SolTrace API

3.4.0

Generated by Doxygen 1.10.0

1 Class Index 1

1.1 Class List 1

2 Class Documentation 3

2.1 pysoltrace.PySolTrace.Stage.Element Class Reference 3

2.1.1 Detailed Description 4

2.1.2 Member Function Documentation 5

2.1.2.1 aperture_annulus() 5

2.1.2.2 aperture_circle() 6

2.1.2.3 aperture_hexagon() 6

2.1.2.4 aperture_irr_triangle() 6

2.1.2.5 aperture_quadrilateral() 7

2.1.2.6 aperture_rectangle() 8

2.1.2.7 aperture_singleax_curve() 8

2.1.2.8 aperture_triangle() 9

2.1.2.9 copy() 9

2.1.2.10 Create() 9

2.1.2.11 surface_conical() 10

2.1.2.12 surface_cubicspline() 10

2.1.2.13 surface_cylindrical() 10

2.1.2.14 surface_finiteelement() 11

2.1.2.15 surface_flat() 11

2.1.2.16 surface_hypellip() 11

2.1.2.17 surface_parabolic() 12

2.1.2.18 surface_polynomialrev() 12

2.1.2.19 surface_spherical() 12

2.1.2.20 surface_toroid() 13

2.1.2.21 surface_vshot() 13

2.1.2.22 surface_zernicke() 14

2.1.3 Member Data Documentation 14

2.1.3.1 interaction 14

2.2 pysoltrace.PySolTrace.Optics.Face Class Reference 14

2.2.1 Detailed Description 15

2.2.2 Member Data Documentation 16

2.2.2.1 dist_type 16

2.2.2.2 refltable 16

2.2.2.3 transtable 16

2.3 pysoltrace.PySolTrace.Optics Class Reference 16

2.3.1 Detailed Description 17

2.3.2 Member Function Documentation 17
2.3.2.1 copy() 17
2.3.2.2 Create() 18
2.4 pysoltrace.Point Class Reference 18
2.4.1 Detailed Description 18
2.4.2 Constructor & Destructor Documentation 19
2.4.2.1 __init__() 19
2.4.3 Member Function Documentation 19
2.4.3.1 __add__() 19
2.4.3.2 __floordiv__() 19
2.4.3.3 __mul__() 20
2.4.3.4 __sub__() 20
2.4.3.5 __truediv__() 20
2.4.3.6 as_list() 21
2.4.3.7 radius() 21
2.4.3.8 unitize() 21
2.5 pysoltrace.PySolTrace Class Reference 21
2.5.1 Detailed Description 23
2.5.2 Member Function Documentation 24
2.5.2.1 add_optic() 24
2.5.2.2 add_stage() 24
2.5.2.3 add_sun() 24
2.5.2.4 copy() 25
2.5.2.5 Create() 25
2.5.2.6 delete_optic() 25
2.5.2.7 delete_stage() 25
2.5.2.8 plot_flux() 26
2.5.2.9 plot_trace() 26
2.5.2.10 run() 27
2.5.2.11 util_calc_euler_angles() 27
2.5.2.12 util_calc_transforms() 28
2.5.2.13 util_calc_unitvect() 28
2.5.2.14 util_calc_zrot_azel() 29
2.5.2.15 util_matrix_transpose() 29
2.5.2.16 util_matrix_vector_mult() 29
2.5.2.17 util_rotation_arbitrary() 30
2.5.2.18 util_transform_to_local() 30
2.5.2.19 util_transform_to_ref() 31
2.5.2.20 write_soltrace_input_file() 31

2.5.3 Member Data Documentation . . . 31

2.5.3.1 sun . . . 31

2.6 pysoltrace.PySolTrace.Stage Class Reference . . . 32

2.6.1 Detailed Description . . . 33

2.6.2 Constructor & Destructor Documentation . . . 33

2.6.2.1 __init__() . . . 33

2.6.3 Member Function Documentation . . . 33

2.6.3.1 add_element() . . . 33

2.6.3.2 copy() . . . 34

2.6.3.3 Create() . . . 34

2.7 pysoltrace.PySolTrace.Sun Class Reference . . . 34

2.7.1 Detailed Description . . . 35

2.7.2 Member Function Documentation . . . 35

2.7.2.1 Create() . . . 35

2.7.3 Member Data Documentation . . . 36

2.7.3.1 shape . . . 36

2.7.3.2 sigma . . . 36

2.7.3.3 user_intensity_table . . . 36

Index **37**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

pysoltrace.PySolTrace.Stage.Element 3
pysoltrace.PySolTrace.Optics.Face 14
pysoltrace.PySolTrace.Optics 16
pysoltrace.Point 18
pysoltrace.PySolTrace 21
pysoltrace.PySolTrace.Stage 32
pysoltrace.PySolTrace.Sun 34

Chapter 2

Class Documentation

2.1 pysoltrace.PySolTrace.Stage.Element Class Reference

Public Member Functions

- **__init__** (self, parent_stage, int element_id)
- copy (self, enew)
- int Create (self, pdll, p_data)
- surface_spherical (self, radius)
- surface_parabolic (self, focal_len_x, focal_len_y)
- surface_flat (self)
- surface_hypellip (self, vertex_curv, kappa)
- surface_conical (self, theta)
- surface_cylindrical (self, radius)
- surface_toroid (self, rad_annulus, rad_ring)
- surface_zernicke (self, file_path)
- surface_polynomialrev (self, file_path)
- surface_cubicspline (self, file_path)
- surface_finiteelement (self, file_path)
- surface_vshot (self, file_path)
- aperture_circle (self, diameter)
- aperture_hexagon (self, diameter)
- aperture_triangle (self, diameter)
- aperture_rectangle (self, length_x, length_y)
- aperture_annulus (self, r_inner, r_outer, theta)
- aperture_singleax_curve (self, x1, x2, L)
- aperture_irr_triangle (self, x1, y1, x2, y2, x3, y3)
- aperture_quadrilateral (self, x1, y1, x2, y2, x3, y3, x4, y4)

Public Attributes

- **stage_id**

  Identifying integer associated with the containing stage.
- **id**

  Identifying integer associated with element.
- **enabled**

  Flag indicating whether the element is included in the model.
- **position**

  Element location in stage coordinates.
- **aim**

  Element coordinate system aim point in stage coordinates.
- **zrot**

  [deg] Rotation of coordinate system around z-axis
- **aperture**

  Charater indicating aperture type.
- **aperture_params**

  Up to 8 coefficients defining aperture – values depend on selection for 'aperture'.
- **surface**

  Character indicating surface type.
- **surface_params**

  Up to 8 coefficients defining surface – values depend on selection for 'surface'.
- **surface_file**

  Name for surface file, if using compatible type.
- interaction

  Flag indicating optical interaction type.
- **optic**

  Reference to Optics instance associated with this element.

