

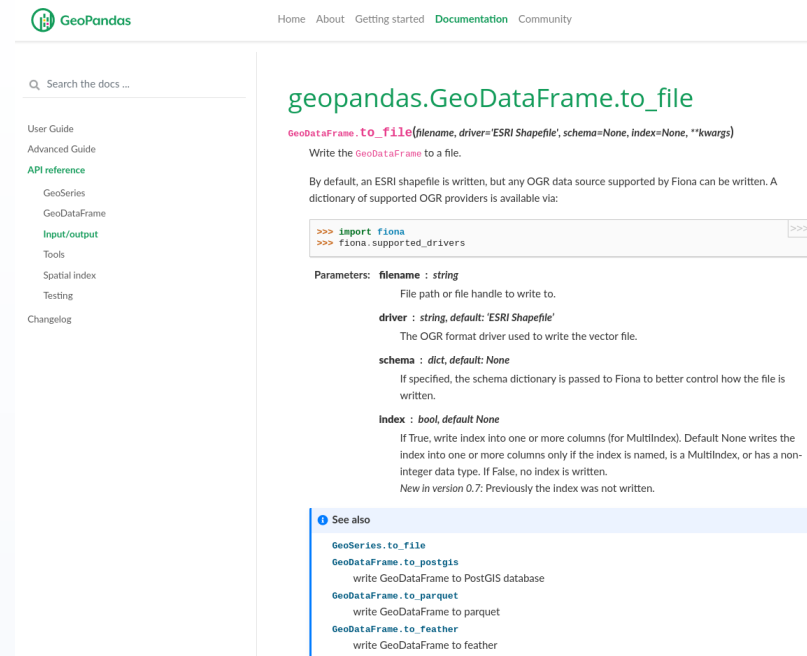
EGM722 – Programming for GIS and Remote Sensing

Week 2, Part 2: Help and Documentation

Using other people's code

- When using modules/packages, we want to know something about them
- For example:
 - Available methods/classes
 - How to use particular methods
 - Object attributes
- Assumes that the package is well-documented (hopefully, but not always, the case)

- Built-in functions and standard library documentation: python.org/doc
 - Check which version!
- Most of the packages we are using have websites
 - Check which version!

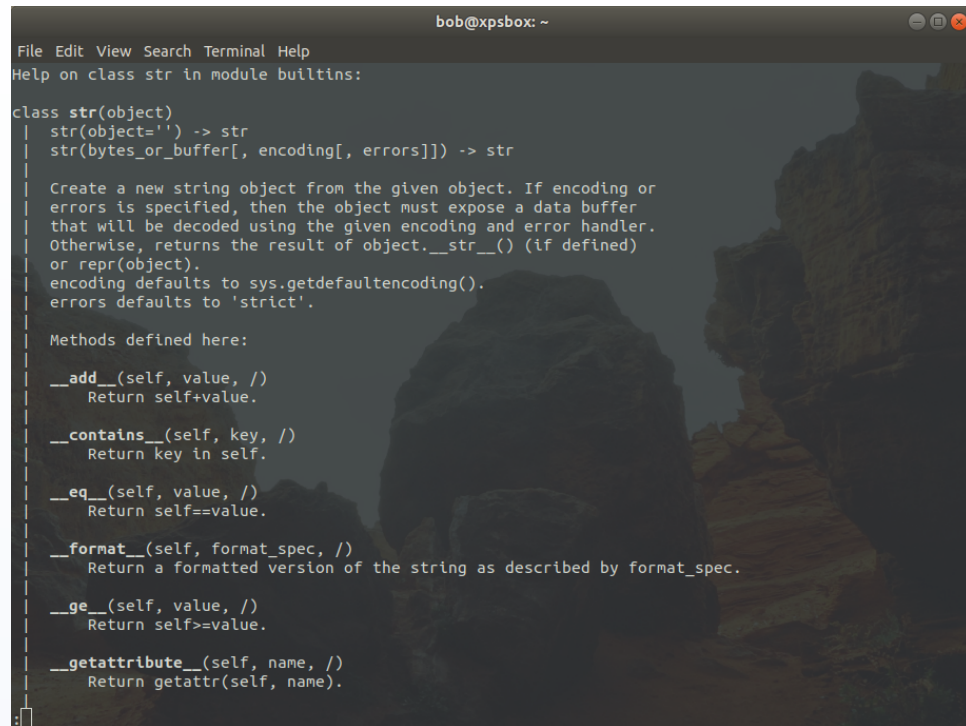


The screenshot shows the GeoPandas documentation page for the `to_file` method. The page has a dark blue header with the GeoPandas logo and navigation links: Home, About, Getting started, Documentation, and Community. A search bar is located below the header. On the left side, there is a sidebar with a list of navigation links: User Guide, Advanced Guide, API reference (highlighted), GeoSeries, GeoDataFrame, Input/output (highlighted), Tools, Spatial index, Testing, and Changelog. The main content area displays the title `geopandas.GeoDataFrame.to_file` in green. Below the title, the method signature is shown: `GeoDataFrame.to_file(filename, driver='ESRI Shapefile', schema=None, index=None, **kwargs)`. A note states: "Write the `GeoDataFrame` to a file." Another note explains: "By default, an ESRI shapefile is written, but any OGR data source supported by Fiona can be written. A dictionary of supported OGR providers is available via:". A code block shows the command to list supported drivers:

```
>>> import fiona
>>> fiona.supported_drivers
```

