ipuz Documentation

Release 0.1

Simeon Visser

December 29, 2015

Contents

Python library for reading and writing ipuz puzzle files. The specification for the ipuz file format can be found at: http://www.ipuz.org/. The ipuz file format supports representing various types of puzzles, including crossword, sudoku and word search. This Python library provides validation and wrapping around the puzzle data.

As the puzzle is inherently JSON data it is the application's responsibility to ensure that the JSON satisfies the constraints of the PuzzleKind prior to writing the puzzle. This library provides validation and additional functionality that you might want to use.

The library supports Python 2.7, Python 3.2, Python 3.3 and Python 3.4.

ipuz is a trademark of Puzzazz, Inc., used with permission.

CHAPTER 1

Contents

1.1 Reading ipuz puzzles

The string `data` contains the puzzle in JSON or JSONP format:

```
import ipuz

try:
    puzzle = ipuz.read(data)
except ipuz.IPUZException:
    # invalid puzzle
```

This function provides:

- Validation of puzzle structure in JSON or JSONP format.
- Validation of missing mandatory fields.
- Sanity checks for fields where possible.

The error messages are not intended as API but only for informative purposes. If you read a puzzle from a string and no exception was raised then your code should not need to perform validation to see whether the puzzle is well-formed. This library will check types of values and perform various sanity checks to see whether the JSON conforms to the ipuz specification.

1.1.1 ipuz.IPUZ_VERSIONS

You can use the constant `ipuz.IPUZ_VERSIONS` to see versions of the ipuz standard that the `ipuz` library can validate. This is a list containing the accepted integer values `X` for `"http://ipuz.org/vX"`.

Puzzles with an ipuz version that is not in this list won't be read by `ipuz.read` and this will result in an `ipuz.IPUZException` exception. The reason is that a future version of the standard may have introduced new features and attempting incomplete validation on such puzzles could lead to invalid puzzles being accepted by `ipuz.read`.

1.1.2 Validation for all puzzles

The `ipuz.read` function performs validation for fields that are common to all PuzzleKinds and validation for fields that are specific to a PuzzleKind. The function expects a puzzle in JSON or JSONP with either the default `ipuz` callback function or a differently named callback function.

Note that `true`, `false` and `null` in JSON / JSONP respectively become `True`, `False` and `None` in Python.

The follows checks are performed for fields that apply to all PuzzleKinds:

Field	Mandatory	Validation
version	Yes	Must be the string `"http://ipuz.org/vX"` where `X` is an integer of at least one.
kind	Yes	Must be a non-empty list of non-empty strings.
copyright	No	Must be a string.
publisher	No	Must be a string.
publication	No	Must be a string.
url	No	Must be a string.
uniqueid	No	Must be a string.
title	No	Must be a string.
intro	No	Must be a string.
explanation	No	Must be a string.
annotation	No	Must be a string.
author	No	Must be a string.
editor	No	Must be a string.
date	No	Must be a string with a date `"mm/dd/yyyy"`.
notes	No	Must be a string.
difficulty	No	Must be a string.
origin	No	Must be a string.
block	No	Must be a string.
empty	No	Must be a string or integer.
styles	No	Must be a dictionary with StyleSpec values.

1.1.3 Validation for Acrostic puzzles

The following checks are performed for PuzzleKinds belonging to `http://ipuz.org/acrostic`:

Field	Mandatory	Validation
puzzle	Yes	Must be a list of lists containing LabeledCell values.
solution	No	Must be a list of lists containing CrosswordValue values.
clues	No	Must be a dictionary with Direction keys and lists of Clue values.

1.1.4 Validation for Answer puzzles

The following checks are performed for PuzzleKinds belonging to `http://ipuz.org/answer`:

Field	Mandatory	Validation
choices	No	Must be a list of strings.
randomize	No	Must be a boolean.
answer	No	Must be a string.
answers	No	Must be a list of strings.
enumeration	No	Must be a string.
enumerations	No	Must be a list of strings.
requiredanswers	No	Must be an integer of at least zero.
misses	No	Must be a dictionary with string keys and string values.
guesses	No	Must be a list of strings.

1.1.5 Validation for Block puzzles

The following checks are performed for PuzzleKinds belonging to `http://ipuz.org/block`:

Field	Mandatory	Validation
dimensions	Yes	Must be a dictionary containing `"width"` and `"height"` keys with integer values of at least one.
slide	No	Must be a boolean.
move	No	Must be a boolean.
rotatable	No	Must be a boolean.
flippable	No	Must be a boolean.
field	No	Must be a list of lists containing StyledCell values.
enter	No	Must be a dictionary with string keys and GroupSpec values.
start	No	Must be a dictionary with string keys and GroupSpec values.
saved	No	Must be a dictionary with string keys and GroupSpec values.
end	No	Must be a dictionary with string keys and GroupSpec values.
exit	No	Must be a dictionary with string keys and GroupSpec values.

