Python Help Documentation
Printed and PDF'd for reference
raserppsprograms@gmail.com

To see this info within IDLE,
Open a Shell Window and Enter

```
help()
```

then enter:

```
pygame
```

DESCRIPTION
Pygame is a set of Python modules designed for writing games.
It is written on top of the excellent SDL library. This allows you
to create fully featured games and multimedia programs in the python
language. The package is highly portable, with games running on
Windows, MacOS, OS X, BeOS, FreeBSD, IRIX, and Linux.

PACKAGE CONTENTS
_camera
_camera_opencv_highgui
_camera_vidcapture
_dummybackend
_freetype
_numericsndarray
_numericsurfarray
_numpysndarray
_numpysurfarray
_vlcbackend
base
bufferproxy
camera
cdrom
color
colordict
compat
constants
cursors
display
docs (package)
draw
event
examples (package)
fastevent
font
freetype
ftfont
gfxdraw
gp2x (package)
image

imageext
joystick
key
locals
macosx
mask
math
midi
mixer
mixer_music
mouse
newbuffer
nmovie
overlay
pixelarray
pixelcopy
pkgdata
pypm
rect
rwobject
scrap
sndarray
sprite
surface
surfarray
surflock
sysfont
tests (package)
threads (package)
time
transform
version

CLASSES
__builtin__.object
BufferProxy
Color
PixelArray
Rect
Surface
overlay
exceptions.BufferError(exceptions.StandardError)
BufferError
exceptions.RuntimeError(exceptions.StandardError)
error

class BufferError(exceptions.BufferError)
| Method resolution order:
| BufferError
| exceptions.BufferError
| exceptions.StandardError
| exceptions.Exception

```
|  exceptions.BaseException
|  __builtin__.object
|
|  Data descriptors defined here:
|
|  __weakref__
|  list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  Methods inherited from exceptions.BufferError:
|
|  __init__(...)
|  x.__init__(...) initializes x; see help(type(x)) for signature
|
|  ----------------------------------------------------------------------
|  Data and other attributes inherited from exceptions.BufferError:
|
|  __new__ = <built-in method __new__ of type object>
|  T.__new__(S, ...) -> a new object with type S, a subtype of T
|
|  ----------------------------------------------------------------------
|  Methods inherited from exceptions.BaseException:
|
|  __delattr__(...)
|  x.__delattr__('name') <==> del x.name
|
|  __getattribute__(...)
|  x.__getattribute__('name') <==> x.name
|
|  __getitem__(...)
|  x.__getitem__(y) <==> x[y]
|
|  __getslice__(...)
|  x.__getslice__(i, j) <==> x[i:j]
|
|  Use of negative indices is not supported.
|
|  __reduce__(...)
|
|  __repr__(...)
|  x.__repr__() <==> repr(x)
|
|  __setattr__(...)
|  x.__setattr__('name', value) <==> x.name = value
|
|  __setstate__(...)
|
|  __str__(...)
|  x.__str__() <==> str(x)
|
|  __unicode__(...)
|
```

```
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from exceptions.BaseException:
 |
 |  __dict__
 |
 |  args
 |
 |  message

class BufferProxy(__builtin__.object)
 |  BufferProxy(<parent>) -> BufferProxy
 |  pygame object to export a surface buffer through an array protocol
 |
 |  Methods defined here:
 |
 |  __repr__(...)
 |      x.__repr__() <==> repr(x)
 |
 |  write(...)
 |      write(buffer, offset=0)
 |      Write raw bytes to object buffer.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __array_interface__
 |      Version 3 array interface, Python level
 |
 |  __array_struct__
 |      Version 3 array interface, C level
 |
 |  __dict__
 |      The object's attribute dictionary, read-only
 |
 |  length
 |      length -> int
 |      The size, in bytes, of the exported buffer.
 |
 |  parent
 |      parent -> Surface
 |      parent -> <parent>
 |      Return wrapped exporting object.
 |
 |  raw
 |      raw -> bytes
 |      A copy of the exported buffer as a single block of bytes.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __new__ = <built-in method __new__ of type object>
 |      T.__new__(S, ...) -> a new object with type S, a subtype of T
```

```
class Color(__builtin__.object)
 |  Color(name) -> Color
 |  Color(r, g, b, a) -> Color
 |  Color(rgbvalue) -> Color
 |  pygame object for color representations
 |
 |  Methods defined here:
 |
 |  __add__(...)
 |      x.__add__(y) <==> x+y
 |
 |  __coerce__(...)
 |      x.__coerce__(y) <==> coerce(x, y)
 |
 |  __delitem__(...)
 |      x.__delitem__(y) <==> del x[y]
 |
 |  __div__(...)
 |      x.__div__(y) <==> x/y
 |
 |  __eq__(...)
 |      x.__eq__(y) <==> x==y
 |
 |  __float__(...)
 |      x.__float__() <==> float(x)
 |
 |  __floordiv__(...)
 |      x.__floordiv__(y) <==> x//y
 |
 |  __ge__(...)
 |      x.__ge__(y) <==> x>=y
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __getslice__(...)
 |      x.__getslice__(i, j) <==> x[i:j]
 |
 |      Use of negative indices is not supported.
 |
 |  __gt__(...)
 |      x.__gt__(y) <==> x>y
 |
 |  __hex__(...)
 |      x.__hex__() <==> hex(x)
 |
 |  __index__(...)
 |      x[y:z] <==> x[y.__index__():z.__index__()]
 |
 |  __int__(...)
 |      x.__int__() <==> int(x)
```

