
gsread Documentation

Release 3.1.0

Anton Burnashev

Dec 05, 2018

Contents

1	Installation	3
2	Example Usage	5
3	Authentication	7
3.1	Using OAuth2 for Authentication	7
4	More Examples	11
4.1	More examples of gspread usage	11
5	API Documentation	15
5.1	API Reference	15
6	How to Contribute	29
6.1	Ask Questions	29
6.2	Report Issues	29
6.3	Contribute code	29
7	Indices and tables	31
	Python Module Index	33

gsread is a Python API for Google Sheets.

Features:

- Google Sheets API v4.
- Open a spreadsheet by its **title** or **url**.
- Extract range, entire row or column values.
- Python 3 support.

CHAPTER 1

Installation

```
pip install gspread
```

Requirements: Python 2.7+ or Python 3+.

CHAPTER 2

Example Usage

1. Obtain OAuth2 credentials from Google Developers Console
2. Get a cell range from a spreadsheet:

```
import gspread

gc = gspread.authorize(credentials)

# Open a worksheet from spreadsheet with one shot
wks = gc.open("Where is the money Lebowski?").sheet1

wks.update_acell('B2', "it's down there somewhere, let me take another look.")

# Fetch a cell range
cell_list = wks.range('A1:B7')
```


3.1 Using OAuth2 for Authentication

3.1.1 OAuth Credentials


OAuth credentials can be generated in several different ways using the `oauth2client` library provided by Google. If you are editing spreadsheets for yourself then the easiest way to generate credentials is to use *Signed Credentials* stored in your application (see example below). If you plan to edit spreadsheets on behalf of others then visit the [Google OAuth2 documentation](#) for more information.

3.1.2 Using Signed Credentials

1. Head to [Google Developers Console](#) and create a new project (or select the one you have.)
2. Under “API & auth”, in the API enable “Drive API”.

API Library [Enabled APIs \(2\)](#)

Some APIs are enabled automatically. You can disable them if you're not using their services.

API ^	Quota	
Drive API	<div style="width: 0%;"></div> 0%	Disable
Drive SDK		Disable 

