LocationInfo(name='London', region='England', timezone='Europe/London',
    latitude=51.473333333333336, longitude=-0.0008333333333333334)
```

> **Note**
>
> Location elevations have been removed from the database. These were added due to a misunderstanding of the affect of elevation on the times of the sun. These are not required for the calculations, only the elevation of the observer above/below the location is needed.
>
> See Effect of Elevation below.

Custom Location

If you only need a single location that is not in the database then you can construct a **`LocationInfo`** and fill in the values, either on initialization

```
from astral import LocationInfo
l = LocationInfo('name', 'region', 'timezone/name', 0.1, 1.2)
```

or set the attributes after initialization:

```
from astral import LocationInfo
l = LocationInfo()
l.name = 'name'
l.region = 'region'
l.timezone = 'US/Central'
l.latitude = 0.1
l.longitude = 1.2
```

> **Note**
>
> *name* and *region* can be anything you like.

Additional Locations

You can add to the list of available locations using the `add_locations()` function and passing either a string with one line per location or by passing a list containing strings, lists or tuples (lists and tuples are passed directly to the LocationInfo constructor).

```
>>> from astral.geocoder import add_locations, database, lookup
>>> db = database()
>>> try:
...     lookup("Somewhere", db)
... except KeyError:
...     print("Somewhere not found")
...
Somewhere not found
>>> add_locations("Somewhere,Secret Location,UTC,24°28'N,39°36'E", db)
>>> lookup("Somewhere", db)
LocationInfo(name='Somewhere', region='Secret Location', timezone='UTC',
    latitude=24.466666666666665, longitude=39.6)
```

Timezone Groups

Timezone groups such as Europe can be accessed via the `group()` function in the `geocoder` module

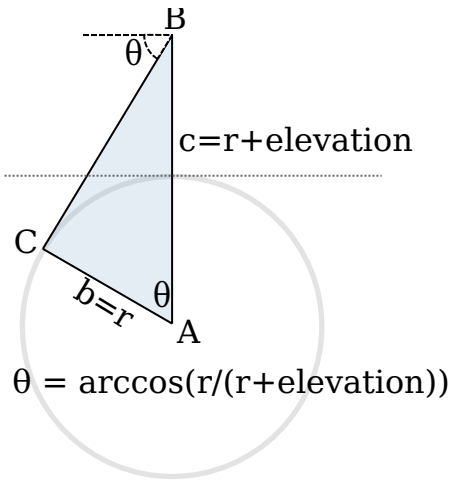
```
from astral.geocoder import group, database
db = database()
europe = group("europe", db)
print(sorted(europe.keys())[:4])
```

```
['aberdeen', 'amsterdam', 'andorra_la_vella', 'ankara']
```

Times Of The Sun

The times of the sun that you experience depend on what obscurs your view of it. It may either be obscured by the horizon or some other geographical feature (e.g. mountains)

1. If what obscures you at ground level is the horizon and you are at a elevation above ground level then the times of the sun depends on how far further round the earth you can see due to your elevation (the sun rises earlier and sets later).
   The extra angle you can see round the earth is determined by calculating the angle α in the image below based on your elevation above ground level, and adding this to the depression angle for the sun calculations.