|
| __invert__(...)
| x.__invert__() <==> ~x
|
| __le__(...)
| x.__le__(y) <==> x<=y
|
| __len__(...)
| x.__len__() <==> len(x)
|
| __long__(...)
| x.__long__() <==> long(x)
|
| __lt__(...)
| x.__lt__(y) <==> x<y
|
| __mod__(...)
| x.__mod__(y) <==> x%y
|
| __mul__(...)
| x.__mul__(y) <==> x*y
|
| __ne__(...)
| x.__ne__(y) <==> x!=y
|
| __oct__(...)
| x.__oct__() <==> oct(x)
|
| __radd__(...)
| x.__radd__(y) <==> y+x
|
| __rdiv__(...)
| x.__rdiv__(y) <==> y/x
|
| __repr__(...)
| x.__repr__() <==> repr(x)
|
| __rfloordiv__(...)
| x.__rfloordiv__(y) <==> y//x
|
| __rmod__(...)
| x.__rmod__(y) <==> y%x
|
| __rmul__(...)
| x.__rmul__(y) <==> y*x
|
| __rsub__(...)
| x.__rsub__(y) <==> y-x
|
| __setitem__(...)
| x.__setitem__(i, y) <==> x[i]=y
|

```
|  __sub__(...)
|      x.__sub__(y) <==> x-y
|
|  correct_gamma(...)
|      correct_gamma (gamma) -> Color
|      Applies a certain gamma value to the Color.
|
|  normalize(...)
|      normalize() -> tuple
|      Returns the normalized RGBA values of the Color.
|
|  set_length(...)
|      set_length(len) -> None
|      Set the number of elements in the Color to 1,2,3, or 4.
|
|  ----------------------------------------------------------------------
|  Data descriptors defined here:
|
|  __array_struct__
|      array structure interface, read only
|
|  a
|      a -> int
|      Gets or sets the alpha value of the Color.
|
|  b
|      b -> int
|      Gets or sets the blue value of the Color.
|
|  cmy
|      cmy -> tuple
|      Gets or sets the CMY representation of the Color.
|
|  g
|      g -> int
|      Gets or sets the green value of the Color.
|
|  hsla
|      hsla -> tuple
|      Gets or sets the HSLA representation of the Color.
|
|  hsva
|      hsva -> tuple
|      Gets or sets the HSVA representation of the Color.
|
|  i1i2i3
|      i1i2i3 -> tuple
|      Gets or sets the I1I2I3 representation of the Color.
|
|  r
|      r -> int
|      Gets or sets the red value of the Color.
```

```
 |
 | ----------------------------------------------------------------------
 | Data and other attributes defined here:
 |
 | __new__ = <built-in method __new__ of type object>
 |     T.__new__(S, ...) -> a new object with type S, a subtype of T

Overlay = class overlay(__builtin__.object)
 | Overlay(format, (width, height)) -> Overlay
 | pygame object for video overlay graphics
 |
 | Methods defined here:
 |
 | display(...)
 |     display((y, u, v)) -> None
 |     display() -> None
 |     set the overlay pixel data
 |
 | get_hardware(...)
 |     get_hardware(rect) -> int
 |     test if the Overlay is hardware accelerated
 |
 | set_location(...)
 |     set_location(rect) -> None
 |     control where the overlay is displayed
 |
 | ----------------------------------------------------------------------
 | Data and other attributes defined here:
 |
 | __new__ = <built-in method __new__ of type object>
 |     T.__new__(S, ...) -> a new object with type S, a subtype of T

class PixelArray(__builtin__.object)
 | PixelArray(Surface) -> PixelArray
 | pygame object for direct pixel access of surfaces
 |
 | Methods defined here:
 |
 | __contains__(...)
 |     x.__contains__(y) <==> y in x
 |
 | __delitem__(...)
 |     x.__delitem__(y) <==> del x[y]
 |
 | __getitem__(...)
 |     x.__getitem__(y) <==> x[y]
 |
 | __iter__(...)
 |     x.__iter__() <==> iter(x)
 |
 | __len__(...)
 |     x.__len__() <==> len(x)
```

```
|
| __repr__(...)
|     x.__repr__() <==> repr(x)
|
| __setitem__(...)
|     x.__setitem__(i, y) <==> x[i]=y
|
| compare(...)
|     compare(array, distance=0, weights=(0.299, 0.587, 0.114)) -> PixelArray
|     Compares the PixelArray with another one.
|
| extract(...)
|     extract(color, distance=0, weights=(0.299, 0.587, 0.114)) -> PixelArray
|     Extracts the passed color from the PixelArray.
|
| make_surface(...)
|     make_surface() -> Surface
|     Creates a new Surface from the current PixelArray.
|
| replace(...)
|     replace(color, repcolor, distance=0, weights=(0.299, 0.587, 0.114)) -> None
|     Replaces the passed color in the PixelArray with another one.
|
| transpose(...)
|     transpose() -> PixelArray
|     Exchanges the x and y axis.
|
| ----------------------------------------------------------------------
| Data descriptors defined here:
|
| __array_interface__
|     Version 3
|
| __array_struct__
|     Version 3
|
| __dict__
|
| itemsize
|     itemsize -> int
|     Returns the byte size of a pixel array item
|
| ndim
|     ndim -> int
|     Returns the number of dimensions.
