formulas Documentation

Release 1.1.1

Vincenzo Arcidiacono

Dec 23, 2021

TABLE OF CONTENTS

2021-10-13 13:30:00

https://github.com/vinci1it2000/formulas

https://pypi.org/project/formulas/

http://formulas.readthedocs.io/

https://github.com/vinci1it2000/formulas/wiki/

http://github.com/vinci1it2000/formulas/releases/

https://donorbox.org/formulas

excel, formulas, interpreter, compiler, dispatch

- Vincenzo Arcidiacono <vinci1it2000@gmail.com>

CHAPTER

ONE

WHAT IS FORMULAS?

formulas implements an interpreter for Excel formulas, which parses and compile Excel formulas expressions.

Moreover, it compiles Excel workbooks to python and executes without using the Excel COM server. Hence, **Excel is not needed**.

CHAPTER

TWO

INSTALLATION

To install it use (with root privileges):

```
$ pip install formulas
```

Or download the last git version and use (with root privileges):

```
$ python setup.py install
```

2.1 Install extras

Some additional functionality is enabled installing the following extras:

- excel: enables to compile Excel workbooks to python and execute using: `ExcelModel`.
- plot: enables to plot the formula ast and the Excel model.

To install formulas and all extras, do:

```
$ pip install formulas[all]
```

2.2 Development version

To help with the testing and the development of *formulas*, you can install the development version:

```
$ pip install https://github.com/vinci1it2000/formulas/archive/dev.zip
```

2.2.1 What is formulas?

formulas implements an interpreter for Excel formulas, which parses and compile Excel formulas expressions.

Moreover, it compiles Excel workbooks to python and executes without using the Excel COM server. Hence, **Excel is not needed**.

2.2.2 Installation

To install it use (with root privileges):

```
$ pip install formulas
```

Or download the last git version and use (with root privileges):

```
$ python setup.py install
```

2.2.2.1 Install extras

Some additional functionality is enabled installing the following extras:

- excel: enables to compile Excel workbooks to python and execute using: `ExcelModel`.
- plot: enables to plot the formula ast and the Excel model.

To install formulas and all extras, do:

```
$ pip install formulas[all]
```

2.2.2.2 Development version

To help with the testing and the development of *formulas*, you can install the development version:

```
$ pip install https://github.com/vinci1it2000/formulas/archive/dev.zip
```

2.2.3 Basic Examples

The following sections will show how to:

- parse a Excel formulas;
- load, compile, and execute a Excel workbook;
- extract a sub-model from a Excel workbook;
- add a custom function.