1.1.6 Validation for Crossword puzzles

The following checks are performed for PuzzleKinds belonging to `http://ipuz.org/crossword`:

Field	Mandatory	Validation
dimensions	Yes	Must be a dictionary containing `"width"` and `"height"` keys with integer values of at least one.
puzzle	Yes	Must be a list of lists containing LabeledCell values.
saved	No	Must be a list of lists containing CrosswordValue values.
solution	No	Must be a list of lists containing CrosswordValue values.
zones	No	Must be a list of GroupSpec values.
clues	No	Must be a dictionary with Direction keys and lists of Clue values.
showenumerations	No	Must be a boolean.
clueplacement	No	Must be an element from `["before", "after", "blocks", null].`
answer	No	Must be a string.
answers	No	Must be a list of strings.
enumeration	No	Must be a string.
enumerations	No	Must be a list of strings.
misses	No	Must be a dictionary with string keys and string values.

1.1.7 Validation for Fill puzzles

The following checks are performed for PuzzleKinds belonging to `http://ipuz.org/fill`:

Field	Mandatory	Validation
start	No	Must be a list of lists containing CrosswordValue values.
solution	No	Must be a list of lists containing CrosswordValue values.
answer	No	Must be a string.
answers	No	Must be a list of strings.
misses	No	Must be a dictionary with string keys and string values.

1.1.8 Validation for Sudoku puzzles

The following checks are performed for PuzzleKinds belonging to `http://ipuz.org/sudoku`:

Field	Mandatory	Validation
charset	No	Must be a string.
displaycharset	No	Must be a boolean.
boxes	No	Must be a boolean.
showoperators	No	Must be a boolean.
cageborder	No	Must be an element from `["thick", "dashed"]`.
puzzle	Yes	Must be a list of lists containing SudokuGiven values.
saved	No	Must be a list of lists containing SudokuGuess values.
solution	No	Must be a list of lists containing SudokuValue values.
zones	No	Must be a list of GroupSpec values.
cages	No	Must be a list of CalcSpec values.

1.1.9 Validation for WordSearch puzzles

The following checks are performed for PuzzleKinds belonging to `http://ipuz.org/wordsearch`:

Field	Mandatory	Validation
dimensions	Yes	Must be a dictionary containing `"width"` and `"height"` keys with integer values of at least one.
puzzle	No	Must be a list of lists containing CrosswordValue values.
solution	No	Must be a string, a list of strings or a dictionary with string keys and GroupSpec values.
dictionary	No	Must be a string or the boolean value `false`.
saved	No	Must be a list of strings.
showanswers	No	Must be an element from `["during", "after", null]`.
time	No	Must be an integer of at least zero.
points	No	Must be an element from `["linear", "log", null]`.
zigzag	No	Must be a boolean.
retrace	No	Must be a boolean.
useall	No	Must be a boolean.
misses	No	Must be a dictionary with string keys and string values.

1.1.10 Parameters

The `ipuz.read` function supports the following keyword parameters to configure what puzzles can be loaded:

- **`puzzlekinds`** Specifies the `"kind"` values that your application supports. This means `ipuz.read` only accepts puzzles where all `"kind"` values are in your list. For example, if your application only loads crossword puzzles you can use `puzzlekinds=["http://ipuz.org/crossword#1"]`. By default `ipuz.read` accepts all official PuzzleKinds in the ipuz specification.

  Note that this is a list of exact strings so if your application supports both version one and two of the Crossword PuzzleKind then you must specify `puzzlekinds=["http://ipuz.org/crossword#1", "http://ipuz.org/crossword#2"]`.

1.1.11 Extensions

For extensions to the ipuz specification the following validation is performed:

Field	Mandatory	Validation
volatile	No	Must be a dictionary with string keys and string values.

It is the application's responsibility to ensure that the volatility of the fields is handled properly.

1.2 Writing ipuz puzzles

The function `ipuz.write` converts a puzzle as a Python dictionary object to a JSON / JSONP string. This function performs no validation on the provided puzzle and it is the application's responsibility to construct a valid puzzle that can be read by `ipuz.read` or other applications.

The following writes a Python dictionary `puzzle` to a string `data`:

```python
import ipuz
data = ipuz.write(puzzle)
```

For security reasons this function encourages the use of JSON and it therefore produces a JSON string by default. You can create a JSONP string by using `jsonp=True`:

```python
import ipuz
data = ipuz.write(puzzle, jsonp=True)
```

By default the callback function `ipuz` is used in the JSONP format. You can specify a different callback function name by using the `callback_name` parameter:

```python
import ipuz
data = ipuz.write(puzzle, jsonp=True, callback_name="ipuz_function")
```

1.3 Contributing

Contributions to the ipuz library are very welcome.

1.3.1 Issues

Please create a new ticket in the issue tracker if you find an issue or if you have a suggestion for a new feature.

1.3.2 Contributing code

Code contributions should follow the PEP 8 style guide, include unit tests and update the documentation where needed.

You can install the necessary dependencies by running:

```
pip install -r dev-requirements.txt
```

1.3.3 Contributing documentation

You can update the documentation by making changes to the `.rst` files in the `docs` directory and running:

```
make html
```

After that you can view the HTML documentation by running:

```
open _build/html/index.html
```

1.3.4 Testing

You can run this library's tests by running:

```
python -m unittest discover
```

You can run the tests in all supported Python version by running:

```
tox
```

You can use coverage to check whether the tests adequately test the code.

Note that all tests run the public interface (`ipuz.read` and `ipuz.write`) to ensure we test both of the following:

- The functionality of any particular validation function.
- Whether the exception is raised properly at the top-level.

1.4 Changelog

1.4.1 0.1 (May 25, 2014)

- Initial release of the library.