Autodrive Documentation

Release 0.6.4

Chris Larabee

Dec 27, 2022

CONTENTS

CHAPTER

ONE

BASIC USAGE

Note: Because Google calls spreadsheets "Sheets", and also refers to the individual sub-sheets in a spreadsheet as "Sheets", Autodrive refers to the sub-sheets as **Tabs** for clarity.

Autodrive is designed to be as easy to navigate right in your IDE as possible, with broad type annotation to facilitate code completion.

1.1 Interacting With Google Drive

Autodrive's most basic class is the *Drive* class, which can be easily instantiated with your downloaded credentials. With a *Drive* instance you can find and/or create files and folders in your Google Drive, or in shared drives if you have access to them.

```
from autodrive import Drive

# Simply instantiate a drive instance to authenticate using your saved credentials
# file.
drive = Drive()

folder = drive.find_folder("Autodrive Folder")
print(folder[0].name)
# Autodrive Folder
```

1.2 Views

Autodrive provides three different views of a Google Sheet, the *GSheet*, which contains properties of the sheet itself and all its *Tabs*. *Tabs* provide convenient access to methods allowing you to manipulate values and formats of individual tabs. Finally, *Range* represents a specific row-column range in a *Tab* (i.e. A1:C3), and provides methods for altering the values and/or formats of that range alone.

These views are designed to be flexible, and you can use any of them to easily interact with the same part of the Google Sheet. For example, below is an example of instantiating a *GSheet*, *Tab*, and *Range*, all of which would, for example, write values to the same cells of the Google Sheet:

```
from autodrive import GSheet, Tab, Range, FullRange

gsheet = GSheet(gsheet_id="19k5cT9Klw1CA8Sum-olP7C0JUo6_kMiOAKDEeHPiSr8")
```

(continues on next page)

(continued from previous page)

```
# Calling write_values on gsheet will write the data to the first tab:
gsheet.write_values(
    [
        [1, 2, 3, 4],
        [5, 6, 7, 8],
    ]
)

tab = Tab(
    gsheet_id="19k5cT9Klw1CA8Sum-olP7C0JUo6_kMiOAKDEeHPiSr8",
    tab_title="Sheet1",
    tab_idx=0,
    tab_id=0
)
# Calling write_values on tab will write the data starting with cell A1:
tab.write_values(
    [
        [1, 2, 3, 4],
        [5, 6, 7, 8],
    ]
)

rng = Range(
    gsheet_range=FullRange("A1:Z1000"),
    gsheet_id="19k5cT9Klw1CA8Sum-olP7C0JUo6_kMiOAKDEeHPiSr8",
    tab_title="Sheet1",
)
# Calling_write values on rng will write the data starting with the first cell
# in the Range:
rng.write_values(
    [
        [1, 2, 3, 4],
        [5, 6, 7, 8],
    ]
)
```

As you can see, these views are nested within one each other as well, so if you have a *Tab* but want to create a *Range* off it for greater convenience, you can easily do so:

```
tab = gsheet.tabs["Sheet1"]

rng = tab.gen_range(FullRange("G1:G"))
```

Note: Integrating with Pandas

While Autodrive views cannot directly accept pandas DataFrames, DataFrames provide methods to make it easy to convert them into a format accepted by Autodrive views:

```
import pandas as pd

df = pd.DataFrame([dict(a=1, b=2, c=3), dict(a=4, b=5, c=6)])
```

(continues on next page)

(continued from previous page)

```
# You can either pass the DataFrame as a list of lists:
tab.write_values(df.values.tolist())

# Or, if you want to automatically include the header, you can pass it
# as a list of dictionaries:
tab.write_values(df.to_dict("records"))
```

1.3 Appending Data

GSheets and *Tabs* can also be used to append values to the end of a tab instead of to a range. Simply switch the available `mode` argument on the `write_values` method to `"a"` or `"append"`. For example, to write more data to the tab from the earlier example:

```
tab.write_values(
    [
        [9, 10, 11, 12],
        [13, 14, 15, 16],
    ],
    mode="a"
)
tab.commit()
tab.get_data()
print(tab.values)
# [
#   [1, 2, 3, 4],
#   [5, 6, 7, 8],
#   [9, 10, 11, 12],
#   [13, 14, 15, 16],
# ]
```

This makes it easy to keep updating a tab's data without having to check what row you should start your ranges at all the time. This will also add new rows to the tab if necessary, so you don't have to check if the tab has enough rows before appending.

CHAPTER

TWO

INTERFACES

Interfaces are essentially argument groupings that are consumed as parameters by many different methods throughout Autodrive. They are designed to provide type hinting and checking, as well as data validation.

2.1 Range Interfaces

Range interfaces serve as guides to constructing valid start and end ranges for a Google Sheet. They essentially correspond to a Google Sheets range like `A1:D50`. There are two types of range interfaces in Autodrive, *FullRanges* and the less common *HalfRanges*.

```
from autodrive.interfaces import FullRange, HalfRange

# FullRanges are much more commonly used, and correspond to a grid of cells in
# Google Sheets. For example, a FullRange covering cells from the upper-left-
# most cell in a Google Sheet to the 4th column (D) and the 50th row could be
# instantiated like:
full = FullRange("A1:D50")

# HalfRanges are less common, only being used by certain formatting methods.
# They represent a single row or column range in Google Sheets. For example,
# you could select the entire top row of a Google Sheet with:
half = HalfRange(start_idx="A", end_index="Z")  # A1:Z
# Or you could select a part of a certain column:
half = HalfRange(start_idx=1, end_index=50)  # A1:A50
# Or the entire column:
half = HalfRange(start_idx="A")  # A1
```

2.2 Formatting Interfaces

Formatting interfaces are fairly straightforward. They simply provide an easy, type-hinted way of helping you construct formatting update requests.

```
from autodrive.interfaces import Color, TextFormat

# You can easily instantiate a Color interface with RGBa color values:
c = Color(235, 229, 52)  # hex: #ebe534
```

(continues on next page)

(continued from previous page)

```
# And TextFormat provides a guide for applying different fonts and the like:
tf = TextFormat(bold=True, font_size=14, font="Calibri")
```

2.3 More Info

For more information about the various interfaces, see the relevant *Interfaces* Documentation.

CHAPTER

THREE

BATCH UPDATING AND REQUESTS

Wherever possible, Autodrive creates batches of update requests and collects them within whatever View you call the relevant methods from. This is to minimize round trips to the Google APIs and back, and give you full control over the timing and size of your interactions with the APIs.

Note: Some methods (particularly on *Connection* objects) require that a request be sent to the Google APIs immediately. The documentation for the method will have a note (like this one) indicating this when a request will be sent upon execution.

All `write_values()` methods on views and all methods provided by *Formatting* properties add a request to themselves or their parent view. These requests can all be executed by calling the view's `commit()` method. For example:

```
from autodrive import Tab

# Adds a request to write the specified values to the tab:
tab.write_values(
    [
        [1, 2, 3],
        [4, 5, 6],
    ]
)
# Adds a request to have every other row have a colorful background:
tab.format_cell.add_alternating_row_background(Color(0.5, 1, 0.5))
# Adds a request to fit the width of the tab's columns to the width of their
# values:
tab.format_grid.auto_column_width()

# All three of the above updates to the Tab can then be executed at once:
tab.commit()

# The commit() method sends the requests to the Google Sheets API and clears
# the view's request queue:
print(tab.requests)
# []
```

Note: If you're curious, you can examine the requests queued on a view through its `requests` property (as seen above). If you'll do, you'll note that the requests are nothing more than dictionaries with all the relevant information required by the Google Sheets API to process their updates.

CHAPTER

FOUR

CUSTOM CONNECTION CONFIGURATION

Autodrive is designed to be fully customizable in terms of connection configuration. Anything that connects to your Google Drive or a Google Sheet can have its default connection behavior overridden. For most use-cases, simply having your credentials saved as a `credentials.json` file on your working directory will meet your needs, but for special cases, you can customize this functionality simply by instantiating your own *AuthConfig* and passing that to your Views, *Folders*, or *Drive*.

4.1 Files

For example, you can customize the path to your credentials file and where the resulting token will get saved:

```
config = AuthConfig(
    token_filepath="custom_token.json",
    creds_filepath="/path/to/somewhere/else/creds.json"
)
```

4.2 Environment Variables

You can also bypass the file-loading behavior entirely and inject your credentialling information with environment variables:

```
# You can get these from your .json token after connecting for the first time:
AUTODRIVE_TOKEN="your_token_here"
AUTODRIVE_REFR_TOKEN="your_refresh_token_here"
# You can find these in GCP or in your credentials.json file:
AUTODRIVE_CLIENT_ID="your_client_id_value"
AUTODRIVE_CLIENT_SECRET="your_client_secret"
```

If these variables are present in your environment, then Autodrive will automatically detect them and use them to connect instead of looking for a file.

4.3 Config Dictionary

Finally, while it is probably not advisable, you can also pass your credentials directly to *AuthConfig* after loading them yourself:

```
config = AuthConfig(
    secrets_config={
        "client_id": "your_client_id_value",
        "client_secret": "your_client_secret",
        "token": "your_token_here",
        "refresh_token": "your_refresh_token_here"
    }
)
```

Warning: If you do need to use this option to load them for some reason, absolulely do **NOT** put your credentials in your source code!

CHAPTER

FIVE

FORMATTING

The *Tabs* and *Ranges* both have a suite of *Formatting* attributes, which provide a wide range of methods that, when called, add a request to update the connected Google Sheet's formatting once committed. They are grouped into three areas, `cell`, `grid`, and `text`. These breakdowns are designed to make it as convenient as possible to locate the desired formatting you wish to apply right in your IDE.

5.1 Cell

Cell formatting methods update the features of the cell(s) targeted, providing easy ways to add backgrounds, borders, and the like:

```
# This will add conditional formatting to the whole tab that will make every
# other row have a different color:
tab.format_cell.add_alternating_row_background(Color(0.5, 1, 0.5))
```

5.2 Grid

Grid formatting methods update the underlying grid of a tab, including the size of cells, the number of rows and columns, and other features of the Google Sheet:

```
# This will set the width of all columns in the tab to the width of their
# contents, as if you'd double-clicked on the right edge of the column
# divider:
rng.format_grid.auto_column_width()
```

Note: Not all formatting methods are duplicated between *Tab* and *Range*. For example, *Tab.format_grid* provides a number of methods relating to inserting and deleting rows that wouldn't make sense to apply to a *Range*.