|
| shape
|     shape -> tuple of int's
|     Returns the array size.
|
| strides
|     strides -> tuple of int's
```

```
|  Returns byte offsets for each array dimension.
|
|  surface
|      surface -> Surface
|      Gets the Surface the PixelArray uses.
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __new__ = <built-in method __new__ of type object>
|      T.__new__(S, ...) -> a new object with type S, a subtype of T

class Rect(__builtin__.object)
|  Rect(left, top, width, height) -> Rect
|  Rect((left, top), (width, height)) -> Rect
|  Rect(object) -> Rect
|  pygame object for storing rectangular coordinates
|
|  Methods defined here:
|
|  __coerce__(...)
|      x.__coerce__(y) <==> coerce(x, y)
|
|  __copy__(...)
|
|  __delitem__(...)
|      x.__delitem__(y) <==> del x[y]
|
|  __delslice__(...)
|      x.__delslice__(i, j) <==> del x[i:j]
|
|      Use of negative indices is not supported.
|
|  __eq__(...)
|      x.__eq__(y) <==> x==y
|
|  __ge__(...)
|      x.__ge__(y) <==> x>=y
|
|  __getitem__(...)
|      x.__getitem__(y) <==> x[y]
|
|  __getslice__(...)
|      x.__getslice__(i, j) <==> x[i:j]
|
|      Use of negative indices is not supported.
|
|  __gt__(...)
|      x.__gt__(y) <==> x>y
|
|  __init__(...)
|      x.__init__(...) initializes x; see help(type(x)) for signature
```

```
|
| __le__(...)
|     x.__le__(y) <==> x<=y
|
| __len__(...)
|     x.__len__() <==> len(x)
|
| __lt__(...)
|     x.__lt__(y) <==> x<y
|
| __ne__(...)
|     x.__ne__(y) <==> x!=y
|
| __nonzero__(...)
|     x.__nonzero__() <==> x != 0
|
| __reduce__(...)
|
| __repr__(...)
|     x.__repr__() <==> repr(x)
|
| __setitem__(...)
|     x.__setitem__(i, y) <==> x[i]=y
|
| __setslice__(...)
|     x.__setslice__(i, j, y) <==> x[i:j]=y
|
|     Use  of negative indices is not supported.
|
| __str__(...)
|     x.__str__() <==> str(x)
|
| clamp(...)
|     clamp(Rect) -> Rect
|     moves the rectangle inside another
|
| clamp_ip(...)
|     clamp_ip(Rect) -> None
|     moves the rectangle inside another, in place
|
| clip(...)
|     clip(Rect) -> Rect
|     crops a rectangle inside another
|
| collidedict(...)
|     collidedict(dict) -> (key, value)
|     test if one rectangle in a dictionary intersects
|
| collidedictall(...)
|     collidedictall(dict) -> [(key, value), ...]
|     test if all rectangles in a dictionary intersect
|
```

```
| collidelist(...)
|     collidelist(list) -> index
|     test if one rectangle in a list intersects
|
| collidelistall(...)
|     collidelistall(list) -> indices
|     test if all rectangles in a list intersect
|
| collidepoint(...)
|     collidepoint(x, y) -> bool
|     collidepoint((x,y)) -> bool
|     test if a point is inside a rectangle
|
| colliderect(...)
|     colliderect(Rect) -> bool
|     test if two rectangles overlap
|
| contains(...)
|     contains(Rect) -> bool
|     test if one rectangle is inside another
|
| copy(...)
|     copy() -> Rect
|     copy the rectangle
|
| fit(...)
|     fit(Rect) -> Rect
|     resize and move a rectangle with aspect ratio
|
| inflate(...)
|     inflate(x, y) -> Rect
|     grow or shrink the rectangle size
|
| inflate_ip(...)
|     inflate_ip(x, y) -> None
|     grow or shrink the rectangle size, in place
|
| move(...)
|     move(x, y) -> Rect
|     moves the rectangle
|
| move_ip(...)
|     move_ip(x, y) -> None
|     moves the rectangle, in place
|
| normalize(...)
|     normalize() -> None
|     correct negative sizes
|
| union(...)
|     union(Rect) -> Rect
|     joins two rectangles into one
```

```
|
| union_ip(...)
|     union_ip(Rect) -> None
|     joins two rectangles into one, in place
|
| unionall(...)
|     unionall(Rect_sequence) -> Rect
|     the union of many rectangles
|
| unionall_ip(...)
|     unionall_ip(Rect_sequence) -> None
|     the union of many rectangles, in place
|
| ----------------------------------------------------------------------
| Data descriptors defined here:
|
| __safe_for_unpickling__
|
| bottom
|
| bottomleft
|
| bottomright
|
| center
|
| centerx
|
| centery
|
| h
|
| height
|
| left
|
| midbottom
|
| midleft
|
| midright
|
| midtop
|
| right
|
| size
|
| top
|
| topleft
|
```

```
 |  topright
 |
 |  w
 |
 |  width
 |
 |  x
 |
 |  y
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __new__ = <built-in method __new__ of type object>
 |      T.__new__(S, ...) -> a new object with type S, a subtype of T

class Surface(__builtin__.object)
 |  Surface((width, height), flags=0, depth=0, masks=None) -> Surface