3. Go to “Credentials” and choose “New Credentials > Service Account Key”.

APIs

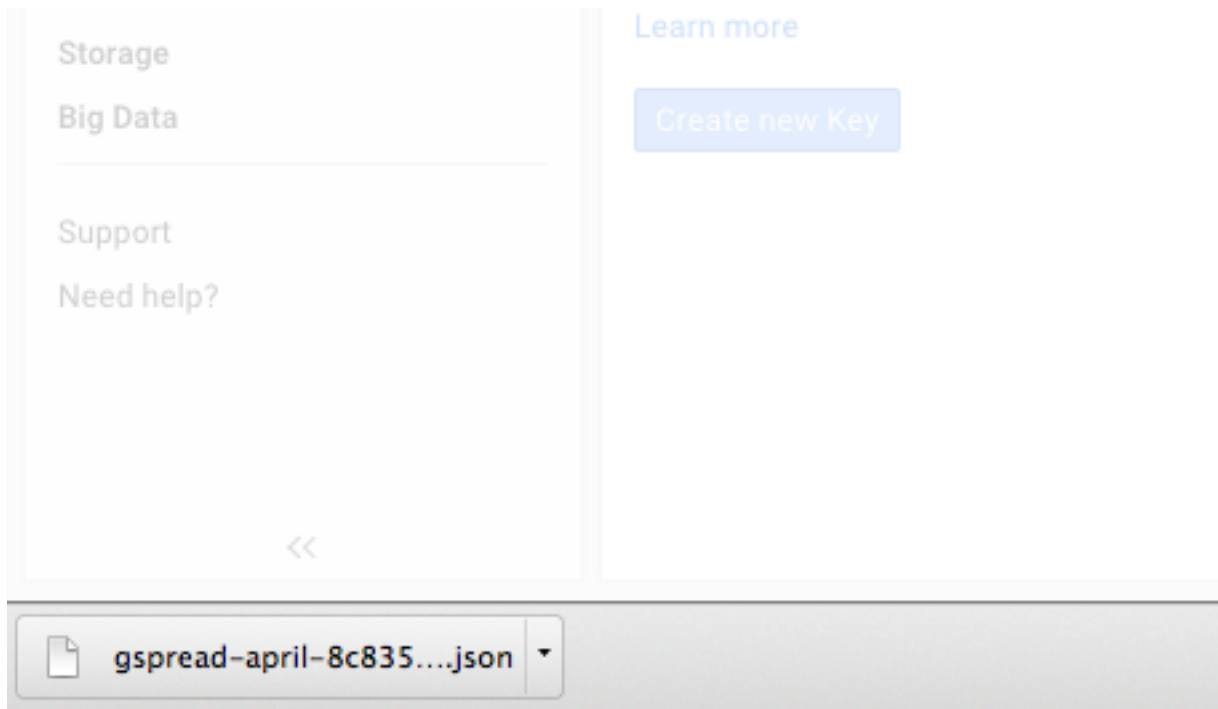
Credentials

You need credentials to access APIs. [Enable the APIs you plan to use](#) and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. [Refer to the API documentation](#) for details.

New credentials ▾

- API key
- OAuth client ID
- Service account key
- Help me choose

You will automatically download a JSON file with this data.



This is how this file may look like:

```
{
  "private_key_id": "2cd ... ba4",
  "private_key": "-----BEGIN PRIVATE KEY-----\nNrDyLw ... jINQh/9\n-----END PRIVATE_\nKEY-----\n",
  "client_email": "473000000000-yoursisdifferent@developer.gserviceaccount.com",
  "client_id": "473 ... hd.apps.googleusercontent.com",
  "type": "service_account"
}
```

In the next step you'll need the value of *client_email* from the file.

4. Go to your spreadsheet and share it with a *client_email* from the step above. Otherwise you'll get a `SpreadsheetNotFound` exception when trying to access this spreadsheet with `gsread`.
5. Install `oauth2client`:

```
pip install --upgrade oauth2client
```

Depending on your system setup you may need to install `PyOpenSSL`:

```
pip install PyOpenSSL
```

6. Now you can read this file, and use the data when constructing your credentials:

```
import gsread
from oauth2client.service_account import ServiceAccountCredentials

scope = ['https://spreadsheets.google.com/feeds',
         'https://www.googleapis.com/auth/drive']

credentials = ServiceAccountCredentials.from_json_keyfile_name('gsread-april-2cd ..._\nba4.json', scope)
```

(continues on next page)

(continued from previous page)

```
gc = gsread.authorize(credentials)
wks = gc.open("Where is the money Lebowski?").sheet1
```

3.1.3 Troubleshooting

oauth2client.client.CryptoUnavailableError: No crypto library available

If you're getting the "No crypto library available" exception, make sure you have PyOpenSSL library installed in your environment.

3.1.4 Custom Credentials Objects

If you have another method of authenticating you can easily hack a custom credentials object.

```
class Credentials (object):
    def __init__ (self, access_token=None):
        self.access_token = access_token

    def refresh (self, http):
        # get new access_token
        # this only gets called if access_token is None
```

4.1 More examples of gspread usage

4.1.1 Opening a Spreadsheet

You can open a spreadsheet by its title as it appears in Google Docs:

```
sh = gc.open('My poor gym results')
```

If you want to be specific, use a key (which can be extracted from the spreadsheet's url):

```
sht1 = gc.open_by_key('0BmgG6nO_6dprdS1MN3d3MkdPa142WFRrdnRRUWl1UFE')
```

Or, if you feel really lazy to extract that key, paste the entire spreadsheet's url

```
sht2 = gc.open_by_url('https://docs.google.com/spreadsheet/ccc?key=0Bm...FE&hl')
```

4.1.2 Creating a Spreadsheet

Use `create()` to create a new blank spreadsheet:

```
sh = gc.create('A new spreadsheet')
```

However, this new spreadsheet will be visible only to your script's account. To be able to access newly created spreadsheet you *must* share it with your email. Which brings us to...

4.1.3 Sharing a Spreadsheet

If your email is `otto@example.com` you can share the newly created spreadsheet with yourself:

```
sh.share('otto@example.com', perm_type='user', role='writer')
```

See `share()` documentation for a full list of accepted parameters.

