|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyOsirix |  |  | pyOsiriX is a python extension to the OsiriX dicom image viewer. The 'osirix' module contains all functions and classes required to interact with OsiriX. It should be noted that, and with good reason, explicit initilisation of most classes is not allowed.   * Generally speaking, the functions contained within should be used to provide access to OsiriX data. * Where exceptions are made this is noted in the documentation for the class. |  |
| currentBrowser | pyOsiriX\_currentBrowser | METH\_NOARGS | Provides access to the OsiriX dicom browser.  Args: None.  Returns:   * BrowserController: The OsiriX dicom browser instance. | Always current? |
| frontmostViewer | pyOsiriX\_frontmostDisplayed2DViewer | METH\_NOARGS | Provides access to the currently selected 2D dicom viewer.  Args: None.  Returns:   * ViewerController: The currently active 2D viewer instance. | Set/get image? |
| getDisplayed2DViewers | pyOsiriX\_getDisplayed2DViewers | METH\_NOARGS | Provides a tuple with each element providing a reference to an open 2D viewer.  Args: None.  Returns:   * A tuple with each element containing a ViewerController instance. | Set/get image? |
| frontmostVRController | pyOsiriX\_frontmostVRController | METH\_NOARGS | Provides access to the currently selected 3D volume render controller.  Args: None.  Returns:   * VRController: The currently active 3D volume controller instance. |  |
| getDisplayedVRControllers | pyOsiriX\_getDisplayedVRControllers | METH\_NOARGS | Provides a tuple with each element providing a reference to an open 3D viewer.  Args: None.  Returns:   * A tuple with each element containing a VRController instance. |  |
| runAlertPanel | (PyCFunction)  pyOsiriX\_runAlertPanel | METH\_VARARGS|METH\_KEYWORDS | Run a modal alert panel and obtain user feedback via up to three customisable buttons. A message and corresponding information can also be displayed and provided to the user.  Args:   * message(str): The message to provide to the user. * informativeText(Optional[str]): Additional information for the user. * firstButton(Optional[str]): The text to display in the first (default) option. * Defaults to **OK**. * secondButton(Optional[str]): The text to display in the second option. Defaults to **None**. * thirdButton(Optional[str]): The text to display in the third option. Defaults to **None**.   Returns:   * str: The text displayed by the button that the user selects. | To insert cocoa status alert inside python code |
| selectPath | (PyCFunction)  pyOsiriX\_selectPath | METH\_VARARGS|METH\_KEYWORDS | Run a modal window prompting the user to select a file/directory path. It is possible to allow selection all file types or a single file type. It is also possible to define whether directories can be returned.  Args:   * dirs(Optional[bool]): Are directory path allowed? * extension(Optional[str]): The extension of allowed file types. Ignore for all file types * title(Optional[str]): Set the displayed title of the path selection window.   Returns:   * str: The selected path. Set to None if user cancels selction. | Atom  Vs Code  PyCharm  …  ?? |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyBrowserController |  |  | A python implementation of the OsiriX 'BrowserController' class. This class is used to obtain access to studies, series and images within the OsiriX database.   * Instances of this class may not be created. * Instead instances are accessed via functions defined in the osirix module |  |
| databaseSelection | (PyCFunction)  pyBrowserController\_databaseSelection | METH\_NOARGS | Return current selection of DicomStudy and DiscomSeries instances currently selected in the database window.  Args: None  Returns:   * tuple: A tuple of all selected DicomStudy and DicomSeries instances. |  |