5.3 Text

Finally, text formatting methods update the textual formatting of the cell contents. This includes anything that can appear within the cell, so despite the name, number formatting is also applied via text formatting methods:

```
from autodrive.interfaces import AccountingFormat

# Will apply the "accounting" mask to any numeric contents in the cell(s):
rng.format_text.apply_format(AccountingFormat)
```

5.4 Chaining

All the formatting methods return their parent Formatting object, so you can easily generate a bunch of different formatting requests by chaining methods together:

```
(
    tab.format_grid
    .insert_rows(num_rows=5, at_row=10)
    .insert_rows(num_rows=1, at_row=40)
    .append_rows(num_rows=20)
)
```

Warning: When queueing multiple formatting requests at once, be aware that the Google API treats batched formatting updates on the same cells in a particular way. If the different requests have partially overlapping ranges, then the later request will *remove* the formatting of any previous requests made to update the overlapping cells' format. Essentially, this means that you should only batch formatting requests together if they all affect the exact same range of cells, or if they all affect completely different rnages of cells.

5.5 More Info

There are many more methods provided by the *Tab* and *Range* objects' `format_cell`, `format_grid`, and `format_text` properties than are covered here. For a complete listing of the currently available methods, see the *Formatting* documentation.

CHAPTER

SIX

SIMPLE OUTPUT

Autodrive offers some simple output functionality for creating csv and json files from the data in your *Range*, *Tab*, and *GSheet* objects.

6.1 Outputting Ranges and Tabs

Outputting the fetched data for *Range* and *Tab* objects is quite easy, all you really need to do is specify a path to write to:

```
tab.to_csv("path/to/your/file.csv")
# Optionally you can insert a header row:
tab.to_csv("path/to/your/file.csv", header=["column_a", "column_b", "column_c"])

tab.to_json("path/to/your/file.json")
# For to_json, you can specifiy the header row manually or with an index
# in the data:
tab.to_json("path/to/your/file.json", header=["column_a", "column_b", "column_c"])
tab.to_json("path/to/your/file.json", header=2)  # for the data in the 3rd row:
```

6.2 Outputting GSheets

Outputting *GSheet* objects is only slightly more complicated. If you only specify a path to a directory (not a file), then every *Tab* in the *GSheet* will be output as individual files. For example, if you have an object with three tabs (named `Sheet1`, `Sheet2`, and `Sheet3`), then this will output 3 files:

```
import os

gsheet.to_csv("path/to/a/directory")
gsheet.to_json("path/to/a/directory")

os.listdir("path/to/a/directory")
"""
[
    "path/to/a/directory/Sheet1.csv",
    "path/to/a/directory/Sheet1.json",
    "path/to/a/directory/Sheet2.csv",
    "path/to/a/directory/Sheet2.json",
    "path/to/a/directory/Sheet3.csv",
```

(continues on next page)

(continued from previous page)

```
    "path/to/a/directory/Sheet3.json",
]
"""
```

6.2.1 Overriding Filenames

By default, the files will be named the same as the corresponding tab titles, but you can override this behavior:

```
# Only the tab names you pass will be overridden:
gsheet.to_csv(
    "path/to/a/directory",
    filename_overrides={"Sheet1": "sums", "Sheet2": "averages"}
)
gsheet.to_json(
    "path/to/a/directory",
    filename_overrides={"Sheet1": "sums"}
)

os.listdir("path/to/a/directory")
"""
[
    "path/to/a/directory/sums.csv",
    "path/to/a/directory/sums.json",
    "path/to/a/directory/averages.csv",
    "path/to/a/directory/Sheet2.json",
    "path/to/a/directory/Sheet3.csv",
    "path/to/a/directory/Sheet3.json",
]
"""
```

6.2.2 Headers and Tab Subsets

You can also simultaneously limit the specific tabs output and/or set the header(s) output with each file:

```
# If you specify any Tab names as kwargs, only the tabs you specify will be
# output:
gsheet.to_csv(
    "path/to/a/directory",
    filename_overrides={"Sheet1": "sums"},
    Sheet1=["column_a", "column_b", "column_c"],
    Sheet2=None  # use None to output Tabs even if you don't supply a header.
)
gsheet.to_json(
    "path/to/a/directory",
    Sheet3=["column_a", "column_b", "column_c"],
    Sheet2=0  # For to_json you MUST specify a header, or pass an index.
)

os.listdir("path/to/a/directory")
"""
```

(continues on next page)

(continued from previous page)

```
[
    "path/to/a/directory/Sheet1.csv",
    "path/to/a/directory/Sheet2.csv",
    "path/to/a/directory/Sheet2.json",
    "path/to/a/directory/Sheet3.json",
]
"""
```

CHAPTER

SEVEN

UPLOADING FILES

Using *Drive* or *Folder*, you can easily upload any file Google Drive will accept via Autodrive. For example, you can upload a bunch of files right to the root of your Google drive, like so.

```
from autodrive import Drive

drive = Drive()

drive.upload_files(
    "path/to/a/file.txt",
    "path/to/a/image.png",
    "path/to/a/spreadsheet.xlsx",
)
```

7.1 Uploading to Folders

You can also upload the files to arbitrary folders, and mix-and-match unspecified and specified target folders using FileUpload objects:

```
from autodrive import Folder, FileUpload

f1 = Folder("folder_id_1")
f2 = Folder("folder_id_2")

drive.upload_files(
    FileUpload("path/to/a/file.txt", f2),   # Will upload to Folder f2.
    FileUpload("path/to/a/image.png", f1),  # Will upload to Folder f1.
    "path/to/a/spreadsheet.xlsx",           # Will upload to the root.
)
```

If you're uploading to a single folder though, this approach is easier:

```
folder = Folder("folder_id_1")

# All of these files will be uploaded to the same folder:
folder.upload_files(
    "path/to/a/file.txt",
    "path/to/a/image.png",
    "path/to/a/spreadsheet.xlsx",
)
```

7.2 Converting to Google Formats

You can also convert the files right to the appropriate Google Drive format during upload.

Note: Google specifies how this conversion must work, so Autodrive will detect the type of the file you upload and use Google's rules to determine what Google format it should be converted to. You may get some unexpected results if you choose to convert a file that doesn't obviously correspond to a Google format.

```
drive.upload_files(
    FileUpload("path/to/a/file.txt", convert=True),          # To Google Doc.
    "path/to/a/image.png",                                   # Format unchanged.
    FileUpload("path/to/a/spreadsheet.xlsx", convert=True), # To Google Sheet.
)
```

CHAPTER

EIGHT

API REFERENCE

This page contains auto-generated API reference documentation[1].

8.1 autodrive

8.1.1 Subpackages

autodrive.formatting

Submodules

autodrive.formatting.format_rng

Module Contents

Classes

RangeCellFormatting	Contains request generation methods related to formatting a Range's grid
RangeGridFormatting	Contains request generation methods relating to formatting a Range's
RangeTextFormatting	Contains request generation methods methods relating to formatting this

class autodrive.formatting.format_rng.**RangeCellFormatting**(*parent*)

Contains request generation methods related to formatting a Range's grid (width and height, etc).

Parameters

parent (*Component[Any, Any, Any]*) – A Component object.

add_alternating_row_background(*colors*)

Queues a request to add an alternating row background of the indicated color to a Range's cells.

Parameters

colors (Color) – The desired Color to apply to every other row.

Returns

This formatting object, so further requests can be queued if desired.

[1] Created with sphinx-autoapi

Return type
RangeCellFormatting

`set_background_color`(*color*)

Queues a request to set the background of the Range's cells to the indicated color.

Parameters
color (Color) – The desired Color to set the background to.

Returns
This formatting object, so further requests can be queued if desired.

Return type
RangeCellFormatting

`set_border_format`(**sides*, *style=None*, *color=None*)

Queues a request to set the border properties of the Range's cells.

Parameters

- ***sides** – (BorderSide): One or more BorderSide objects, indicating which side(s) of the cells you want to apply the border properties to. If no sides are provided, set_border_format will apply border properties to all sides.
- **style** (BorderStyle, *optional*) – The style to apply to all the indicated sides. Defaults to None, for the default border style.
- **color** (Color, *optional*) – The color to set the border(s) to. Defaults to None, for black.

Returns
This formatting object, so further requests can be queued if desired.

Return type
RangeCellFormatting

`add_request`(*request*)

Adds the passed request to the Formatting object's parent component.

Parameters
request (*Dict[str, Any]*) – An api-ready request.

Return type
None

`ensure_full_range`(*rng=None*)

Convenience method for ensuring that all requests generated by this Formatting object have a FullRange attached to them, if one isn't manually supplied.

Parameters
rng (FullRange | *str, optional*) – A manually generated FullRange, defaults to None.

Returns
The passed FullRange, or a range generated from the Formatting object's parent Component.

Return type
FullRange

`class autodrive.formatting.format_rng.RangeGridFormatting`(*parent*)

Contains request generation methods relating to formatting a Range's cells (like adding borders and backgrounds and such).

Parameters
parent (*Component[Any, Any, Any]*) – A Component object.

auto_column_width()

Queues a request to set the column width of the Range's columns equal to the width of the values in the cells.

Returns
This formatting object, so further requests can be queued if desired.

Return type
RangeGridFormatting

add_request(*request*)

Adds the passed request to the Formatting object's parent component.

Parameters
request (*Dict[str, Any]*) – An api-ready request.

Return type
None

ensure_full_range(*rng=None*)

Convenience method for ensuring that all requests generated by this Formatting object have a FullRange attached to them, if one isn't manually supplied.

Parameters
rng (FullRange | *str, optional*) – A manually generated FullRange, defaults to None.

Returns
The passed FullRange, or a range generated from the Formatting object's parent Component.

Return type
FullRange

class autodrive.formatting.format_rng.**RangeTextFormatting**(*parent*)

Contains request generation methods methods relating to formatting this Range's text (the text format of any cells, even those containing non-text values like integers or null values).

Parameters
parent (*Component[Any, Any, Any]*) – A Component object.

apply_format(*format*)

Queues a request to set the text/number format of the Range's cells.

