CCL Scratch Tools

Jul 16, 2020

Contents:

A Python package with which to work with Scratch JSON files.

The project repository is hosted here: https://github.com/GSE-CCL/scratch-tools

CHAPTER 1

Usage

To use this package, make sure you have Python 3.7 or higher. Run pip install ccl-scratch-tools to install.

Then use the API, as described in this documentation.

CHAPTER 2

API documentation

class ccl_scratch_tools.**Parser**

A parser with which to parse Scratch projects.

Typical usage example:

from ccl_scratch_tools import Parser

parser = Parser()

blocks = parser.blockify("555555555.json")

blockify(*file_name=None*, *scratch_data=None*)

Gets the statistics about a Scratch project.

Parameters

- **file_name** (*str*) – the name of the Scratch JSON file to report on. (optional)
- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON. (optional)

Returns

A dictionary mapping function names (as in this class) to their results.

If the data are invalid Scratch 3, returns False.

Raises ValueError – If neither file_name or scratch_data parameter is set.

get_block(*block_id*, *scratch_data*)

Returns the block object in the Scratch object given block ID.

Parameters

- **block_id** (*str*) – the Scratch block ID in the project data structure.
- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns A dictionary containing the block's information from the project data structure. Returns False if doesn't exist or if trouble accessing the data.

get_block_comments(*scratch_data*)
Gets the comments left in a Scratch project, organized by block.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns A dictionary mapping blocks to a list of comments associated with them. If the input data is invalid, returns False.

get_block_name(*opcode*)
Gets the human-readable name of a Scratch block.

Parameters **opcode** (*str*) – the Scratch opcode of the block.

Returns

A string containing the block's human-readable name.

If the opcode isn't listed in our block information, returns None.

get_block_names(*items*, *scratch_data=None*)
Gets the human-readable name of a list of Scratch blocks.

Parameters

- **items** – a list of blocks to translate, whether a list of opcodes or a list of block IDs.
- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON. Include this only if items is a list of block IDs.

Returns

A list containing the blocks' human-readable names.

If the opcode isn't listed in our block information, that item of the list will be None.

get_block_text(*scratch_data*)
Gets the user-added block text, e.g. in Say blocks.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON. Include this only if items is a list of block IDs.

Returns

A dictionary mapping the user-added block text to a list of block IDs of blocks where it appears.

False if unsuccessful.

get_blocks(*scratch_data*, *include_orphans=True*)
Gets the blocks used in a Scratch project.

Parameters

- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.
- **include_orphans** (*bool) (optional*) – whether to include orphan blocks. Defaults to True.

Returns

A dictionary mapping used block opcodes to a list of block IDs (the IDs of the blocks of this type). Excludes unused blocks.

If the input data is invalid, returns False.

get_categories(*scratch_data*)
Gets the categories of blocks used in a Scratch project.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns

A dictionary mapping used block categories to the number of times they've been used.

If the input data is invalid, returns False.

get_child_blocks(*block_id*, *scratch_data*)
Gets the child blocks of a given block.

Parameters

- **block_id** (*str*) – The ID of the block whose children we're retrieving.
- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns A list of block IDs, each of which corresponds to a child block of block_id. The first element of the list will be the block_id argument passed in. Returns False if data are invalid.

get_comments(*scratch_data*)
Gets the comments left in a Scratch project.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns

A list of comments in the project.

If the input data is invalid, returns False.

get_costumes(*scratch_data*)
Gets the costumes used in a Scratch project.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns

A list of the names of the costumes used.

If the input data is invalid, returns False.

get_orphan_blocks(*scratch_data*)

Gets all the orphan blocks in a Scratch project. An orphan block is defined as an event listener with no children, or a non-event listener with no parents.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns A set of the block IDs that are orphan blocks. False if invalid data.

get_sounds(*scratch_data*)
Gets the sounds used in a Scratch project.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns

A list of the names of the sounds used.

If the input data is invalid, returns False.

get_sprite(*block_id*, *scratch_data*)
Gets the sprite with which a block is associated.

Parameters

- **block_id** (*str*) – the Scratch block ID in the project data structure.
- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns A dictionary containing the sprite's information from the project data structure, including sprite name, current costume asset ID, and current costume image URL. Returns False if doesn't exist or if trouble accessing the data.

get_sprite_names(*scratch_data*)
Get a list of sprite names, not including the stage targets.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns A list of sprite names. Stage targets are excluded. False if unsuccessful.

get_surrounding_blocks(*block_id*, *scratch_data*, *count=5*, *delve=False*)
Gets the surrounding blocks given a block ID.

Parameters

- **block_id** (*str*) – The ID of the block of which to capture surrounding blocks.
- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.
- **count** (*int*) – The maximum number of blocks you want to capture, including block_id. Defaults to 5.
- **delve** (*bool*) – Whether we should only return child blocks if this block has children, as with loops or event listeners.

Returns An ordered list of the IDs of the blocks surrounding block_id, including block_id in the middle. Returns False if the block_id doesn't exist, or if the data are invalid.

get_target(*block_id*, *scratch_data*)
Returns the target a block is part of.

Parameters

- **block_id** (*str*) – find the target this block is part of.
- **scratch_data** (*dict*) – the Scratch project to search through.

Returns

A tuple. First, a dictionary representing the relevant target; second, the index of this target in the project's target list.