| copyFilesIntoDatabaseIfNeeded | (PyCFunction)  pyBrowserController\_copyFilesIntoDatabaseIfNeeded | METH\_VARARGS | Import a list of dicom files into the database. If they are already present, this operation is a no-op. Note that this method will make a COPY of the files and store them in the currently active database.  Args:   * filenames (list): A list of absolute paths for the dicom files to be added.   Returns: None. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyViewerController |  |  | A python implementation of the OsiriX 'ViewerController' class.   * This class is used to obtain access to many of the viewer properties and its contained data. * Instances of this class may not be created. Instead, instances are accessed via functions defined in the osirix module |  |
| title (get) | pyViewerController\_getTitle | NULL | The string title of the ViewerController window |  |
| title (set) | pyViewerController\_setTitle | NULL |  |  |
| movieIdx (set) | pyViewerController\_getMovieIdx | NULL | The current 4D index of the ViewerController as an integer |  |
| movieIdx (set) | pyViewerController\_setMovieIdx | NULL |  |  |
| idx (get) | pyViewerController\_getIdx | NULL | The current image index of the ViewerController as an integer |  |
| idx (set) | pyViewerController\_setIdx | NULL |  |  |
| WLWW (get) | pyViewerController\_getWLWW | NULL | The current WLWW (window-level/window-width) settings of the ViewerController as an tuple of floats (WL, WW) |  |
| WLWW (set) | pyViewerController\_setWLWW | NULL |  |  |
| modality (get) | pyViewerController\_getModality | NULL | A string representation of the viewed image modality. This property cannot be set |  |
| modality (set) | pyViewerController\_setModality | NULL |  |  |
|  |  |  |  |  |
| maxMovieIndex | (PyCFunction)  pyViewerController\_maxMovieIdx | METH\_NOARGS | Return the number of 4D viewer frames contained within the ViewerController.  Args: None  Returns:   * int: The number of 4D movie frames. |  |
| closeViewer | (PyCFunction)  pyViewerController\_closeViewer | METH\_NOARGS | Close the ViewerController instance. Note: To truly destroy the ViewerController, it should also be deletd via del().  Args: None  Returns: None. |  |
| pixList | (PyCFunction)  pyViewerConroller\_pixList | METH\_VARARGS|METH\_KEYWORDS | Provides a tuple containing the DCMPix objects represented in the ViewerController.  Args:   * movieIdx (Optional[int]): The 4D index (starts at 0) from which to obtain the tuple of DCMPix instances. Defaults to the currently displayed frame.   Returns:   * tuple: A tuple with each element containing a DCMPix instance. |  |
| startWaitProgressWindow | (PyCFunction)  pyViewerController\_startWaitProgressWindow | METH\_VARARGS | Starts a progress bar window with a given message and maximum number of steps.  Args:   * message (str): The message to display in the progress window * max (int): The maximum number of incremental steps to use until the bar is full.   Returns:   * Wait: An instance of the Wait class that can be used fro progress management. |  |
| endWaitWindow | (PyCFunction)  pyViewerController\_endWaitWindow | METH\_VARARGS | Ends a progress window and closes it.  Args:   * w (Wait): The instance of a Wait class that is to closed   Returns: None. |  |
| needsDisplayUpdate | (PyCFunction)  pyViewerController\_needsDisplayUpdate | METH\_NOARGS | Tells a ViewerController that it should be updated. This method should be called after every change to the viewer's content.  Args: None.  Returns: None. |  |
| setROI | (PyCFunction)  pyViewerController\_roiList | METH\_VARARGS|METH\_KEYWORDS | Adds a ROI instance to the viewer at a specified 4D frame and image number. Note: The needsDisplayUpdate() method should be called afterwards.  Args:   * roi (ROI): The ROI to add to the viewer * position (Optional[int]): The image position of the ROI. Defaults to the currently displayed position. * movieIdx (Optional[int]): The 4D frame of the ROI. Defaults to the currently displayed frame.   Returns: None. |  |
