Hora-Utils

[Package Structure: 1](#_Toc105341235)

[utils module - functions 2](#_Toc105341236)

[Required imports 2](#_Toc105341237)

[get\_place\_from\_user\_location () 3](#_Toc105341238)

[scrap\_google\_map\_for\_latlongtz\_from\_city\_with\_country (city\_with\_country) 3](#_Toc105341239)

[get\_latitude\_longitude\_from\_place\_name (place\_with\_country\_code) 3](#_Toc105341240)

[get\_place\_latitude\_longitude (place\_name) 3](#_Toc105341241)

[get\_place\_timezone\_offset (latitude, longitude) 3](#_Toc105341242)

[get\_house\_to\_planet\_dict\_from\_planet\_to\_house\_dict (planet\_to\_house\_dict) 4](#_Toc105341243)

[get\_planet\_to\_house\_dict\_from\_chart (house\_to\_planet\_list) 4](#_Toc105341244)

[get\_planet\_house\_dictionary\_from\_planet\_positions (planet\_positions) 4](#_Toc105341245)

[get\_house\_planet\_list\_from\_planet\_positions (planet\_positions): 4](#_Toc105341246)

[get\_resource\_messages (language\_message\_file = const.\_DEFAULT\_LANGUAGE\_MSG\_FILE): 5](#_Toc105341247)

[get\_resource\_lists (language\_list\_file=const.\_DEFAULT\_LANGUAGE\_LIST\_FILE): 5](#_Toc105341248)

[to\_dms\_prec (deg) 5](#_Toc105341249)

[to\_dms (deg,as\_string=False, is\_lat\_long=None) 5](#_Toc105341250)

[unwrap\_angles (angles) 6](#_Toc105341251)

[inverse\_lagrange (x, y, ya) 6](#_Toc105341252)

[julian\_day\_number (date\_of\_birth\_as\_tuple, time\_of\_birth\_as\_tuple) 6](#_Toc105341253)

[deeptaamsa\_range\_of\_planet (planet, planet\_longitude\_within\_raasi) 6](#_Toc105341254)

# Package Structure:

hora

!- data - contains program configuration data, world cities data, marriage compatibility table

!- ephe - contains swiss ephimeride compressed JPL data

!- images - contains images

!- lang - contains language resource files

!- panchanga - panchanga module to calculate daily panchanga

!- horoscope

!- horoscope.py - horoscope package

!- chart - chart package

!- arudhas.py - arudhas, argala, virodhargal

!- ashtavarga.py - ashtavarga, trikona sodhana, ekadhipatya\_sodhana, sodhaya pinda

!- charts.py - divisional charts, planet combustion, retrograde

!- house.py - aspects, drishti,stronger planets/raasi, kaarakas

!- yoga.py - 100+ yogas

!- raja\_yoga.py - raja\_yoga and its sub-types

!- dhasa - dhasa package

!- ashtottari.py - ashtottari dhasa-bhuthi

!- drig.py - drigdhasa-bhuthi

!- kalachakra.py - kalachakra dhasa-bhuthi

!- moola.py - moola dhasa-bhuthi

!- mudda.py - mudda dhasa-bhuthi

!- narayana.py - narayana dhasa-bhuthi

!- nirayana.py - nirayana dhasa-bhuthi

!- patyayini.py - patyayini dhasa-bhuthi

!- shoola.py - shoola dhasa-bhuthi

!- sudasa.py - sudasa dhasa-bhuthi

!- sudharsana\_chakra.py - sudharsana\_chakra dhasa-bhuthi

!- vimsottari.py - vimsottari dhasa-bhuthi

!- match - marriage compatibility package

!- compatibility.py - marriage compatibility

!- transit - tajaka package

!- tajaka.py - annual, monthly and 60 hour charts, muntha, vargeeya balas, tajaka lord

!- tajaka\_yoga.py - tajaka yogas

!- saham.py - 36 sahams

!- ui - user interface package

!- horo\_chart.py - simple horoscope chart Raasi/Navamsa and calendar information

!- horo\_chart\_tabs.py - horoscope with lot of details

!- match\_ui.py - ui for marriage compatibility

!- utils.py - utility functions

!- const.py - constants related to PyHora package

!- tests - unit/integration tests

!- unit\_tests.py - unit tests for the features based on examples from the book

!- pvr\_tests.py - Exercise problems from book.

# utils module - functions

## Required imports

import os

import sys

import codecs

import warnings

import datetime

import geocoder

from pytz import timezone, utc

from timezonefinder import TimezoneFinder

import pandas as pd

import numpy as np

import swisseph as swe

from geopy.geocoders import Nominatim

from hora import const

## get\_place\_from\_user\_location ()

function to get place from user's IP address

@param - None

@return place,latitude,longitude,time\_zone\_offset

## scrap\_google\_map\_for\_latlongtz\_from\_city\_with\_country (city\_with\_country)

function to scrap google maps to get latitude/longitude of a city/country

@param city\_with\_country: city name <comma> country name

Example: Chennai, India

@return [city,latitude,longitude,time\_zone\_offset]

## get\_latitude\_longitude\_from\_place\_name (place\_with\_country\_code)

function to get latitude/longitude from city with country code using Nominatim

requires geopy installed

@param place\_with\_country\_code: city name <comma> country name

Example: Chennai, IN

@return [city,latitude,longitude,time\_zone\_offset]

## get\_place\_latitude\_longitude (place\_name)

this is the top level function to be used to get latitude/longitude from the place name

This internally calls other functions to try out the in the following order

First checks if city is found in the CSV file comes up with the package

Then checks google maps, and then Open Street Map / Nominatim

@param place\_name: city name <comma> country name

Example: Chennai, IN

@return [city,latitude,longitude,time\_zone\_offset]

## get\_place\_timezone\_offset (latitude, longitude)

This can be used when latitude/longitude are known but not the time zone offset of the place.