 |  Surface((width, height), flags=0, Surface) -> Surface
 |  pygame object for representing images
 |
 |  Methods defined here:
 |
 |  __copy__(...)
 |      copy() -> Surface
 |      create a new copy of a Surface
 |
 |  __init__(...)
 |      x.__init__(...) initializes x; see help(type(x)) for signature
 |
 |  __repr__(...)
 |      x.__repr__() <==> repr(x)
 |
 |  blit(...)
 |      blit(source, dest, area=None, special_flags = 0) -> Rect
 |      draw one image onto another
 |
 |  convert(...)
 |      convert(Surface) -> Surface
 |      convert(depth, flags=0) -> Surface
 |      convert(masks, flags=0) -> Surface
 |      convert() -> Surface
 |      change the pixel format of an image
 |
 |  convert_alpha(...)
 |      convert_alpha(Surface) -> Surface
 |      convert_alpha() -> Surface
 |      change the pixel format of an image including per pixel alphas
 |
 |  copy(...)
 |      copy() -> Surface
 |      create a new copy of a Surface
```

```
|
| fill(...)
|     fill(color, rect=None, special_flags=0) -> Rect
|     fill Surface with a solid color
|
| get_abs_offset(...)
|     get_abs_offset() -> (x, y)
|     find the absolute position of a child subsurface inside its top level parent
|
| get_abs_parent(...)
|     get_abs_parent() -> Surface
|     find the top level parent of a subsurface
|
| get_alpha(...)
|     get_alpha() -> int_value or None
|     get the current Surface transparency value
|
| get_at(...)
|     get_at((x, y)) -> Color
|     get the color value at a single pixel
|
| get_at_mapped(...)
|     get_at_mapped((x, y)) -> Color
|     get the mapped color value at a single pixel
|
| get_bitsize(...)
|     get_bitsize() -> int
|     get the bit depth of the Surface pixel format
|
| get_bounding_rect(...)
|     get_bounding_rect(min_alpha = 1) -> Rect
|     find the smallest rect containing data
|
| get_buffer(...)
|     get_buffer() -> BufferProxy
|     acquires a buffer object for the pixels of the Surface.
|
| get_bytesize(...)
|     get_bytesize() -> int
|     get the bytes used per Surface pixel
|
| get_clip(...)
|     get_clip() -> Rect
|     get the current clipping area of the Surface
|
| get_colorkey(...)
|     get_colorkey() -> RGB or None
|     Get the current transparent colorkey
|
| get_flags(...)
|     get_flags() -> int
|     get the additional flags used for the Surface
```

```
|
| get_height(...)
|     get_height() -> height
|     get the height of the Surface
|
| get_locked(...)
|     get_locked() -> bool
|     test if the Surface is current locked
|
| get_locks(...)
|     get_locks() -> tuple
|     Gets the locks for the Surface
|
| get_losses(...)
|     get_losses() -> (R, G, B, A)
|     the significant bits used to convert between a color and a mapped integer
|
| get_masks(...)
|     get_masks() -> (R, G, B, A)
|     the bitmasks needed to convert between a color and a mapped integer
|
| get_offset(...)
|     get_offset() -> (x, y)
|     find the position of a child subsurface inside a parent
|
| get_palette(...)
|     get_palette() -> [RGB, RGB, RGB, ...]
|     get the color index palette for an 8bit Surface
|
| get_palette_at(...)
|     get_palette_at(index) -> RGB
|     get the color for a single entry in a palette
|
| get_parent(...)
|     get_parent() -> Surface
|     find the parent of a subsurface
|
| get_pitch(...)
|     get_pitch() -> int
|     get the number of bytes used per Surface row
|
| get_rect(...)
|     get_rect(**kwargs) -> Rect
|     get the rectangular area of the Surface
|
| get_shifts(...)
|     get_shifts() -> (R, G, B, A)
|     the bit shifts needed to convert between a color and a mapped integer
|
| get_size(...)
|     get_size() -> (width, height)
|     get the dimensions of the Surface
```

```
|
| get_view(...)
|     get_view(<kind>='2') -> BufferProxy
|     return a buffer view of the Surface's pixels.
|
| get_width(...)
|     get_width() -> width
|     get the width of the Surface
|
| lock(...)
|     lock() -> None
|     lock the Surface memory for pixel access
|
| map_rgb(...)
|     map_rgb(Color) -> mapped_int
|     convert a color into a mapped color value
|
| mustlock(...)
|     mustlock() -> bool
|     test if the Surface requires locking
|
| scroll(...)
|     scroll(dx=0, dy=0) -> None
|     Shift the surface image in place
|
| set_alpha(...)
|     set_alpha(value, flags=0) -> None
|     set_alpha(None) -> None
|     set the alpha value for the full Surface image
|
| set_at(...)
|     set_at((x, y), Color) -> None