2.1.1 Detailed Description

```
 *Element* is a subclass of PySolTrace.Stage, and represents a set of properties and
geometric settings related to a single optical element in SolTrace. Elements are associ-
ated with a single stage, and are stored in the respective stage’s Stage.elements[] list.

Attributes
----------
stage_id : int
    Identifying integer associated with the containing stage
id : int
    Identifying integer associated with element
enabled : bool
    Flag indicating whether the element is included in the model
position : Point
    Element location in stage coordinates
aim : Point
    Element coordinate system aim point in stage coordinates
zrot : float
    [deg] Rotation of coordinate system around z-axis
aperture : char
    Charater indicating aperture type. One of:
    {’c’:circle, ’h’:hexagon, ’t’:triangle, ’r’:rectangle, ’a’:annulus,
    ’l’:single-axis curvature, ’i’:irregular triangle, ’q’:quadrilateral}
```

```
aperture_params : [float,]
    Up to 8 coefficients defining aperture -- values depend on selection for ’aperture’
surface : char
    Character indicating surface type. One of:
    {’s’:spherical, ’p’:parabolic, ’f’:flat plane, ’o’:hyperboloid/ellipsoid,
    ’c’:conical, ’t’:cylindrical, ’d’:toroid, ’m’:Zernicke monomial,
    ’r’:Polynomial revolution, ’i’:cubic spline interpolation,
    ’e’:finite element data, ’v’:VSHOT data}
surface_params : [float,]
    Up to 8 coefficients defining surface -- values depend on selection for ’surface’
surface_file : string
    Name for surface file, if using compatible type. File extension:
    *.mon --> ’m’ / Zernicke monomial
    *.sht --> ’v’ / VSHOT data
    *.ply --> ’r’ / Polynomial revolution
    *.csi --> ’i’ / Cubic spline interpolation
    *.fed --> ’e’ / Finite element data
interaction : int
    Flag indicating optical interaction type. {1:refraction, 2:reflection}
optic : Optics
    Reference to *Optics* instance associated with this element

Methods
----------
Create
    Calls methods to instantiate and construct element in the SolTrace context
surface_XXXXXX
    Family of methods that compute surface coefficients. Options include:
    surface_spherical, surface_parabolic, surface_flat, surface_hypellip,
    surface_conical, surface_cylindrical, surface_toroid, surface_zernicke,
    surface_polynomialrev, surface_cubicspline, surface_finiteelement,
    surface_vshot
aperture_XXXXXX
    Family of methods that compute aperture coefficients. Options include:
    aperture_circle, aperture_hexagon, aperture_triangle, aperture_rectangle,
    aperture_annulus, aperture_singleax_curve, aperture_irr_triangle,
    aperture_quadrilateral
```

2.1.2 Member Function Documentation

2.1.2.1 aperture_annulus()

```
pysoltrace.PySolTrace.Stage.Element.aperture_annulus (
              self,
              r_inner,
              r_outer,
              theta )
```

```
Set up the aperture as annular, where aperture is the annulus between to specified radii
and within an angular slice ’theta’ which is centered around the x-axis.

Aim: The X and Y directions lie in the plane of the annulus. Z is normal to the plane.

Parameters
----------
r_inner
    Inner radius of annular region
r_outer
    Outer radius of annular region
theta : deg
    Slice of the circle contained, centered around x-axis
```

2.1.2.2 aperture_circle()

```
pysoltrace.PySolTrace.Stage.Element.aperture_circle (
              self,
              diameter )
```

```
Set up the aperture as circular with 'diameter'.

Aim: The X and Y directions lie in the plane of the circle. Z is normal to the plane.

Parameters
----------
diameter
    Diameter of the circle
```

2.1.2.3 aperture_hexagon()

```
pysoltrace.PySolTrace.Stage.Element.aperture_hexagon (
              self,
              diameter )
```

```
Set up the aperture as a hexagon centered at x=0,y=0. The hexagon is circumscribed
by a circle of 'diameter'.

Aim: The X and Y directions lie in the plane of the hexagon. X crosses through a vertex
between two segments, while Y bisects an edge segment. Z is normal to the plane.

 y^
 __
/  \ x->
\__/

Parameters
----------
diameter
    Diameter of the circumscribing circle.
```

2.1.2.4 aperture_irr_triangle()

```
pysoltrace.PySolTrace.Stage.Element.aperture_irr_triangle (
              self,
              x1,
              y1,
              x2,
              y2,
              x3,
              y3 )
```

```
Set up the aperture as a triangle given by three (x,y) coordinate pairs.

Aim: X and Y are in the plane containing the coordinates. Z is normal to the plane.

Parameters
----------
x1
    x-coordinate, point 1
y1
    y-coordinate, point 1
x2
    x-coordinate, point 2
y2
    y-coordinate, point 2
x3
    x-coordinate, point 3
y3
    y-coordinate, point 3
```

2.1.2.5 aperture_quadrilateral()

```
pysoltrace.PySolTrace.Stage.Element.aperture_quadrilateral (
              self,
              x1,
              y1,
              x2,
              y2,
              x3,
              y3,
              x4,
              y4 )
```

```
Set up the aperture as a quadrilateral given by four (x,y) coordinate pairs.

Aim: X and Y are in the plane containing the coordinates. Z is normal to the plane.

Parameters
----------
x1
    x-coordinate, point 1
y1
    y-coordinate, point 1
x2
    x-coordinate, point 2
y2
    y-coordinate, point 2
x3
    x-coordinate, point 3
y3
    y-coordinate, point 3
x4
    x-coordinate, point 4
y4
    y-coordinate, point 4
```

2.1.2.6 aperture_rectangle()

```
pysoltrace.PySolTrace.Stage.Element.aperture_rectangle (
              self,
              length_x,
              length_y )
```

```
Set up the aperture as a rectangle.

Aim: The X and Y directions lie in the plane of the rectangle. Y crosses bisects a horzontal leg
of width 'W', while X bisects a vertical leg of height 'H'. Z is normal to the plane. The
coordinates are centered x=W/2, y=H/2.

Parameters
----------
length_x
    Width in x-coordinate direction
length_y
    Height in y-coordinate direction
```

2.1.2.7 aperture_singleax_curve()

```
pysoltrace.PySolTrace.Stage.Element.aperture_singleax_curve (
              self,
              x1,
              x2,
              L )
```

```
Set up the aperture as revolved around a single axis. Revolved window is between two
coordinates x1->x1, both non-negative and with x2 > x1. The aperture has
length 'L' in the y-direction.

This aperture is often used with a cylindrical surface. In this case,
both x1 and x2 should be zero, and the cylinder height specified with 'L'.

Aim: X and Z follow radial lines and cross through the curvature section. Y lies along the
centerline/axis of the cylindrical section at X=0, Z=0. The radial positions are with
respect to the X and Z coordinates.

^ y
|    ___  ....L
|   |   |
|---|---|---> X
|   |___| ....
|   x1  x2

Parameters
----------
x1
    inner coordinate of revolved section
x2
    outer coordinate of revolved section
L
    length of revolved section along axis of revolution
```