| roiList | (PyCFunction)  pyViewerController\_setROI | METH\_VARARGS|METH\_KEYWORDS | Provides a tuple of tuples, each containing the ROIs within each slice of the ViewerController.  Args:   * movieIdx (Optional[int]): The 4D index (starts at 0) from which to obtain the tuple of ROI instances. Defaults to the currently displayed frame. * Returns:tuple: A tuple with each element containing a tuple of ROI instance. |  |
| curDCM | (PyCFunction)pyViewerController\_curDCM | METH\_NOARGS | Provide a reference to the currently displayed DCMPix.  Args: None.  Returns:   * DCMPix: The currently displayed DCMPix of the ViewerController. |  |
| roisWithName | (PyCFunction)  pyViewerController\_roisWithName | METH\_VARARGS|METH\_KEYWORDS | Returns a tuple of ROIs within the ViewerController with a given name.  Args:   * name (str): The name of the ROIs to look for. * movieIdx (Optional[int]): The 4D index (starts at 0) in which to search. Defaults to the currently displayed frame. * in4D (Optional[bool]): Defines whether to search over all 4D frames. Defaults to False.   Returns:   * tuple: A tuple with each element containing a ROI instance. |  |
| selectedROIs | (PyCFunction)  pyViewerController\_selectedROIs | METH\_NOARGS | Returns a tuple of the currently selected ROIs.  Args: None.  Returns:   * tuple: A tuple with each element containing a ROI instance. |  |
| isDataVolumic | (PyCFunction)  pyViewerController\_isDataVolumic | METH\_VARARGS|METH\_KEYWORDS | Identifies whether the data within the viewer window is volumic (i.e. can be displayed in 3D view).  Args:   * in4D (Optional[bool]): Defines whether to check over all 4D frames. Defaults to False.   Returns:   * bool: A boolean value determining whether the data is volumic. |  |
| copyViewerWindow | (PyCFunction)  pyViewerController\_copyViewerWindow | METH\_VARARGS|METH\_KEYWORDS | Duplicates the current viewer window and displays it.  Args:   * in4D (Optional[bool]): Defines whether to copy all 4D frames. Defaults to False.   Returns:   * ViewerController: A reference to the newly created viewer. |  |
| resampleViewerController | (PyCFunction)  pyViewerController\_resampleViewerController | METH\_VARARGS | Create a copy of the the ViewerController instance with images resampled  to the same resolution as a another viewer.  Args:   * vc (ViewerController): The viewer to which the current resolution should be matched.   Returns:   * ViewerController: The reference to the new viewer in which the data is resampled. |  |
| blendingController | (PyCFunction)  pyViewerController\_blendingController | METH\_NOARGS | Returns a reference to the ViewerController currently fused with this one.  Args: None.  Returns: ViewerController: The fused viewer window. |  |
| VRControllers | (PyCFunction)pyViewerController\_VRControllers | METH\_NOARGS | Returns all volume render controllers associated with this viewer.  Args: None.  Returns:   * tuple: A tuple of all associated VRControllers. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyLog |  |  | Output log to scripting window |  |
| write | (PyCFunction)  pyLog\_write | METH\_VARARGS | Print the input string to the log window |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyWait |  |  | A python implementation of the OsiriX 'Wait' class. This class is used to update a progress indicator / progress bar.   * Instances of this class may not be created. * Instead see the documentation for osirix.ViewerController for creation methods therein |  |
| maxValue (get) | pyWait\_getMax | NULL | A float representing the maximum value of the progress indicator. This property cannot be set. | implemented? |
| floatValue (get) | pyWait\_getFloat | NULL | A float representing the current value of the progress indicator. This property cannot be set | implemented? |
|  |  |  |  |  |