Returns False if unsuccessful.

get_variables(*scratch_data*)
Gets the variables used in a Scratch project.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns

A list of the names of the variables used.

If the input data is invalid, returns False.

is_scratch3(*scratch_data*)
Checks a supposed Scratch data file against the Scratch 3 schema.

Parameters **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.

Returns True if the data matches the Scratch 3 schema. Otherwise False.

loop_through_blocks(*block_id*, *scratch_data*, *mode='next'*)
Loops through blocks in forward or backward direction.

Parameters

- **block_id** (*str*) – The ID of the block where we start our loop.
- **scratch_data** (*dict*) – a Python dictionary representing the imported Scratch JSON.
- **mode** (*str*) – The mode in which we're looping – "next" or "parent". Defaults to "next".

Returns A list of block IDs, each of which corresponds to a block related to block_id. The first element of the list will be the block_id argument passed in. Returns False if data or arguments are invalid.

class ccl_scratch_tools.**Scraper**(*studio_url=None*, *project_url=None*, *project_meta_url=None*, *comments_url=None*, *user_url=None*, *studio_meta_url=None*)
A scraper with which to scrape Scratch projects.

Typical usage example:

from ccl_scratch_tools import Scraper

scraper = Scraper()

project = scraper.download_project(555555555)

download_project(*id*)
Downloads an individual project JSON and returns it as a Python object.

Parameters **id** – An integer Scratch project ID.

Returns A dictionary object representing the Scratch project JSON.

Raises RuntimeError – An error occurred accessing the Scratch API, or the project couldn't be downloaded in/converted to JSON format.

download_projects(*ids*, *projects_to_studio={}*, *output_directory=None*, *file_name=None*)
Given project IDs, download the JSON files.

Parameters

- **ids** – array-like collection of Scratch project IDs.
- **projects_to_studio** – dictionary mapping project IDs to studio IDs. If set, creates subdirectories for each studio.
- **output_directory** (*str*) – directory for output; if not set, defaults to current working directory.
- **file_name** (*str*) – if set, combines projects into one JSON file with file_name; else creates a separate JSON file for each project.

Returns None.

get_id(*url*)
Returns the integer ID from a string that may be a URL or an ID.

Parameters **url** – The string representing the URL, or ID, to be extracted.

Returns

An integer ID of a Scratch object, whether a studio or project.

In case of error, returns None.

get_ids_from_file(*filename*)
Returns a list of IDs from a newline-separated file. Project/studio link agnostic. Works with links and IDs.

Parameters **filename** – String file name of a text file with line-separated URLs or IDs.

Returns A list of integer IDs. Empty if error reading file.

get_project_comments(*id*)
Returns the comments on a given Scratch project.

Parameters **id** (*int*) – a Scratch project ID.

Returns A list of dictionaries, each with keys for author, comment, and timestamp.

Raises RuntimeError – An error occurred accessing the Scratch API, or the project doesn't exist.

get_project_meta(*id*)
Returns the publicly-available metadata about a given Scratch project.

Parameters **id** (*int*) – a Scratch project ID.

Returns A dictionary with the entire API response from project meta API endpoint. None if the studio doesn't exist.

Raises RuntimeError – An error occurred accessing the Scratch API.

get_projects_in_studio(*id*)
Returns the set of project IDs contained in a given Scratch studio.

Parameters **id** – An integer Scratch studio ID.

Returns A set of project IDs.

Raises RuntimeError – An error occurred accessing the Scratch API.

get_studio_meta(*id*)
Returns the metadata for a given Scratch studio.

Parameters **id** – An integer Scratch studio ID.

Returns A dictionary with the studio's metadata. None if the studio doesn't exist.

Raises RuntimeError – An error occurred accessing the Scratch API.

get_user_info(*username*)
Gets a Scratch user's publicly-available information.

Parameters **username** (*str*) – the username to look up.

Returns A dictionary with the results of the API call.

Raises RuntimeError – An error occurred accessing the Scratch API, or the user doesn't exist.

make_dir(*path*)
Creates a directory given path.

Parameters **path** (*str*) – A file path on the current system.

Returns True, if directory was successfully created or already existed.

Raises RuntimeError – Failed to create the directory.

class ccl_scratch_tools.**Visualizer**
A way to visualize blocks by exporting to Scratchblocks syntax.

Typical usage example:

from ccl_scratch_tools import Visualizer

visualizer = Visualizer()

scratchblocks = visualizer.generate_scratchblocks(scratch_data)

generate_scratchblocks(*scratch_data*)
Generates all blocks in a project.

Parameters **scratch_data** (*dict*) – the Scratch project to process.

Returns A list of scripts in the project that can be turned into text. False if unsuccessful.

generate_script(*block_id*, *block_list*, *block_ids=None*, *text=False*, *find_block=True*)
Generates a script dictionary, nesting the blocks as appropriate.

Parameters

- **block_id** (*str*) – generate the script starting from this block.
- **block_list** (*array-like*) – the full list of blocks within this target.
- **block_ids** (*array-like) (optional*) – the set of block IDs that are allowed to be added. If not provided, then all blocks are allowed.
- **text** (*boolean) (optonal*) – whether to return this in Scratchblocks text syntax. If not provided, defaults to False - it'll just return the script dictionary.
- **find_block** (*bool) (optional*) – whether to find the closest parent block that can form a script, as when a script uses input blocks. Defaults to True.

Returns A dictionary with the nesting of this script as appropriate. False if unsuccessful.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index