2.1.2.8 aperture_triangle()

```
pysoltrace.PySolTrace.Stage.Element.aperture_triangle (
              self,
              diameter )
```

```
Set up the aperture as a equilateral triangle with centroid at x=0,y=0. The triangle
is circumscribed by a circle of 'diameter'.

Aim: The X and Y directions lie in the plane of the triangle. Y crosses through a vertex between
two segments, while X crosses at an intermediate position on one leg of the triangle. Z is normal
to the plane. The coordinates are centered at the middle of a circle of diameter 'D' that circumscribes
the isoceles triangle

  y^
  /\
 /  \  x ->
/____\

Parameters
----------
diameter
    Diameter of the circumscribing circle.
```

2.1.2.9 copy()

```
pysoltrace.PySolTrace.Stage.Element.copy (
              self,
              enew )
```

```
Deep copy of the current Element instance

Inputs
---------
enew : Stage.Element
    Reference to new Element object to which data will be copied
```

2.1.2.10 Create()

```
int pysoltrace.PySolTrace.Stage.Element.Create (
              self,
              pdll,
              p_data )
```

```
Create Element instance in the SolTrace context.

Returns
----------
int
    1 if successful, 0 otherwise
```

2.1.2.11 surface_conical()

```
pysoltrace.PySolTrace.Stage.Element.surface_conical (
              self,
              theta )
```

```
Set up the surface described by cone with half-angle theta.

The axis of the cone coincides with the z-axis. The function of the surface is:
    Z(x,y) = sqrt(x^2 + y^2)/tan(theta)

Parameters
----------
theta : float
    (degrees) half-angle of cone
```

2.1.2.12 surface_cubicspline()

```
pysoltrace.PySolTrace.Stage.Element.surface_cubicspline (
              self,
              file_path )
```

```
Set up the surface from a file as a rotationally symmetric cubic spline. Accepts *csi file extension.
File format should be two tab-separated columns:
    N
    r1      Z1
    r2      Z2
    r3      Z3
    ...
    rN      ZN
    dZ/dr1  dZ/drN

Parameters
----------
file_path
    Path to the file containing the data.
```

2.1.2.13 surface_cylindrical()

```
pysoltrace.PySolTrace.Stage.Element.surface_cylindrical (
              self,
              radius )
```

```
Set up the surface as cylindrical.

The surface centroid is located at x=0, y=0, z=radius. The cylinder's axis
is parallel to the Y-axis.

Parameters
----------
radius
    Radius of the cylinder
```

2.1.2.14 surface_finiteelement()

```
pysoltrace.PySolTrace.Stage.Element.surface_finiteelement (
              self,
              file_path )
```

```
Set up the surface from a file using finite element data specifying the vertices of the elements in
x,y,z coordinates.

Accepts the *.fed file extension. File format should be 3 tab-separated
columns:
    N
    x1      y1      z1
    x2      y2      z2
    x3      y3      z3
    ...
    xN      yN      zN

Parameters
----------
file_path
    Path to the file containing the data.
```

2.1.2.15 surface_flat()

```
pysoltrace.PySolTrace.Stage.Element.surface_flat (
              self )
```

```
Set up the surface as flat
```

2.1.2.16 surface_hypellip()

```
pysoltrace.PySolTrace.Stage.Element.surface_hypellip (
              self,
              vertex_curv,
              kappa )
```

```
Set up the surface described by equation:
    Z(x,y) = ( vertex_curv*(x^2 + y^2) ) /
                (1 + sqrt(1-kappa*vertex_curv^2*(x^2 + y^2)))
Parameters
----------
vertex_curv
    Curvature parameter
kappa
    Form parameter. Value of parameter determines geometry as follows:
    kappa < 0 --> tall hyperboloid
    kappa 0..1 --> ellipsoid
    kappa > 1 --> stout ellipsoid
```

2.1.2.17 surface_parabolic()

```
pysoltrace.PySolTrace.Stage.Element.surface_parabolic (
              self,
              focal_len_x,
              focal_len_y )
```

```
Set up the surface as parabolic.

Surface function is:
    Z(x,y) = 1/2 * (c_x * x^2 + c_y * y^2)
    where
    c_x = 1 / (2 * focal_len_x)
    c_y = 1 / (2 * focal_len_y)

The surface value is z=0 at x=y=0.

Parameters
==========
focal_len_x : float
    Focal length of the surface in the x-direction. If infinite, use float('inf')
focal_len_y : float
    Focal length of the surface in the y-direction. If infinite, use float('inf')
```

2.1.2.18 surface_polynomialrev()

```
pysoltrace.PySolTrace.Stage.Element.surface_polynomialrev (
              self,
              file_path )
```

```
Set up the surface from a file as a rotationally symmetric polynomial, where the surface is described by
the equation:
Z(r) = sum_i=0^N  C_i * r^i,  where r=sqrt(x^2 + y^2)

Accepts *ply file extension specifying equation coefficients.
File format should be a single data column:
    N
    C0
    C1
    C2
    ...
    C,N

Parameters
----------
file_path
    Path to the file containing the data.
```

2.1.2.19 surface_spherical()

```
pysoltrace.PySolTrace.Stage.Element.surface_spherical (
              self,
              radius )
```

```
Set up the surface as spherical type.

Surface centroid is at x=0, y=0, z=radius.

Parameters
==========
radius : float
    Radius of the spherical surface
```

2.1.2.20 surface_toroid()

```
pysoltrace.PySolTrace.Stage.Element.surface_toroid (
              self,
              rad_annulus,
              rad_ring )
```

```
Set up the surface as a toroid "donut".

Parameters
----------
rad_annulus
    Radius of the ’tube’, the distance between the min and max radii of the torus
rad_ring
    The radius of the centerpoint of the annular tube
```

2.1.2.21 surface_vshot()

```
pysoltrace.PySolTrace.Stage.Element.surface_vshot (
              self,
              file_path )
```

```
Set up the surface from a file using VSHOT data specifying matrix coefficients generated by a VSHOT test.

Accepts the *.sht file extension. File format should be:
    First line - file name (skipped)
    Radius      Focal length       Target-dist
    0           order              num points
    rmsslope    rmsscale
    b00
    b10
    b11
    b20
    b21
    b22
    ...
    bDD    || where ’D’ is order
    a1      b1      c1      d1      e1
    a2      b2      c2      d2      e2
    a3      b3      c3      d3      e3
    ...
    aN      bN      cN      dN      eN  || where ’N’ is num points

Parameters
----------
file_path
    Path to the file containing the data.
```

2.1.2.22 surface_zernicke()

```
pysoltrace.PySolTrace.Stage.Element.surface_zernicke (
              self,
              file_path )
```

```
Set up the surface from a file as a Zernicke surface, where the surface is described by the equation:
Z(x,y) = sum_i=0^N
            sum_j=0^i  Bi,j * x^j * y^(i-j)

Accepts *mon file extension specifying the Zernicke coefficients.
File format should be a single data column:
    N
    B0,0
    B1,0
    B1,1
    B2,1
    B2,2
    B2,3
    ...
    BN,N

Parameters
----------
file_path
    Path to the file containing the data.
```

2.1.3 Member Data Documentation

2.1.3.1 interaction

```
pysoltrace.PySolTrace.Stage.Element.interaction
```

Flag indicating optical interaction type.