Parameters
format (Format) – A format instance, such as TextFormat or NumberFormat.

Returns
This formatting object, so further requests can be queued if desired.

Return type
RangeTextFormatting

set_alignment(**aligns*)

Queues a request to set the horizontal and/or vertical text alignment of the Range's cells.

Parameters
aligns (HorizontalAlign | VerticalAlign) – The desired horizontal and/or vertical alignment properties. Note that if you specify a HorizontalAlign more than once, or a VerticalAlign more than once, only the last of each will be used.

Returns
This formatting object, so further requests can be queued if desired.

Return type
RangeTextFormatting

`add_request`(*request*)

Adds the passed request to the Formatting object's parent component.

Parameters
request (*Dict[str, Any]*) – An api-ready request.

Return type
None

`ensure_full_range`(*rng=None*)

Convenience method for ensuring that all requests generated by this Formatting object have a FullRange attached to them, if one isn't manually supplied.

Parameters
rng (*FullRange | str, optional*) – A manually generated FullRange, defaults to None.

Returns
The passed FullRange, or a range generated from the Formatting object's parent Component.

Return type
FullRange

autodrive.formatting.format_tab

Module Contents

Classes

TabCellFormatting	Contains request generation methods relating to formatting this Tab's cells
TabGridFormatting	Contains request generation methods related to formatting this Tab's grid
TabTextFormatting	Contains request generation methods relating to formatting this Tab's text

`class` autodrive.formatting.format_tab.`TabCellFormatting`(*parent*)

Contains request generation methods relating to formatting this Tab's cells (like adding borders and backgrounds and such).

Parameters
parent (*Component[Any, Any, Any]*) – A Component object.

`add_alternating_row_background`(*colors*, *rng=None*)

Queues a request to add an alternating row background of the indicated color to this Tab's cells, or to a range within the Tab.

Parameters

- **colors** (Color) – The desired Color to apply to every other row.
- **rng** (FullRange | *str, optional*) – Optional range within the Tab to apply the format to, defaults to None, for all cells in the Tab.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabCellFormatting

`set_background_color`(*color*, *rng=None*)

Queues a request to set the background of the Tab's cells (or the specified cells within the Tab) to the indicated color

Parameters

- **color** (Color) – The desired Color to set the background to.
- **rng** (FullRange | *str, optional*) – Optional range within the Tab to apply the format to, defaults to None, for all cells in the Tab.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabCellFormatting

`set_border_format`(**sides*, *style=None*, *color=None*, *rng=None*)

Queues a request to set the border properties of the Tab's cells (or the specified cells within the Tab).

Parameters

- ***sides** – (BorderSide): One or more BorderSide objects, indicating which side(s) of the cells you want to apply the border properties to. If no sides are provided, set_border_format will apply border properties to all sides.
- **style** (BorderStyle, *optional*) – The style to apply to all the indicated sides. Defaults to None, for the default border style.
- **color** (Color, *optional*) – The color to set the border(s) to. Defaults to None, for black.
- **rng** (FullRange | *str, optional*) – Optional range within the Tab to apply the format to, defaults to None, for all cells in the Tab.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabCellFormatting

`add_request`(*request*)

Adds the passed request to the Formatting object's parent component.

Parameters
request (*Dict[str, Any]*) – An api-ready request.

Return type
None

`ensure_full_range`(*rng=None*)

Convenience method for ensuring that all requests generated by this Formatting object have a FullRange attached to them, if one isn't manually supplied.

Parameters
rng (FullRange | *str, optional*) – A manually generated FullRange, defaults to None.

Returns
The passed FullRange, or a range generated from the Formatting object's parent Component.

Return type
FullRange

class autodrive.formatting.format_tab.**TabGridFormatting**(*parent*)

Contains request generation methods related to formatting this Tab's grid (number of columns, rows, width and height, etc).

Parameters
parent (*Component[Any, Any, Any]*) – A Component object.

auto_column_width(*rng=None*)

Queues a request to set the column width of the Tab's columns equal to the width of the values in the cells.

Parameters
rng (HalfRange, *optional*) – The range of columns to be affected, defaults to None for all columns in the Tab.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabGridFormatting

append_rows(*num_rows*)

Queues a request to add new empty rows at the bottom of the Tab.

Parameters
num_rows (*int*) – The number of rows to add to the bottom of the Tab.

Returns

This formatting object, so further requests can be
queued if desired.

Return type
TabGridFormatting

insert_rows(*num_rows*, *at_row*)

Queues a request to insert new empty rows at the specified row number.

Parameters

- **num_rows** (*int*) – The number of rows to insert.
- **at_row** (*int*) – The row number to insert after.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabGridFormatting

delete_rows(*rng*)

Queues a request to delete rows in the selected row range.

Parameters
rng (HalfRange) – The range of rows to delete.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabGridFormatting

`append_columns`(*num_cols*)

Queues a request to add new empty columns at the right of the Tab.

Parameters
num_cols (*int*) – The number of columns to add to the right of the Tab.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabGridFormatting

`insert_columns`(*num_cols*, *at_col*)

Queues a request to insert new empty columns at the specified column number.

Parameters

- **num_cols** (*int*) – The number of columns to insert.
- **at_col** (*int*) – The column number to insert after.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabGridFormatting

`delete_columns`(*rng*)

Queues a request to delete columns in the selected column range.

Parameters
rng (HalfRange) – The range of columns to delete.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabGridFormatting

`add_request`(*request*)

Adds the passed request to the Formatting object's parent component.

Parameters
request (*Dict[str, Any]*) – An api-ready request.

Return type
None

`ensure_full_range`(*rng=None*)

Convenience method for ensuring that all requests generated by this Formatting object have a FullRange attached to them, if one isn't manually supplied.

Parameters
rng (FullRange | *str, optional*) – A manually generated FullRange, defaults to None.

Returns
The passed FullRange, or a range generated from the Formatting object's parent Component.

Return type
FullRange

class autodrive.formatting.format_tab.**TabTextFormatting**(*parent*)

Contains request generation methods relating to formatting this Tab's text (the text format of any cells, even those containing non-text values like integers or null values).

Parameters
parent (*Component[Any, Any, Any]*) – A Component object.

apply_format(*format*, *rng=None*)

Queues a request to set the text/number format of the Tab's cells (or the specified cells within the Tab).

Parameters

- **format** (Format) – A format instance, such as TextFormat or NumberFormat.
- **rng** (FullRange | *str, optional*) – Optional range within the Tab to apply the format to, defaults to None, for all cells in the Tab.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabTextFormatting

set_alignment(**aligns*, *rng=None*)

Queues a request to set the horizontal and/or vertical text alignment of the Tab's cells (or the specified cells within the Tab).

Parameters

- **aligns** (HorizontalAlign | VerticalAlign) – The desired horizontal and/or vertical alignment properties. Note that if you specify a HorizontalAlign more than once, or a VerticalAlign more than once, only the last of each will be used.
- **rng** (FullRange | *str, optional*) – Optional range within the Tab to apply the format to, defaults to None, for all cells in the Tab.

Returns
This formatting object, so further requests can be queued if desired.

Return type
TabTextFormatting

add_request(*request*)

Adds the passed request to the Formatting object's parent component.

Parameters
request (*Dict[str, Any]*) – An api-ready request.

Return type
None

ensure_full_range(*rng=None*)

Convenience method for ensuring that all requests generated by this Formatting object have a FullRange attached to them, if one isn't manually supplied.

Parameters
rng (FullRange | *str, optional*) – A manually generated FullRange, defaults to None.

Returns
The passed FullRange, or a range generated from the Formatting object's parent Component.

Return type
FullRange

Package Contents

Classes

RangeCellFormatting	Contains request generation methods related to formatting a Range's grid
RangeGridFormatting	Contains request generation methods relating to formatting a Range's
RangeTextFormatting	Contains request generation methods methods relating to formatting this
TabCellFormatting	Contains request generation methods relating to formatting this Tab's cells
TabGridFormatting	Contains request generation methods related to formatting this Tab's grid
TabTextFormatting	Contains request generation methods relating to formatting this Tab's text

8.1.2 Submodules

autodrive.connection

Module Contents

Classes

FileUpload	Use FileUploads when you need more complicated file upload instructions.
DriveConnection	Provides a connection to the Google Drive api, and methods to send requests
SheetsConnection	Provides a connection to the Google Sheets api, and methods to send requests

class autodrive.connection.**FileUpload**(*path*, *to_folder_id=None*, *convert=False*, *name_override=None*)

Use FileUploads when you need more complicated file upload instructions.

Parameters

- **path** (*Path, str*) – The pathlike to the file.
- **to_folder_id** (*str, optional*) – A folder id, if you want to upload this file to a specific folder in Google Drive. Defaults to None.
- **convert** (*bool, optional*) – Set to True if you want to convert the file to a Google Drive format. The format will be automatically determined based on Google Drive's rules for conversion. Defaults to False.
- **name_override** (*str | None*) –

property do_conv: bool

Return type
bool

class autodrive.connection.**DriveConnection**(*, *auth_config=None*, *api_version='v3'*)

Provides a connection to the Google Drive api, and methods to send requests to it.

This class is documented primarily for reference, if you want to connect to Google Drive, it's easier to just use a *Drive* instance.

Parameters

- **auth_config** (*AuthConfig, optional*) – Optional custom AuthConfig object, defaults to None.
- **api_version** (*str, optional*) – The version of the Drive api to connect to, defaults to "v3".

get_import_formats()

Returns

The mapping between standard MIMEtypes and the
MIMEtypes for Google Drive's types (Docs, Sheets, etc) as provided by the Google Drive API.

Return type
Dict[str, str]

find_object(*obj_name*, *obj_type=None*, *shared_drive_id=None*)

Searches for a Google Drive Object via the connected api.

Parameters

- **obj_name** (*str*) – The name of the object, or part of its name.
- **obj_type** (*Literal["sheet", "folder", "file"], optional*) – The type of object to restrict the search to.
- **shared_drive_id** (*str, optional*) – The id of a Shared Drive to search within, if desired, defaults to None.

Returns
A list of object properties, if any matches are found.

Return type
List[Dict[str, Any]]

create_object(*obj_name*, *obj_type*, *parent_id=None*)

Creates a file or folder via the Google Drive connection.

