

---

# **solarsystem**

***Release 0.1.4***

**Apr 13, 2020**



---

## Contents:

---

<b>1</b>	<b>solarsystem package</b>	<b>1</b>
1.1	Submodules . . . . .	1
1.1.1	solarsystem.functions module . . . . .	1
1.1.2	solarsystem.geocentric module . . . . .	3
1.1.3	solarsystem.heliocentric module . . . . .	4
1.1.4	solarsystem.moon module . . . . .	5
1.1.5	solarsystem.sunriseset module . . . . .	6
1.2	Module contents . . . . .	6
1.2.1	SolarSystem . . . . .	6
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## 1.1 Submodules

### 1.1.1 solarsystem.functions module

`solarsystem.functions.Planet_Sun` ( $M, e, a, N, w, i$ )

Helper Function. From planet's trajectory elements to position around sun

**Returns** position elements

**Return type** tuple

`solarsystem.functions.demical2arcs` ( $num$ )

Convert Demical view to Degrees and minutes.

**Parameters** `num` ( $float$ ) – degrees to be converted.

**Returns** one string representation in degrees and minutes format.

**Return type** str

`solarsystem.functions.demical2clock` ( $demicaltime$ )

Convert demical time view to Hours, Minutes and Seconds.

**Parameters** `demicaltime` ( $float$ ) – time to be converted.

**Returns** one string representation in hours, minutes format.

**Return type** str

`solarsystem.functions.demical2hms` ( $degrees$ )

Convert degrees to string representation of hours, minutes and seconds.

**Parameters** `degrees` ( $float$ ) – degrees to be converted.

**Returns** one string representation in hours, minutes and seconds format.

**Return type** str

`solarsystem.functions.ecliptic2equatorial` (*xeclip, yeclip, zeclip, oblecl*)

Transform ecliptic to equatorial projection.

**Parameters**

- **xeclip** – value on x axis of ecliptic plane.
- **yeclip** – value on y axis of ecliptic plane.
- **zeclip** – value on z axis of ecliptic plane.
- **oblecl** – obliquity of the ecliptic, approximately 23.4 degrees for earth

**Returns** x, y, z equatorial projection

**Return type** tuple

`solarsystem.functions.equatorial2ecliptic` (*xequat, yequat, zequat, oblecl*)

Transform equatorial to ecliptic projection.

**Parameters**

- **xequat** – value on x axis of equatorial plane
- **yequat** – value on y axis of equatorial plane
- **zequat** – value on z axis of equatorial plane
- **oblecl** – obliquity of the ecliptic, approximately 23.4 degrees for earth

**Returns** x, y, z ecliptic projection

**Return type** tuple

`solarsystem.functions.normalize` (*degrees*)

set degrees always between 0 - 360

**Parameters** **degrees** (*float*) – degrees to be adjusted

**Returns** degrees between 0-360

**Return type** float

`solarsystem.functions.rectangular2spherical` (*x, y, z*)

Transform rectangular to spherical projection.

From rectangular(x,y,z) coordinates system to spherical (RA,Decl, r) or by replacing x with azimuth and y with altitude we can tranform horizontal coordinates to ecliptic (longitude, latitude).

**Parameters**

- **x** – value on x axis of a rectangular projection.
- **y** – value on y axis of a rectangular projection.
- **z** – value on z axis of a rectangular projection.

**Returns** RA, Decl, r spherical coordinate system.

**Return type** tuple

`solarsystem.functions.spherical2rectangular` (*RA, Decl, r*)

Transform spherical to rectangular projection.

From spherical (RA,Decl) coordinates system to rectangular(x,y,z) or by replacing RA with longitude and Decl with latitude we can tranform ecliptic coordinates to horizontal (azimuth,altitude).

**Parameters**

- **RA** – Right Ascension.
- **Decl** – Declination.
- **r** – Distance in astronomical units.

**Returns** x, y, z rectangular coordinate system.

**Return type** tuple

`solarsystem.functions.spherical_ecliptic2equatorial` (*long, lat, distance, oblecl*)

Transform ecliptic to spherical projection for given obliquity.

From spherical (RA, Decl, distance) coordinates system to ecliptic(long, lat, distance).