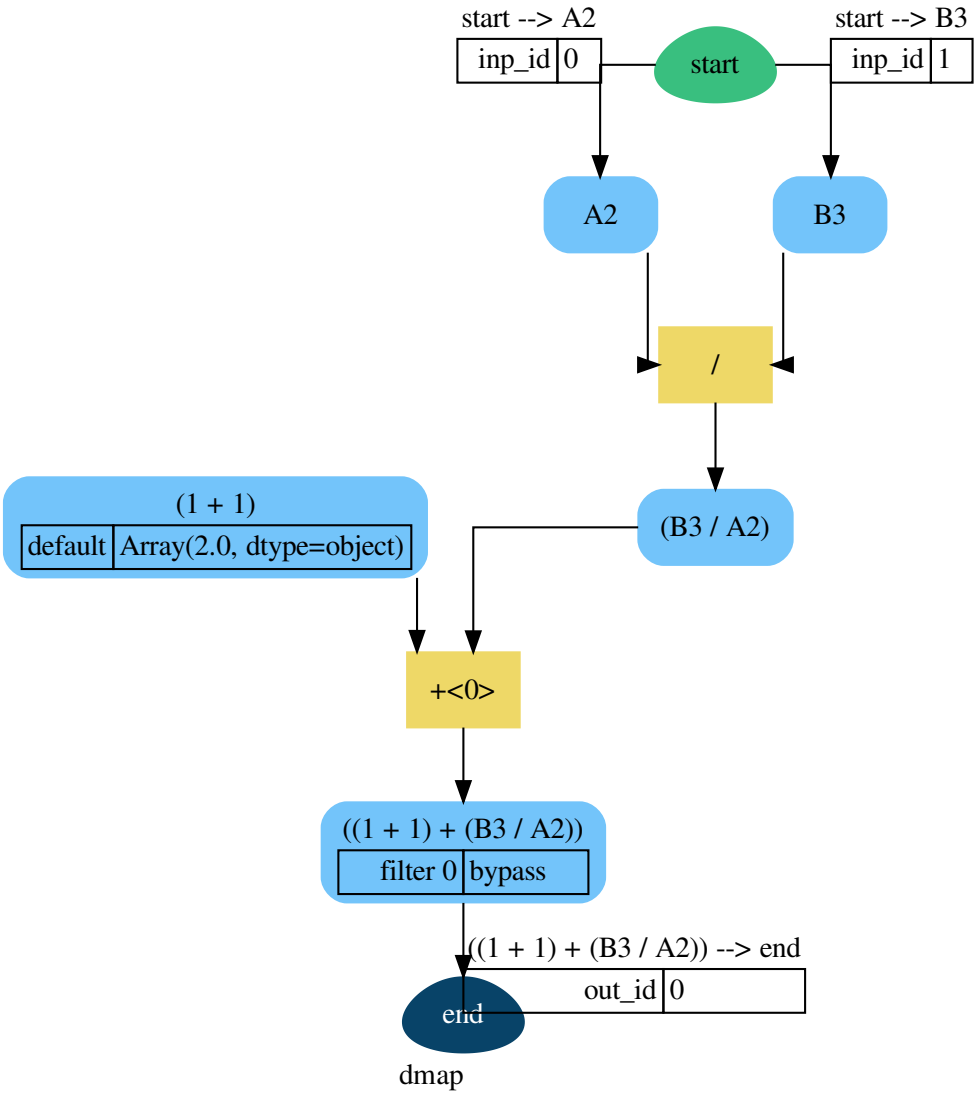
2.2.3.1 Parsing formula

An example how to parse and execute an Excel formula is the following:

```
>>> import formulas
>>> func = formulas.Parser().ast('=(1 + 1) + B3 / A2')[1].compile()
```