Parameters

- **obj_name** (*str*) – The desired name of the object to create.
- **obj_type** (*Literal["sheet", "folder"]*) – The type of object to create.
- **parent_id** (*str, optional*) – The id of the folder or shared drive to create the object within, defaults to None.

Returns
The new id of the created object.

Return type
str

`delete_object`(*object_id*)

Deletes the passed Google object id from the connected Google Drive.

Parameters
object_id (*str*) – A Google object id.

Return type
None

`upload_files`(**filepaths*)

Uploads files to the root drive or to a folder.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Parameters
***filepaths** (*Path | str | FileUpload*) – An arbitrary number of Path objects, path strings, or FileUpload objects (for more complex cases).

Returns

The names of the uploaded files and their new ids in
Google Drive.

Return type
Dict[str, str]

`detect_conv_format`(*p*)

Detects the MIMEtype for the passed filepath and determines its corresponding Google Drive format.

Parameters
p (*Path*) – Any filepath.

Raises

- **ValueError** – If the MIMEtype cannot be determined from the path at all.
- **ValueError** – If the MIMEtype cannot be converted to a Google Drive format MIMEtype.

Returns
The Google Drive format MIMEtype for the file.

Return type
str

class autodrive.connection.**SheetsConnection**(*, *auth_config=None*, *api_version='v4'*)

Provides a connection to the Google Sheets api, and methods to send requests to it.

This class is documented primarily for reference, if you want to connect to Google Sheets, it's easier to just use a View.

Parameters

- **auth_config** (*AuthConfig, optional*) – Optional custom AuthConfig object, defaults to None.
- **api_version** (*str, optional*) – The version of the Sheets api to connect to, defaults to "v4".

execute_requests(*spreadsheet_id*, *requests*)

Sends the passed list of request dictionaries to the Sheets api to be applied to the spreadsheet_id via batch update.

Parameters

- **spreadsheet_id** (*str*) – The id of the Google Sheet to update.
- **requests** (*List[Dict[str, Any]]*) – A list of dictionaries formatted as requests.

Returns
The resulting response from the Sheets api as a dictionary.

Return type
Dict[str, Any]

get_properties(*spreadsheet_id*)

Gets the metadata properties of the indicated Google Sheet.

Parameters
spreadsheet_id (*str*) – The id of the Google Sheet to collect properties from.

Returns
A dictionary of the Google Sheet's properties.

Return type
Dict[str, Any]

get_data(*spreadsheet_id*, *ranges=None*)

Collects data from cells in the passed spreadsheet.

Parameters

- **spreadsheet_id** (*str*) – The id of the Google Sheet to collect data from.
- **ranges** (*List[str], optional*) – A list of range strings (e.g. Sheet1!A1:C3), defaults to None, which prompts get_data to fetch all data from all cells.

Returns
The collected data from the spreadsheet, raw and unparsed.

Return type
Dict[str, Any]

autodrive.drive

Module Contents

Classes

Folder	Stores all the necessary properties of a Google Drive Folder and provides
Drive	A one-stop shop for finding and managing Google Drive objects, including

class autodrive.drive.**Folder**(*folder_id*, *name*, *, *parents=None*, *auth_config=None*, *drive_conn=None*, *sheets_conn=None*)

Stores all the necessary properties of a Google Drive Folder and provides methods for interacting with its subfolders and files.

Parameters

- **folder_id** (*str*) – The id string of the folder.
- **name** (*str*) – The name of the folder.
- **parents** (*List[str], optional*) – A list of the Folder's parents, usually just its parent folder or parent shared drive, if any, defaults to None, signifying that the folder is at the root of an unshared drive.
- **auth_config** (AuthConfig, *optional*) – Optional custom AuthConfig object, defaults to None.
- **drive_conn** (DriveConnection, *optional*) – Optional manually created DriveConnection, defaults to None
- **sheets_conn** (SheetsConnection, *optional*) – Optional manually created SheetsConnection, defaults to None.

property id: str

Returns: str: The id string of the Folder.

Return type
str

property name: str

Returns: str: The name of the Folder.

Return type
str

property parent_ids: List[str]

Returns: List[str]: The list of the Folder's parents, usually just its parent folder or parent shared drive, if any.

Return type
List[str]

create_folder(*folder_name*)

Generates a new Folder with this Folder as its parent.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Parameters
folder_name (*str*) – The name of the Folder to create.

Returns
The newly created Folder.

Return type
Folder

`create_gsheet`(*gsheet_name*)

Creates a new Google Sheet with this Folder as its parent.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Parameters
gsheet_name (*str*) – The desired name of the new Google Sheet.

Returns
The newly created GSheet.

Return type
GSheet

`upload_files`(**filepaths*)

Uploads files to the folder.

Parameters
***filepaths** (*Path | str | FileUpload*) – An arbitrary number of Path objects, path strings, or FileUpload objects (if you want to convert the file to a Google Drive format (e.g. Docs, Sheets, etc).

Returns

The names of the uploaded files and their new ids in
Google Drive.

Return type
Dict[str, str]

class autodrive.drive.**Drive**(*, *auth_config=None*, *drive_conn=None*, *sheets_conn=None*)

A one-stop shop for finding and managing Google Drive objects, including folders and Google Sheets.

Note: All the methods on this class will cause a request to be posted to the relevant Google API immediately.

Parameters

- **auth_config** (*AuthConfig, optional*) – Optional custom AuthConfig object, defaults to None.
- **drive_conn** (*DriveConnection, optional*) – Optional manually created DriveConnection, defaults to None.
- **sheets_conn** (*SheetsConnection, optional*) – Optional manually created SheetsConnection, defaults to None.

`create_folder`(*folder_name*, *parent=None*)

Creates a folder in the connected Google Drive.

Parameters

- **folder_name** (*str*) – The desired name of the new folder
- **parent** (*str | Folder, optional*) – The parent folder/id to create this folder within, or the id of the shared drive to create within, defaults to None.

Returns
The newly created Folder.

Return type
Folder

`create_gsheet`(*gsheet_name*, *parent=None*)

Creates a Google Sheet in the connected Google Drive.

Parameters

- **gsheet_name** (*str*) – The desired name of the new Google Sheet.
- **parent** (*str | Folder, optional*) – The parent folder/id to create this gsheet within, or the id of the shared drive to create within, defaults to None.

Returns
The newly created GSheet.

Return type
GSheet

`find_gsheet`(*gsheet_name*)

Search for Google Sheets that match the passed name. GSheets found in this way will need to have their properties gathered with a GSheet.fetch () call. To save round trips this method only collects the basic details of the Google Sheets it matches. These properties cannot be gathered via the Google Drive api, unfortunately.

Parameters
gsheet_name (*str*) – The name (or part of the name) to search for.

Returns
A list of GSheet objects with basic details (id, name) that matched the name parameter.

Return type
List[*GSheet*]

`find_folder`(*folder_name*, *shared_drive_id=None*)

Search for folders that match the passed name.

Parameters

- **folder_name** (*str*) – The name (or part of the name) to search for.
- **shared_drive_id** (*str, optional*) – The shared drive id to search within, defaults to None

Returns
A list of Folder object that matched the name parameter.

Return type
List[*Folder*]

`upload_files`(**filepaths*)

Uploads files to the root drive or to a folder.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Parameters
***filepaths** (*Path | str | FileUpload*) – An arbitrary number of Path objects, path strings, or FileUpload objects (for more complex cases).

Returns

The names of the uploaded files and their new ids in
Google Drive.

Return type
Dict[str, str]

autodrive.dtypes

Module Contents

Classes

String	String datatype, covers all non-numeric/date text that isn't a formula.
Formula	Formula datatype, essentially a string, but begins with =.
Number	Number datatype, covers both floats and integers, though internally Number is
Boolean	Boolean datatype, appears in Google Sheets as FALSE or TRUE.
GoogleValueType	Metaclass for the three different ways values are stored in Google Sheets.
UserEnteredVal	UserEnteredVal is the value as entered by the user, without any calculations
EffectiveVal	EffectiveVal is the value as displayed in Google Sheets, and is appropriately
FormattedVal	The untyped string value of the cell as displayed in Google Sheets. Essentially
BorderStyle	A property describing the style of border to apply to part of a cell.
BorderSide	A property describing which side of a cell to apply border properties to.
VerticalAlign	A property describing vertical text alignment.
HorizontalAlign	A property describing horizontal text alignment.

Attributes

GOOGLE_VAL_TYPES	A tuple of UserEnteredVal, EffectiveVal, and FormattedVal google value types.
GOOGLE_DTYPES	A tuple of String, Formula, Number, and Boolean google data types.
TYPE_MAP	Dictionary mapping python data types to corresponding google data types.
REV_TYPE_MAP	Dictionary mapping google data types to corresponding python types.
UserEnteredFmt	The formatting properties the user as applied to the cell.
EffectiveFmt	The final formatting information about a cell and its values, includes the
DefaultFmt	The default formatting properties of a cell, dictated by the settings of the tab.
BorderSolid	The default border style, a thin, solid line.
BorderSolidMedium	Same as BorderSolid, but slightly thicker.
BorderSolidThick	Same as BorderSolid, but much thicker.
BorderDashed	A thin line comprised of dashes.
BorderDotted	A thin line comprised of dots.
BorderDoubleLine	A set of two parallel lines.
BorderLeft	The border for the left side of a cell.
BorderRight	The border for the right side of a cell.
BorderTop	The border for the top side of a cell.
BorderBottom	The border for the bottom side of a cell.
BorderSides	Convenience reference for all BorderSide objects.
AlignTop	Align text to the top of the cell(s).
AlignMiddle	Align text to the middle of the cell(s).
AlignBottom	Align text to the middle of the cell(s).
AlignLeft	Align text to the left of the cell(s).
AlignCenter	Align text to the center of the cell(s).
AlignRight	Align text to the right of the cell(s).

class autodrive.dtypes.**String**

String datatype, covers all non-numeric/date text that isn't a formula.

python_type

type_key = stringValue

classmethod parse(*value*)

Converts a string into a string.

Parameters
value (*str*) – Any string.

Returns
The passed value as a string.

Return type
str

class autodrive.dtypes.**Formula**

Formula datatype, essentially a string, but begins with =.

python_type

type_key = formulaValue

classmethod parse(*value*)

Converts a string into a formula (string).