#### Parameters

- **long** – Longitude.
- **lat** – Latitude.
- **distance** – Distance in astronomical units.
- **oblecl** – obliquity (axial tilt).

**Returns** RA, Decl, distance spherical coordinate system.

**Return type** tuple

`solarsystem.functions.spherical_equatorial2ecliptic` (*RA, Decl, distance, oblecl*)

Transform spherical to ecliptic projection for given obliquity.

From spherical (RA, Decl, distance) coordinates system to ecliptic(long, lat, distance).

#### Parameters

- **RA** – Right Ascension.
- **Decl** – Declination.
- **distance** – Distance in astronomical units.
- **oblecl** – obliquity (axial tilt).

**Returns** long, lat, distance ecliptic coordinate system.

**Return type** tuple

`solarsystem.functions.sun2planet` (*xeclip, yeclip, zeclip, x, y, z*)

Helper Function. From Heliocentric to Geocentric position

**Returns** geocentric view of object.

**Return type** tuple

## 1.1.2 solarsystem.geocentric module

**class** `solarsystem.geocentric.Geocentric` (*year, month, day, hour, minute, UT=0, dst=0, plane='ecliptic'*)

Bases: `object`

Import date data outputs planets positions around Earth.

#### Parameters

- **year** (*int*) – Year (4 digits) ex. 2020
- **month** (*int*) – Month (1-12)

- **day** (*int*) – Day (1-31)
- **hour** (*int*) – Hour (0-23)
- **minute** (*int*) – Minute (0-60)
- **UT** – Time Zone (deviation from UT, -12:+14), ex. for Greece (GMT + 2) enter UT = 2
- **dst** (*int*) – daylight saving time (0 or 1). Wheather dst is applied at given time and place
- **plane** – desired output format. Should be one of: ecliptic, equatorial. Default: ecliptic

**objectnames** ()

Names of solar system objects used.

**Returns** A list of solar system objects

**Return type** list

**position** ()

Main method which returns a dictionary of geocentric positions.

**Returns**

**Planet positions around earth: Each row represents a planet and each column the position of that planet.**

**Return type** dictionary

### 1.1.3 solarsystem.heliocentric module

**class** solarsystem.heliocentric.**Heliocentric** (*year, month, day, hour, minute, UT=0, dst=0, view='horizontal'*)

Bases: object

Import date data outputs planets positions around Sun.

**Parameters**

- **year** (*int*) – Year (4 digits) ex. 2020
- **month** (*int*) – Month (1-12)
- **day** (*int*) – Day (1-31)
- **hour** (*int*) – Hour (0-23)
- **minute** (*int*) – Minute (0-60)
- **UT** – Time Zone (deviation from UT, -12:+14), ex. for Greece (GMT + 2) enter UT = 2
- **dst** (*int*) – daylight saving time (0 or 1). Wheather dst is applied at given time and place
- **view** – desired output format. Should be one of: horizontal (long in degrees, lat in degrees, distance in AU) or rectangular (x, y, z, all in AU). Default: horizontal.

**planetnames** ()

Names of solar system objects used.

**Returns** A list of solar system objects.

**Return type** list

**planets** ()

Main method which returns a dictionary of Heliocentric positions.



**Returns** Planet positions around sun: Dictionary of tuples. Each row represents a planet and each column the position of that planet.

**Return type** dictionary

### 1.1.4 solarsystem.moon module

**class** solarsystem.moon.**Moon** (*year, month, day, hour, minute, UT=0, dst=0, longitude=0.0, latitude=51.48, topographic=False*)

Bases: object

Import date and place outputs moons position, phase and rise-set time.

Moon is a class that feeded with date data as well as geocoordinates outputs moons position around Earth, moon phase and moonrise-moonset/

#### Parameters

- **year** (*int*) – Year (4 digits) ex. 2020
- **month** (*int*) – Month (1-12)
- **day** (*int*) – Day (1-31)
- **hour** (*int*) – Hour (0-23)
- **minute** (*int*) – Minute (0-60)
- **UT** – Time Zone (deviation from UT, -12:+14), ex. for Greece (GMT + 2) enter UT = 2
- **dst** (*int*) – daylight saving time (0 or 1). Wheather dst is applied at given time and place
- **longitude** (*float*) – longitude of place of Moonrise - Moonset in demical format
- **latitude** (*float*) – latitude of place of Moonrise-Moonset in demical format
- **topographic** (*bool*) – Wheather or not moon's position around earth will be calculated regarding earth surface or center