4.1.4 Selecting a Worksheet

Select worksheet by index. Worksheet indexes start from zero:

```
worksheet = sh.get_worksheet(0)
```

Or by title:

```
worksheet = sh.worksheet("January")
```

Or the most common case: *Sheet1*:

```
worksheet = sh.sheet1
```

To get a list of all worksheets:

```
worksheet_list = sh.worksheets()
```

4.1.5 Creating a Worksheet

```
worksheet = sh.add_worksheet(title="A worksheet", rows="100", cols="20")
```

4.1.6 Deleting a Worksheet

```
sh.del_worksheet(worksheet)
```

4.1.7 Getting a Cell Value

Using A1 notation:

```
val = worksheet.acell('B1').value
```

Or row and column coordinates:

```
val = worksheet.cell(1, 2).value
```

If you want to get a cell formula:

```
cell = worksheet.acell('B1', value_render_option='FORMULA').value  
  
# or  
  
cell = worksheet.cell(1, 2, value_render_option='FORMULA').value
```


4.1.8 Getting All Values From a Row or a Column

Get all values from the first row:

```
values_list = worksheet.row_values(1)
```

Get all values from the first column:

```
values_list = worksheet.col_values(1)
```

4.1.9 Getting All Values From a Worksheet as a List of Lists

```
list_of_lists = worksheet.get_all_values()
```

4.1.10 Finding a Cell

Find a cell matching a string:

```
cell = worksheet.find("Dough")  
  
print("Found something at R%sC%s" % (cell.row, cell.col))
```

Find a cell matching a regular expression

```
amount_re = re.compile(r'(Big|Enormous) dough')  
cell = worksheet.find(amount_re)
```

4.1.11 Finding All Matched Cells

Find all cells matching a string:

```
cell_list = worksheet.findall("Rug store")
```

Find all cells matching a regexp:

```
criteria_re = re.compile(r'(Small|Room-tiering) rug')  
cell_list = worksheet.findall(criteria_re)
```

4.1.12 Cell Object

Each cell has a value and coordinates properties:

```
value = cell.value  
row_number = cell.row  
column_number = cell.col
```

4.1.13 Updating Cells

Using A1 notation:

```
worksheet.update_acell('B1', 'Bingo!')
```

Or row and column coordinates:

```
worksheet.update_cell(1, 2, 'Bingo!')
```

A more complicated example: fetch all cells in a range, change their values and send an API request that update cells in batch:

```
cell_list = worksheet.range('A1:C7')  
  
for cell in cell_list:  
    cell.value = 'O_o'  
  
# Update in batch  
worksheet.update_cells(cell_list)
```

5.1 API Reference

5.1.1 Top level

`gsread.authorize(credentials, client_class=<class 'gsread.client.Client'>)`

Login to Google API using OAuth2 credentials. This is a shortcut function which instantiates `gsread.client.Client` and performs login right away.

Returns `gsread.Client` instance.

5.1.2 Client

class `gsread.Client(auth, session=None)`

An instance of this class communicates with Google API.

Parameters

- **auth** – An OAuth2 credential object. Credential objects are those created by the `oauth2client` library. <https://github.com/google/oauth2client>
- **session** – (optional) A session object capable of making HTTP requests while persisting some parameters across requests. Defaults to `requests.Session`.

```
>>> c = gsread.Client(auth=OAuthCredentialObject)
```

copy (`file_id, title=None, copy_permissions=False`)

Copies a spreadsheet.

Parameters

- **file_id** – A key of a spreadsheet to copy.
- **title** (`str`) – (optional) A title for the new spreadsheet.

- **copy_permissions** (*bool*) – (optional) If True, copy permissions from original spreadsheet to new spreadsheet.

Returns a *Spreadsheet* instance.

New in version 3.1.0.

Note: In order to use this method, you need to add `https://www.googleapis.com/auth/drive` to your OAuth scope.

Example:

```
scope = [  
    'https://spreadsheets.google.com/feeds',  
    'https://www.googleapis.com/auth/drive'  
]
```

Otherwise you will get an `Insufficient Permission` error when you try to copy a spreadsheet.

create (*title*)

Creates a new spreadsheet.

Parameters **title** (*str*) – A title of a new spreadsheet.

Returns a *Spreadsheet* instance.

Note: In order to use this method, you need to add `https://www.googleapis.com/auth/drive` to your OAuth scope.

Example:

```
scope = [  
    'https://spreadsheets.google.com/feeds',  
    'https://www.googleapis.com/auth/drive'  
]
```

Otherwise you will get an `Insufficient Permission` error when you try to create a new spreadsheet.

del_spreadsheet (*file_id*)

Deletes a spreadsheet.

Parameters **file_id** (*str*) – a spreadsheet ID (aka file ID.)

import_csv (*file_id, data*)

Imports data into the first page of the spreadsheet.

Parameters **data** (*str*) – A CSV string of data.