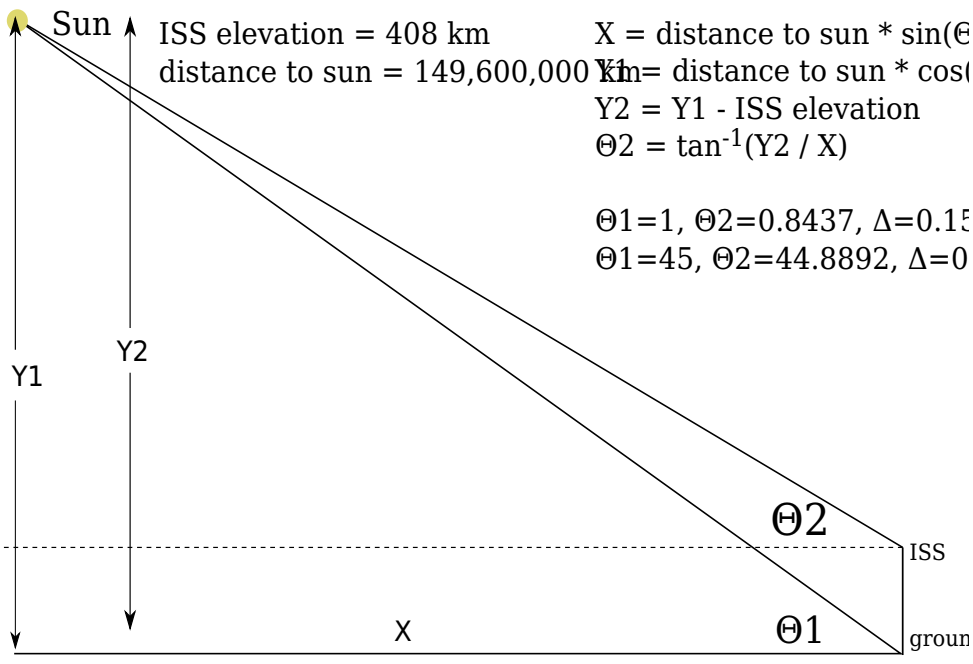
2. If your view is obscured by some other geographical feature than the horizon, then the adjustment angle is based on how far you are above or below the feature and your distance to it.

For the first case i.e. obscured by the horizon you need to pass a single float to the Observer as its elevation. For the second case pass a tuple of 2 floats. The first being the vertical distance to the top of the feature and the second the horizontal distance to the feature.

Elevation Of The Sun

Even though an observer's elevation can significantly affect the times of the sun the same is not true for the elevation angle from the observer to the sun.

As an example the diagram below shows the difference in angle between an observer at ground level and one on the ISS orbiting 408 km above the earth.



The largest difference between the two angles is when the angle at ground level is 1 degree. The difference then is approximately 0.15 degrees.

At the summit of mount Everest (8,848 m) the maximum difference is 0.00338821 degrees.

Due to the very small difference the astral package does not currently adjust the solar elevation for changes in observer elevation.

When viewing the sun the position you see it at is different from its actual position due to the effect of atmospheric refraction which makes the sun appear to be higher in the sky. The calculations in the package take this refraction into account.

The **`sunrise()`** and **`sunset()`** functions use the refraction at an angle when the sun is half of its apparent diameter below the horizon. This is between about 30 and 32 arcminutes and for the astral package a value of 32" is used.

> **Note**
>
> The refraction calculation does not take into account temperature and pressure which can affect the angle of refraction.

This module is licensed under the terms of the Apache V2.0 license.

To install Astral you should use the pip tool:

```
pip3 install astral
```

> **Note**
>
> Now that we are Python 3 only and pip provides a versioned executable on Windows you should use the *pip3* command on all operating systems to ensure you are targetting the right Python version.

The module includes location and time zone data for the following cities. The list includes all capital cities plus some from the UK. The list also includes the US state capitals and some other US cities.

