Python IDD/IDF Utilities for EnergyPlus Documentation

Release 0.1

Edwin Lee

Nov 21, 2018

Contents:

This is a Python library useful for reading/writing/validating EnergyPlus input files.

CHAPTER 1

Library Usage

This page will describe the work flow for this library.

CHAPTER 2

IDD Object Module Documentation

class pyiddidf.idd.objects.**IDDField**(*an_index*)

A simple class that defines a single field for an IDD object. Relevant members are listed here:

Variables

- **field_an_index** (*str*) – Main identifier for this field
- **meta_data** (*dict(str,[str])*) – A dictionary, where each key is a string metadata type, such as "note", and each value is a list of strings for each entry in the metadata of the key type. So if the field has 3 note lines, the dictionary value for key "note" would be a 3 element list, holding the 3 note lines.
- **field_name** (*str*) – A convenience variable holding the field name, if it is found in the metadata

Constructor parameters:

Parameters **an_index** (*str*) – The A_i or N_i descriptor for this field in the IDD, where i is an integer 1-...

class pyiddidf.idd.objects.**IDDGroup**(*name*)

A simple class that defines a single IDD group. An IDD group is simply a container for IDD objects. Relevant members are listed here:

Variables

- **name** (*str*) – IDD Type, or name, of this group
- **objects** (*list(IDDObject)*) – A list of all objects found in the IDD within this group.

Constructor parameters:

Parameters **name** (*str*) – The group's name

class pyiddidf.idd.objects.**IDDObject**(*name*)

A simple class that defines a single IDD object. Relevant members are listed here:

Variables

- **name** (*str*) – IDD Type, or name, of this object

- **meta_data** (*dict(str, [str])*) – A dictionary, where each key is a string metadata type, such as "memo", and each value is a list of strings for each entry in the metadata of the key type. So if the object has 3 memo lines, the dictionary value for key "memo" would be a 3 element list, holding the 3 memo lines.
- **fields** (*list(IDDField)*) – A list of IDDField instances in order as read from the IDD

Constructor parameters:

Parameters **name** (*str*) – The object's type, or name

class pyiddidf.idd.objects.**IDDStructure**(*file_path*)

An IDD structure representation. This includes containing all the IDD objects (either inside groups or as standalone "single line objects"), as well as meta data such as the version ID for this IDD, and finally providing worker functions for accessing the IDD data

Relevant "public" members are listed here:

Variables

- **file_path** (*str*) – The path given when instantiating this IDD, not necessarily an actual path
- **version_float** (*float*) – The floating point representation of the version of this IDD (for 8.6.0 it is 8.6)
- **build_string** (*str*) – The abbreviated git SHA used when generating this IDD
- **single_line_objects** (*[str]*) – A list of strings, each representing a raw, single-token, name-only IDD object
- **groups** (*list(IDDGroup)*) – A list of all groups found in the IDD, each of which will contain IDD objects

Constructor parameters:

Parameters **file_path** (*str*) – A file path for this IDD; not necessarily a valid path as it is never used, just available for bookkeeping purposes.

get_object_by_type(*type_to_get*)

Given a type name, this returns the IDD object instance, or a single string if it is a single-line object

Parameters **type_to_get** – The name of the object to get, case-insensitive as it is compared insensitively inside

Returns If the object is a single-line object, simply the name; if the object is a full IDDObject instance, that instance is returned. If a match is not found, this returns None.

get_objects_with_meta_data(*meta_data*)

Given an object-level metadata string (required-object, e.g.), this returns objects that contain that metadata

Parameters **meta_data** – An object-level metadata string, such as required-object

Returns A list of IDDObjects that contain this metadata

CHAPTER 3

IDD Processor Module Documentation

class pyiddidf.idd.processor.**CurrentReadType**

Internal class containing constants for the different states of the actual IDD Processor engine

EncounteredComment_ReadToCR = 0

LookingForFieldMetaDataOrNextField = 11

LookingForFieldMetaDataOrNextObject = 10

LookingForObjectMetaDataOrNextField = 4

ReadAnything = 1

ReadingFieldANValue = 7

ReadingFieldMetaData = 8

ReadingFieldMetaDataOrNextANValue = 9

ReadingGroupDeclaration = 2

ReadingObjectMetaData = 5

ReadingObjectMetaDataContents = 6

ReadingObjectName = 3

class pyiddidf.idd.processor.**IDDProcessor**

The core IDD Processor class. Given an IDD via stream or path, this class has workers to robustly process the IDD into a rich IDDStructure instance.