Example:

```
# Read CSV file contents  
content = open('file_to_import.csv', 'r').read()  
  
gc.import_csv(spreadsheet.id, content)
```

Note: This method removes all other worksheets and then entirely replaces the contents of the first worksheet.

insert_permission (*file_id*, *value*, *perm_type*, *role*, *notify=True*, *email_message=None*, *with_link=False*)

Creates a new permission for a file.

Parameters

- **file_id** (*str*) – a spreadsheet ID (aka file ID.)
- **value** (*str*, *None*) – user or group e-mail address, domain name or *None* for ‘default’ type.
- **perm_type** (*str*) – (optional) The account type. Allowed values are: *user*, *group*, *domain*, *anyone*
- **role** – (optional) The primary role for this user. Allowed values are: *owner*, *writer*, *reader*
- **notify** (*str*) – (optional) Whether to send an email to the target user/domain.
- **email_message** (*str*) – (optional) An email message to be sent if *notify=True*.
- **with_link** (*bool*) – (optional) Whether the link is required for this permission to be active.

Examples:

```
# Give write permissions to otto@example.com

gc.insert_permission(
    '0BmgG6nO_6dprnRRUWl1UFE',
    'otto@example.org',
    perm_type='user',
    role='writer'
)

# Make the spreadsheet publicly readable

gc.insert_permission(
    '0BmgG6nO_6dprnRRUWl1UFE',
    None,
    perm_type='anyone',
    role='reader'
)
```

list_permissions (*file_id*)

Retrieve a list of permissions for a file.

Parameters **file_id** (*str*) – a spreadsheet ID (aka file ID.)

login ()

Authorize client.

open (*title*)

Opens a spreadsheet.

Parameters **title** (*str*) – A title of a spreadsheet.

Returns a *Spreadsheet* instance.

If there's more than one spreadsheet with same title the first one will be opened.

Raises `gsread.SpreadsheetNotFound` – if no spreadsheet with specified *title* is found.

```
>>> c = gsread.authorize(credentials)
>>> c.open('My fancy spreadsheet')
```

open_by_key (*key*)

Opens a spreadsheet specified by *key*.

Parameters *key* (*str*) – A key of a spreadsheet as it appears in a URL in a browser.

Returns a *Spreadsheet* instance.

```
>>> c = gsread.authorize(credentials)
>>> c.open_by_key('0BmgG6nO_6dprdS1MN3d3MkdPa142WFRrdnRRUW11UFE')
```

open_by_url (*url*)

Opens a spreadsheet specified by *url*.

Parameters *url* (*str*) – URL of a spreadsheet as it appears in a browser.

Returns a *Spreadsheet* instance.

Raises `gsread.SpreadsheetNotFound` – if no spreadsheet with specified *url* is found.

```
>>> c = gsread.authorize(credentials)
>>> c.open_by_url('https://docs.google.com/spreadsheet/ccc?key=0Bm...FE&hl')
```

openall (*title=None*)

Opens all available spreadsheets.

Parameters *title* (*str*) – (optional) If specified can be used to filter spreadsheets by title.

Returns a list of *Spreadsheet* instances.

remove_permission (*file_id*, *permission_id*)

Deletes a permission from a file.

Parameters

- **file_id** (*str*) – a spreadsheet ID (aka file ID.)
- **permission_id** (*str*) – an ID for the permission.

5.1.3 Models

The models represent common spreadsheet objects: a `spreadsheet`, a `worksheet` and a `cell`.

Note: The classes described below should not be instantiated by end-user. Their instances result from calling other objects' methods.

class `gsread.models.Spreadsheet` (*client*, *properties*)

The class that represents a spreadsheet.

add_worksheet (*title*, *rows*, *cols*)

Adds a new worksheet to a spreadsheet.

Parameters

- **title** (*str*) – A title of a new worksheet.

- **rows** (*int*) – Number of rows.
- **cols** (*int*) – Number of columns.

Returns a newly created `worksheets`.

batch_update (*body*)

Lower-level method that directly calls `spreadsheets.batchUpdate`.

Parameters **body** (*dict*) – Request body.

Returns Response body.

Return type `dict`

New in version 3.0.

del_worksheet (*worksheet*)

Deletes a worksheet from a spreadsheet.

Parameters **worksheet** (`Worksheet`) – The worksheet to be deleted.

duplicate_sheet (*source_sheet_id*, *insert_sheet_index=None*, *new_sheet_id=None*, *new_sheet_name=None*)

Duplicates the contents of a sheet.