{1:refraction, 2:reflection}

The documentation for this class was generated from the following file:

- pysoltrace.py

2.2 pysoltrace.PySolTrace.Optics.Face Class Reference

Public Member Functions

- **__init__** (self)
- **copy** (self, fnew)

Public Attributes

- dist_type

  Distribution type for surface interactions.
- **refraction_real**

  Real component of the refraction index.
- **reflectivity**

  [0..1] Surface reflectivity
- **transmissivity**

  [0..1] Surface transmissivity
- **slope_error**

  [mrad] Surface RMS slope error, half-angle
- **spec_error**

  [mrad] Surface specularity error, half-angle
- **userefltable**

  Flag specifying use of user reflectivity table to modify reflectivity as a function of incidence angle.
- refltable

  [mrad,0..1] 2D list containing pairs of [angle,reflectivity] values.
- **usetranstable**

  Flag specifying use of user transmissivity table to modify transmissivity as a function of incidence angle.
- transtable

  [mrad,0..1] 2D list containing pairs of [angle,transmissivity] values.

2.2.1 Detailed Description

```
Subclass of Optics, contains properties associated with one of the optical faces.

Attributes
----------
dist_type : char
    Distribution type for surface interactions. One of:
    {'g':Gaussian, 'p':Pillbox, 'd':Diffuse }
refraction_real : float
    Real component of the refraction index
reflectivity : float
    [0..1] Surface reflectivity
transmissivity : float
    [0..1] Surface transmissivity
slope_error : float
    [mrad] Surface RMS slope error, half-angle
spec_error : float
    [mrad] Surface specularity error, half-angle
userefltable : bool
    Flag specifying use of user reflectivity table to modify reflectivity as a function of incidence angle
refltable : [[float,float],]
    [mrad,0..1] 2D list containing pairs of [angle,reflectivity] values.
usetranstable : bool
    Flag specifying use of user transmissivity table to modify transmissivity as a function of incidence angle
transtable : [[float,float],]
    [mrad,0..1] 2D list containing pairs of [angle,transmissivity] values.

Methods
----------
copy
    Deep copy of the current Face instance
```

2.2.2 Member Data Documentation

2.2.2.1 dist_type

```
pysoltrace.PySolTrace.Optics.Face.dist_type
```

Distribution type for surface interactions.

One of: {'g':Gaussian, 'p':Pillbox, 'd':Diffuse }

2.2.2.2 refltable

```
pysoltrace.PySolTrace.Optics.Face.refltable
```

[mrad,0..1] 2D list containing pairs of [angle,reflectivity] values.

2.2.2.3 transtable

```
pysoltrace.PySolTrace.Optics.Face.transtable
```

[mrad,0..1] 2D list containing pairs of [angle,transmissivity] values.

The documentation for this class was generated from the following file:

- pysoltrace.py

2.3 pysoltrace.PySolTrace.Optics Class Reference

Classes

- class Face

Public Member Functions

- **__init__** (self, int id)
- copy (self, onew)
- int Create (self, pdll, p_data)

Public Attributes

- **name**

  Unique name for the optical property set.
- **id**

  Identifying integer associated with the property set.
- **front**

  properties associated with the front of the optical surface
- **back**

  properties associated with the back of the optical surface

2.3.1 Detailed Description

```
 *Optics* is a subclass of PySolTrace, and represents an optical property set.
A PySolTrace instance may have multiple Optics member instances, which are stored in
the PySolTrace.optics list.

Optics contains a subclass *Face*, which collects properties associated with the front
or back face of an optical surface.

Attributes
----------
name : str
    Unique name for the optical property set
id : int
    Identifying integer associated with the property set
front : Face
    properties associated with the front of the optical surface
back : Face
    properties associated with the back of the optical surface

Methods
----------
copy
    Deep copy of the current Optics instance
Create
    Calls methods to instantiate and construct optical surface in the SolTrace context.
```

2.3.2 Member Function Documentation

2.3.2.1 copy()

```
pysoltrace.PySolTrace.Optics.copy (
              self,
              onew )
```

```
Deep copy of the current Optics instance

Inputs
---------
onew : Optics.Face
    Reference to new Optics object to which data will be copied
```

2.3.2.2 Create()

```
int pysoltrace.PySolTrace.Optics.Create (
              self,
              pdll,
              p_data )
```

```
Create Optics instance in the SolTrace context.

Returns
----------
int
    1 if successful, 0 otherwise
```

The documentation for this class was generated from the following file:

- pysoltrace.py

2.4 pysoltrace.Point Class Reference

Public Member Functions

- __init__ (self, x=0, y=0, z=0)
- **copy** (self)
- **__str__** (self)
- __add__ (self, obj)
- __sub__ (self, obj)
- __mul__ (self, obj)
- __floordiv__ (self, obj)
- __truediv__ (self, obj)
- radius (self)
- unitize (self, bool inplace=False)
- as_list (self)

Public Attributes

- **x**

  (float) x-coordinate
- **y**

  (float) y-coordinate
- **z**

  (float) z-coordinate

2.4.1 Detailed Description

```
A simple class to manage points in Cartesian coordinates.
```

2.4.2 Constructor & Destructor Documentation

2.4.2.1 __init__()

```
pysoltrace.Point.__init__ (
              self,
              x = 0,
              y = 0,
              z = 0 )
```

```
Parameters
==========
x : float
    x-coordinate
y : float
    y-coordinate
z : float
    z-coordinate
```

2.4.3 Member Function Documentation

2.4.3.1 __add__()

```
pysoltrace.Point.__add__ (
              self,
              obj )
```

```
Add to the current point coordinate values.

Parameters
==========
obj : variant
    If obj = (Point), adds component-wise to the current point
    If obj = (float), adds obj to each component
Returns
=======
Point
    Reference to this point
```

2.4.3.2 __floordiv__()

```
pysoltrace.Point.__floordiv__ (
              self,
              obj )
```

```
Divides the current point coordinate values, taking the floor of the result.

Parameters
==========
obj : variant
    If obj = (Point), divides current point component-wise
    If obj = (float), divides components of current point by obj
Returns
=======
Point
    Reference to this point
```

2.4.3.3 __mul__()

```
pysoltrace.Point.__mul__ (
              self,
              obj )
```

```
Multiplies the current point coordinate values.

Parameters
==========
obj : variant
    If obj = (Point), multiplies the current point component-wise by obj
    If obj = (float), multiplies each component of the current point by obj
Returns
=======
Point
    Reference to this point
```