The constructor takes no arguments but sets up instance variables. Relevant "public" members are listed here:

Variables

- **idd** (*IDDStructure*) – The resulting IDDStructure instance after processing the IDD file/stream
- **file_path** (*str*) – A file path for this IDD, although it may be just a simple descriptor

peek_one_char()
Internal worker function that reads a single character from the internal IDD stream but resets the stream to the former position

Returns A single character, the one immediately following the cursor, or None if it can't peek ahead.

process_file()
Internal worker function that reads the IDD stream, whether it was constructed from a file path, stream or string. This state machine worker moves character by character reading tokens and processing them into a meaningful IDD structure.

Returns An IDD structure describing the IDD contents

Raises **ProcessingException** – for any erroneous conditions encountered during processing

process_file_given_file_path(*file_path*)
This worker allows processing of an IDD file at a specific path on disk.

Parameters **file_path** – The path to an IDD file on disk.

Returns An IDDStructure instance created from processing the IDD file

Raises **ProcessingException** – if the specified file does not exist

process_file_via_stream(*idd_file_stream*)
This worker allows processing of an IDD snippet via stream. Most useful for unit testing, but possibly for other situations.

Parameters **idd_file_stream**(*file-like-object*) – An IDD snippet that responds to typical file-like commands such as read(). A common object would be the StringIO object.

Returns An IDDStructure instance created from processing the IDD snippet

process_file_via_string(*idd_string*)
This worker allows processing of an IDD snippet string. Most useful for unit testing, but possibly for other situations.

Parameters **idd_string**(*str*) – An IDD snippet string

Returns An IDDStructure instance created from processing the IDD string

read_one_char()
Internal worker function that reads a single character from the internal IDD stream, advancing the cursor.

Returns A single character, the one immediately following the cursor, or None if it can't read.

CHAPTER 4

IDF Object Module Documentation

class pyiddidf.idf.objects.**IDFObject**(*tokens*, *comment_blob=False*)

Bases: object

This class defines a single IDF object. An IDF object is either a comma/semicolon delimited list of actual object data, or a block of line delimited comments. Blocks of comment lines are treated as IDF objects so they can be intelligently written back out to a new IDF file after transition in the same location.

Relevant members are listed here:

Variables

- **object_name** (*str*) – IDD Type, or name, of this object
- **fields** (*[str]*) – A list of strings, one per field, found for this object in the IDF file

Constructor parameters:

Parameters

- **tokens** (*[str]*) – A list of tokens defining this idf object, the first token in the list is the object type.
- **comment_blob** (*bool*) – A signal that this list is comment data, and not an actual IDF object; default is False. indicating it is meaningful IDF data.

object_string(*idd_object=None*)

This function creates an intelligently formed IDF object. If the current instance is comment data, it simply writes the comment block out, line delimited, otherwise it writes out proper IDF syntax. If the matching IDD object is passed in as an argument, the field names are matched from that to create a properly commented IDF object.

Parameters **idd_object** (*IDDObject*) – The IDDObject structure that matches this IDFObject

Returns A string representation of the IDF object or comment block

validate(*idd_object*)

This function validates the current IDF object instance against standard IDD field tags such as minimum and maximum, etc.