| incrementBy | (PyCFunction)  pyWait\_incrementBy | METH\_VARARGS | Increment the progress by the amount specified.  Args:   * floatValue (float): The value by which to increase the progress bar   Returns: None. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyDCMPix |  |  | A python Call of the OsiriX 'DCMPix' class. A seperate DCMPix instance is stored for each image displayed by the OsiriX 2D viewer.   * Instances of this class should not be created. * Instead instances are accessed via functions defined in the ViewerContoller class |  |
| image (get) | pyDCMPix\_getImage | NULL | A numpy array represeting the pixel data stored by the DCMPix instance.   * For BW DCMPix instances, the array has shape [columns, rows] and is 32-bit floating point ('int32'). * For RGB DCMPix insatnces, the resulting array has shape [4, columns, rows] with the first dimension providing the RGBA values as an 8-bit unsigned byte array at each pixel location ('byte'). |  |
| image (set) | pyDCMPix\_setImage | NULL | (see above)   * When setting the image array, the array dimensions must not change. * Conversion should be achieved via explicit numpy conversion ('int32' for BW and 'byte' for RGBA). |  |
| shape (get) | pyDCMPix\_getShape | NULL | A tuple representing the shape of the contained image data in the form: (width, height).  This property cannot be set. |  |
| pixelSpacing (get) | pyDCMPix\_getPixelSpacing | NULL | A tuple representing the pixelSpacing of the contained image data in the form: (col. Spacing, row spacing). This property cannot be set. |  |
| origin (get) | pyDCMPix\_getOrigin | NULL | A tuple representing the origin of the top-left pixel of the image in patient coordinates. Returned in the form: (x, y, z). This property cannot be set. |  |
| location (get) | pyDCMPix\_getSliceLocation | NULL | A float representing the slice-location of the image in patient coordinates. This property cannot be set. |  |
| sourceFile (get) | pyDCMPix\_getSourceFile | NULL | A string providing the location of the source dicom file. This property cannot be set. |  |
| orientation (get) | pyDCMPix\_getOrientation | NULL | A 9-element tuple providing the orientation of the image as defined by the dicom standard. See dicom element (0020, 0037) and descriptions provided by the standard: <http://dicom.nema.org>. This property cannot be set. |  |
| isRGB (get) | pyDCMPix\_getIsRGB | NULL | A bool determining whether the DCMPix instance represents a grayscale image or RGBA.  This property cannot be set. |  |
|  |  |  |  |  |
|  |  |  |  |  |
| computeROI | (PyCFunction)  pyDCMPix\_computeROI | METH\_VARARGS | Returns a dictionary of statistics for the pixel values within a given ROI.  Args:   * ROI: The ROI from within which to compute the image statistics.   Returns:   * dict: A dictionary of ROI statistics with representative keys:   + {‘Mean’, ‘Total’, ‘Std. Dev.’, ‘Min.’, ‘Max.’, ‘Skewness’, ‘Kurtosis’} |  |
| getROIValues | (PyCFunction)  pyDCMPix\_getROIValues | METH\_VARARGS | Provides the pixel values within a given ROI  Args:   * ROI: The ROI from within which to provide the image values.   Returns:   * tuple: A two-element tuple containing numpy arrays of the pixel values and their locations in the form (row, col) |  |
| getMapFromROI | (PyCFunction)  pyDCMPix\_getMapFromROI | METH\_VARARGS | Provides a mask for the image representing regions within a given ROI.  Args:   * ROI: The ROI from which to compute the mask.   Returns:   * Numpy array (bool): A numpy array with the same dimensions as the contained image. Elements with value ‘True’ ooleant regions within the ROI. |  |
| convertToRGB | (PyCFunction)  pyDCMPix\_convertToRGB | METH\_VARARGS | Convert the contained pixel data from a grayscale (int32) array to RGBA (byte)  Args:   * type (Optional[int]): Convert greyscale values to:   + 0: Red, 1: Green, 2: Blue, 0: BW   Returns: None. |  |
|  |  |  |  |  |
| convertToBW | (PyCFunction)  pyDCMPix\_convertToBW | METH\_VARARGS | Convert the contained pixel data from an RGBA (byte) array to grayscale (byte).  Args: None.  Returns: None. |  |