2.4.3.4 __sub__()

```
pysoltrace.Point.__sub__ (
              self,
              obj )
```

```
Subtract from the current point coordinate values.

Parameters
==========
obj : variant
    If obj = (Point), subtracts component-wise from the current point
    If obj = (float), subtracts obj from each component
Returns
=======
Point
    Reference to this point
```

2.4.3.5 __truediv__()

```
pysoltrace.Point.__truediv__ (
              self,
              obj )
```

```
Divides the current point coordinate values/

Parameters
==========
obj : variant
    If obj = (Point), divides current point component-wise
    If obj = (float), divides components of current point by obj
Returns
=======
Point
    Reference to this point
```

2.4.3.6 as_list()

```
pysoltrace.Point.as_list (
              self )
```

```
Returns:
--------
coordinates as a python list [x,y,z]
```

2.4.3.7 radius()

```
pysoltrace.Point.radius (
              self )
```

```
Computes distance between current point and the origin (0,0,0)

Returns
=======
float
    Calculated radius
```

2.4.3.8 unitize()

```
pysoltrace.Point.unitize (
              self,
             bool  inplace = False )
```

```
Converts current point into a unit vector with magnitude 1

Parameters
==========
inplace : bool = False
    Specifies whether the current point is converted to a unit vector in place, or whether the
    current point remains unchanged and a unitized copy of the vector is returned.
```

The documentation for this class was generated from the following file:

- pysoltrace.py

2.5 pysoltrace.PySolTrace Class Reference

Classes

- class Optics
- class Stage
- class Sun

Public Member Functions

- **__init__** (self)
- copy (self)
- Create (self, pdll, p_data)
- add_optic (self, str optic_name)
- int delete_optic (self, int optic_id)
- add_sun (self)
- add_stage (self)
- int delete_stage (self, int stage_id)
- run (self, int seed=-1, as_power_tower=False, nthread=1, thread_id=0)
- plot_trace (self, int nrays=100000, int ntrace=100)
- plot_flux (self, element, int nx=25, int ny=25, str figpath=None, display=True, figsize=(9, 6), bool absorbed_↩ only=True, levels=25, int dpi=300, str xlabel=None, str ylabel=None)
- numpy.array util_calc_euler_angles (self, numpy.array origin, numpy.array aimpoint, zrot)
- util_transform_to_local (self, numpy.array posref, numpy.array cosref, numpy.array origin, numpy.array rreftoloc)
- util_transform_to_ref (self, posloc, cosloc, origin, rloctoref)
- util_matrix_vector_mult (self, m, v)
- util_calc_transforms (self, euler)
- util_matrix_transpose (self, m)
- util_rotation_arbitrary (self, theta, axis, axloc, pt)
- util_calc_unitvect (self, vect)
- float util_calc_zrot_azel (self, vect)
- write_soltrace_input_file (self, str path)

Public Attributes

- **optics**

  List of Optics instances.
- **stages**

  List of Stage instances.
- sun

  sun
- **num_ray_hits**

  Minimum number of simulation ray hits.
- **max_rays_traced**

  Maximum number of ray hits in a simulation.
- **is_sunshape**

  Flag indicating whether sunshape should be included.
- **is_surface_errors**

  Flag indicating whether surface errors should be included.
- **raydata**
- **sunstats**
- **powerperray**
- **dni**

2.5.1 Detailed Description

```
A class to access PySolTrace (SolTrace's Python API)

Attributes
----------
optics : [PySolTrace.Optics,]
    List of Optics instances
stages : [PySolTrace.Stage,]
    List of Stage instances
sun : PySolTrace.Sun
    Instance of the Sun class
num_ray_hits : int
    Minimum number of simulation ray hits
max_rays_traced : int
    Maximum number of ray hits in a simulation
is_sunshape : bool
    Flag indicating whether sunshape should be included
is_surface_errors : bool
    Flag indicating whether surface errors should be included
raydata : Pandas.dataframe
    Dataframe with ray hit information
sunstats : dict
    Dict containing information on the ray trace bounding box
powerperray : float
    Calculated value of power associated with a single ray hit
dni : float
    Specified direct normal irradiance

Methods
-------
copy
    Deep copy of the current PySolTrace instance
Create
    Create soltrace context from data structures
add_optics
    Instantiates a new PySolTrace.Optics object
delete_optic
    Delete Optics instance
add_sun
    Adds Sun instance
add_stage
    Adds Stage instance
delete_stage
    Deletes stage instance
run
    Runs SolTrace simulation
plot_trace
    Creates and (optionally) displays a 3D scatter and trace plot.
plot_flux
    Creates and (optionally) displays a flux plot for a given stage element.
util_calc_euler_angles
    Calculate Euler angles for a position, aimpoint, and rotation
util_transform_to_local
    Transform a coordinate system from reference to a local system
util_transform_to_ref
    Transform a coordinate system from local to reference system
util_matrix_vector_mult
    Calculate product of a square matrix and a vector
util_calc_transforms
    Calculate matrix transforms
util_matrix_transpose
    Compute the transpose of a matrix
util_rotation_arbitrary
    Rotation of a point about an arbitrary axis
util_calc_unitvect
    Scales a vector to have total magnitude of 1
util_calc_zrot_azel
```

```
    Compute the z-rotation of a vector
write_soltrace_input_file
    Write a SolTrace input file based on the current objects
```

2.5.2 Member Function Documentation

2.5.2.1 add_optic()

```
pysoltrace.PySolTrace.add_optic (
              self,
             str optic_name )
```

```
Instantiates a new PySolTrace.Optics object, adding it to the optics list.
This method does not set optics properties, which instead is done using the Optics.Create method.

Parameters
----------
optic_name : string
    Unique name for this Optics instance.

Returns
----------
Optics
    Reference to the Optics object that was just created.
```

2.5.2.2 add_stage()

```
pysoltrace.PySolTrace.add_stage (
              self )
```

```
Adds a new Stage instance to the PySolTrace.stages list. The Stage ID is automatically generated based on the numb
of current stages.

Returns
----------
PySolTrace.Stage
    Reference to the newly created Stage object.
```

2.5.2.3 add_sun()

```
pysoltrace.PySolTrace.add_sun (
              self )
```

```
Instantiates a PySolTrace.Sun object and associates it with the PySolTrace.sun member.
This does not create or modify the Sun data in the SolTrace context.

Returns
----------
PySolTrace.Sun
    Reference to newly created Sun instance.
```

2.5.2.4 copy()

```
pysoltrace.PySolTrace.copy (
              self )
```

```
Deep copy of the current PySolTrace instance

Returns:
========
Copy of the current PySolTrace instance
```

2.5.2.5 Create()

```
pysoltrace.PySolTrace.Create (
              self,
              pdll,
              p_data )
```

```
Create soltrace context from data structures
```