This is an internal function that returns a location's time zone offset from UTC in minutes - using latitude/longitude of the place.

@param latitude: latitude of the place

@param longitude: longitude of the place

@return [city,latitude,longitude,time\_zone\_offset]

## get\_house\_to\_planet\_dict\_from\_planet\_to\_house\_dict (planet\_to\_house\_dict)

function to get house\_to\_planet list from planet\_to\_house dictionary

@param planet\_to\_house dict: Format {planet\_id : raasi\_number, ....}

Example: {0:0, 1:1,2:1,...} Sun in Aries, Moon in Tarus, Mars in Gemini etc

@return: house\_to\_planet list

- in the format ['0','1/2',...] Aries has Sun, Tarus has Moon/Mars etc

## get\_planet\_to\_house\_dict\_from\_chart (house\_to\_planet\_list)

function to get planet\_to\_house dictionary from house\_to\_planet list

@param house\_to\_planet list - in the format ['0','1/2',...] Aries has Sun, Tarus has Moon/Mars etc

@return: house\_to\_planet\_list:

Format {planet\_id : raasi\_number, ....}

Example: {0:0, 1:1,2:1,...} Sun in Aries, Moon in Tarus, Mars in Gemini etc

## get\_planet\_house\_dictionary\_from\_planet\_positions (planet\_positions)

Get Planet\_to\_House Dictionary {p:h} from Planet\_Positions {p:(h,long)}

@param planet\_positions: Format: {planet\_index:(raasi\_index,planet\_longitude\_in\_the\_raasi),...

@return: planet\_to\_house\_dictionary in the format {planet\_index:raasi\_index,...}

## get\_house\_planet\_list\_from\_planet\_positions (planet\_positions):

to convert from the format [planet,(house,planet\_longitude,...]

into a dict of {house\_1:planet\_1/planet\_2,house\_2:Lagnam/planet\_2,....}

@param planet\_positions: Format: {planet\_index:(raasi\_index,planet\_longitude\_in\_the\_raasi),...

@return: house\_to\_planet list - in the format ['0','1/2',...] Aries has Sun, Tarus has Moon/Mars etc

## get\_resource\_messages (language\_message\_file = const.\_DEFAULT\_LANGUAGE\_MSG\_FILE):

Retrieve message strings from language specific message resource file

@param param:language\_message\_file -language specific message resource file name

Default: const.\_LANGUAGE\_PATH + 'msg\_strings\_' + const.\_DEFAULT\_LANGUAGE + '.txt'

Defualt: ./lang/msg\_strings\_en.txt

@return: dictionary of message keys with language specific values

## get\_resource\_lists (language\_list\_file=const.\_DEFAULT\_LANGUAGE\_LIST\_FILE):

Retrieve resource list from language specific resource list file

list values in resource language are read and returned

@param param:language\_message\_file -language specific message resource file name

Default: \_DEFAULT\_LANGUAGE\_LIST\_FILE = \_LANGUAGE\_PATH + 'list\_values\_' + \_DEFAULT\_LANGUAGE + '.txt'

Defualt: ./lang/list\_values\_en.txt

@return: [PLANET\_NAMES,NAKSHATRA\_LIST,TITHI\_LIST,RAASI\_LIST,KARANA\_LIST,DAYS\_LIST,PAKSHA\_LIST,

YOGAM\_LIST,MONTH\_LIST,YEAR\_LIST,DHASA\_LIST,BHUKTHI\_LIST,PLANET\_SHORT\_NAMES,RAASI\_SHORT\_LIST]

## to\_dms\_prec (deg)

convert float degrees to (int)degrees, (int) minutes, (float) seconds tuple

## to\_dms (deg,as\_string=False, is\_lat\_long=None)

convert float degrees to (int)degrees, (int) minutes, (int) seconds tuple

@param deg: degrees as float

@param as\_string: True - output will be single string with degree symbols

False - output will be tuple (int)degrees, (int) minutes, (float) seconds tuple

@param is\_lat\_long: works with as\_string=True

= plong - degree symbol shown

= lat - N / S symbol is shown for degrees

= long - E / W symbol is shown for degrees

@return: degrees/minutes/seconds as string or tuple depending on as\_string=True and is\_lat\_long values

## unwrap\_angles (angles)

Add 360 to those elements in the input list so that all elements are sorted in ascending order.

## inverse\_lagrange (x, y, ya)

Given two lists x and y, find the value of x = xa when y = ya, i.e., f(xa) = ya

## julian\_day\_number (date\_of\_birth\_as\_tuple, time\_of\_birth\_as\_tuple)

return julian day number for give date of birth and time of birth as tuples

@param date\_of\_birth\_as\_tuple: date of birth as tuple. e.g. (2000,1,1)

Note: For BC Dates use negative year e.g. (-3114,1,1) means 1-Jan of 3114 BC

Note: There is no 0 BC or 0 AD so avoid Zero year

@param time\_of\_birth\_as\_tuple: time of birth as tuple e.g. (18,0,30)

@return julian day number

## deeptaamsa\_range\_of\_planet (planet, planet\_longitude\_within\_raasi)

get deeptaaamsa range of the planet

@param planet: the index of the planet 0 for Sun, 1 for moon, ... 7 for Rahu and 8 for Ketu

@param planet\_longitude\_within\_raasi: longitude of the planet within the raasi (0.0 to 30.0 degrees)

@return: deeptaamsa range of the planet as a tuple (deeptaamsa\_minimum, deeptaamsa\_maximum)