| imageObj | (PyCFunction)  pyDCMPix\_imageObj | METH\_NOARGS | Provides the instance of the associated DicomImage stored by the OsiriX browser.  Args: None.  Returns: DicomImage |  |
| seriesObj | (PyCFunction)  pyDCMPix\_seriresObj | METH\_NOARGS | Provides the instance of the associated DicomSeries stored by the OsiriX browser.  Args: None.  Returns: DicomSeries |  |
| studyObj | (PyCFunction)  pyDCMPix\_studyObj | METH\_NOARGS | Provides the instance of the associated DicomStudy stored by the OsiriX browser.  Args: None.  Returns: DicomStudy |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyROI |  |  | A python implementation of the OsiriX ‘ROI’ class. Instances of this class can be created with the following signature: osirix.ROI(\*args)  Args:   * itype (Optional[str]): The type of ROI to create. Currently can only be none of:   + tPlain, tCPolygon (default), tOPolygon, tPencil. * buffer (npy\_array): A 2D numpy oolean array representing pixels contained with the ROI. This keyword is required if ROI type is tPlain and must not be specified otherwise. * origin (Optional[tuple]): The position of the top-left most pixel of the ROI buffer. * DCMPix (Optional[DCMPix]): A DCMPix instance from which to extract tha arguments ipixelSpacing and iimageOrigin. * ipixelSpacing (Optional[tuple]): A 2-element tuple, (x, y), with the pixel spacing of the image to which the ROI will be associated. Ignored if DCMPix is set. * iimageOrigin (Optional[tuple]): A 2-element tuple, (x, y), with the position of the top-left image pixel to which the ROI will be associated. Ignored if DCMPix is set. |  |
| type (get) | pyROI\_getType |  | A string representing the ROI type. This property cannot be set. |  |
| thickness (get) | pyROI\_getThickness |  | A float value for the thickness of the ROI. |  |
| thickness (set) | pyROI\_setThickness |  |  |  |
| opacity (get) | pyROI\_getOpacity |  | A float value for the opacity of the ROI. Must be within the range 0.0 -> 1.0. |  |
| opacity (set) | pyROI\_setOpacity |  |  |  |
| color (get) | pyROI\_getColor |  | A three element (R, G, B) tuple representing the ROI color. Each element must be an integer in the range 0 -> 255. |  |
| color (set) | pyROI\_setColor |  |  |  |
| name (get) | pyROI\_getName |  | A string representing the name of the ROI. |  |
| name (set) | pyROI\_setName |  |  |  |
| pix (get) | pyROI\_getPix |  | The DCMPix instance associated with the ROI. This property cannot be set. |  |
| points (get) | pyROI\_getPoints |  | A numpy float array with shape [N, 2] for N points in the ROI. |  |
| points (set) | pyROI\_setPoints |  | * Setting this property for some ROI types result in a no-op. | (which??) |
|  |  |  |  |  |
| centroid | (PyCFunction)  pyROI\_centroid | METH\_NOARGS | Returns a two-element tuple representing the centroid of the ROI.  Args: None.  Returns:   * tuple: A 2-element tuple representing the ROI centroid in the form: (rows, columns) |  |
| roiMove | (PyCFunction)  pyROI\_roiMove | METH\_VARARGS | Move the ROI by a specified number of pixels in the image plane.  Args:   * c (float): The number of columns to move the ROI. Positive values move left -> right. * r (float): The number of rows to move the ROI. Positive values move up -> down.   Returns: None. |  |
| rotate | (PyCFunction)  pyROI\_rotate | METH\_VARARGS | Rotate the ROI by a specified angle about a point (x, y) in the image plane.  Note: This method is a no-op for brush ROIs.  Args:   * theta (float): The angle (in degrees) by which to rotate the ROI. * pt (tuple): A 2-element tuple of float values representing the point about which to rotate the ROI. The order of the tuple should be (columns, rows).   Returns: None. |  |
| flipVertically | (PyCFunction)  pyROI\_flipVertically | METH\_NOARGS | Flip the ROI vertically. Note: This method is a no-op for brush ROIs.  Args: None.  Returns: None. |  |