. Below the code block, the parameters are listed: `filename` (string, File path or file handle to write to), `driver` (string, default: 'ESRI Shapefile', The OGR format driver used to write the vector file), `schema` (dict, default: None, If specified, the schema dictionary is passed to Fiona to better control how the file is written), and `index` (bool, default: None, If True, write index into one or more columns (for Multindex). Default None writes the index into one or more columns only if the index is named, is a Multindex, or has a non-integer data type. If False, no index is written. New in version 0.7: Previously the index was not written.). At the bottom, there is a "See also" section with links to `GeoSeries.to_file`, `GeoDataFrame.to_postgis` (write GeoDataFrame to PostGIS database), `GeoDataFrame.to_parquet` (write GeoDataFrame to parquet), and `GeoDataFrame.to_feather` (write GeoDataFrame to feather).

- Within interpreter, use help():
 - e.g., help(str)
 - Remember: help() reads the docstring
 - Can call on functions, modules, objects...
- In ipython/jupyter notebook, can also use ?
 - e.g., str?



```
bob@xpsbox: ~  
File Edit View Search Terminal Help  
Help on class str in module builtins:  
  
class str(object)  
|   str(object='') -> str  
|   str(bytes_or_buffer[, encoding[, errors]]) -> str  
|  
|   Create a new string object from the given object. If encoding or  
|   errors is specified, then the object must expose a data buffer  
|   that will be decoded using the given encoding and error handler.  
|   Otherwise, returns the result of object.__str__() (if defined)  
|   or repr(object).  
|   encoding defaults to sys.getdefaultencoding().  
|   errors defaults to 'strict'.  
|  
|   Methods defined here:  
|  
|   __add__(self, value, /)  
|       Return self+value.  
|  
|   __contains__(self, key, /)  
|       Return key in self.  
|  
|   __eq__(self, value, /)  
|       Return self==value.  
|  
|   __format__(self, format_spec, /)  
|       Return a formatted version of the string as described by format_spec.  
|  
|   __ge__(self, value, /)  
|       Return self>=value.  
|  
|   __getattr__(self, name, /)  
|       Return getattr(self, name).  
|  
|  
|
```

- The dir() built-in lists all of the methods and attributes associated with an object
 - e.g., dir(str)
 - dir() lists names in current scope (packages, variables, functions, etc.)
- Note the “magic” or “special” methods:
 - __add__, __init__, etc
 - These are normally only used by the interpreter – we don’t tend to use them in programming

```

bob@xpsbox: ~
File Edit View Search Terminal Help
>>> dir(str)
['_add_', '__class__', '_contains_', '_delattr_', '_dir_', '_doc_', '_eq_', '_format_',
'_ge_', '_getattr_', '_getitem_', '_getnewargs_', '_gt_', '_hash_', '_init_',
'_init_subclass_', '_iter_', '_le_', '_len_', '_lt_', '_mod_', '_mul_', '_ne_', '_ne
w_', '_reduce_', '_reduce_ex_', '_repr_', '_rmod_', '_rmul_', '_setattr_', '_sizeof_',
'_str_', '_subclasshook_', '_capitalize_', '_casefold_', '_center_', '_count_', '_endswith_',
'_expandtabs_', '_find_', '_format_', '_format_map_', '_index_', '_isalnum_', '_isalpha_', '_isascii_', '_isdecimal_',
'_isdigit_', '_isidentifier_', '_islower_', '_isnumeric_', '_isprintable_', '_isspace_', '_istitle_', '_isupper_',
'_join_', '_ljust_', '_lower_', '_lstrip_', '_maketrans_', '_partition_', '_replace_', '_rfind_', '_rindex_', '_rjust_',
'_rpartition_', '_rsplit_', '_rstrip_', '_split_', '_splitlines_', '_startswith_', '_strip_', '_swapcase_', '_title_',
'_translate_', '_upper_', '_zfill_']
>>> 

```

```

bob@xpsbox: ~
File Edit View Search Terminal Help
>>> dir()
['_annotations_', '_builtins_', '_doc_', '_loader_', '_name_', '_package_', '_spec_',
'gpd', 'math', 'np', 'plt', 'randint', 'random', 'x']
>>> 

```

- Well-developed packages will have documentation available
- In the interpreter:
 - Use `help()` to learn more about a given object, method, or function
 - Use `dir(object)` to list methods/attributes of the object
 - Use `dir()` to list all objects in current scope