|     set the color value for a single pixel
|
| set_clip(...)
|     set_clip(rect) -> None
|     set_clip(None) -> None
|     set the current clipping area of the Surface
|
| set_colorkey(...)
|     set_colorkey(Color, flags=0) -> None
|     set_colorkey(None) -> None
|     Set the transparent colorkey
|
| set_masks(...)
|     set_masks((r,g,b,a)) -> None
|     set the bitmasks needed to convert between a color and a mapped integer
|
| set_palette(...)
|     set_palette([RGB, RGB, RGB, ...]) -> None
|     set the color palette for an 8bit Surface
|
```

```
 |  set_palette_at(...)
 |      set_palette_at(index, RGB) -> None
 |      set the color for a single index in an 8bit Surface palette
 |
 |  set_shifts(...)
 |      set_shifts((r,g,b,a)) -> None
 |      sets the bit shifts needed to convert between a color and a mapped integer
 |
 |  subsurface(...)
 |      subsurface(Rect) -> Surface
 |      create a new surface that references its parent
 |
 |  unlock(...)
 |      unlock() -> None
 |      unlock the Surface memory from pixel access
 |
 |  unmap_rgb(...)
 |      unmap_rgb(mapped_int) -> Color
 |      convert a mapped integer color value into a Color
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __new__ = <built-in method __new__ of type object>
 |      T.__new__(S, ...) -> a new object with type S, a subtype of T

SurfaceType = class Surface(__builtin__.object)
 |  Surface((width, height), flags=0, depth=0, masks=None) -> Surface
 |  Surface((width, height), flags=0, Surface) -> Surface
 |  pygame object for representing images
 |
 |  Methods defined here:
 |
 |  __copy__(...)
 |      copy() -> Surface
 |      create a new copy of a Surface
 |
 |  __init__(...)
 |      x.__init__(...) initializes x; see help(type(x)) for signature
 |
 |  __repr__(...)
 |      x.__repr__() <==> repr(x)
 |
 |  blit(...)
 |      blit(source, dest, area=None, special_flags = 0) -> Rect
 |      draw one image onto another
 |
 |  convert(...)
 |      convert(Surface) -> Surface
 |      convert(depth, flags=0) -> Surface
 |      convert(masks, flags=0) -> Surface
 |      convert() -> Surface
```

```
|     change the pixel format of an image
|
| convert_alpha(...)
|     convert_alpha(Surface) -> Surface
|     convert_alpha() -> Surface
|     change the pixel format of an image including per pixel alphas
|
| copy(...)
|     copy() -> Surface
|     create a new copy of a Surface
|
| fill(...)
|     fill(color, rect=None, special_flags=0) -> Rect
|     fill Surface with a solid color
|
| get_abs_offset(...)
|     get_abs_offset() -> (x, y)
|     find the absolute position of a child subsurface inside its top level parent
|
| get_abs_parent(...)
|     get_abs_parent() -> Surface
|     find the top level parent of a subsurface
|
| get_alpha(...)
|     get_alpha() -> int_value or None
|     get the current Surface transparency value
|
| get_at(...)
|     get_at((x, y)) -> Color
|     get the color value at a single pixel
|
| get_at_mapped(...)
|     get_at_mapped((x, y)) -> Color
|     get the mapped color value at a single pixel
|
| get_bitsize(...)
|     get_bitsize() -> int
|     get the bit depth of the Surface pixel format
|
| get_bounding_rect(...)
|     get_bounding_rect(min_alpha = 1) -> Rect
|     find the smallest rect containing data
|
| get_buffer(...)
|     get_buffer() -> BufferProxy
|     acquires a buffer object for the pixels of the Surface.
|
| get_bytesize(...)
|     get_bytesize() -> int
|     get the bytes used per Surface pixel
|
| get_clip(...)
```

```
|     get_clip() -> Rect
|     get the current clipping area of the Surface
|
| get_colorkey(...)
|     get_colorkey() -> RGB or None
|     Get the current transparent colorkey
|
| get_flags(...)
|     get_flags() -> int
|     get the additional flags used for the Surface
|
| get_height(...)
|     get_height() -> height
|     get the height of the Surface
|
| get_locked(...)
|     get_locked() -> bool
|     test if the Surface is current locked
|
| get_locks(...)
|     get_locks() -> tuple
|     Gets the locks for the Surface
|
| get_losses(...)
|     get_losses() -> (R, G, B, A)
|     the significant bits used to convert between a color and a mapped integer
|
| get_masks(...)
|     get_masks() -> (R, G, B, A)
|     the bitmasks needed to convert between a color and a mapped integer
|
| get_offset(...)
|     get_offset() -> (x, y)
|     find the position of a child subsurface inside a parent
|
| get_palette(...)
|     get_palette() -> [RGB, RGB, RGB, ...]