| flipHorizontally | (PyCFunction)  pyROI\_flipHorizontally | METH\_NOARGS | Flip the ROI horizontally. Note: This method is a no-op for brush ROIs.  Args: None.  Returns: None. |  |
| roiArea | (PyCFunction)  pyROI\_roiArea | METH\_NOARGS | Return the area with the ROI.  Args: None.  Returns:   * float: The area within the ROI in units of cm^2. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyROIVolume |  |  | A python implementation of the OsiriX ‘ROIVolume’ class.   * Instances of this class may not be created. * Rather they are accessed through the methods defined in VRController. |  |
| name (get) | pyROIVolume\_getName | NULL | A string representing the name of the ROIVolume. This will match those of the contructing ROIs. This property cannot be set. |  |
| visible (get) | pyROIVolume\_getVisible |  | A bool determining whether the ROIVolume is visible within the VRController. |  |
| visible (set) | pyROIVolume\_setVisible |  |  |  |
| texture (get) | pyROIVolume\_getTexture |  | A bool determining whether the ROIVolume is textured. |  |
| texture (set) | pyROIVolume\_setTexture |  |  |  |
| color (get) | pyROIVolume\_getColor |  | A 4-element (R,G,B,A) tuple representing the ROIVolume color. Each element must be in the range 0 -> 1. |  |
| color (set) | pyROIVolume\_setColor |  |  |  |
|  |  |  |  |  |
| volume | (PyCFunction)  pyROIVolume\_volume | METH\_NOARGS | Returns float representing the volume of the 3D ROI.  Args: None.  Returns:  float: The volume in cm^3. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyDicomImage |  |  | A python implementation of the OsiriX 'DicomImage' class. DicomImage is a convienience class used by the main browser within OsiriX to organise dicom instances.   * Instances of this class should not be created. * Instead instances are accessed via functions defined in the BrowserController class |  |
| date (get) | pyDicomImage\_getDate | NULL | The date of image acquisition. This property cannot be set |  |
| numberOfFrames (get) | pyDicomImage\_getNumberOfFrames | NULL | The integer number of frames contained within the associated dicom SOP instance. This property cannot be set. |  |
| modality (get) | pyDicomImage\_getModality | NULL | The modality of the image in string representation. This property cannot be set. |  |
| series (get) | pyDicomImage\_getSeries | NULL | Returns the DicomSeries associated with the DicomImage. This property cannot be set. | string? |
| sliceLocation (get) | pyDicomImage\_getSliceLocation | NULL | The floating-point slice location in patient coordinates. This property cannot be set. |  |
| instanceNumber (get) | pyDicomImage\_getInstanceNumber | NULL | The integer instance number. This property cannot be set. |  |
|  |  |  |  |  |
| width | (PyCFunction)  pyDicomImage\_width | METH\_NOARGS | The number of columns within the contained image.  Args: None.  Returns:   * int: The number of columns. |  |
| height | (PyCFunction)  pyDicomImage\_height | METH\_NOARGS | The number of rows within the contained image.  Args: None.  Returns:   * int: The number of rows. |  |
| sopInstanceUID | (PyCFunction)  pyDicomImage\_sopInstanceUID | METH\_NOARGS | The SOPInstance unique identifier of the image object represented as a string.  Args: None.  Returns:   * str: The UID value represented by dicom tag (0008, 0018) . |  |
| completePath | (PyCFunction)  pyDicomImage\_completePath | METH\_NOARGS | The path of the associated dicom file.  Args: None.  Returns:   * str: The path of dicom file conforming to RFC 1808. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyDicomSeries |  |  | A python implementation of the OsiriX 'DicomSeries' class. DicomSeries is a convienience class used by the main browser within OsiriX to organise and group dicom instances within the same series.   * Instances of this class may not be created. * Instead instances are accessed via functions defined in the BrowserController class |  |