Aberdeen, Abu Dhabi, Abu Dhabi, Abuja, Accra, Addis Ababa, Adelaide, Al Jubail, Albany, Albuquerque, Algiers, Amman, Amsterdam, Anchorage, Andorra la Vella, Ankara, Annapolis, Antananarivo, Apia, Ashgabat, Asmara, Astana, Asuncion, Athens, Atlanta, Augusta, Austin, Avarua, Baghdad, Baku, Baltimore, Bamako, Bandar Seri Begawan, Bangkok, Bangui, Banjul, Barrow-In-Furness, Basse-Terre, Basseterre, Baton Rouge, Beijing, Beirut, Belfast, Belgrade, Belmopan, Berlin, Bern, Billings, Birmingham, Birmingham, Bishkek, Bismarck, Bissau, Bloemfontein, Bogota, Boise, Bolton, Boston, Bradford, Brasilia, Bratislava, Brazzaville, Bridgeport, Bridgetown, Brisbane, Bristol, Brussels, Bucharest, Bucuresti, Budapest, Buenos Aires, Buffalo, Bujumbura, Burlington, Cairo, Canberra, Cape Town, Caracas, Cardiff, Carson City, Castries, Cayenne, Charleston, Charlotte, Charlotte Amalie, Cheyenne, Chicago, Chisinau, Cleveland, Columbia, Columbus, Conakry, Concord, Copenhagen, Cotonou, Crawley, Dakar, Dallas, Damascus, Dammam, Denver, Des Moines, Detroit, Dhaka, Dili, Djibouti, Dodoma, Doha, Douglas, Dover, Dublin, Dushanbe, Edinburgh, El Aaiun, Fargo, Fort-de-France, Frankfort, Freetown, Funafuti, Gaborone, George Town, Georgetown, Gibraltar, Glasgow, Greenwich, Guatemala, Hanoi, Harare, Harrisburg, Hartford, Havana, Helena, Helsinki, Hobart, Hong Kong, Honiara, Honolulu, Houston, Indianapolis, Islamabad, Jackson, Jacksonville, Jakarta, Jefferson City, Jerusalem, Juba, Jubail, Juneau, Kabul, Kampala, Kansas City, Kathmandu, Khartoum, Kiev, Kigali, Kingston, Kingston, Kingstown, Kinshasa, Koror, Kuala Lumpur, Kuwait, La Paz, Lansing, Las Vegas, Leeds, Leicester, Libreville, Lilongwe, Lima, Lincoln, Lisbon, Little Rock, Liverpool, Ljubljana, Lome, London, Los Angeles, Louisville, Luanda, Lusaka, Luxembourg, Macau, Madinah, Madison, Madrid, Majuro, Makkah, Malabo, Male, Mamoudzou, Managua, Manama, Manchester, Manchester, Manila, Maputo, Maseru, Masqat, Mbabane, Mecca, Medina, Melbourne, Memphis, Mexico, Miami, Milwaukee, Minneapolis, Minsk, Mogadishu, Monaco, Monrovia, Montevideo, Montgomery, Montpelier, Moroni, Moscow, Moskva, Mumbai, Muscat, N’Djamena, Nairobi, Nashville, Nassau, Naypyidaw, New Delhi, New Orleans, New York, Newark, Newcastle, Newcastle Upon Tyne, Ngerulmud, Niamey, Nicosia, Norwich, Nouakchott, Noumea, Nuku’alofa, Nuuk, Oklahoma City, Olympia, Omaha, Oranjestad, Orlando, Oslo, Ottawa, Ouagadougou, Oxford, P’yongyang, Pago Pago, Palikir, Panama, Papeete, Paramaribo, Paris, Perth, Philadelphia, Phnom Penh, Phoenix, Pierre, Plymouth, Podgorica, Port Louis, Port Moresby, Port of Spain, Port-Vila, Port-au-Prince, Portland, Portland, Porto-Novo, Portsmouth, Prague, Praia, Pretoria, Pristina, Providence, Quito, Rabat, Raleigh, Reading, Reykjavik, Richmond, Riga, Riyadh, Road Town, Rome, Roseau, Sacramento, Saint Helier, Saint Paul, Saint Pierre, Saipan, Salem, Salt Lake City, San Diego, San Francisco, San Jose, San Juan, San Marino, San Salvador, Sana, Sana’a, Santa Fe, Santiago, Santo Domingo, Sao Tome, Sarajevo, Seattle, Seoul, Sheffield, Singapore, Sioux Falls, Skopje, Sofia, Southampton, Springfield, Sri Jayawardenapura Kotte, St. George’s, St. John’s, St. Peter Port, Stanley, Stockholm, Sucre, Suva, Swansea, Swindon, Sydney, T’bilisi, Taipei, Tallahassee, Tallinn, Tarawa, Tashkent, Tbilisi, Tegucigalpa, Tehran, Thimphu, Tirana, Tirane, Tokyo, Toledo, Topeka, Torshavn, Trenton, Tripoli, Tunis, Ulaanbaatar, Ulan Bator, Vaduz, Valletta, Vienna, Vientiane, Vilnius, Virginia Beach, W. Indies, Warsaw, Washington DC, Wellington, Wichita, Willemstad, Wilmington, Windhoek, Wolverhampton, Yamoussoukro, Yangon, Yaounde, Yaren, Yerevan, Zagreb, Zurich

US Cities