Parameters

- **source_sheet_id** (*int*) – The sheet ID to duplicate.
- **insert_sheet_index** (*int*) – (optional) The zero-based index where the new sheet should be inserted. The index of all sheets after this are incremented.
- **new_sheet_id** (*int*) – (optional) The ID of the new sheet. If not set, an ID is chosen. If set, the ID must not conflict with any existing sheet ID. If set, it must be non-negative.
- **new_sheet_name** (*str*) – (optional) The name of the new sheet. If empty, a new name is chosen for you.

Returns a newly created `<gspread.models.Worksheet>`.

New in version 3.1.0.

get_worksheet (*index*)

Returns a worksheet with specified *index*.

Parameters **index** (*int*) – An index of a worksheet. Indexes start from zero.

Returns an instance of `gspread.models.Worksheet` or `None` if the worksheet is not found.

Example. To get first worksheet of a spreadsheet:

```
>>> sht = client.open('My fancy spreadsheet')
>>> worksheet = sht.get_worksheet(0)
```

id

Spreadsheet ID.

list_permissions ()

Lists the spreadsheet's permissions.

remove_permissions (*value*, *role='any'*)

Remove permissions from a user or domain.

Parameters

- **value** (*str*) – User or domain to remove permissions from
- **role** (*str*) – (optional) Permission to remove. Defaults to all permissions.

Example:

```
# Remove Otto's write permission for this spreadsheet
sh.remove_permissions('otto@example.com', role='writer')

# Remove all Otto's permissions for this spreadsheet
sh.remove_permissions('otto@example.com')
```

share (*value, perm_type, role, notify=True, email_message=None, with_link=False*)

Share the spreadsheet with other accounts.

Parameters

- **value** (*str, None*) – user or group e-mail address, domain name or None for ‘default’ type.
- **perm_type** (*str*) – The account type. Allowed values are: user, group, domain, anyone.
- **role** (*str*) – The primary role for this user. Allowed values are: owner, writer, reader.
- **notify** (*str*) – (optional) Whether to send an email to the target user/domain.
- **email_message** (*str*) – (optional) The email to be sent if notify=True
- **with_link** (*bool*) – (optional) Whether the link is required for this permission

Example:

```
# Give Otto a write permission on this spreadsheet
sh.share('otto@example.com', perm_type='user', role='writer')

# Transfer ownership to Otto
sh.share('otto@example.com', perm_type='user', role='owner')
```

sheet1

Shortcut property for getting the first worksheet.

title

Spreadsheet title.

updated

Deprecated since version 2.0.

This feature is not supported in Sheets API v4.

values_append (*range, params, body*)

Lower-level method that directly calls `spreadsheets.values.append`.

Parameters

- **range** (*str*) – The A1 notation of a range to search for a logical table of data. Values will be appended after the last row of the table.
- **params** (*dict*) – Query parameters.
- **body** (*dict*) – Request body.

Returns

Response body.

Return type dict

New in version 3.0.

values_clear (*range*)

Lower-level method that directly calls `spreadsheets.values.clear`.

Parameters **range** (*str*) – The A1 notation of the values to clear.

Returns

Response body.

Return type dict

New in version 3.0.

values_get (*range*, *params=None*)

Lower-level method that directly calls `spreadsheets.values.get`.

Parameters

- **range** (*str*) – The A1 notation of the values to retrieve.
- **params** (*dict*) – (optional) Query parameters.

Returns

Response body.

Return type dict

New in version 3.0.

values_update (*range*, *params=None*, *body=None*)

Lower-level method that directly calls `spreadsheets.values.update`.

Parameters

- **range** (*str*) – The A1 notation of the values to update.
- **params** (*dict*) – (optional) Query parameters.
- **body** (*dict*) – (optional) Request body.

Returns

Response body.

Return type dict

Example:

```
sh.values_update(
    'Sheet1!A2',
    params={
        'valueInputOption': 'USER_ENTERED'
    },
    body={
        'values': [[1, 2, 3]]
    }
)
```

New in version 3.0.

worksheet (*title*)

Returns a worksheet with specified *title*.

Parameters **title** (*int*) – A title of a worksheet. If there're multiple worksheets with the same title, first one will be returned.

Returns an instance of `gsperad.models.Worksheet`.

Example. Getting worksheet named 'Annual bonuses'

```
>>> sht = client.open('Sample one')
>>> worksheet = sht.worksheet('Annual bonuses')
```

worksheets ()

Returns a list of all worksheets in a spreadsheet.

class `gsread.models.Worksheet` (*spreadsheet, properties*)

The class that represents a single sheet in a spreadsheet (aka "worksheet").

acell (*label, value_render_option='FORMATTED_VALUE'*)

Returns an instance of a `gsread.models.Cell`.

