amaptor Documentation

Release 0.1.2.3

Nick Santos

Feb 24, 2018

Contents

Contents:

CHAPTER 1

Amaptor Classes

1.1 amaptor.Project

class amaptor.**Project**(*path*)

An ArcGIS Pro Project or an ArcMap map document - maps in ArcGIS Pro and data frames in ArcMap are Map class attached to this project Access to the underlying object is provided using name ArcGISProProject and ArcMapDocument

active_map

Returns the active map object or data frame as determined by get_active_map()

Returns

check_layout_name(*name*)

PRO ONLY. Given the name of a layout, confirms it doesn't exist and raised amaptor.LayoutExists if it's found

Parameters **name** – the case sensitive name of an existing layout to find

Returns None. Raises amaptor.LayoutExists if layout with name exists.

check_map_name(*name*)

Checks to see if the project or map document already has a map or data frame with a given name. Since names must be unique in ArcGIS Pro, this code helps check before adding new maps

Parameters **name** – name of map to check for

Returns None. Raises an error if name is taken

default_geodatabase

Returns the Project's default geodatabase in Pro, and the current workspace (arcpy.env.workspace) in ArcMap. If arcpy.env.workspace is None, creates a GDB in same folder as map document and returns that value, to ensure that this function always returns a usable workspace. If a GDB is created, this function does NOT set arcpy.env.workspace to that, so as not to interfere with other operations. Do that explicitly if that behavior is desired.

Returns

find_layer(*path*, *find_all=True*)

Finds a layer in all maps by searching for the path. By default finds all, but can find just the first one too

Parameters

- **path** – the full path of the data source for the layer
- **find_all** – When True, reutrns a list of amaptor.Map instances that match. When false, returns only the first match

Returns list of amaptor.map instances or a single amaptor.map instance.

find_layout(*name*)

PRO ONLY. Given a layout name, returns the amaptor.Layout object or raises LayoutNotFoundError

Parameters **name** – the name of the layout to find.

Returns amaptor.Layout instance with given name.

find_map(*name*)

Given a map name, returns the map object or raises MapNotFoundError

Parameters **name** – name of map to find.

Returns amaptor.Map instance

get_active_map(*use_pro_backup=True*)

Emulates functionality of arcpy.mapping(mxd).activeDataFrame. In ArcMap, it returns the amaptor.Map object that corresponds to that active Data Frame. In Pro, which doesn't have the concept of active maps, it by default returns the first map in the document. If use_pro_backup is set to False, it will instead raise amaptor.MapNotImplementedError

Parameters **use_pro_backup** – When True, it uses the first map in the ArcGIS Pro project, since Pro doesn't have a way to get the active map.

Returns amaptor.Map instance

list_maps()

Provided to give a similar interface to ArcGIS Pro - Project.maps is also publically accessible

Returns

map_names

A convenience function to get a list of map names

Returns

new_layout(*name*, *template_layout='/home/docs/checkouts/readthedocs.org/user_builds/amaptor/envs/latest/local/lib/python packages/amaptor-0.1.2.4-py2.7.egg/amaptor/templates/pro/blank_layout.pagx'*, *template_name='_pro_blank_layout_template'*)

PRO ONLY. Adds a new layout to an ArcGIS Pro Project by importing a saved blank layout. Alternatively, you can provide an importable layout document (.pagx) for ArcGIS Pro, and then provide that layout's name as template_name so that it can be renamed, and the provided template will be used instead of a blank.

Parameters

- **name** – The name to give the new layout
- **template_layout** – The template to use for creating the layout (an ArcGIS Pro .pagx file). If none is provided, uses a blank template
- **template_name** – The name of the layout in the template. Only define this value if you also provide a new template layout and the name should match the layout name in the template. This parameter is used to find the inserted template and rename it. Strange things will happen if this value does not match the name of the layout in the template_layout.

Returns amaptor.Layout instance. This layout will already have been added to the project, but is returned for convenience.

new_map(*name*, *template_map='/home/docs/checkouts/readthedocs.org/user_builds/amaptor/envs/latest/local/lib/python2.7/sit packages/amaptor-0.1.2.4-py2.7.egg/amaptor/templates/arcmap/pro_import_map_template.mxd'*, *template_df_name='_rename_template_amaptor'*)

PRO ONLY. Creates a new map in the current project using a hack (importing a blank map document, and renaming data frame) Warning: Only works in Pro due to workaround. There isn't a way to add a data frame from arcpy.mapping. In the future, this could potentially work in arcmap by transparently working with a separate map document in the background (creating a project, map, and layout for those items and linking them into this project).