|     get the color index palette for an 8bit Surface
|
| get_palette_at(...)
|     get_palette_at(index) -> RGB
|     get the color for a single entry in a palette
|
| get_parent(...)
|     get_parent() -> Surface
|     find the parent of a subsurface
|
| get_pitch(...)
|     get_pitch() -> int
|     get the number of bytes used per Surface row
|
| get_rect(...)
```

```
|      get_rect(**kwargs) -> Rect
|      get the rectangular area of the Surface
|
|  get_shifts(...)
|      get_shifts() -> (R, G, B, A)
|      the bit shifts needed to convert between a color and a mapped integer
|
|  get_size(...)
|      get_size() -> (width, height)
|      get the dimensions of the Surface
|
|  get_view(...)
|      get_view(<kind>='2') -> BufferProxy
|      return a buffer view of the Surface's pixels.
|
|  get_width(...)
|      get_width() -> width
|      get the width of the Surface
|
|  lock(...)
|      lock() -> None
|      lock the Surface memory for pixel access
|
|  map_rgb(...)
|      map_rgb(Color) -> mapped_int
|      convert a color into a mapped color value
|
|  mustlock(...)
|      mustlock() -> bool
|      test if the Surface requires locking
|
|  scroll(...)
|      scroll(dx=0, dy=0) -> None
|      Shift the surface image in place
|
|  set_alpha(...)
|      set_alpha(value, flags=0) -> None
|      set_alpha(None) -> None
|      set the alpha value for the full Surface image
|
|  set_at(...)
|      set_at((x, y), Color) -> None
|      set the color value for a single pixel
|
|  set_clip(...)
|      set_clip(rect) -> None
|      set_clip(None) -> None
|      set the current clipping area of the Surface
|
|  set_colorkey(...)
|      set_colorkey(Color, flags=0) -> None
|      set_colorkey(None) -> None
```

```
|  Set the transparent colorkey
|
| set_masks(...)
|  set_masks((r,g,b,a)) -> None
|  set the bitmasks needed to convert between a color and a mapped integer
|
| set_palette(...)
|  set_palette([RGB, RGB, RGB, ...]) -> None
|  set the color palette for an 8bit Surface
|
| set_palette_at(...)
|  set_palette_at(index, RGB) -> None
|  set the color for a single index in an 8bit Surface palette
|
| set_shifts(...)
|  set_shifts((r,g,b,a)) -> None
|  sets the bit shifts needed to convert between a color and a mapped integer
|
| subsurface(...)
|  subsurface(Rect) -> Surface
|  create a new surface that references its parent
|
| unlock(...)
|  unlock() -> None
|  unlock the Surface memory from pixel access
|
| unmap_rgb(...)
|  unmap_rgb(mapped_int) -> Color
|  convert a mapped integer color value into a Color
|
| ----------------------------------------------------------------------
| Data and other attributes defined here:
|
| __new__ = <built-in method __new__ of type object>
|  T.__new__(S, ...) -> a new object with type S, a subtype of T

class error(exceptions.RuntimeError)
| Method resolution order:
|  error
|  exceptions.RuntimeError
|  exceptions.StandardError
|  exceptions.Exception
|  exceptions.BaseException
|  __builtin__.object
|
| Data descriptors defined here:
|
| __weakref__
|  list of weak references to the object (if defined)
|
| ----------------------------------------------------------------------
| Methods inherited from exceptions.RuntimeError:
```

```
 |
 | __init__(...)
 |     x.__init__(...) initializes x; see help(type(x)) for signature
 |
 | ----------------------------------------------------------------------
 | Data and other attributes inherited from exceptions.RuntimeError:
 |
 | __new__ = <built-in method __new__ of type object>
 |     T.__new__(S, ...) -> a new object with type S, a subtype of T
 |
 | ----------------------------------------------------------------------
 | Methods inherited from exceptions.BaseException:
 |
 | __delattr__(...)
 |     x.__delattr__('name') <==> del x.name
 |
 | __getattribute__(...)
 |     x.__getattribute__('name') <==> x.name
 |
 | __getitem__(...)
 |     x.__getitem__(y) <==> x[y]
 |
 | __getslice__(...)
 |     x.__getslice__(i, j) <==> x[i:j]
 |
 |     Use of negative indices is not supported.
 |
 | __reduce__(...)
 |
 | __repr__(...)
 |     x.__repr__() <==> repr(x)
 |
 | __setattr__(...)
 |     x.__setattr__('name', value) <==> x.name = value
 |
 | __setstate__(...)
 |
 | __str__(...)
 |     x.__str__() <==> str(x)
 |
 | __unicode__(...)
 |
 | ----------------------------------------------------------------------
 | Data descriptors inherited from exceptions.BaseException:
 |
 | __dict__
 |
 | args
 |
 | message

FUNCTIONS
```