2.5.2.6 delete_optic()

```
int pysoltrace.PySolTrace.delete_optic (
              self,
             int optic_id )
```

```
Delete Optics instance. The optics object is removed from the PySolTrace.optics list and
from the SolTrace context.

Parameters
----------
optic_id : int
    ID associated with the optics to be deleted

Returns
----------
int
    1 if successful, 0 otherwise
```

2.5.2.7 delete_stage()

```
int pysoltrace.PySolTrace.delete_stage (
              self,
             int stage_id )
```

```
Delete Stage instance. The stage object is removed from the PySolTrace.stages list and
from the SolTrace context.

Parameters
----------
stage_id : int
    ID associated with the stage to be deleted

Returns
----------
int
    1 if successful, 0 otherwise
```

2.5.2.8 plot_flux()

```
pysoltrace.PySolTrace.plot_flux (
              self,
              element,
             int  nx = 25,
             int  ny = 25,
             str figpath = None,
              display = True,
              figsize = (9,6),
             bool  absorbed_only = True,
              levels = 25,
             int dpi = 300,
             str xlabel = None,
             str  ylabel = None )
```

```
Creates and (optionally) displays a flux plot for a given stage element.

Parameters
----------
element : PySolTrace:Stage:Element
    Reference to the element for which the plot will be generated
nx : int (default 25)
    Number of flux bins along the aperture x-coordinate
ny : int (default 25)
    Number of flux bins along the aperture y-coordinate
figpath : str (default None)
    Path to file location where figure will be saved. If None, figure is not saved.
display : bool (default True)
    Flag indicating whether the figure should be displayed at runtime
figsize : tuple (default (9,6))
    Figure size in inches
absorbed_only : bool (default True)
    Only include rays that are absorbed by the element, omitting reflected rays
levels : int (default 25)
    Number of contour levels to include in the flux map
dpi : int (default 300)
    Resolution of the saved image
xlabel : str (default None)
    String specifying label to use on x-axis of plot
ylabel : str (default None)
    String specifying label to use on y-axis of plot

Returns
------------
None
```

2.5.2.9 plot_trace()

```
pysoltrace.PySolTrace.plot_trace (
              self,
             int  nrays = 100000,
             int ntrace = 100 )
```

```
Creates and (optionally) displays a 3D scatter and trace plot. This
function requires that the Python package `plotly` be installed.
```

```
Parameters
------------
nrays : int
    Number of individual rays to include in the scatter plot. Very
    large values may render slowly.
ntrace : int
    Number of rays for which traces will be displayed. Large values
    may render slowly
```

2.5.2.10 run()

```
pysoltrace.PySolTrace.run (
              self,
             int  seed = -1,
              as_power_tower = False,
              nthread = 1,
              thread_id = 0 )
```

```
Run SolTrace simulation.

If calling this function in multithread mode, note that the run() function
**must** be called inside an import guard, e.g.:
> if __name__ == "__main__":
>    mypst_obj.run(...)
Otherwise, you'll receive an error.

Parameters
----------
seed : int
    Seed for random number generator. [-1] for random seed.
as_power_tower : bool
    Flag indicating simulation should be processed as power
    tower / central receiver type, with corresponding efficiency adjustments.
nthread : int
    Number of threads to execute. Will be limited by the method to the number
    available on the machine.
thread_id : int
    Argument used by the multi-threading call. Do not manually specify this value.

Returns
----------
int
    Simulation return value
```

2.5.2.11 util_calc_euler_angles()

```
numpy.array pysoltrace.PySolTrace.util_calc_euler_angles (
              self,
             numpy.array origin,
             numpy.array aimpoint,
              zrot )
```

```
Calculate the Euler angles associated with a given origin, aimpoint, and z-axis rotation.

Parameters
----------
origin : [float,*3]
    Origin of the coordinate system
aimpoint : [float,*3]
    Aimpoint of the vector originating at the origin
zrot : float
    Rotation around the z-axis coordinate (degr)

Returns
----------
list
    Calculated Euler angles (rad)
```

2.5.2.12 util_calc_transforms()

```
pysoltrace.PySolTrace.util_calc_transforms (
              self,
              euler )
```

```
Calculate matrix transforms

Parameters
----------
euler : [float,]*3
    Euler angles

Returns
----------
(dict) A dictionary containing the keys:
    rreftoloc : Transformation matrix from Reference to Local system
    rloctoref : Transformation matrix from Local to Reference system
```

2.5.2.13 util_calc_unitvect()

```
pysoltrace.PySolTrace.util_calc_unitvect (
              self,
              vect )
```

```
Scales a vector to have total magnitude of 1

Parameters
----------
vect : list | Point
    list or Point containing the vector

Returns
----------
list | Point
    Unitized vector of type list or Point, depending on input type
```

2.5.2.14 util_calc_zrot_azel()

```
float pysoltrace.PySolTrace.util_calc_zrot_azel (
              self,
              vect )
```

```
Compute the z-rotation of a vector, assuming the vector's deviation from (0,0,1)
has been realized using azimuth-elevation transforms.

Parameters
----------
vect : (list OR Point)
    i,j,k components of a vector

Returns
----------
float
    Computed z-rotation (degrees)
```

2.5.2.15 util_matrix_transpose()

```
pysoltrace.PySolTrace.util_matrix_transpose (
              self,
              m )
```

```
Calculate matrix transpose

Parameters
----------
m : [[float,]*3]*3
    Square matrix 3x3 to be transposed.

Returns
----------
[[float,]*3]*3
    Square matrix 3x3, transpose of 'm'
```

2.5.2.16 util_matrix_vector_mult()

```
pysoltrace.PySolTrace.util_matrix_vector_mult (
              self,
              m,
              v )
```

```
Perform multiplication of a 3x3 matrix and a length-3 vector, returning the result vector.

Parameters
----------
m : array
    m[3][3] - a 3x3 matrix
v : array
    v[3] - a list, length 3

Returns
----------
list
    m x v [3]
```

2.5.2.17 util_rotation_arbitrary()

```
pysoltrace.PySolTrace.util_rotation_arbitrary (
              self,
              theta,
              axis,
              axloc,
              pt )
```

```
Rotation of a point 'pt' about an arbitrary axis with direction 'axis' centered at point 'axloc'.
The point is rotated through 'theta' radians.

Parameters
----------
theta : float
    Angle of rotation (rad)
axis : Point()
    Vector (x=i,y=j,z=k) indicating direction of axis for rotation
axloc : Point()
    Location of the axis origin
pt : Point()
    Location of the point that is to be rotated

Returns
-----------
Point
    Point after rotation
```

2.5.2.18 util_transform_to_local()

```
pysoltrace.PySolTrace.util_transform_to_local (
              self,
             numpy.array posref,
             numpy.array cosref,
             numpy.array origin,
             numpy.array rreftoloc )
```

```
Perform coordinate transformation from reference system to local system.

Parameters
----------
PosRef : numpy.array([float,]*3)
    X,Y,Z coordinates of ray point in reference system
CosRef : numpy.array([float,]*3)
    Direction cosines of ray in reference system
Origin : numpy.array([float,]*3)
    X,Y,Z coordinates of origin of local system as measured in reference system
RRefToLoc : numpy.array([float,]*3)
    Rotation matrices required for coordinate transform from reference to local

Returns
----------
(dict)  Keys in return dictionary include:
    posloc : ([float,]*3) X,Y,Z coordinates of ray point in local system
    cosloc : ([float,]*3) Direction cosines of ray in local system
```