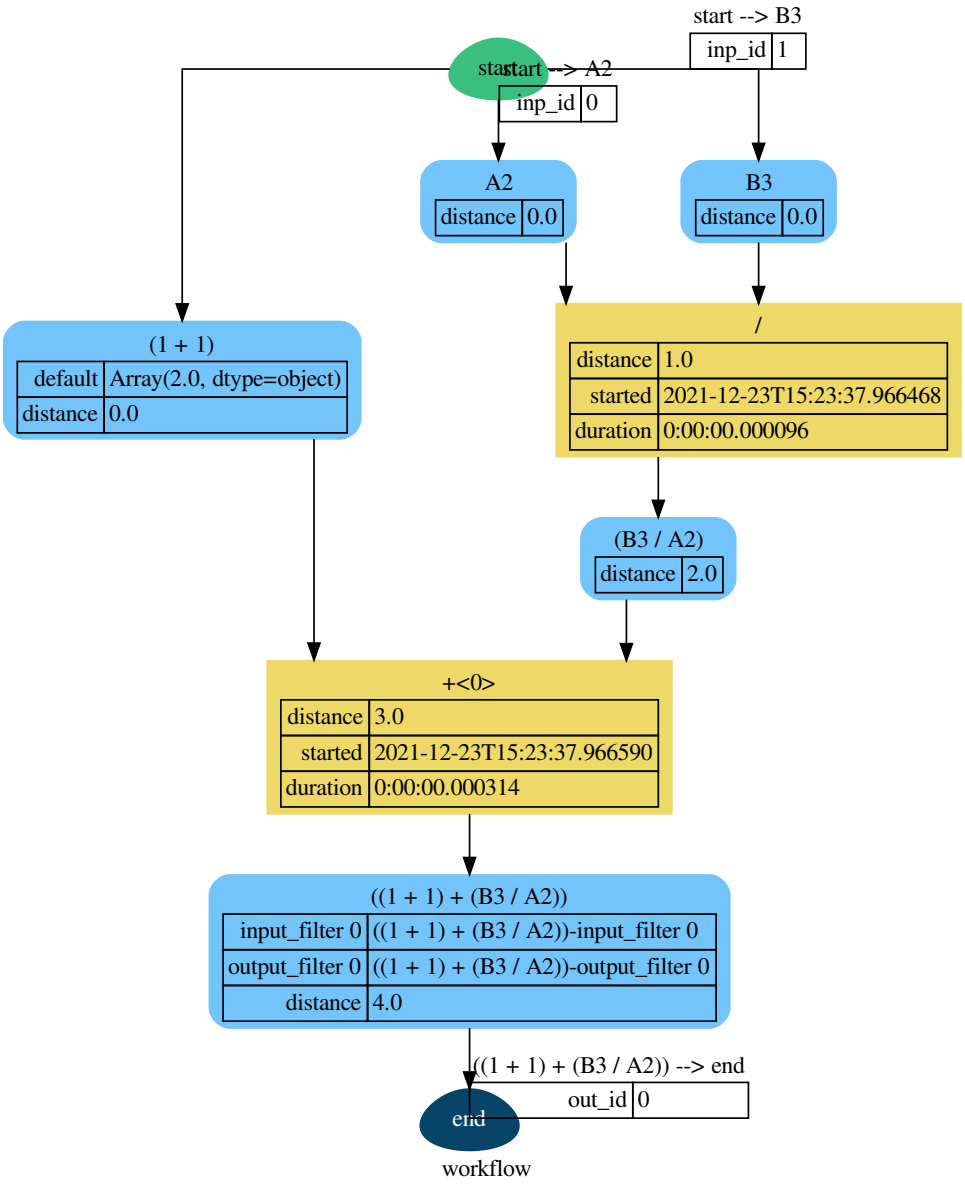
To visualize formula model and get the input order you can do the following:

```
>>> list(func.inputs)
['A2', 'B3']
>>> func.plot(view=False)  # Set view=True to plot in the default browser.
SiteMap([(=((1 + 1) + (B3 / A2)), SiteMap())])
```



Finally to execute the formula and plot the workflow:

```
>>> func(1, 5)
Array(7.0, dtype=object)
>>> func.plot(workflow=True, view=False)  # Set view=True to plot in the default browser.
SiteMap([(=((1 + 1) + (B3 / A2)), SiteMap())])
```



2.2.3.2 Excel workbook

An example how to load, calculate, and write an Excel workbook is the following:

```
>>> import formulas
>>> fpath, dir_output = 'excel.xlsx', 'output'
>>> xl_model = formulas.ExcelModel().loads(fpath).finish()
>>> xl_model.calculate()
Solution(...)
>>> xl_model.write(dirpath=dir_output)
```

(continues on next page)