Parameters
value (*str*) – Any string.

Returns
The passed value as a string.

Return type
str

class autodrive.dtypes.**Number**

Number datatype, covers both floats and integers, though internally Number is treated as a float so as to not lose significant digits.

python_type

type_key = numberValue

classmethod parse(*value*)

Converts a string into an integer (if it has no decimal point) or float.

Parameters
value (*str*) – Any numeric string.

Returns
The passed value as an integer or float.

Return type
float | int

class autodrive.dtypes.**Boolean**

Boolean datatype, appears in Google Sheets as FALSE or TRUE.

python_type

type_key = boolValue

classmethod parse(*value*)

Converts a string into a boolean.

Parameters
value (*str*) – Any string.

Returns
False if the string is some variation of FALSE, otherwise True for all other strings.

Return type
bool

class autodrive.dtypes.**GoogleValueType**

Metaclass for the three different ways values are stored in Google Sheets.

value_key :str

The API key for the GoogleValueType.

has_dtype :bool = True

Whether dtype information can be extracted from the GoogleValueType.

class autodrive.dtypes.**UserEnteredVal**

UserEnteredVal is the value as entered by the user, without any calculations applied to it. It is always a string, but is accompanied by metadata that indicates what datatype the value is.

value_key = userEnteredValue

class autodrive.dtypes.**EffectiveVal**

EffectiveVal is the value as displayed in Google Sheets, and is appropriately typed when read from the api. So if the formula "=A1+A2" would equal 3, then the EffectiveVal of that cell is 3.

value_key = effectiveValue

class autodrive.dtypes.**FormattedVal**

The untyped string value of the cell as displayed in Google Sheets. Essentially equivalent to EffectiveVal, but without appropriate typing.

value_key = formattedValue

has_dtype = False

autodrive.dtypes.**GOOGLE_VAL_TYPES**

A tuple of UserEnteredVal, EffectiveVal, and FormattedVal google value types.

autodrive.dtypes.**GOOGLE_DTYPES**

A tuple of String, Formula, Number, and Boolean google data types.

autodrive.dtypes.**TYPE_MAP**

Dictionary mapping python data types to corresponding google data types.

autodrive.dtypes.**REV_TYPE_MAP**

Dictionary mapping google data types to corresponding python types.

autodrive.dtypes.**UserEnteredFmt**

The formatting properties the user as applied to the cell.

autodrive.dtypes.**EffectiveFmt**

The final formatting information about a cell and its values, includes the effects of conditional formatting and the like.

autodrive.dtypes.**DefaultFmt**

The default formatting properties of a cell, dictated by the settings of the tab.

class autodrive.dtypes.**BorderStyle**(*style*)

A property describing the style of border to apply to part of a cell.

Parameters

style (*str*) – The name of the style.

autodrive.dtypes.**BorderSolid**

The default border style, a thin, solid line.

autodrive.dtypes.**BorderSolidMedium**

Same as BorderSolid, but slightly thicker.

autodrive.dtypes.**BorderSolidThick**

Same as BorderSolid, but much thicker.

autodrive.dtypes.**BorderDashed**

A thin line comprised of dashes.

autodrive.dtypes.**BorderDotted**

A thin line comprised of dots.

autodrive.dtypes.**BorderDoubleLine**

A set of two parallel lines.

class autodrive.dtypes.**BorderSide**(*side*)

A property describing which side of a cell to apply border properties to.

Parameters
side (*str*) – The name of the side.

autodrive.dtypes.**BorderLeft**

The border for the left side of a cell.

autodrive.dtypes.**BorderRight**

The border for the right side of a cell.

autodrive.dtypes.**BorderTop**

The border for the top side of a cell.

autodrive.dtypes.**BorderBottom**

The border for the bottom side of a cell.

autodrive.dtypes.**BorderSides**

Convenience reference for all BorderSide objects.

class autodrive.dtypes.**VerticalAlign**(*align_str*)

A property describing vertical text alignment.

Parameters
align_str (*str*) –

class autodrive.dtypes.**HorizontalAlign**(*align_str*)

A property describing horizontal text alignment.

Parameters
align_str (*str*) –

autodrive.dtypes.**AlignTop**

Align text to the top of the cell(s).

autodrive.dtypes.**AlignMiddle**

Align text to the middle of the cell(s).

autodrive.dtypes.**AlignBottom**

Align text to the middle of the cell(s).

autodrive.dtypes.**AlignLeft**

Align text to the left of the cell(s).

autodrive.dtypes.**AlignCenter**

Align text to the center of the cell(s).

autodrive.dtypes.**AlignRight**

Align text to the right of the cell(s).

autodrive.gsheet

Module Contents

Classes

GSheet	Provides a connection to a single Google Sheet and access to its properties

class autodrive.gsheet.**GSheet**(*gsheet_id*, *title=None*, ***, *tabs=None*, *auth_config=None*, *sheets_conn=None*, *autoconnect=True*)

Provides a connection to a single Google Sheet and access to its properties and Tabs.

Parameters

- **gsheet_id** (*str*) – The id string of the target Google Sheet; can be found in the link to the Google Sheet.
- **title** (*str, optional*) – The name of the Google Sheet, defaults to None.
- **tabs** (*List[Tab], optional*) – A list of Tabs attached to the Google Sheet. You should probably manage Tabs with GSheet.fetch () or by getting the GSheet directly from a Drive, rather than using this parameter, defaults to None.
- **auth_config** (*AuthConfig, optional*) – Optional custom AuthConfig object, defaults to None.
- **sheets_conn** (*SheetsConnection, optional*) – Optional manually created SheetsConnection, defaults to None.
- **autoconnect** (*bool, optional*) – If you want to instantiate a GSheet without immediately checking your authentication credentials and connecting to the Google Sheets api, set this to False, defaults to True.

property **requests: List[Dict[str, Any]]**

List of accumulated (uncommitted) requests on this GSheet.

Returns
List of update request dictionaries that have been created for this GSheet.

Return type
List[Dict[str, Any]]

property **tabs: Dict[str, *autodrive.tab.Tab*]**

Dictionary of fetched Tabs on this GSheet by title.

Returns
Tab titles as keys and corresponding Tabs as values.

Return type
Dict[str, *Tab*]

property **title: Optional[str]**

The name of the GSheet.

Returns
The name of the GSheet, or None if its name hasn't been fetched.

Return type
Optional[str]

`fetch`()

Gets the latest metadata from the API for this GSheet. Populates title and tab properties.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Returns
This GSheet

Return type
GSheet

`add_tab`(*tab*)

Adds a Tab to the GSheet.

Parameters
tab (Tab) – The Tab instance you want to add.

Returns
This Gsheet.

Return type
GSheet

Raises
ValueError – If the GSheet already has a Tab with that title.

`gen_range`(*rng*, *tab=None*)

Convenience method for generating a new Range object from a Tab in this GSheet.

Parameters

- **rng** (FullRange) – The desired FullRange of the new Range object.
- **tab** (*str | int, optional*) – The name of the Tab to generate from, or its index. Defaults to None, which will generate a range from the first Tab.

Returns
The newly generated Range object.

Return type
Range

`write_values`(*data*, *to_tab=None*, *rng=None*, *mode='write'*)

Adds a request to write data. GSheet.commit () to commit the requests.

Parameters

- **data** (*Sequence[Sequence[Any] | Dict[str, Any]]*) – The data to write. Each sequence or dictionary in the passed data is a row, with each value in that sub-iterable being a column. Dictionary keys will be used as a header row in the written data.
- **to_tab** (*str, optional*) – The name of the tab to write to, defaults to None, which will write to whatever tab is first in the Sheet.
- **rng** (FullRange *| str, optional*) – The range to which the data will be written, starting with the top-left-most cell in the range, defaults to None, which will write to the top-left-most cell in the passed tab, or the first tab.

- **mode** (*Literal, optional*) – Whether to append the data after any populated rows already present in the tab or to write to the passed rng. Overrides rng if specified. Defaults to "write".

Returns
This GSheet.

Return type
GSheet

Raises
KeyError – If the passed tab name (to_tab) isn't present in the GSheet's current tabs property.

get_data(*tab=None*, *rng=None*, *value_type=EffectiveVal*)

Gets the data from the cells of the GSheet. GSheet.fetch() will be called automatically if the GSheet's tabs are not populated.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Parameters

- **tab** (*str | int, optional*) – The name of the tab, or its (0-based) index (from left to right), defaults to None, which will collect data from the first tab in the Sheet.
- **rng** (FullRange | *str, optional*) – The specific range to fetch data from, defaults to None, for all data in the target tab.
- **value_type** (GoogleValueType, *optional*) – Allows you to toggle the type of the values returned by the Google Sheets API. See the *dtypes* documentation for more info on the different GoogleValueTypes.

Returns
This GSheet.

Return type
GSheet

Raises

- **KeyError** – If the passed tab name is not found in this GSheet's tabs.
- **TypeError** – If anything other than the displayed types is passed for the tab parameter.

keys()

Gets the keys (tab titles) for the fetched tabs on this GSheet.

Returns
The tab titles on this GSheet.

Return type
KeysView[str]

values()

Gets the values (Tabs) for the fetched tabs on this GSheet.

Returns
The Tabs on this GSheet.

Return type
ValuesView[*Tab*]

get_tab_index_by_title(*tab_title*)

Parameters
tab_title (*str*) –

Return type
Optional[int]

to_csv(*root_path*, *filename_overrides=None*, ***tabs_and_headers*)

Convenience method for calling *.to_csv()* on some or all of the GSheet's tabs.

Parameters

- **root_path** (*str*) – The root directory path to save tab files to.
- **filename_overrides** (*Dict[str, str], optional*) – By default, this method will name each file after the corresponding tab's title. To override some or all of the resulting filenames, pass a dictionary with keys equal to the names of the tab you want to override and the values equal to the name of the filename you want. Defaults to None, for all tabs being treated with default behavior.
- **tabs_and_headers** (*Sequence[Any] | None*) – (Sequence[Any], optional): If you want to only output the data for some of the tabs, you can pass the names of the desired tabs as kwargs. If you wish, you can also pass a header row for those tabs, which will be inserted as the first row of the file. If you don't want to pass a header row, simply pass `tabname=None` for that tab.