Parameters

- **label** (*str*) – Cell label in A1 notation Letter case is ignored.
- **value_render_option** (*str*) – (optional) Determines how values should be rendered in the the output. See [ValueRenderOption](#) in the Sheets API.

Example:

```
>>> worksheet.acell('A1')
<Cell R1C1 "I'm cell A1">
```

add_cols (*cols*)

Adds columns to worksheet.

Parameters **cols** (*int*) – Number of new columns to add.

add_rows (*rows*)

Adds rows to worksheet.

Parameters **rows** (*int*) – Number of new rows to add.

append_row (*values, value_input_option='RAW'*)

Adds a row to the worksheet and populates it with values. Widens the worksheet if there are more values than columns.

Parameters

- **values** – List of values for the new row.
- **value_input_option** (*str*) – (optional) Determines how input data should be interpreted. See [ValueInputOption](#) in the Sheets API.

cell (*row, col, value_render_option='FORMATTED_VALUE'*)

Returns an instance of a `gsread.models.Cell` located at *row* and *col* column.

Parameters

- **row** (*int*) – Row number.
- **col** (*int*) – Column number.

- **value_render_option** (*str*) – (optional) Determines how values should be rendered in the the output. See [ValueRenderOption](#) in the Sheets API.

Example:

```
>>> worksheet.cell(1, 1)
<Cell R1C1 "I'm cell A1">
```

clear ()

Clears all cells in the worksheet.

col_count

Number of columns.

col_values (*col*, *value_render_option*='FORMATTED_VALUE')

Returns a list of all values in column *col*.

Empty cells in this list will be rendered as None.

Parameters

- **col** (*int*) – Column number.
- **value_render_option** (*str*) – (optional) Determines how values should be rendered in the the output. See [ValueRenderOption](#) in the Sheets API.

delete_row (*index*)

“Deletes the row from the worksheet at the specified index.

Parameters **index** (*int*) – Index of a row for deletion.

duplicate (*insert_sheet_index*=None, *new_sheet_id*=None, *new_sheet_name*=None)

Duplicate the sheet.

Parameters

- **insert_sheet_index** (*int*) – (optional) The zero-based index where the new sheet should be inserted. The index of all sheets after this are incremented.
- **new_sheet_id** (*int*) – (optional) The ID of the new sheet. If not set, an ID is chosen. If set, the ID must not conflict with any existing sheet ID. If set, it must be non-negative.
- **new_sheet_name** (*str*) – (optional) The name of the new sheet. If empty, a new name is chosen for you.

Returns a newly created `<gsread.models.Worksheet>`.

New in version 3.1.0.

export (*format*)

Deprecated since version 2.0.

This feature is not supported in Sheets API v4.

find (*query*)

Finds the first cell matching the query.

Parameters **query** (*str*, *re.RegexObject*) – A literal string to match or compiled regular expression.

findall (*query*)

Finds all cells matching the query.

Parameters **query** (*str*, *re.RegexObject*) – A literal string to match or compiled regular expression.

get_all_records (*empty2zero=False*, *head=1*, *default_blank=""*, *allow_underscores_in_numeric_literals=False*)

Returns a list of dictionaries, all of them having the contents of the spreadsheet with the head row as keys and each of these dictionaries holding the contents of subsequent rows of cells as values.

Cell values are numericised (strings that can be read as ints or floats are converted).

Parameters

- **empty2zero** (*bool*) – (optional) Determines whether empty cells are converted to zeros.
- **head** (*int*) – (optional) Determines which row to use as keys, starting from 1 following the numeration of the spreadsheet.
- **default_blank** (*str*) – (optional) Determines whether empty cells are converted to something else except empty string or zero.
- **allow_underscores_in_numeric_literals** (*bool*) – (optional) Allow underscores in numeric literals, as introduced in PEP 515

get_all_values ()

Returns a list of lists containing all cells' values as strings.

Note: Empty trailing rows and columns will not be included.

id

Worksheet ID.

insert_row (*values*, *index=1*, *value_input_option='RAW'*)

Adds a row to the worksheet at the specified index and populates it with values.

Widens the worksheet if there are more values than columns.