Mask(...)
    Mask((width, height)) -> Mask
    pygame object for representing 2d bitmasks

encode_file_path(...)
    encode_file_path([obj [, etype]]) -> bytes or None
    Encode a unicode or bytes object as a file system path

encode_string(...)
    encode_string([obj [, encoding [, errors [, etype]]]]) -> bytes or None
    Encode a unicode or bytes object

get_array_interface(...)
    return an array struct interface as an interface dictionary

get_error(...)
    get_error() -> errorstr
    get the current error message

get_sdl_byteorder(...)
    get_sdl_byteorder() -> int
    get the byte order of SDL

get_sdl_version(...)
    get_sdl_version() -> major, minor, patch
    get the version number of SDL

init(...)
    init() -> (numpass, numfail)
    initialize all imported pygame modules

packager_imports()
    Some additional things that py2app/py2exe will want to see

quit(...)
    quit() -> None
    uninitialize all pygame modules

register_quit(...)
    register_quit(callable) -> None
    register a function to be called when pygame quits

segfault(...)
    crash

set_error(...)
    set_error(error_msg) -> None
    set the current error message

warn_unwanted_files()
    Used to warn about unneeded old files.

DATA
ACTIVEEVENT = 1
ANYFORMAT = 268435456
ASYNCBLIT = 4
AUDIO_S16 = 32784
AUDIO_S16LSB = 32784
AUDIO_S16MSB = 36880
AUDIO_S16SYS = 32784
AUDIO_S8 = 32776
AUDIO_U16 = 16
AUDIO_U16LSB = 16
AUDIO_U16MSB = 4112
AUDIO_U16SYS = 16
AUDIO_U8 = 8
BIG_ENDIAN = 4321
BLEND_ADD = 1
BLEND_MAX = 5
BLEND_MIN = 4
BLEND_MULT = 3
BLEND_PREMULTIPLIED = 17
BLEND_RGBA_ADD = 6
BLEND_RGBA_MAX = 16
BLEND_RGBA_MIN = 9
BLEND_RGBA_MULT = 8
BLEND_RGBA_SUB = 7
BLEND_RGB_ADD = 1
BLEND_RGB_MAX = 5
BLEND_RGB_MIN = 4
BLEND_RGB_MULT = 3
BLEND_RGB_SUB = 2
BLEND_SUB = 2
BUTTON_X1 = 6
BUTTON_X2 = 7
DOUBLEBUF = 1073741824
FULLSCREEN = -2147483648
GL_ACCELERATED_VISUAL = 15
GL_ACCUM_ALPHA_SIZE = 11
GL_ACCUM_BLUE_SIZE = 10
GL_ACCUM_GREEN_SIZE = 9
GL_ACCUM_RED_SIZE = 8
GL_ALPHA_SIZE = 3
GL_BLUE_SIZE = 2
GL_BUFFER_SIZE = 4
GL_DEPTH_SIZE = 6
GL_DOUBLEBUFFER = 5
GL_GREEN_SIZE = 1
GL_MULTISAMPLEBUFFERS = 13
GL_MULTISAMPLESAMPLES = 14
GL_RED_SIZE = 0
GL_STENCIL_SIZE = 7
GL_STEREO = 12
GL_SWAP_CONTROL = 16

HAT_CENTERED = 0
HAT_DOWN = 4
HAT_LEFT = 8
HAT_LEFTDOWN = 12
HAT_LEFTUP = 9
HAT_RIGHT = 2
HAT_RIGHTDOWN = 6
HAT_RIGHTUP = 3
HAT_UP = 1
HAVE_NEWBUF = 1
HWACCEL = 256
HWPALETTE = 536870912
HWSURFACE = 1
IYUV_OVERLAY = 1448433993
JOYAXISMOTION = 7
JOYBALLMOTION = 8
JOYBUTTONDOWN = 10
JOYBUTTONUP = 11
JOYHATMOTION = 9
KEYDOWN = 2
KEYUP = 3
KMOD_ALT = 768
KMOD_CAPS = 8192
KMOD_CTRL = 192
KMOD_LALT = 256
KMOD_LCTRL = 64
KMOD_LMETA = 1024
KMOD_LSHIFT = 1
KMOD_META = 3072
KMOD_MODE = 16384
KMOD_NONE = 0
KMOD_NUM = 4096
KMOD_RALT = 512
KMOD_RCTRL = 128
KMOD_RMETA = 2048
KMOD_RSHIFT = 2
KMOD_SHIFT = 3
K_0 = 48
K_1 = 49
K_2 = 50
K_3 = 51
K_4 = 52
K_5 = 53
K_6 = 54
K_7 = 55
K_8 = 56
K_9 = 57
K_AMPERSAND = 38
K_ASTERISK = 42
K_AT = 64
K_BACKQUOTE = 96
K_BACKSLASH = 92