Raises
ValueError – If root_path does not lead to a directory.

Return type
None

to_json(*root_path*, *filename_overrides=None*, ***tabs_and_headers*)

Convenience method for calling *.to_json()* on some or all of the GSheet's tabs.

Parameters

- **root_path** (*str*) – The root directory path to save tab files to.
- **filename_overrides** (*Dict[str, str], optional*) – By default, this method will name each file after the corresponding tab's title. To override some or all of the resulting filenames, pass a dictionary with keys equal to the names of the tab you want to override and the values equal to the name of the filename you want. Defaults to None, for all tabs being treated with default behavior.
- **tabs_and_headers** (*Sequence[str] | int*) – (Sequence[str] | int, optional): If you want to only output the data for some of the tabs, you can pass the names of the desired tabs as kwargs. You must also indicate what keys should be used when creating the jsons for those tabs. For each header value, you may either pass a row index to pull for the keys, or a list of keys.

Raises
ValueError – If root_path does not lead to a directory.

Return type
None

autodrive.interfaces

Module Contents

Classes

AuthConfig	Optional custom configuration for Autodrive's authentication processes using
HalfRange	A partial range used in requests to the Google Sheets API when the axis of the
FullRange	A complete Google Sheets range (indicating start and end row and column as
BorderFormat	The settings for the border of a single side of a cell.
Color	An RGBA color value.
Format	Underlying class for text/number formats.
TextFormat	Provides parameters available in updating the text formatting of a cell.
NumericFormat	A Google Sheet Number Format, which the value of the cell (s) will be parsed

Attributes

DEFAULT_TOKEN	Filepath for the token json file. Default="gdrive_token.json".
DEFAULT_CREDS	Filepath for the credentials json file. Default="credentials.json"
AutomaticFormat	Corresponds to the Automatic `1000.12` format.
NumberFormat	Corresponds to the Number `1,000.12` format.
AccountingFormat	Corresponds to the Accounting `$ (1,000.12)` format.
PercentFormat	Corresponds to the Percent `10.12%` format.
ScientificFormat	Corresponds to the Scientific `1.01E+03` format.
FinancialFormat	Corresponds to the Financial `(1,000.12)` format.
CurrencyFormat	Corresponds to the Currency `$1,000.12` format.
CurRoundFormat	Corresponds to the Currency (rounded) `$1,000` format.
DateFormat	Corresponds to the Date `9/26/2008` format.
TimeFormat	Corresponds to the Time `3:59:00 PM` format.
DatetimeFormat	Corresponds to the Date time `9/26/2008 15:59:00` format.
DurationFormat	Corresponds to the Duration `24:01:00` format.

autodrive.interfaces.**DEFAULT_TOKEN = gdrive_token.json**

Filepath for the token json file. Default="gdrive_token.json".

autodrive.interfaces.**DEFAULT_CREDS = credentials.json**

Filepath for the credentials json file. Default="credentials.json"

exception autodrive.interfaces.**ParseRangeError**(*rng*, *msg_addon=None*, **args*)

Error raised when an invalid range string is passed to a FullRange.

Parameters

- **rng** (*str*) – The original range string.
- **msg_addon** (*str, optional*) – An addendum to the error message. Defaults to None.
- **args** (*object*) –

class args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class autodrive.interfaces.**AuthConfig**(*secrets_config=None*, *token_filepath=DEFAULT_TOKEN*, *creds_filepath=DEFAULT_CREDS*)

Optional custom configuration for Autodrive's authentication processes using your Google api credentials.

Parameters

- **secrets_config** (*Dict[str, Any], optional*) – The dictionary of configuration used by a google.oath2.credentials Credentials object. Generally only useful if for some reason you are reading in your own token file, defaults to None.
- **token_filepath** (*str | Path, optional*) – The filepath to your gdrive_token json file. This doesn't have to exist at time of authentication, and will be saved to this path when the authorization flow completes, defaults to DEFAULT_TOKEN, which is "gdrive_token.json" in your cwd.
- **creds_filepath** (*str | Path, optional*) – The filepath to your api credentials json file. This file *does* need to exist at time of authentication, unless you pass a secrets_config dictionary, defaults to DEFAULT_CREDS, which is "credentials.json" from your cwd.

property token_filepath: pathlib.Path

Returns: Path: The Path to your token json file.

Return type
pathlib.Path

property creds_filepath: pathlib.Path

Returns: Path: The Path to your credentials json file.

Return type
pathlib.Path

class autodrive.interfaces.**HalfRange**(*start_idx=None*, *end_idx=None*, ***, *tab_title=None*, *base0_idxs=False*, *column=False*)

A partial range used in requests to the Google Sheets API when the axis of the range is obvious from context (such as when inserting rows or columns).

Note that HalfRange parses your inputs into Google Sheets API row/column indices, which are 0-based, so if you call `start_idx` or `end_idx` expect them to be 1 lower than the value you passed.

Parameters

- **start_idx** (*str | int, optional*) – The first column/row in the range, assumed to not be 0-based if it's an integer (row), defaults to None.
- **end_idx** (*str | int, optional*) – The last column/row in the range, assumed to not be 0-based if it's an integer (row), defaults to None.
- **tab_title** (*str, optional*) – The name of the tab this range resides in, defaults to None.
- **base0_idxs** (*bool, optional*) – Set to True if you'd like to pass 0-based indices to start_idx and end_idx params, above, defaults to False.

- **column** (*bool, optional*) – Set to True if this HalfRange is a column range and you supplied the column indexes as integers, defaults to False.

class Mapping

get(*key*, *default=None*)

D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.

keys()

D.keys() -> a set-like object providing a view on D's keys

items()

D.items() -> a set-like object providing a view on D's items

values()

D.values() -> an object providing a view on D's values

to_dict()

The Google Sheets api ranges are end-value exclusive, so this method will produce a dictionary with an endIndex value 1 higher than the FullRange's attribute.

Returns
Outputs the HalfRange as a dictionary of properties usable in generating an api request to affect the target range of cells.

Return type
Dict[str, int]

get(*key*, *default=None*)

D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.

keys()

D.keys() -> a set-like object providing a view on D's keys

items()

D.items() -> a set-like object providing a view on D's items

values()

D.values() -> an object providing a view on D's values

class autodrive.interfaces.**FullRange**(*range_str=None*, *, *start_row=None*, *end_row=None*, *start_col=None*, *end_col=None*, *base0_idxs=False*, *tab_title=None*)

A complete Google Sheets range (indicating start and end row and column as well as at least an end column, if not an end row).

Note that FullRange parses your inputs into Google Sheets API row/column indices, which are 0-based, so if you call `start_row`, `end_row`, `start_col`, or `end_col`, expect them to be 1 lower than the value you passed.

Parameters

- **range_str** (*str, optional*) – A range string (e.g. Sheet1!A1:B3, A1:B3, A1).
- **start_row** (*int, optional*) – The first row in the range, assumed to not be 0-based, defaults to None.
- **end_row** (*int, optional*) – The last row in the range, assumed to not be 0-based, defaults to None.
- **start_col** (*int | str, optional*) – The first column in the range, assumed to not be 0-based if it's an integer, defaults to None.

- **end_col** (*int | str, optional*) – The last column in the range, assumed to not be 0-based if it's an integer, defaults to None.
- **base0_idxs** (*bool, optional*) – Set to True if you'd like to pass 0-based indices to start/end_row and start/end_col, defaults to False.
- **tab_title** (*str, optional*) – The name of the tab this range resides in, defaults to None.

class Mapping

get(*key*, *default=None*)

D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.

keys()

D.keys() -> a set-like object providing a view on D's keys

items()

D.items() -> a set-like object providing a view on D's items

values()

D.values() -> an object providing a view on D's values

property row_range: *HalfRange*

Returns: HalfRange: The FullRange's row range as a HalfRange.

Return type
HalfRange

property col_range: *HalfRange*

Returns: HalfRange: The FullRange's column range as a HalfRange.

Return type
HalfRange

to_dict()

The Google Sheets api ranges are end-value exclusive, so this method will produce a dictionary with endRowIndex and endColumnIndex values 1 higher than the FullRange's attribute.

Returns
Outputs the FullRange as a dictionary of properties usable in generating an api request to affect the target range of cells.

Return type
Dict[str, int]

get(*key*, *default=None*)

D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.

keys()

D.keys() -> a set-like object providing a view on D's keys

items()

D.items() -> a set-like object providing a view on D's items

values()

D.values() -> an object providing a view on D's values

class autodrive.interfaces.**BorderFormat**(*side*, *color=None*, *style=None*)

The settings for the border of a single side of a cell.

Parameters

- **side** (BorderSide) – The BorderSide to apply settings to.
- **color** (Color, *optional*) – The color of the side of the border. Defaults to None, for a black border.
- **style** (BorderStyle, *optional*) – The BorderStyle of the side of the border. Defaults to None, for the default border style.

to_dict()

Returns
Outputs the BorderFormat as a dictionary of properties usable in generating an api request to affect cell border properties.

Return type
Dict[str, Any]

class autodrive.interfaces.Color(*red=0*, *green=0*, *blue=0*, *alpha=100*)

An RGBA color value.

Parameters

- **red** (*int | float, optional*) – Either an integer from 0 to 255 (from which a float value will be calculated), or the float representation of same. defaults to 0.
- **green** (*int | float, optional*) – Either an integer from 0 to 255 (from which a float value will be calculated), or the float representation of same. defaults to 0.
- **blue** (*int | float, optional*) – Either an integer from 0 to 255 (from which a float value will be calculated), or the float representation of same. defaults to 0.
- **alpha** (*int, optional*) – Either an integer from 0 to 100 (from which a float value will be calculated), or the float representation of same. defaults to 100.

class Mapping

get(*key*, *default=None*)

D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.

keys()

D.keys() -> a set-like object providing a view on D's keys

items()

D.items() -> a set-like object providing a view on D's items

values()

D.values() -> an object providing a view on D's values

to_dict()

Returns
Outputs the Color as a dictionary of properties usable in generating an api request to affect the color of text or cell background.

Return type
Dict[str, float]

classmethod from_hex(*hex_code*, *alpha=100*)

Instantiates a Color object from the supplied hex code.