Parameters

- **name** – The name to give the imported map
- **template_map** – The map document to import. If we're just going with a blank new map, leave as default. To import some other template as your base, provide a path to a document importable to ArcGIS Pro' .importDocument function for projects.
- **template_df_name** – The current name of the imported map document for renaming. Only needs to be set if template_map is overridden

Returns amaptor.Map instance - also added to the map document, but returned for immediate use.

replace_text(*text*, *replacement*)

Given a string and a replacement value, finds all instances (in the current map document or project) of that text in text elements and titles, and replaces those instances with the new value. Useful for creating your own variables like {species} or {field_id} in map templates. Careful when using this - in Pro, it will search all Layouts and replace the string. If you are concerned and want single layout behavior, use the same function on the Layout class.

Parameters

- **text** –
- **replacement** –

Returns

save()

Saves the project or map document in place.

Returns None

save_a_copy(*path*)

Saves the project or map document to the provided path.

Parameters **path** – the new path to save the copy of the document to.

Returns None

to_package(*output_file*, *summary*, *tags*, ***kwargs*)

Though it's not normally a mapping method, packaging concepts need translation between the two versions, so we've included to_package for maps and projects. In ArcGIS Pro, project.to_package will create a Project Package and map.to_package will create a map package. In ArcMap, both will create a map package. Extra **kwargs beyond output path are only passed through to Pro Package command, not to map packages. To pass kwargs through to a map package, use a map object's to_package method.

Parameters

- **output_file** – the path to output the package to
- **kwargs** – dictionary of kwargs to pass through to project packaging in Pro.

Returns None

1.2 amaptor.Map

class amaptor.**Map**(*project*, *map_object*)

Corresponds to an ArcMap Data Frame or an ArcGIS Pro Map

add_layer(*add_layer*, *add_position='AUTO_ARRANGE'*)

Straight replication of addLayer API in arcpy.mp and arcpy.mapping. Adds a layer to a specified position in the table of contents.

Parameters

- **add_layer** – The layer to add to the map
- **add_position** – The position to add the layer into. The default is AUTO_ARRANGE, and available options are the same as those available on addLayer.

Returns None

export_pdf(*out_path*, *layout='ALL'*, ***kwargs*)

See documentation for _export for description of behavior in each version. kwargs that apply to exporting to PDF in ArcMap and ArcGIS Pro apply here.

Parameters

- **out_path** – The full path to export the document to. Will be modified in the case of layout="ALL". New generated paths will be returned by the function as a list.
- **layout** – PRO only, safely ignored in ArcMap. The mp.Layout or amaptor.Layout object to export, or the keyword "ALL"
- ****kwargs** – accepts the set of parameters that works for both arcmap and arcgis pro. resolution, image_quality, image_compression, embed_fonts, layers_attributes, georef_info, jpeg_compression_quality. In the future, this may be reengineered to translate parameters with common goals but different names

Returns

export_png(*out_path*, *resolution=300*, *layout='ALL'*)

See documentation for _export for description of behavior in each version. The specific option here is only the resolution to export at.

Parameters

- **out_path** – The full path to export the document to. Will be modified in the case of layout="ALL". New generated paths will be returned by the function as a list.
- **resolution** – the resolution to export the map at
- **layout** – PRO only, safely ignored in ArcMap. The mp.Layout or amaptor.Layout object to export, or the keyword "ALL"

Returns

find_layer(*name=None*, *path=None*, *find_all=False*)

Given the name OR Path of a layer in the map, returns the layer object. If both are provided, returns based on path. If multiple layers with the same name/path exist, returns the first one, unless find_all is True - then it returns a list with all instances. Automatically converted to work with new Layer object because self.layers uses them

Parameters

- **name** –
- **path** –
- **find_all** –

Returns arcpy.Layer object

insert_feature_class_with_symbology(*feature_class*, *layer_file*, *layer_name=None*, *near_name=None*, *near_path=None*, *insert_position='BEFORE'*)

Given a path to a feature calss, and a path to a layer file, creates a layer with layer file symbology and inserts it using insert_layer_by_name_or_path's approach

Parameters

- **feature_class** –
- **layer_file** –
- **layer_name** –

- **near_name** –
- **near_path** –
- **insert_position** –

Returns

insert_layer(*reference_layer*, *insert_layer_or_layerfile*, *insert_position='BEFORE'*)

Inserts a layer to a specific position in the table of contents, based on a reference layer.