Parameters

- **values** – List of values for the new row.
- **index** (*int*) – (optional) Offset for the newly inserted row.
- **value_input_option** (*str*) – (optional) Determines how input data should be interpreted. See [ValueInputOption](#) in the Sheets API.

range (**args*, ***kwargs*)

Returns a list of *Cell* objects from a specified range.

Parameters name (*str*) – A string with range value in A1 notation, e.g. 'A1:A5'.

Alternatively, you may specify numeric boundaries. All values index from 1 (one):

Parameters

- **first_row** (*int*) – Row number
- **first_col** (*int*) – Row number
- **last_row** (*int*) – Row number
- **last_col** (*int*) – Row number

Example:

```

>>> # Using A1 notation
>>> worksheet.range('A1:B7')
[<Cell R1C1 "42">, ...]

>>> # Same with numeric boundaries
>>> worksheet.range(1, 1, 7, 2)
[<Cell R1C1 "42">, ...]

```

resize (*rows=None, cols=None*)

Resizes the worksheet. Specify one of *rows* or *cols*.

Parameters

- **rows** (*int*) – (optional) New number of rows.
- **cols** (*int*) – (optional) New number columns.

row_count

Number of rows.

row_values (*row, value_render_option='FORMATTED_VALUE'*)

Returns a list of all values in a *row*.

Empty cells in this list will be rendered as `None`.

Parameters

- **row** (*int*) – Row number.
- **value_render_option** (*str*) – (optional) Determines how values should be rendered in the the output. See [ValueRenderOption](#) in the Sheets API.

title

Worksheet title.

update_acell (*label, value*)

Updates the value of a cell.

Parameters

- **label** (*str*) – Cell label in A1 notation. Letter case is ignored.
- **value** – New value.

Example:

```
worksheet.update_acell('A1', '42')
```

update_cell (*row, col, value*)

Updates the value of a cell.

Parameters

- **row** (*int*) – Row number.
- **col** (*int*) – Column number.
- **value** – New value.

Example:

```
worksheet.update_cell(1, 1, '42')
```

update_cells (*cell_list, value_input_option='RAW'*)

Updates many cells at once.

Parameters

- **cell_list** – List of *Cell* objects to update.
- **value_input_option** (*str*) – (optional) Determines how input data should be interpreted. See [ValueInputOption](#) in the Sheets API.

Example:

```
# Select a range
cell_list = worksheet.range('A1:C7')

for cell in cell_list:
    cell.value = 'O_o'

# Update in batch
worksheet.update_cells(cell_list)
```

update_title (*title*)

Renames the worksheet.

Parameters **title** (*str*) – A new title.

updated

Deprecated since version 2.0.

This feature is not supported in Sheets API v4.

class `gsread.models.Cell` (*row, col, value=""*)

An instance of this class represents a single cell in a *worksheet*.

col

Column number of the cell.

input_value

Deprecated since version 2.0.

This feature is not supported in Sheets API v4.

row

Row number of the cell.

value = None

Value of the cell.

5.1.4 Utils

gsread.utils

This module contains utility functions.

`gsread.utils.rowcol_to_a1` (*row, col*)

Translates a row and column cell address to A1 notation.

Parameters

- **row** (*int, str*) – The row of the cell to be converted. Rows start at index 1.
- **col** – The column of the cell to be converted. Columns start at index 1.

Returns a string containing the cell's coordinates in A1 notation.

Example:

```
>>> rowcol_to_a1(1, 1)
A1
```

`gsread.utils.a1_to_rowcol` (*label*)

Translates a cell's address in A1 notation to a tuple of integers.

Parameters `label` (*str*) – A cell label in A1 notation, e.g. 'B1'. Letter case is ignored.

Returns a tuple containing *row* and *column* numbers. Both indexed from 1 (one).

Example:

```
>>> a1_to_rowcol('A1')
(1, 1)
```

5.1.5 Exceptions

exception `gsread.exceptions.GSreadException`

A base class for gsread's exceptions.

exception `gsread.exceptions.APIError` (*response*)

Please make sure to take a moment and read the [Code of Conduct](#).

6.1 Ask Questions

The best way to get an answer to a question is to ask on [Stack Overflow](#) with a `gspread` tag.

6.2 Report Issues

Please report bugs and suggest features via the [GitHub Issues](#).

Before opening an issue, search the tracker for possible duplicates. If you find a duplicate, please add a comment saying that you encountered the problem as well.

6.3 Contribute code

Please make sure to read the [Contributing Guide](#) before making a pull request.

CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

g

`gsread`, 15

`gsread.utils`, 26

A

`al_to_rowcol()` (in module `gsread.utils`), 27
`acell()` (`gsread.models.Worksheet` method), 22
`add_cols()` (`gsread.models.Worksheet` method), 22
`add_rows()` (`gsread.models.Worksheet` method), 22
`add_worksheet()` (`gsread.models.Spreadsheet` method), 18
`APIError`, 27
`append_row()` (`gsread.models.Worksheet` method), 22
`authorize()` (in module `gsread`), 15

B

`batch_update()` (`gsread.models.Spreadsheet` method), 19

C

`Cell` (class in `gsread.models`), 26
`cell()` (`gsread.models.Worksheet` method), 22
`clear()` (`gsread.models.Worksheet` method), 23
`Client` (class in `gsread`), 15
`col` (`gsread.models.Cell` attribute), 26
`col_count` (`gsread.models.Worksheet` attribute), 23
`col_values()` (`gsread.models.Worksheet` method), 23
`copy()` (`gsread.Client` method), 15
`create()` (`gsread.Client` method), 16

D

`del_spreadsheet()` (`gsread.Client` method), 16
`del_worksheet()` (`gsread.models.Spreadsheet` method), 19
`delete_row()` (`gsread.models.Worksheet` method), 23
`duplicate()` (`gsread.models.Worksheet` method), 23
`duplicate_sheet()` (`gsread.models.Spreadsheet` method), 19

E

`export()` (`gsread.models.Worksheet` method), 23

F

`find()` (`gsread.models.Worksheet` method), 23
`findall()` (`gsread.models.Worksheet` method), 23

G

`get_all_records()` (`gsread.models.Worksheet` method), 23
`get_all_values()` (`gsread.models.Worksheet` method), 24
`get_worksheet()` (`gsread.models.Spreadsheet` method), 19
`gsread` (module), 15
`gsread.utils` (module), 26
`GSreadException`, 27

I

`id` (`gsread.models.Spreadsheet` attribute), 19
`id` (`gsread.models.Worksheet` attribute), 24
`import_csv()` (`gsread.Client` method), 16
`input_value` (`gsread.models.Cell` attribute), 26
`insert_permission()` (`gsread.Client` method), 17
`insert_row()` (`gsread.models.Worksheet` method), 24

L

`list_permissions()` (`gsread.Client` method), 17
`list_permissions()` (`gsread.models.Spreadsheet` method), 19
`login()` (`gsread.Client` method), 17

O

`open()` (`gsread.Client` method), 17
`open_by_key()` (`gsread.Client` method), 18
`open_by_url()` (`gsread.Client` method), 18
`openall()` (`gsread.Client` method), 18

R

`range()` (`gsread.models.Worksheet` method), 24
`remove_permission()` (`gsread.Client` method), 18

`remove_permissions()` (*gsread.models.Spreadsheet method*), 19
`resize()` (*gsread.models.Worksheet method*), 25
`row` (*gsread.models.Cell attribute*), 26
`row_count` (*gsread.models.Worksheet attribute*), 25
`row_values()` (*gsread.models.Worksheet method*), 25
`rowcol_to_a1()` (*in module gsread.utils*), 26

S

`share()` (*gsread.models.Spreadsheet method*), 20
`sheet1` (*gsread.models.Spreadsheet attribute*), 20
`Spreadsheet` (*class in gsread.models*), 18

T

`title` (*gsread.models.Spreadsheet attribute*), 20
`title` (*gsread.models.Worksheet attribute*), 25

U

`update_acell()` (*gsread.models.Worksheet method*), 25
`update_cell()` (*gsread.models.Worksheet method*), 25
`update_cells()` (*gsread.models.Worksheet method*), 25
`update_title()` (*gsread.models.Worksheet method*), 26
`updated` (*gsread.models.Spreadsheet attribute*), 20
`updated` (*gsread.models.Worksheet attribute*), 26

V

`value` (*gsread.models.Cell attribute*), 26
`values_append()` (*gsread.models.Spreadsheet method*), 20
`values_clear()` (*gsread.models.Spreadsheet method*), 21
`values_get()` (*gsread.models.Spreadsheet method*), 21
`values_update()` (*gsread.models.Spreadsheet method*), 21

W

`Worksheet` (*class in gsread.models*), 22
`worksheet()` (*gsread.models.Spreadsheet method*), 21
`worksheets()` (*gsread.models.Spreadsheet method*), 22