Albany, Albuquerque, Anchorage, Annapolis, Atlanta, Augusta, Austin, Baltimore, Baton Rouge, Billings, Birmingham, Bismarck, Boise, Boston, Bridgeport, Buffalo, Burlington, Carson City, Charleston, Charlotte, Cheyenne, Chicago, Cleveland, Columbia, Columbus, Concord, Dallas, Denver, Des Moines, Detroit, Dover, Fargo, Frankfort, Harrisburg, Hartford, Helena, Honolulu, Houston, Indianapolis, Jackson, Jacksonville, Jefferson City, Juneau, Kansas City, Lansing, Las Vegas, Lincoln, Little Rock, Los Angeles, Louisville, Madison, Manchester, Memphis, Miami, Milwaukee, Minneapolis, Montgomery, Montpelier, Nashville, New Orleans, New York, Newark, Oklahoma City, Olympia, Omaha, Orlando, Philadelphia, Phoenix, Pierre, Portland, Portland, Providence, Raleigh, Richmond, Sacramento, Saint Paul, Salem, Salt Lake City, San Diego, San Francisco, Santa Fe, Seattle, Sioux Falls, Springfield, Tallahassee, Toledo, Topeka, Trenton, Virginia Beach, Wichita, Wilmington

The sun calculations in this module were adapted, for Python, from the spreadsheets on the following page.

> https://www.esrl.noaa.gov/gmd/grad/solcalc/calcdetails.html

Refraction calculation is taken from

> Sun-Pointing Programs and Their Accuracy
> John C. Zimmerman Of Sandia National Laboratones
> https://www.osti.gov/servlets/purl/6377969

Which cites the following as the original source

> In Solar Energy Vol 20 No.5-C
> Robert Walraven Of The University Of California, Davis

Moon position calculations from

> LOW-PRECISION FORMULAE FOR PLANETARY POSITIONS
> T. C. Van Flandern and K. F. Pulkkinen

And from

> Astronomical Algorithms
> Jean Meeus

The moon phase calculation is based on javascript code from Sky and Telescope magazine

> Moon-phase calculation
> Roger W. Sinnott, Sky & Telescope, June 16, 2006.
> https://skyandtelescope.org/observing/the-phase-of-the-moon/

Also to Sphinx for making doc generation an easy thing (not that the writing of the docs is any easier.)

Simon Kennedy <sffjunkie+code@gmail.com>

Version	Description
3.2	Dropped Python 3.6 support as it has reached “End of Life” Documentation now hosted on Github Pages
3.1	Fix for issue #77
3.0	Added moon rise, set, azimuth and zenith functions.https://github.com/sffjunkie/astral/issues/77 Switched from pytz to *zoneinfo* provided as part of Python >= 3.9 or *backports.zoneinfo* for older versions. In some circumstances the result of the calculation of rise and set times would return information for a different date. This has now been fixed.
2.2	Fix for bug #48 - As per the bug report the angle to adjust for the effect of elevation should have been θ (not α). The sun functions can now also be passed a timezone name as a string. Previously only a pytz timezone was accepted.
2.1	Fix for bug #44 - Incorrectly raised exception when UTC sun times were on the day previous to the day asked for. Only manifested for timezones with a large positive offset.
2.0	This is a code refactor as well as an update so it is highly likely that you will need to adapt your code to suit. Astral, AstralGeocoder & GoogleGeocoder classes removed Now only compatible with Python 3.6 and greater due to the use of data classes New `Observer` data class to store a latitude, longitude & elevation New `LocationInfo` data class to store a location name, region, timezone, latitude & longitude Geocoder functions return a `LocationInfo` instead of a `Location` All calculations now automatically adjust for refraction. For elevation you can return the true angle by setting the *with_refraction* parameter to False. The solar_noon and solar_midnight functions have been renamed to `noon()` and `midnight()` respectively. Rahukaalam can now be calculated for night times.
1.10.1	Keyword args are now passed to the geocoder class from Astral __init__ in order to allow the Google Maps API key to be passed to the GoogleGeocoder.
1.10	Added support to AstralGeocoder to add additional locations to the database.
1.9.2	1.9 broke the sun_utc method. Sun UTC calculation passed incorrect parameter to more specific methods e.g. sunrise, sunset etc.