| study (get) | pyDicomSeries\_getStudy | NULL | The DicomStudy associated with the DicomSeries. This property cannot be set. |  |
| images (get) | pyDicomSeries\_getImages | NULL | A tuple of (Osirix defined) DicomImage objects associated with the DicomSeries. This property cannot be set. |  |
| numberOfImages (get) | pyDicomSeries\_getNumberOfImages | NULL | The integer number of images in the DicomSeries. This property cannot be set. |  |
| seriesInstanceUID (get) | pyDicomSeries\_getSeriesInstanceUID | NULL | A string of the series UID for the DicomSeries. This property cannot be set. |  |
| seriesSOPClassUID (get) | pyDicomSeries\_getSeriesSOPClassUID | NJLL | A string of the series SOP class UID for the DicomSeries. This property cannot be set. |  |
| seriesDescription (get) | pyDicomSeries\_getSeriesDescription | NJLL | A string of the series UID for the DicomSeries. This property cannot be set. |  |
| modality (get) | pyDicomSeries\_getModality | NJLL | A string of the DicomSeries modality. This property cannot be set. |  |
| name (get) | pyDicomSeries\_getName | NJLL | A string of the DicomSeries name. This property cannot be set. |  |
| date (get) | pyDicomSeries\_getDate | NJLL | A date object for the DicomSeries. This property cannot be set. |  |
|  |  |  |  |  |
| paths | (PyCFunction)  pyDicomSeries\_paths | METH\_NOARGS | Returns a tuple containing the paths of all associated dicom files for the series.  Args: None.  Returns:   * tuple: The filepaths to all associated dicom SOP instances. |  |
| previousSeries | (PyCFunction)  pyDicomSeries\_previousSeries | METH\_NOARGS | Returns a reference to the previous series in the OsiriX browser.  Args: None.  Returns: DicomSeries |  |
| nextSeries | (PyCFunction)  pyDicomSeries\_nextSeries | METH\_NOARGS | Returns a reference to the next series in the OsiriX browser.  Args: None.  Returns: DicomSeries |  |
| sortedImages | (PyCFunction)  pyDicomSeries\_sortedImages | METH\_NOARGS | Returns a tuple of DicomImage instances sorted accorrding to the cirteria currently set by the OsiriX browser.  Args: None.  Returns:   * tuple: A tuple of DicomImage instances. |  |
| Class/method | Call | Args | Notes | Status |
| pyDicomStudy |  |  | A python implementation of the OsiriX 'DicomStudy' class. DicomStudy is a convenience class used by the main browser within OsiriX to organise and group dicom instances within the same study.   * Instances of this class may not be created. * Instead instances are accessed via functions defined in the BrowserController class |  |
| numberOfImages (get) | pyDicomStudy\_getNumberOfImages | NULL | The integer number of images contained by the DicomStudy. This property cannot be set |  |
| series (get) | pyDicomStudy\_getSeries | NULL | A tuple containing all DicomSeries within the DicomStudy. This property cannot be set |  |
| name (get) | pyDicomStudy\_getName | NULL | The patient name for DicomStudy as a string. This property cannot be set |  |
| date (get) | pyDicomStudy\_getDate | NULL | The date of the DicomStudy as a datetime object. This property cannot be set |  |
| dateAdded (get) | pyDicomStudy\_getDateAdded | NULL | The date the DicomStudy was added to OsiriX as a datetime object. This property cannot be set |  |
| dateOfBirth (get) | pyDicomStudy\_getDateOfBirth | NULL | The patient DOB for the DicomStudy as a datetime object. This property cannot be set |  |
| institutionName (get) | pyDicomStudy\_getInstitutionName | NULL | The institution name of the DicomStudy as a string. This property cannot be set |  |
| modality (get) | pyDicomStudy\_getModality | NULL | The modality of the DicomStudy as a string. This property cannot be set |  |
| patientID (get) | pyDicomStudy\_getPatientID | NULL | The patient ID of the DicomStudy as a string. This property cannot be set |  |
| patientUID (get) | pyDicomStudy\_getPatientUID | NULL | The patient UID of the DicomStudy as a string. This property cannot be set |  |