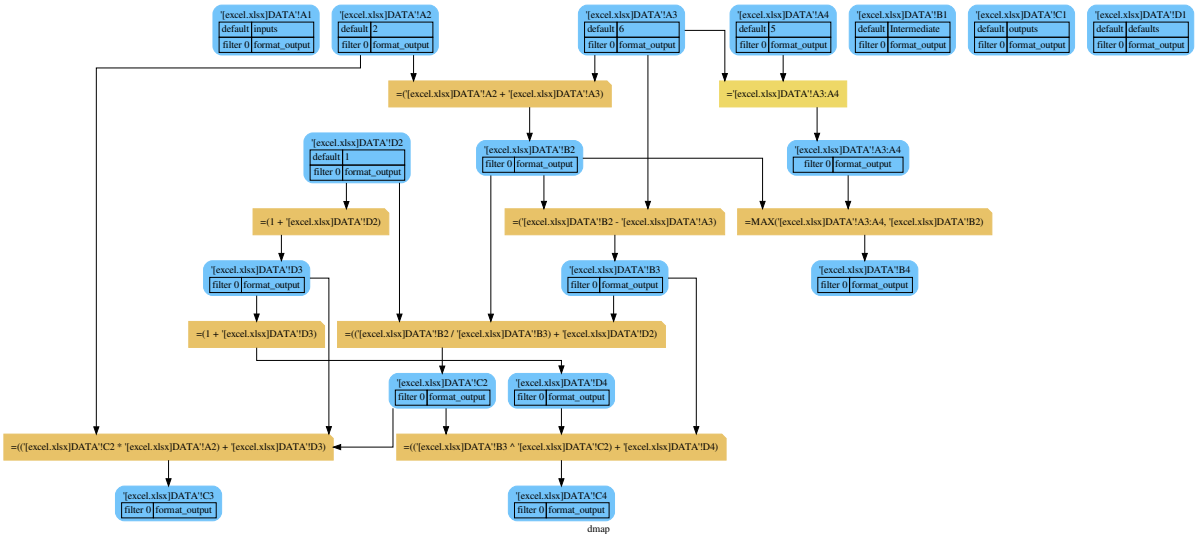
(continued from previous page)

```
{'EXCEL.XLSX': {Book: <openpyxl.workbook.workbook.Workbook ...>}}
```

Tip: If you have or could have **circular references**, add *circular=True* to *finish* method.

To plot the dependency graph that depict relationships between Excel cells:

```
>>> dsp = xl_model.dsp
>>> dsp.plot(view=False)  # Set view=True to plot in the default browser.
SiteMap([(ExcelModel, SiteMap(...))])
```