Version	Description
1.9.1	Correct version number in astral.py
1.9	Now takes elevation into account.
1.8	Location methods now allow the timezone to be None which returns all times as UTC. Added command line interface to return ‘sun’ values
1.7.1	Changed GoogleGeocoder test to not use raise…from as this is not valid for Python 2
1.7	Requests is now only needed when using GoogleGeocoder GoogleGeocoder now requires the *api_key* parameter to be passed to the constructor as Google now require it for their API calls.
1.6.1	Updates for Travis CI integration / Github signed release.
1.6	Added api_key parameter to the GoogleGeocoder `__init__()` method
1.5	Added parameter *rtype* to `moon_phase()` to determine the return type of the method. Added example for calculating the phase of the moon.
1.4.1	Using versioneer to manage version numbers
1.4	Changed to use calculations from NOAA spreadsheets Changed some exception error messages for when sun does not reach a requested elevation. Added more tests
1.3.4	Changes to project configuration files. No user facing changes.
1.3.3	Fixed call to twilight_utc as date and direction parameters were reversed.
1.3.2	Updated URL to point to gitgub.com Added Apache 2.0 boilerplate to source file
1.3.1	Added LICENSE file to sdist
1.3	Corrected solar zenith to return the angle from the vertical. Added solar midnight calculation.
1.2	Added handling for when unicode literals are used. This may possibly affect your code if you’re using Python 2 (there are tests for this but they may not catch all uses.) (Bug 1588198) Changed timezone for Phoenix, AZ to America/Phoenix (Bug 1561258)
1.1	Added methods to calculate Twilight, the Golden Hour and the Blue Hour.

Version	Description
1.0	It's time for a version 1.0 Added examples where the location you want is not in the Astral geocoder.
0.9	Added a method to calculate the date and time when the sun is at a specific elevation, for either a rising or a setting sun. Added daylight and night methods to Location and Astral classes. Rahukaalam methods now return a tuple.
0.8.2	Fix for moon phase calcualtions which were off by 1. Use pytz.timezone().localize method instead of passing tzinfo parameter to datetime.datetime. See the pytz docs for info
0.8.1	Fix for bug 1417641: `solar_elevation()` and `solar_azimuth()` fail when a naive `datetime` object is used. Added `solar_zenith()` methods to `Astral` and `Location` as an alias for `solar_elevation()` Added *tzinfo* as an alias for *tz*
0.8	Fix for bug 1407773: Moon phase calculation changed to remove time zone parameter (tz) as it is not required for the calculation.
0.7.5	Fix for bug 1402103: Buenos Aires incorrect timezone
0.7.4	Added Canadian cities from Yip Shing Ho
0.7.3	Fix for bug 1239387 submitted by Torbjörn Lönnemark
0.7.2	Minor bug fix in `GoogleGeocoder`. location name and region are now stripped of whitespace
0.7.1	Bug fix. Missed a vital return statement in the `GoogleGeocoder`
0.7	Added ability to lookup location information from Google's mapping APIs (see `GoogleGeocoder`) Renamed `City` class to `Location` Renamed `CityDB` to `AstralGeocoder` Added elevations of cities to database and property to obtain elevation from `Location` class
0.6.2	Added various cities to database as per https://bugs.launchpad.net/astral/+bug/1040936
0.6.1	Docstrings were not updated to match changes to code. Other minor docstring changes made

Version	Description
0.6	Fix for bug 884716 submitted by Martin Heemskerk regarding moon phase calculations Fixes for bug report 944754 submitted by Hajo Werder • Changed co-ordinate system so that eastern longitudes are now positive • Added solar_depression property to City class
0.5	Changed `City` to accept unicode name and country. Moved city information into a database class `CityDB` Added attribute access to database for timezone groups
0.4	Duplicate city names could not be accessed. Sun calculations for some cities failed with times outside valid ranges. Fixes for city data. Added calculation for moon phase.
0.3	Changed to Apache V2.0 license. Fix for bug 555508 submitted by me. US state capitals and other cities added.
0.2	Fix for bug 554041 submitted by Derek_ / John Dimatos
0.1	First release