| patientSex (get) | pyDicomStudy\_getPatientSex | NULL | The patient sex of the DicomStudy as a string. This property cannot be set |  |
| performingPhysician (get) | pyDicomStudy\_getPerformingPhysician | NULL | The name of the performing physician for the DicomStudy as a string. This property cannot be set |  |
| referringPhysician (get) | pyDicomStudy\_getReferringPhysician | NULL | The name of the referring physician for the DicomStudy as a string. This property cannot be set |  |
| studyInstanceUID (get) | pyDicomStudy\_getStudyInstanceUID | NULL | The DicomStudy UID as a string. This property cannot be set |  |
| studyName (get) | pyDicomStudy\_getStudyName | NULL | The DicomStudy name as a string. This property cannot be set |  |
|  |  |  |  |  |
| paths | (PyCFunction)  pyDicomStudy\_paths | METH\_NOARGS | Returns a tuple containing the paths of all associated dicom files for the study.  Args: None.  Returns:   * tuple: The filepaths to all associated dicom SOP instances. |  |
| images | (PyCFunction)  pyDicomStudy\_images | METH\_NOARGS | Returns a tuple containing the DicomImages contained within the study.  Args: None.  Returns:   * tuple: The DicomImage instances contained within the study. |  |
| modalities | (PyCFunction)  pyDicomStudy\_modalities | METH\_NOARGS | Returns a string with all modelities in the study.  Args: None.  Returns: str: See above. |  |
| imageSeries | (PyCFunction)  pyDicomStudy\_imageSeries | METH\_NOARGS | Returns a tuple containing the DicomSeries contained within the study.  Args: None.  Returns:   * tuple: The DicomSeries instances contained within the study. |  |
| noFiles | (PyCFunction)  pyDicomStudy\_noFiles | METH\_NOARGS | Returns the number of dicom files associated with the study.  Args: None.  Returns: int: See above. |  |
| rawNoFiles | (PyCFunction)  pyDicomStudy\_rawNoFiles | METH\_NOARGS | Returns the raw number of dicom files associated with the study.  Args: None.  Returns: int: See above. |  |
| noFilesExcludingMultiFrames | (PyCFunction)  pyDicomStudy\_noFilesExcludingMultiFrames | METH\_NOARGS | Returns the number of dicom files associated with the study, excluding any multiframe instances.  Args: None.  Returns: int: See above. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/method | Call | Args | Notes | Status |
| pyVRController |  |  | A python implementation of the OsiriX 'VRController' class. This class is used to obtain limited access to some of the volume rendering window properties.   * Instances of this class may not be created. * Instead instances are accessed via functions defined in the osirix module |  |
| renderingMode (get) | pyVRController\_renderingMode | NULL | The rendering mode the ViewerController window as a string.   * Must be one of two possible values: **VR** (volume rendering) or **MIP** |  |
| renderingMode (set) | pyVRController\_setRenderingMode | NULL |  |  |
| WLWW (get) | pyViewerController\_getWLWW | NULL | The current WLWW (window-level/window-width) settings of the ViewerController as an tuple of floats (WL, WW) |  |
| WLWW (set) | pyViewerController\_setWLWW | NULL |  |  |
| title (get) | pyViewerController\_getTitle | NULL | The string title of the VRController window. This property cannot be set. |  |
| style (get) | pyViewerController\_setStyle | NULL | The style of the VRController window (either **panel** or **standard**). This property cannot be set. |  |
|  |  |  |  |  |
| viewer2D | (PyCFunction)  pyVRController\_viewer2D | METH\_NOARGS | Return the ViewerController instance containing the original data currently displayed.  Args: None  Returns:   * ViewerController: The associated OsiriX 2D viewer window. |  |
| blendingController | (PyCFunction)  pyVRController\_blendingController | METH\_NOARGS | If the data is fused, return the ViewerController instance containing the fused image data.  Args: None  Returns:   * ViewerController: The fused OsiriX 2D viewer window if available. Set to 'None' if no fusion active. |  |