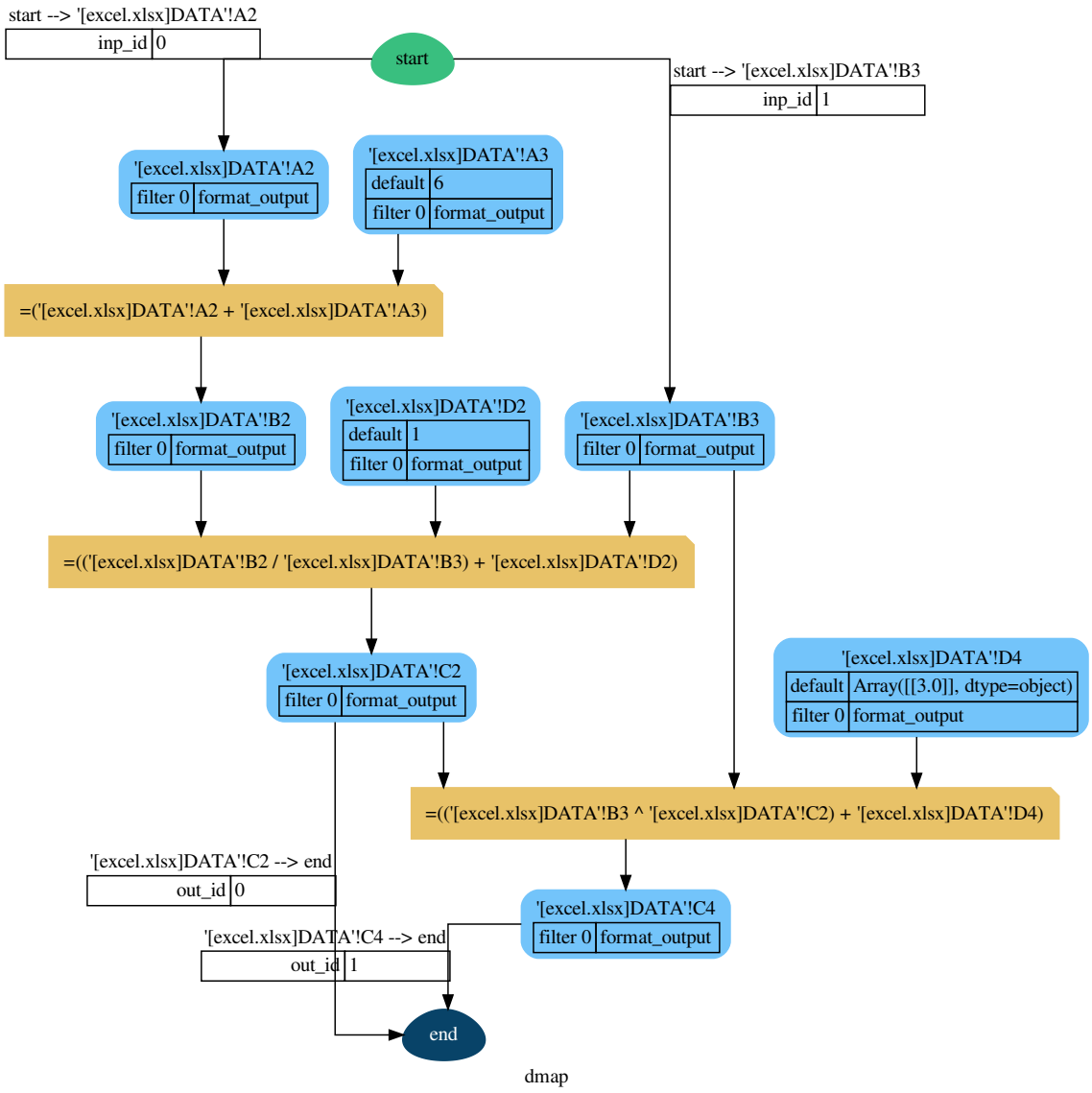
To overwrite the default inputs that are defined by the excel file or to impose some value to a specific cell:

```
>>> xl_model.calculate(
...     inputs={
...         "'[excel.xlsx]DATA'!A2": 3,  # To overwrite the default value.
...         "'[excel.xlsx]DATA'!B3": 1  # To impose a value to B3 cell.
...     },
...     outputs=[
...         "'[excel.xlsx]DATA'!C2", "'[excel.xlsx]DATA'!C4"
...     ] # To define the outputs that you want to calculate.
... )
Solution([("'[excel.xlsx]DATA'!A2", <Ranges>('[excel.xlsx]DATA'!A2)=[[3]]),
          ("'[excel.xlsx]DATA'!A3", <Ranges>('[excel.xlsx]DATA'!A3)=[[6]]),
          ("'[excel.xlsx]DATA'!B3", <Ranges>('[excel.xlsx]DATA'!B3)=[[1]]),
          ("'[excel.xlsx]DATA'!D2", <Ranges>('[excel.xlsx]DATA'!D2)=[[1]]),
          ("'[excel.xlsx]DATA'!B2", <Ranges>('[excel.xlsx]DATA'!B2)=[[9.0]]),
          ("'[excel.xlsx]DATA'!D3", <Ranges>('[excel.xlsx]DATA'!D3)=[[2.0]]),
          ("'[excel.xlsx]DATA'!C2", <Ranges>('[excel.xlsx]DATA'!C2)=[[10.0]]),
          ("'[excel.xlsx]DATA'!D4", <Ranges>('[excel.xlsx]DATA'!D4)=[[3.0]]),
          ("'[excel.xlsx]DATA'!C4", <Ranges>('[excel.xlsx]DATA'!C4)=[[4.0]])])
```