2.5.2.19 util_transform_to_ref()

```
pysoltrace.PySolTrace.util_transform_to_ref (
              self,
              posloc,
              cosloc,
              origin,
              rloctoref )
```

```
Perform coordinate transformation from local system to reference system.

Parameters
----------
PosLoc : [float,]*3
    X,Y,Z coordinates of ray point in local system
CosLoc : [float,]*3
    Direction cosines of ray in local system
Origin : [float,]*3
    X,Y,Z coordinates of origin of local system as measured in reference system
RLocToRef
    Rotation matrices required for coordinate transform from local to reference
    -- inverse of reference to local transformation

Returns
----------
dict
    Keys in return dictionary include:
    posref : ([float,]*3) X,Y,Z coordinates of ray point in reference system
    cosref : ([float,]*3) Direction cosines of ray in reference system
```

2.5.2.20 write_soltrace_input_file()

```
pysoltrace.PySolTrace.write_soltrace_input_file (
              self,
             str path )
```

```
Write a SolTrace input file (.stinput) based on the currently created API objects. This file is written
using the objects and data in the PySolTrace instance, not necessarily on what has been created in the
coretrace 'context' data space. The 'context' may not match the PySolTrace instance if not all 'Create()'
methods have been called.

Parameters
==========
path : str
    Path and file name to be used to write the resulting .stinput file

Returns
=======
None
```

2.5.3 Member Data Documentation

2.5.3.1 sun

```
pysoltrace.PySolTrace.sun
```

sun

Object containing Sun class data.

The documentation for this class was generated from the following file:

- pysoltrace.py

2.6 pysoltrace.PySolTrace.Stage Class Reference

Classes

- class Element

Public Member Functions

- __init__ (self, int id)
- copy (self, snew)
- int Create (self, pdll, p_data)
- int add_element (self)

Public Attributes

- **id**

  Identifying integer associated with the stage.
- **position**

  Stage location in global coordinates.
- **aim**

  Coordinate system aim point in global coordinates.
- **zrot**

  [deg] Rotation of coordinate system around z-axis
- **is_virtual**

  Flag indicating virtual stage.
- **is_multihit**

  Flag indicating that rays can have multiple interactions within a single stage.
- **is_tracethrough**

  Flag indicating the stage is in trace-through mode.
- **name**

  Descriptive name for this stage.
- **elements**

  list of all elements in the stage

2.6.1 Detailed Description

```
 *Stage* is a subclass of PySolTrace, and represents a grouping of elements.
A PySolTrace instance may have multiple Stage member instances, which are stored in
the PySolTrace.stages list.

Stage contains a subclass *Element*, which collects properties and geometry associated
with individual geometric elements.

Attributes
----------
id : int
    Identifying integer associated with the stage
position : Point
    Stage location in global coordinates
aim : Point
    Coordinate system aim point in global coordinates
zrot : float
    [deg] Rotation of coordinate system around z-axis
is_virtual : bool
    Flag indicating virtual stage
is_multihit : bool
    Flag indicating that rays can have multiple interactions within a single stage.
is_tracethrough : bool
    Flag indicating the stage is in trace-through mode
name : str
    Descriptive name for this stage
elements : [Stage.Element,]
    list of all elements in the stage

Methods
----------
copy
    Creates a deepcopy of the current Stage instance
Create
    Calls methods to instantiate and construct a stage in the context.
add_elements
    Creates new element in Stage.element[] list
```

2.6.2 Constructor & Destructor Documentation

2.6.2.1 __init__()

```
pysoltrace.PySolTrace.Stage.__init__ (
              self,
             int id )
```

2.6.3 Member Function Documentation

2.6.3.1 add_element()

```
int pysoltrace.PySolTrace.Stage.add_element (
              self )
```

```
Add one element to the stage. This method appends an Element object to the
stage’s Stage.elements list.
To update element properties and settings, call the Element.Create method
on each element.

Returns
----------
PySolTrace.Stage.Element
    Reference to the newly created element
```

2.6.3.2 copy()

```
pysoltrace.PySolTrace.Stage.copy (
              self,
              snew )
```

```
Deep copy of the current Stage instance

Inputs
---------
snew : Stage
    Reference to new Stage object to which data will be copied
```

2.6.3.3 Create()

```
int pysoltrace.PySolTrace.Stage.Create (
              self,
              pdll,
              p_data )
```

```
Create Stage instance in the SolTrace context.
Note: This does not create any associated Elements, which must have their Create method called separately.

Returns
----------
int
    1 if successful, 0 otherwise
```

The documentation for this class was generated from the following file:

- pysoltrace.py

2.7 pysoltrace.PySolTrace.Sun Class Reference

Public Member Functions

- **__init__** (self)
- **copy** (self, snew)
- Create (self, pdll, p_data)

Public Attributes

- **point_source**

  Flag indicating whether the sun is modeled as a point source at a finite distance.
- shape

  Sun shape model.
- sigma

  [mrad] Half-width or std.
- **position**

  Location of the sun/sun vector in global coordinates.
- user_intensity_table

  [mrad, 0..1] 2D list containing pairs of angle deviation from sun vector and irradiation intensity.

2.7.1 Detailed Description

```
 *Sun* is a subclass of PySolTrace, and represents a sun property set.
A PySolTrace instance may have a single Sun member instance, which is stored as the
PySolTrace.sun member.

Attributes
----------
point_source : bool
    Flag indicating whether the sun is modeled as a point source at a finite distance.
shape : char
    Sun shape model. One of: {'p':Pillbox, 'g':Gaussian, 'd':data table, 'f':gray diffuse}
sigma : float
    [mrad] Half-width or std. dev. of the error distribution
position : Point
    Location of the sun/sun vector in global coordinates
user_intensity_table : [[float,float],]
    [mrad, 0..1] 2D list containing pairs of
    angle deviation from sun vector and irradiation intensity.
    A typical table will have angles spanning 0->~5mrad, and inten-
    sities starting at 1 and decreasing to zero. The table must
    contain at least 2 entries.

Methods
----------
copy
    Deep copy of the current Sun instance
Create
    Calls methods to instantiate and construct optical surface in the SolTrace context.
```

2.7.2 Member Function Documentation

2.7.2.1 Create()

```
pysoltrace.PySolTrace.Sun.Create (
              self,
              pdll,
              p_data )
```

```
Create Sun instance in the SolTrace context.

Returns
----------
int
    1 if successful, 0 otherwise
```

2.7.3 Member Data Documentation

2.7.3.1 shape

```
pysoltrace.PySolTrace.Sun.shape
```

Sun shape model.