**moonriseset** ()

Method which returns moon's rise and set time

**Returns** Moon's time of given date where moon rises and sets

**Return type** tuple

**phase** ()

Method which returns moon's phase

**Returns** Moon's phase (percent of illumination)

**Return type** float

**position** ()

Method which returns moon's position around Earth

**Returns** Moon's positions around earth in horizontal projection (long, lat and distance in multiple of earth radius)

**Return type** tuple

## 1.1.5 solarsystem.sunriseset module

**class** solarsystem.sunriseset.**Sunriseset** (*year, month, day, UT=0, dst=0, longitude=0.0, latitude=51.48*)

Bases: object

Import date outputs Sunrise and Sunset time.

### Parameters

- **year** (*int*) – Year (4 digits) ex. 2020.
- **month** (*int*) – Month (1-12).
- **day** (*int*) – Day (1-31).
- **UT** – Time Zone (deviation from UT, -12:+14), ex. for Greece (GMT + 2) enter UT = 2.
- **dst** (*int*) – daylight saving time (0 or 1). Wheather dst is applied at given time and place.
- **longitude** (*float*) – longitude of place of Sunrise - Sunset in demical format.
- **latitude** (*float*) – latitude of place of Sunrise - Sunset in demical format.

**riseset** ()

Get the time of sun rise and set within given date.

**Returns** Sunrise - Sunset time of given date

**Return type** tuple

## 1.2 Module contents

### 1.2.1 SolarSystem

- Planets positions around Sun.
- Planets positions around Earth.
- Moon positions around Earth.
- Sunrise/Sunset.
- Moonrise/Moonset, Moon Phase.
- Convert between different coordinate systems.

## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### S

- `solarsystem`, 6
- `solarsystem.functions`, 1
- `solarsystem.geocentric`, 3
- `solarsystem.heliocentric`, 4
- `solarsystem.moon`, 5
- `solarsystem.sunriseset`, 6



## D

`demical2arcs()` (in module *solarsystem.functions*), 1  
`demical2clock()` (in module *solarsystem.functions*), 1  
`demical2hms()` (in module *solarsystem.functions*), 1

## E

`ecliptic2equatorial()` (in module *solarsystem.functions*), 1  
`equatorial2ecliptic()` (in module *solarsystem.functions*), 2

## G

`Geocentric` (class in *solarsystem.geocentric*), 3

## H

`Heliocentric` (class in *solarsystem.heliocentric*), 4

## M

`Moon` (class in *solarsystem.moon*), 5  
`moonriset` (in *solarsystem.moon.Moon* method), 5

## N

`normalize()` (in module *solarsystem.functions*), 2

## O

`objectnames()` (*solarsystem.geocentric.Geocentric* method), 4

## P

`phase()` (*solarsystem.moon.Moon* method), 5  
`Planet_Sun()` (in module *solarsystem.functions*), 1  
`planetnames()` (*solarsystem.heliocentric.Heliocentric* method), 4  
`planets()` (*solarsystem.heliocentric.Heliocentric* method), 4  
`position()` (*solarsystem.geocentric.Geocentric* method), 4

`position()` (*solarsystem.moon.Moon* method), 5

## R

`rectangular2spherical()` (in module *solarsystem.functions*), 2  
`riset` (in *solarsystem.sunriset.Sunriset* method), 6

## S

`solarsystem` (module), 6  
`solarsystem.functions` (module), 1  
`solarsystem.geocentric` (module), 3  
`solarsystem.heliocentric` (module), 4  
`solarsystem.moon` (module), 5  
`solarsystem.sunriset` (module), 6  
`spherical2rectangular()` (in module *solarsystem.functions*), 2  
`spherical_ecliptic2equatorial()` (in module *solarsystem.functions*), 3  
`spherical_equatorial2ecliptic()` (in module *solarsystem.functions*), 3  
`sun2planet()` (in module *solarsystem.functions*), 3  
`Sunriset` (class in *solarsystem.sunriset*), 6