To build a single function out of an excel model with fixed inputs and outputs, you can use the *compile* method of the *ExcelModel* that returns a DispatchPipe. This is a function where the inputs and outputs are defined by the data node

ids (i.e., cell references).

```
>>> func = xl_model.compile(
...     inputs=[
...         "'[excel.xlsx]DATA'!A2",  # First argument of the function.
...         "'[excel.xlsx]DATA'!B3"   # Second argument of the function.
...     ], # To define function inputs.
...     outputs=[
...         "'[excel.xlsx]DATA'!C2", "'[excel.xlsx]DATA'!C4"
...     ] # To define function outputs.
... )
>>> func
<schedula.utils.dsp.DispatchPipe object at ...>
>>> [v.value[0, 0] for v in func(3, 1)]  # To retrieve the data.
[10.0, 4.0]
>>> func.plot(view=False)  # Set view=True to plot in the default browser.
SiteMap([(ExcelModel, SiteMap(...))])
```



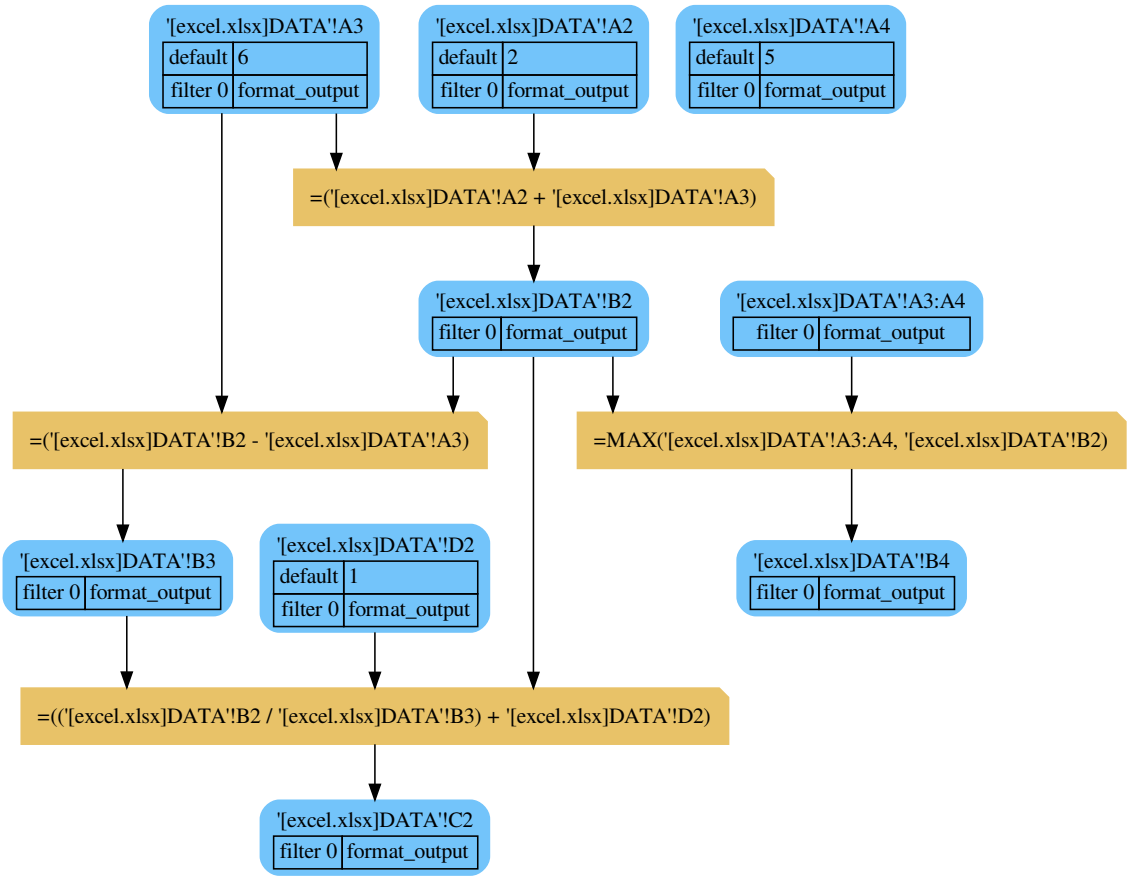
Alternatively, to load a partial excel model from the output cells, you can use the *from_ranges* method of the *ExcelModel*:

```
>>> xl = formulas.ExcelModel().from_ranges(
...     "'[%s]DATA'!C2:D2" % fpath,  # Output range.
...     "'[%s]DATA'!B4" % fpath,  # Output cell.
... )
>>> dsp = xl.dsp
>>> sorted(dsp.data_nodes)
["'[excel.xlsx]DATA'!A2",
 "'[excel.xlsx]DATA'!A3",
 "'[excel.xlsx]DATA'!A3:A4",
 "'[excel.xlsx]DATA'!A4",
```