K_BACKSPACE = 8
K_BREAK = 318
K_CAPSLOCK = 301
K_CARET = 94
K_CLEAR = 12
K_COLON = 58
K_COMMA = 44
K_DELETE = 127
K_DOLLAR = 36
K_DOWN = 274
K_END = 279
K_EQUALS = 61
K_ESCAPE = 27
K_EURO = 321
K_EXCLAIM = 33
K_F1 = 282
K_F10 = 291
K_F11 = 292
K_F12 = 293
K_F13 = 294
K_F14 = 295
K_F15 = 296
K_F2 = 283
K_F3 = 284
K_F4 = 285
K_F5 = 286
K_F6 = 287
K_F7 = 288
K_F8 = 289
K_F9 = 290
K_FIRST = 0
K_GREATER = 62
K_HASH = 35
K_HELP = 315
K_HOME = 278
K_INSERT = 277
K_KP0 = 256
K_KP1 = 257
K_KP2 = 258
K_KP3 = 259
K_KP4 = 260
K_KP5 = 261
K_KP6 = 262
K_KP7 = 263
K_KP8 = 264
K_KP9 = 265
K_KP_DIVIDE = 267
K_KP_ENTER = 271
K_KP_EQUALS = 272
K_KP_MINUS = 269
K_KP_MULTIPLY = 268
K_KP_PERIOD = 266

K_KP_PLUS = 270
K_LALT = 308
K_LAST = 323
K_LCTRL = 306
K_LEFT = 276
K_LEFTBRACKET = 91
K_LEFTPAREN = 40
K_LESS = 60
K_LMETA = 310
K_LSHIFT = 304
K_LSUPER = 311
K_MENU = 319
K_MINUS = 45
K_MODE = 313
K_NUMLOCK = 300
K_PAGEDOWN = 281
K_PAGEUP = 280
K_PAUSE = 19
K_PERIOD = 46
K_PLUS = 43
K_POWER = 320
K_PRINT = 316
K_QUESTION = 63
K_QUOTE = 39
K_QUOTEDBL = 34
K_RALT = 307
K_RCTRL = 305
K_RETURN = 13
K_RIGHT = 275
K_RIGHTBRACKET = 93
K_RIGHTPAREN = 41
K_RMETA = 309
K_RSHIFT = 303
K_RSUPER = 312
K_SCROLLOCK = 302
K_SEMICOLON = 59
K_SLASH = 47
K_SPACE = 32
K_SYSREQ = 317
K_TAB = 9
K_UNDERSCORE = 95
K_UNKNOWN = 0
K_UP = 273
K_a = 97
K_b = 98
K_c = 99
K_d = 100
K_e = 101
K_f = 102
K_g = 103
K_h = 104
K_i = 105

K_j = 106
K_k = 107
K_l = 108
K_m = 109
K_n = 110
K_o = 111
K_p = 112
K_q = 113
K_r = 114
K_s = 115
K_t = 116
K_u = 117
K_v = 118
K_w = 119
K_x = 120
K_y = 121
K_z = 122
LIL_ENDIAN = 1234
MOUSEBUTTONDOWN = 5
MOUSEBUTTONUP = 6
MOUSEMOTION = 4
NOEVENT = 0
NOFRAME = 32
NUMEVENTS = 32
OPENGL = 2
OPENGLBLIT = 10
PREALLOC = 16777216
QUIT = 12
RESIZABLE = 16
RLEACCEL = 16384
RLEACCELOK = 8192
SCRAP_BMP = 'image/bmp'
SCRAP_CLIPBOARD = 0
SCRAP_PBM = 'image/pbm'
SCRAP_PPM = 'image/ppm'
SCRAP_SELECTION = 1
SCRAP_TEXT = 'text/plain'
SRCALPHA = 65536
SRCCOLORKEY = 4096
SWSURFACE = 0
SYSWMEVENT = 13
TIMER_RESOLUTION = 10
USEREVENT = 24
USEREVENT_DROPFILE = 4096
UYVY_OVERLAY = 1498831189
VIDEOEXPOSE = 17
VIDEORESIZE = 16
YUY2_OVERLAY = 844715353
YV12_OVERLAY = 842094169
YVYU_OVERLAY = 1431918169
__version__ = '1.9.2a0'
movie = <MissingModule instance>

```
    rev = ''
    ver = '1.9.2a0'
    vernum = (1, 9, 2)

VERSION
    1.9.2a0
```