Parameters

- **hex_code** (*str*) – A hexadecimal color code.

- **alpha** (*int | float, optional*) – Optional alpha parameter (not included in hex codes). Defaults to 100.

Returns
The new Color object.

Return type
Color

get(*key*, *default=None*)

D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.

keys()

D.keys() -> a set-like object providing a view on D's keys

items()

D.items() -> a set-like object providing a view on D's items

values()

D.values() -> an object providing a view on D's values

class autodrive.interfaces.**Format**(*format_key*)

Underlying class for text/number formats.

Parameters
format_key (*str*) – The format key as dictated by the Google Sheets api.

property **format_key: str**

Returns: str: The format's Google Sheets api key, used to generate the userEnteredFormat value.

Return type
str

to_dict()

Returns
Outputs the Format as a dictionary of properties usable in generating an api request to affect the text/number format of one or more cells.

Return type
Dict[str, Any]

class autodrive.interfaces.**TextFormat**(*, *font=None*, *color=None*, *font_size=None*, *bold=None*, *italic=None*, *underline=None*, *strikethrough=None*)

Provides parameters available in updating the text formatting of a cell.

Parameters

- **font** (*str, optional*) – The name of a font to change to, defaults to None.
- **color** (*Color, optional*) – The Color properties to apply to the text, defaults to None.
- **font_size** (*int, optional*) – The size to apply to the text, defaults to None.
- **bold** (*bool, optional*) – Whether to turn bold on (True) or off (False), defaults to None, for unchanged.
- **italic** (*bool, optional*) – Whether to turn italic on (True) or off (False), defaults to None, for unchanged.
- **underline** (*bool, optional*) – Whether to turn underline on (True) or off (False), defaults to None, for unchanged.

- **strikethrough** (*bool, optional*) – Whether to turn strikethrough on (True) or off (False), defaults to None, for unchanged.

property format_key: str

Returns: str: The format's Google Sheets api key, used to generate the userEnteredFormat value.

Return type
str

to_dict()

Returns
Outputs the Format as a dictionary of properties usable in generating an api request to affect the text/number format of one or more cells.

Return type
Dict[str, Any]

class autodrive.interfaces.NumericFormat(*pattern=''*)

A Google Sheet Number Format, which the value of the cell (s) will be parsed into.

Parameters
pattern (*str, optional*) – A pattern valid as a Google Sheet Number Format, defaults to "", for automatic number format.

property format_key: str

Returns: str: The format's Google Sheets api key, used to generate the userEnteredFormat value.

Return type
str

to_dict()

Returns
Outputs the Format as a dictionary of properties usable in generating an api request to affect the text/number format of one or more cells.

Return type
Dict[str, Any]

autodrive.interfaces.AutomaticFormat

Corresponds to the Automatic `1000.12` format.

autodrive.interfaces.NumberFormat

Corresponds to the Number `1,000.12` format.

autodrive.interfaces.AccountingFormat

Corresponds to the Accounting `$ (1,000.12)` format.

autodrive.interfaces.PercentFormat

Corresponds to the Percent `10.12%` format.

autodrive.interfaces.ScientificFormat

Corresponds to the Scientific `1.01E+03` format.

autodrive.interfaces.FinancialFormat

Corresponds to the Financial `(1,000.12)` format.

autodrive.interfaces.CurrencyFormat

Corresponds to the Currency `$1,000.12` format.

autodrive.interfaces.**CurRoundFormat**

Corresponds to the Currency (rounded) `$1,000` format.

autodrive.interfaces.**DateFormat**

Corresponds to the Date `9/26/2008` format.

autodrive.interfaces.**TimeFormat**

Corresponds to the Time `3:59:00 PM` format.

autodrive.interfaces.**DatetimeFormat**

Corresponds to the Date time `9/26/2008 15:59:00` format.

autodrive.interfaces.**DurationFormat**

Corresponds to the Duration `24:01:00` format.

autodrive.range

Module Contents

Classes

Range	Provides a connection to the data in a specific range in a Google Sheet Tab.

class autodrive.range.**Range**(*gsheet_range*, *gsheet_id*, *tab_title*, *tab_id*, ***, *auth_config=None*, *sheets_conn=None*, *autoconnect=True*)

Provides a connection to the data in a specific range in a Google Sheet Tab.

Parameters

- **gsheet_range** (FullRange | *str*) – The range (i.e. A5:C10) to associate with this Range.
- **gsheet_id** (*str*) – The id string of the target Google Sheet that the Range resides in; can be found in the Google Sheet url.
- **tab_title** (*str*) – The name of the Tab this Range is within.
- **auth_config** (AuthConfig, *optional*) – Optional custom AuthConfig, defaults to None.
- **sheets_conn** (SheetsConnection, *optional*) – Optional manually created SheetsConnection, defaults to None.
- **autoconnect** (*bool, optional*) – If you want to instantiate a Range without immediately checking your authentication credentials and connection to the Google Sheets api, set this to False, defaults to True.
- **tab_id** (*int*) –

property format_grid: *autodrive.formatting.format_rng.RangeGridFormatting*

Returns: RangeGridFormatting: An object with grid formatting methods.

Return type

autodrive.formatting.format_rng.RangeGridFormatting

property format_text: *autodrive.formatting.format_rng.RangeTextFormatting*

Returns: RangeTextFormatting: An object with text formatting methods.

Return type
autodrive.formatting.format_rng.RangeTextFormatting

property format_cell: *autodrive.formatting.format_rng.RangeCellFormatting*

Returns: RangeCellFormatting: An object with cell formatting methods.

Return type
autodrive.formatting.format_rng.RangeCellFormatting

property tab_id: int

Returns: int: The id of the linked tab.

Return type
int

property range_str: str

The string representation of the range specified by this Component.

Type
str

Return type
str

property range: *autodrive.interfaces.FullRange*

Returns: FullRange: The FullRange representation of the range specified by this Component.

Return type
autodrive.interfaces.FullRange

property values: List[List[Any]]

Returns: List[List[Any]]: The fetched data values in this Component's cells.

Return type
List[List[Any]]

property formats: List[List[Dict[str, Any]]]

Returns: List[List[Dict[str, Any]]]: The fetched formatting properties of this Component's cells.

Return type
List[List[Dict[str, Any]]]

property data_shape: Tuple[int, int]

Returns: Tuple[int, int]: The row length and column width of this Component's data.

Return type
Tuple[int, int]

property requests: List[Dict[str, Any]]

Returns: List[Dict[str, Any]]: The list of current (uncommitted) requests.

Return type
List[Dict[str, Any]]

property conn: *autodrive.connection.SheetsConnection*

Returns
The view's SheetsConnection.

Return type
SheetsConnection

Raises
NoConnectionError – If the view's connection is null.

property auth: *autodrive.interfaces.AuthConfig*

Returns
The view's AuthConfig.

Return type
AuthConfig

Raises
NoConnectionError – If the view's auth config is null.

property gsheet_id: str

Returns: str: The id of the Google Sheet this view is connected to.

Return type
str

get_data(*value_type=EffectiveVal*)

Gets the data from the cells of this Range.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Parameters
value_type (*GoogleValueType, optional*) – Allows you to toggle the type of the values returned by the Google Sheets API. See the *dtypes* documentation for more info on the different GoogleValueTypes.

Returns
This Range.

Return type
Range

write_values(*data*)

Adds a request to write data. Range.commit () to commit the requests.

Parameters
data (*Sequence[Sequence[Any] | Dict[str, Any]]*) – The data to write. Each sequence or dictionary in the passed data is a row, with each value in that sub-iterable being a column. Dictionary keys will be used as a header row in the written data.

Returns
This Tab.

Return type
Range

to_csv(*p*, *header=None*)

Saves values to a csv file.

Parameters

- **p** (*str | Path*) – The path-like for the file to save the data to.

- **header** (*Sequence[Any], optional*) – A header row. If supplied, must be the same number of columns as the data values. Defaults to None.

Return type
None

`to_json`(*p*, *header=0*)

Saves values to a json file, with one json per line.

Parameters

- **p** (*str | Path*) – The path-like for the file to save the data to.
- **header** (*Sequence[Any] | int*) – A header row or the index of a row in the values data to use as the header row. The header will be used as the keys for the json-formatted dictionaries. Defaults to 0, using the first row as the header.

Return type
None

`commit`()

Commits the amassed requests on this view, sending them to the Sheets api as a batch update request.

Returns
The response from the api.

Return type
Dict[str, Any]

Raises
NoConnectionError – If the view's SheetsConnection is null.

`ensure_full_range`(*backup*, *rng=None*)

Convenience method for ensuring that a range argument that could be None or a string is always a FullRange.

Parameters

- **rng** (FullRange | *str, optional*) – A manually generated FullRange or valid FullRange string, defaults to None.
- **backup** (autodrive.interfaces.FullRange) –

Returns
The passed FullRange, or the backup FullRange.

Return type
FullRange

`static gen_alpha_keys`(*num*)

Generates a list of characters from the Latin alphabet a la gsheet/excel headers.

Parameters
num (*int*) – The desired length of the list.

Returns
A list containing as many letters and letter combos as desired. Can be used to generate sets up to 676 in length.

Return type
List[str]

autodrive.tab

Module Contents

Classes

Tab	Provides a connection to a single Google Sheet Tab, its properties, and its

class autodrive.tab.**Tab**(*gsheet_id*, *tab_title*, *tab_idx*, *tab_id*, *column_count=26*, *row_count=1000*, ***, *auth_config=None*, *sheets_conn=None*, *autoconnect=True*)

Provides a connection to a single Google Sheet Tab, its properties, and its data.

Parameters

- **gsheet_id** (*str*) – The id string of the target Google Sheet that the Tab resides in; can be found in the Google Sheet url.
- **tab_title** (*str*) – The name of the Tab.
- **tab_idx** (*int*) – The index (0-based) of the Tab in the Google Sheet.
- **tab_id** (*int*) – The id of the Tab, can be found in the Google Sheet url (after gid=).
- **column_count** (*int, optional*) – The starting number of columns in the Tab, defaults to 26.
- **row_count** (*int, optional*) – The starting number of rows in the Tab, defaults to 1000.
- **auth_config** (AuthConfig, *optional*) – Optional custom AuthConfig, defaults to None.
- **sheets_conn** (SheetsConnection, *optional*) – Optional manually created SheetsConnection, defaults to None.
- **autoconnect** (*bool, optional*) – If you want to instantiate a Tab without immediately checking your authentication credentials and connection to the Google Sheets api, set this to False, defaults to True.

property tab_id: int

Returns: int: This Tab's id. Matches the gid of the parent GSheet's url.