Parameters

- **reference_layer** – The arcpy Layer instance to use as the reference layer
- **insert_layer_or_layerfile** – The arcpy Layer instance to insert
- **insert_position** – the position relative to the reference layer to insert the new layer. Default is "BEFORE" (above). options correspond to those available on insertLayer in arcpy.mapping and arcpy.mp

Returns None

insert_layer_by_name_or_path(*insert_layer_or_layer_file*, *near_name=None*, *near_path=None*, *insert_position='BEFORE'*)

Not a standard arcpy.mapping or arcpy.mp function - given a name or data source path of a layer, finds it in the layers, and inserts it. Only provide either near_name or near_path. If both are provided, near_path will be used because it's more specifci :param insert_layer_or_layer_file: :param near_name: :param near_path: :param insert_position:

Returns None

list_layers()

Returns the list of layers in the map or data frame. Also available as map.layers

Returns

name

replace_text(*text*, *replacement*)

Given a string and a replacement value, finds all instances of that text in text elements and titles, and replaces those instances with the new value. Useful for creating your own variables like {species} or {field_id} in map templates.

Similar to the project-level replace_text, but behaves slightly differently. In ArcMap, replaces all occurrences in the current map document. In Pro, searches all layouts linked to the current layout and replaces the string in any text element in those layouts.

Parameters

- **text** –
- **replacement** –

Returns

set_extent(*extent_object*, *set_frame='ALL'*, *add_buffer=True*, *buffer_factor=0.05*)

Sets map frames to a provided extent object. In ArcMap, just sets the data frame's extent. In Pro, it has many potential behaviors. If set_frame == "ALL" it sets all map frames linked to this map to this extent (default behavior) and sets the default camera for this map so that future map frames will use the same extent. If set_frame is an arcpy.mp MapFrame object instance, then it only sets the extent on that map frame.

Parameters

- **extent_object** – an arcpy.Extent object. It will be reprojected to the spatial reference of the map frame or data frame automatically.
- **set_frame** – ignored in arcmap, behavior described in main method description.
- **add_buffer** – adds an empty space of 5% of the distance across the feature class around the provided extent
- **buffer_factor** – if add_buffer is True, then this factor controls how much space to add around the layer (default=.05)

Returns None

to_package(*output_file*, ***kwargs*)

Though it's not normally a mapping method, packaging concepts need translation between the two versions, so we've included to_package for maps and projects. In ArcGIS Pro, project.to_package will create a Project Package and map.to_package will create a map package. In ArcMap, both will create a map package. Calling to_package on the map will pass through all kwargs to map packaging because the signatures are the same between ArcMap and ArcGIS Pro. Sending kwargs to project.to_package will only send to project package since they differ.

Parameters

- **output_file** –
- **kwargs** –

Returns

zoom_to_layer(*layer*, *set_frame='ALL'*, *add_buffer=True*, *buffer_factor=0.05*)

Given a name of a layer as a string or a layer object, zooms the map extent to that layer WARNING: In Pro, see the parameter information for set_layout on the set_extent method for a description of how this option behaves. Since maps don't correspond 1:1 to layouts, in some cases multiple layouts will be changed.

Parameters

- **layer** – can be a string name of a layer, or a layer object
- **set_layout** – PRO ONLY, but ignored in ArcMap, so can be safe to use. This parameter controls which map frames are changed by the Zoom to Layer. By default, all linked map frames are updated. If an arcpy.mp.MapFrame instance or an amaptor.MapFrame instance is provided, it zooms only that map frame to the layer.
- **add_buffer** – adds an empty space of 5% of the distance across the feature class around the provided extent
- **buffer_factor** – if add_buffer is True, then this factor controls how much space to add around the layer (default=.05)

Returns None

1.3 amaptor.MapFrame

class amaptor.**MapFrame**(*map_frame_object*, *layout*)

get_extent()

map

name

set_extent(*extent_object*)

1.4 amaptor.Layout

class amaptor.**Layout**(*layout_object*, *project*)

Replicates Layouts so that we can do some nice things behind the scenes.