Parameters **idd_object** (*IDDObject*) – The IDDObject structure that matches this IDFObject

Returns A list of ValidationIssue instances, each describing an issue encountered

write_object(*file_object*)
This function simply writes out the idf string to a file object

Parameters **file_object** – A file-type object that responds to a write command

Returns None

class pyiddidf.idf.objects.**IDFStructure**(*file_path*)
Bases: object

An IDF structure representation. This includes containing all the IDF objects in the file, as well as meta data such as the version ID for this IDD, and finally providing worker functions for accessing the IDD data

Relevant "public" members are listed here:

Variables

- **file_path** (*str*) – The path given when instantiating this IDF, not necessarily an actual path
- **version_float** (*float*) – The floating point representation of the version of this IDD (for 8.6.0 it is 8.6)
- **objects** (*[IDFObject]*) – A list of all IDF objects found in the IDF

Constructor parameters:

Parameters **file_path** (*str*) – A file path for this IDF; not necessarily a valid path as it is never used, just available for bookkeeping purposes.

get_idf_objects_by_type(*type_to_get*)
This function returns all objects of a given type found in this IDF structure instance

Parameters **type_to_get** (*str*) – A case-insensitive object type to retrieve

Returns A list of all objects of the given type

global_swap(*dict_of_swaps*)

validate(*idd_structure*)
This function validates the current IDF structure instance against standard IDD object tags such as required and unique objects.

Parameters **idd_structure** – An IDDStructure instance representing an entire IDD file

Returns A list of ValidationIssue instances, each describing an issue encountered

whole_idf_string(*idd_structure=None*)
This function returns a string representation of the entire IDF contents. If the idd structure argument is passed in, it is passed along to object worker functions in order to generate an intelligent representation.

Parameters **idd_structure** (*IDDStructure*) – An optional IDDStructure instance representing an entire IDD file

Returns A string of the entire IDF contents, ready to write to a file

write_idf(*idf_path*, *idd_structure=None*)
This function writes the entire IDF contents to a file. If the idd structure argument is passed in, it is passed along to object worker functions in order to generate an intelligent representation.

Parameters

- **idf_path** (*str*) – The path to the file to write
- **idd_structure** (*IDDStructure*) – An optional IDDStructure instance representing an entire IDD file

Returns None

class pyiddidf.idf.objects.**ValidationIssue**(*object_name*, *severity*, *message*, *field_name=None*)

This class stores information about any issue that occurred when reading an IDF file.

Parameters

- **object_name** (*str*) – The object type that was being validated when this issue arose
- **severity** (*int*) – The severity of this issue, from the class constants
- **message** (*str*) – A descriptive message for this issue
- **field_name** (*str*) – The field name that was being validated when this issue arose, if available.

ERROR = 2

INFORMATION = 0

WARNING = 1

static severity_string(*severity_integer*)

Returns a string version of the severity of this issue

Parameters **severity_integer** (*int*) – One of the constants defined in this class (INFORMATION, etc.)

Returns A string representation of the severity

CHAPTER 5

IDF Processor Module Documentation

class pyiddidf.idf.processor.**IDFProcessor**

The core IDF Processor class. Given an IDF via stream or path, this class has workers to robustly process the IDF into a rich IDFStructure instance.

The constructor takes no arguments but sets up instance variables. Relevant "public" members are listed here:

Variables

- **idf** (*IDFStructure*) – The resulting IDFStructure instance after processing the IDF file/stream
- **file_path** (*str*) – A file path for this IDF, although it may be just a simple descriptor

process_file()

Internal worker function that reads the IDF stream, whether it was constructed from a file path, stream or string. This processor then processes the file line by line looking for IDF objects and comment blocks, and parsing them into a meaningful structure

Returns An IDF structure describing the IDF contents

Raises **ProcessingException** – for any issues encountered during the processing of the idf

process_file_given_file_path(*file_path*)

This worker allows processing of an IDF file at a specific path on disk.

Parameters **file_path** – The path to an IDF file on disk.

Returns An IDFStructure instance created from processing the IDF file

Raises **ProcessingException** – if the specified file does not exist

process_file_via_stream(*idf_file_stream*)

This worker allows processing of an IDF snippet via stream. Most useful for unit testing, but possibly for other situations.

Parameters **idf_file_stream** (*file-like-object*) – An IDF snippet that responds to typical file-like commands such as read(). A common object would be the StringIO object.

Returns An IDFStructure instance created from processing the IDF snippet

process_file_via_string(*idf_string*)

This worker allows processing of an IDF snippet string. Most useful for unit testing, but possibly for other situations.

Parameters **idf_string** (*str*) – An IDF snippet string

Returns An IDFStructure instance created from processing the IDF string

CHAPTER 6

Indexes and tables

- genindex
- modindex
- search