Return type
int

property format_grid: *autodrive.formatting.format_tab.TabGridFormatting*

Returns: TabGridFormatting: An object with grid formatting methods.

Return type
autodrive.formatting.format_tab.TabGridFormatting

property format_text: *autodrive.formatting.format_tab.TabTextFormatting*

Returns: TabTextFormatting: An object with text formatting methods.

Return type
autodrive.formatting.format_tab.TabTextFormatting

property format_cell: *autodrive.formatting.format_tab.TabCellFormatting*

Returns: TabCellFormatting: An object with cell formatting methods.

Return type
autodrive.formatting.format_tab.TabCellFormatting

`property title: str`

Returns: str: The name of this Tab.

Return type
str

`property index: int`

Returns: int: The (0-based) index location of this Tab among the other Tabs on the parent GSheet.

Return type
int

`property column_count: int`

Returns: int: The number of columns in this Tab.

Return type
int

`property row_count: int`

Returns: int: The number of rows in this Tab.

Return type
int

`property range_str: str`

The string representation of the range specified by this Component.

Type
str

Return type
str

`property range: autodrive.interfaces.FullRange`

Returns: FullRange: The FullRange representation of the range specified by this Component.

Return type
autodrive.interfaces.FullRange

`property values: List[List[Any]]`

Returns: List[List[Any]]: The fetched data values in this Component's cells.

Return type
List[List[Any]]

`property formats: List[List[Dict[str, Any]]]`

Returns: List[List[Dict[str, Any]]]: The fetched formatting properties of this Component's cells.

Return type
List[List[Dict[str, Any]]]

`property data_shape: Tuple[int, int]`

Returns: Tuple[int, int]: The row length and column width of this Component's data.

Return type
Tuple[int, int]

property requests: List[Dict[str, Any]]

Returns: List[Dict[str, Any]]: The list of current (uncommitted) requests.

Return type
List[Dict[str, Any]]

property conn: *autodrive.connection.SheetsConnection*

Returns
The view's SheetsConnection.

Return type
SheetsConnection

Raises
NoConnectionError – If the view's connection is null.

property auth: *autodrive.interfaces.AuthConfig*

Returns
The view's AuthConfig.

Return type
AuthConfig

Raises
NoConnectionError – If the view's auth config is null.

property gsheet_id: str

Returns: str: The id of the Google Sheet this view is connected to.

Return type
str

fetch()

Gets the latest metadata from the API for this Tab. Re-populates tab properties like row and column count.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Returns
This GSheet

Return type
Tab

classmethod from_properties(*gsheet_id*, *properties*, *auth_config=None*, *sheets_conn=None*, *autoconnect=True*)

Generates a Tab assigned to the passed gsheet_id with the passed tab properties dictionary from a SheetsConnection.get_properties call.

Unless you have a special use-case, it is probably more trouble than it's worth to try to instantiate a Tab with this method, as it is designed for use by other Autodrive objects.

Parameters

- **gsheet_id** (*str*) – The id of the parent GSheet.
- **properties** (*Dict[str, Any]*) – A properties dictionary, which must contain index, SheetId, title, and gridProperties keys. The gridProperties must be a dictionary containing columnCount and rowCount keys.

- **auth_config** (AuthConfig, *optional*) – Optional custom AuthConfig, defaults to None.
- **sheets_conn** (SheetsConnection, *optional*) – Optional manually created SheetsConnection, defaults to None.
- **autoconnect** (*bool, optional*) – If you want to instantiate a Tab without immediately checking your authentication credentials and connection to the Google Sheets api, set this to False, defaults to True.

Returns
A Tab with the values from the passed properties dictionary.

Return type
Tab

full_range()

Generates a FullRange object corresponding to the full range of the Tab.

Returns
A FullRange from A1:the end of the Tab.

Return type
FullRange

get_data(*rng=None*, *value_type=EffectiveVal*)

Gets the data from the cells of this Tab.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Parameters

- **rng** (FullRange | HalfRange, *optional*) – An optional range value, to specify a subset of the Tab's values to get, defaults to None, which fetches all values in the Tab.
- **value_type** (GoogleValueType, *optional*) – Allows you to toggle the type of the values returned by the Google Sheets API. See the *dtypes* documentation for more info on the different GoogleValueTypes.

Returns
This Tab.

Return type
Tab

write_values(*data*, *rng=None*, *mode='write'*)

Adds a request to write data. Tab.commit () to commit the requests.

Parameters

- **data** (*Sequence[Sequence[Any] | Dict[str, Any]]*) – The data to write. Each sequence or dictionary in the passed data is a row, with each value in that sub-iterable being a column. Dictionary keys will be used as a header row in the written data.
- **rng** (FullRange, *optional*) – A specific range to write to, starting with the top-left-most cell in the range, defaults to None, which will write to the top-left-most cell of the Tab.

- **mode** (*Literal, optional*) – Whether to append the data after any populated rows already present in the Tab or to write to the passed rng. Overrides rng if specified. Defaults to "write".

Returns
This Tab.

Return type
Tab

classmethod new_tab_request(*tab_title*, *tab_id=None*, *tab_idx=None*, *num_rows=1000*, *num_cols=26*)

Creates a dictionary request to create a new tab in a Google Sheet.

Parameters

- **tab_title** (*str*) – The name of the tab.
- **tab_id** (*int, optional*) – The desired id of the tab, which cannot already exist in the Google Sheet, defaults to None, which will allow the Google Sheet to generate the tab_id.
- **tab_idx** (*int, optional*) – The (0-based) index to create the tab at, defaults to None, meaning the tab will be created as the last tab in the Google Sheet.
- **num_rows** (*int, optional*) – The starting number of rows in the new tab, defaults to 1000.
- **num_cols** (*int, optional*) – The starting number of columns in the new tab, defaults to 26.

Returns
A dictionary ready to be passed as a request via a request commit.

Return type
Dict[str, Any]

gen_add_tab_request()

Generates a new tab request dictionary for this Tab. Useful when you have manually instantiated a Tab object instead of fetching it and want to add it to the parent GSheet.

Returns
A dictionary ready to be passed as a request via a request commit.

Return type
Dict[str, Any]

create()

Convenience method for generating a new tab request based on this Tab and immediately committing it, thus adding it to the parent Google Sheet.

Note: This method will cause a request to be posted to the relevant Google API immediately.

Returns
This Tab.

Return type
Tab

gen_range(*rng*)

Convenience method for generating a new Range object in this Tab.

Parameters
rng (FullRange) – The desired FullRange of the new Range object.

Returns
The newly generated Range.

Return type
Range

to_csv(*p*, *header=None*)

Saves values to a csv file.

Parameters

- **p** (*str | Path*) – The path-like for the file to save the data to.
- **header** (*Sequence[Any], optional*) – A header row. If supplied, must be the same number of columns as the data values. Defaults to None.

Return type
None

to_json(*p*, *header=0*)

Saves values to a json file, with one json per line.

Parameters

- **p** (*str | Path*) – The path-like for the file to save the data to.
- **header** (*Sequence[Any] | int*) – A header row or the index of a row in the values data to use as the header row. The header will be used as the keys for the json-formatted dictionaries. Defaults to 0, using the first row as the header.

Return type
None

commit()

Commits the amassed requests on this view, sending them to the Sheets api as a batch update request.

Returns
The response from the api.

Return type
Dict[str, Any]

Raises
NoConnectionError – If the view's SheetsConnection is null.

ensure_full_range(*backup*, *rng=None*)

Convenience method for ensuring that a range argument that could be None or a string is always a FullRange.

Parameters

- **rng** (FullRange | *str, optional*) – A manually generated FullRange or valid FullRange string, defaults to None.
- **backup** (autodrive.interfaces.FullRange) –

Returns
The passed FullRange, or the backup FullRange.

Return type
FullRange

static gen_alpha_keys(*num*)

Generates a list of characters from the Latin alphabet a la gsheet/excel headers.

Parameters
num (*int*) – The desired length of the list.

Returns
A list containing as many letters and letter combos as desired. Can be used to generate sets up to 676 in length.

Return type
List[str]

autodrive.typing

Module Contents

autodrive.typing.**View**

Supertype for Ranges, Tabs, and GSheets.

autodrive.typing.**Connection**

Supertype for SheetsConnections and DriveConnections.

8.1.3 Package Contents

Classes

Drive	A one-stop shop for finding and managing Google Drive objects, including
Folder	Stores all the necessary properties of a Google Drive Folder and provides
Boolean	Boolean datatype, appears in Google Sheets as FALSE or TRUE.
EffectiveVal	EffectiveVal is the value as displayed in Google Sheets, and is appropriately
FormattedVal	The untyped string value of the cell as displayed in Google Sheets. Essentially
Formula	Formula datatype, essentially a string, but begins with =.
Number	Number datatype, covers both floats and integers, though internally Number is
String	String datatype, covers all non-numeric/date text that isn't a formula.
UserEnteredVal	UserEnteredVal is the value as entered by the user, without any calculations
GSheet	Provides a connection to a single Google Sheet and access to its properties
AuthConfig	Optional custom configuration for Autodrive's authentication processes using
Color	An RGBA color value.
NumericFormat	A Google Sheet Number Format, which the value of the cell (s) will be parsed
HalfRange	A partial range used in requests to the Google Sheets API when the axis of the
TextFormat	Provides parameters available in updating the text formatting of a cell.
FullRange	A complete Google Sheets range (indicating start and end row and column as
Range	Provides a connection to the data in a specific range in a Google Sheet Tab.
Tab	Provides a connection to a single Google Sheet Tab, its properties, and its
FileUpload	Use FileUploads when you need more complicated file upload instructions.

Attributes

AccountingFormat	Corresponds to the Accounting `$ (1,000.12)` format.
NumberFormat	Corresponds to the Number `1,000.12` format.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

a

INDEX

C

D

E

F

G

H

I

K

M

N

P

R

S

T

U

V

W