In ArcMap, a single layout is created - that way, a layout can safely be retrieved for all documents and modifying properties of a layout modifies corresponding map document properties.

export_to_pdf(*out_path*, ***kwargs*)

export_to_png(*out_path*, *resolution=300*)

Currently Pro only - needs refactoring to support ArcMap and Pro (should export map document in ArcMap). Also needs refactoring to combine Map and Layout export code.

Parameters

- **out_path** –
- **resolution** –

Returns

find_element(*name*)

Finds the first element matching the provided name

Parameters **name** –

Returns

find_map_frame(*name*)

Finds the map frame with a given name

Parameters **name** – the name of the frame to find

Returns MapFrame object

list_elements()

name

Corresponds to the name of a layout in Pro and the Map Document's "title" property in ArcMap

Returns

replace_text (*text*, *replacement*)

Single layout analogue of Project.replace_text. Given a string and a replacement value, replaces all instances of that string in all text elements in the layout. Useful for having template strings in a map document

Parameters

- **text** –
- **replacement** –

Returns

toggle_element (*name_or_element*, *visibility='TOGGLE'*)

Given an element name, toggles, makes visible, or makes invisible that element.

Parameters

- **name_or_element** – a string name of an element, or an element object
- **visibility** – Controls the action. Valid values are boolean (True, False) or the keyword "TOGGLE" which switches its current visibility state.

Returns

CHAPTER 2

amaptor.functions

amaptor.functions.**get_workspace_factory_of_dataset**(*dataset_path*)

Provides the workspace factory type of a provided dataset - a convenience function that calls get_workspace type with the appropriate flag in the backgroun

Parameters **dataset_path** – path to dataset to return workspace_factory value of

Returns

amaptor.functions.**get_workspace_type**(*dataset_path*, *factory_type=False*)

Gives us a workspace type that's usable for layer.replaceDataSource in ArcMap based on a dataset path

Parameters

- **dataset_path** – path to dataset to return workspace type from
- **factory_type** – boolean flag indicating whether to return the workspace_factory type or the standard dataset type - workspace factory values are not yet fully implemented

Returns

amaptor.functions.**make_layer_with_file_symbology**(*feature_class*, *layer_file*, *layer_name=None*)

Given a feature class or raster and a template layer file with symbology, returns a new Layer object that has the layer from the layer file with the feature class as its data source. Optionally, layer can be renamed with layer_name

This function will be scheduled for deprecation as it's superceded by using the Layer object with a template layer. More testing needs to be done and maybe more code written in ArcMap to make the alternative apply to both cases. This will be deprecated when that's complete

Parameters

- **feature_class** –

- **layer_file** –
- **layer_name** –

Returns

amaptor.functions.**reproject_extent**(*extent*, *current_extent*)

Changes an extent from its current spatial reference to the spatial reference on another extent object

Parameters

- **extent** –
- **current_extent** –

Returns

CHAPTER 3

amaptor.errors

exception amaptor.errors.**ElementNotFoundError**(**args*, ***kwargs*)
Raised when searching a layout for an element, and it's not found

exception amaptor.errors.**EmptyFieldError**(*field*, *description*, ***kwargs*)

exception amaptor.errors.**LayerNotFoundError**
Raised when searching for a specific layer and it isn't found.

exception amaptor.errors.**LayoutExists**(*layout_name*, ***kwargs*)
Raised when a layout (in Pro only) already exists because a new one cannot be added with the same name

exception amaptor.errors.**LayoutNotFoundError**(**args*, ***kwargs*)
Raised when a process is trying to find a specific layout (Pro only) and it is not found.

exception amaptor.errors.**MapExists**(*map_name*, ***kwargs*)
Raised when a map or data frame already exists because a new one cannot be added with the same name

exception amaptor.errors.**MapFrameNotFoundError**(**args*, ***kwargs*)
Raised when a process is trying to find a specific layout (Pro only) and it is not found.

exception amaptor.errors.**MapNotFoundError**(**args*, ***kwargs*)
Raised when a process is trying to find a specific map and it is not found.

exception amaptor.errors.**MapNotImplementedError**(*feature_name*, *arcmap_or_pro*, ***kwargs*)
Raised when a feature is only implemented for either ArcMap or ArcGIS Pro and code tries to use it in the environment it is not applicable to.

exception amaptor.errors.**NotSupportedError**(*message*, ***kwargs*)
Raised when a feature is not supported by ArcGIS itself.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

S

T

Z