One of: {'p':Pillbox, 'g':Gaussian, 'd':data table, 'f':gray diffuse}

2.7.3.2 sigma

```
pysoltrace.PySolTrace.Sun.sigma
```

[mrad] Half-width or std.

dev. of the error distribution

2.7.3.3 user_intensity_table

```
pysoltrace.PySolTrace.Sun.user_intensity_table
```

[mrad, 0..1] 2D list containing pairs of angle deviation from sun vector and irradiation intensity.

A typical table will have angles spanning 0->~5mrad, and inten- sities starting at 1 and decreasing to zero. The table must contain at least 2 entries.

The documentation for this class was generated from the following file:

- pysoltrace.py

Index

__add__
pysoltrace.Point, 19
__floordiv__
pysoltrace.Point, 19
__init__
pysoltrace.Point, 19
pysoltrace.PySolTrace.Stage, 33
__mul__
pysoltrace.Point, 19
__sub__
pysoltrace.Point, 20
__truediv__
pysoltrace.Point, 20

add_element
pysoltrace.PySolTrace.Stage, 33
add_optic
pysoltrace.PySolTrace, 24
add_stage
pysoltrace.PySolTrace, 24
add_sun
pysoltrace.PySolTrace, 24
aperture_annulus
pysoltrace.PySolTrace.Stage.Element, 5
aperture_circle
pysoltrace.PySolTrace.Stage.Element, 5
aperture_hexagon
pysoltrace.PySolTrace.Stage.Element, 6
aperture_irr_triangle
pysoltrace.PySolTrace.Stage.Element, 6
aperture_quadrilateral
pysoltrace.PySolTrace.Stage.Element, 7
aperture_rectangle
pysoltrace.PySolTrace.Stage.Element, 7
aperture_singleax_curve
pysoltrace.PySolTrace.Stage.Element, 8
aperture_triangle
pysoltrace.PySolTrace.Stage.Element, 8
as_list
pysoltrace.Point, 20

copy
pysoltrace.PySolTrace, 24
pysoltrace.PySolTrace.Optics, 17
pysoltrace.PySolTrace.Stage, 34
pysoltrace.PySolTrace.Stage.Element, 9
Create
pysoltrace.PySolTrace, 25
pysoltrace.PySolTrace.Optics, 17
pysoltrace.PySolTrace.Stage, 34
pysoltrace.PySolTrace.Stage.Element, 9
pysoltrace.PySolTrace.Sun, 35

delete_optic
pysoltrace.PySolTrace, 25
delete_stage
pysoltrace.PySolTrace, 25
dist_type
pysoltrace.PySolTrace.Optics.Face, 16

interaction
pysoltrace.PySolTrace.Stage.Element, 14

plot_flux
pysoltrace.PySolTrace, 25
plot_trace
pysoltrace.PySolTrace, 26
pysoltrace.Point, 18
__add__, 19
__floordiv__, 19
__init__, 19
__mul__, 19
__sub__, 20
__truediv__, 20
as_list, 20
radius, 21
unitize, 21
pysoltrace.PySolTrace, 21
add_optic, 24
add_stage, 24
add_sun, 24
copy, 24
Create, 25
delete_optic, 25
delete_stage, 25
plot_flux, 25
plot_trace, 26
run, 27
sun, 31
util_calc_euler_angles, 27
util_calc_transforms, 28
util_calc_unitvect, 28

util_calc_zrot_azel, 28
util_matrix_transpose, 29
util_matrix_vector_mult, 29
util_rotation_arbitrary, 29
util_transform_to_local, 30
util_transform_to_ref, 30
write_soltrace_input_file, 31
pysoltrace.PySolTrace.Optics, 16
copy, 17
Create, 17
pysoltrace.PySolTrace.Optics.Face, 14
dist_type, 16
refltable, 16
transtable, 16
pysoltrace.PySolTrace.Stage, 32
__init__, 33
add_element, 33
copy, 34
Create, 34
pysoltrace.PySolTrace.Stage.Element, 3
aperture_annulus, 5
aperture_circle, 5
aperture_hexagon, 6
aperture_irr_triangle, 6
aperture_quadrilateral, 7
aperture_rectangle, 7
aperture_singleax_curve, 8
aperture_triangle, 8
copy, 9
Create, 9
interaction, 14
surface_conical, 9
surface_cubicspline, 10
surface_cylindrical, 10
surface_finiteelement, 10
surface_flat, 11
surface_hypellip, 11
surface_parabolic, 11
surface_polynomialrev, 12
surface_spherical, 12
surface_toroid, 13
surface_vshot, 13
surface_zernicke, 13
pysoltrace.PySolTrace.Sun, 34
Create, 35
shape, 36
sigma, 36
user_intensity_table, 36

radius
pysoltrace.Point, 21
refltable
pysoltrace.PySolTrace.Optics.Face, 16
run
pysoltrace.PySolTrace, 27

shape
pysoltrace.PySolTrace.Sun, 36
sigma
pysoltrace.PySolTrace.Sun, 36
sun
pysoltrace.PySolTrace, 31
surface_conical
pysoltrace.PySolTrace.Stage.Element, 9
surface_cubicspline
pysoltrace.PySolTrace.Stage.Element, 10
surface_cylindrical
pysoltrace.PySolTrace.Stage.Element, 10
surface_finiteelement
pysoltrace.PySolTrace.Stage.Element, 10
surface_flat
pysoltrace.PySolTrace.Stage.Element, 11
surface_hypellip
pysoltrace.PySolTrace.Stage.Element, 11
surface_parabolic
pysoltrace.PySolTrace.Stage.Element, 11
surface_polynomialrev
pysoltrace.PySolTrace.Stage.Element, 12
surface_spherical
pysoltrace.PySolTrace.Stage.Element, 12
surface_toroid
pysoltrace.PySolTrace.Stage.Element, 13
surface_vshot
pysoltrace.PySolTrace.Stage.Element, 13
surface_zernicke
pysoltrace.PySolTrace.Stage.Element, 13

transtable
pysoltrace.PySolTrace.Optics.Face, 16

unitize
pysoltrace.Point, 21
user_intensity_table
pysoltrace.PySolTrace.Sun, 36
util_calc_euler_angles
pysoltrace.PySolTrace, 27
util_calc_transforms
pysoltrace.PySolTrace, 28
util_calc_unitvect
pysoltrace.PySolTrace, 28
util_calc_zrot_azel
pysoltrace.PySolTrace, 28
util_matrix_transpose
pysoltrace.PySolTrace, 29
util_matrix_vector_mult
pysoltrace.PySolTrace, 29
util_rotation_arbitrary
pysoltrace.PySolTrace, 29
util_transform_to_local

pysoltrace.PySolTrace, 30
util_transform_to_ref
pysoltrace.PySolTrace, 30

write_soltrace_input_file
pysoltrace.PySolTrace, 31