(continues on next page)

(continued from previous page)

```
 "'[excel.xlsx]DATA'!B2",
 "'[excel.xlsx]DATA'!B3",
 "'[excel.xlsx]DATA'!B4",
 "'[excel.xlsx]DATA'!C2",
 "'[excel.xlsx]DATA'!D2"]
```



dmap

JSON export/import

The *ExcelModel* can be exported/imported to/from a readable JSON format. The reason of this functionality is to have format that can be easily maintained (e.g. using version control programs like *git*). Follows an example on how to export/import to/from JSON an *ExcelModel*:

```
>>> import json
>>> xl_dict = xl_model.to_dict()  # To JSON-able dict.
>>> xl_dict  # Exported format.
{
 "'[excel.xlsx]DATA'!A1": "inputs",
 "'[excel.xlsx]DATA'!B1": "Intermediate",
```

(continues on next page)

(continued from previous page)

```
 "'[excel.xlsx]DATA'!C1": "outputs",
 "'[excel.xlsx]DATA'!D1": "defaults",
 "'[excel.xlsx]DATA'!A2": 2,
 "'[excel.xlsx]DATA'!D2": 1,
 "'[excel.xlsx]DATA'!A3": 6,
 "'[excel.xlsx]DATA'!A4": 5,
 "'[excel.xlsx]DATA'!B2": "=('[excel.xlsx]DATA'!A2 + '[excel.xlsx]DATA'!A3)",
 "'[excel.xlsx]DATA'!C2": "=(('[excel.xlsx]DATA'!B2 / '[excel.xlsx]DATA'!B3) + '[excel.
↪xlsx]DATA'!D2)",
 "'[excel.xlsx]DATA'!B3": "=('[excel.xlsx]DATA'!B2 - '[excel.xlsx]DATA'!A3)",
 "'[excel.xlsx]DATA'!C3": "=(('[excel.xlsx]DATA'!C2 * '[excel.xlsx]DATA'!A2) + '[excel.
↪xlsx]DATA'!D3)",
 "'[excel.xlsx]DATA'!D3": "=(1 + '[excel.xlsx]DATA'!D2)",
 "'[excel.xlsx]DATA'!B4": "=MAX('[excel.xlsx]DATA'!A3:A4, '[excel.xlsx]DATA'!B2)",
 "'[excel.xlsx]DATA'!C4": "=(('[excel.xlsx]DATA'!B3 ^ '[excel.xlsx]DATA'!C2) + '[excel.
↪xlsx]DATA'!D4)",
 "'[excel.xlsx]DATA'!D4": "=(1 + '[excel.xlsx]DATA'!D3)"
}
>>> xl_json = json.dumps(xl_dict, indent=True)  # To JSON.
>>> xl_model = formulas.ExcelModel().from_dict(json.loads(xl_json))  # From JSON.
```

2.2.3.3 Custom functions

An example how to add a custom function to the formula parser is the following:

```
>>> import formulas
>>> FUNCTIONS = formulas.get_functions()
>>> FUNCTIONS['MYFUNC'] = lambda x, y: 1 + y + x
>>> func = formulas.Parser().ast('=MYFUNC(1, 2)')[1].compile()
>>> func()
4
```

2.2.4 Next moves

Things yet to do: implement the missing Excel formulas.

2.2.5 Contributing to formulas

If you want to contribute to **formulas** and make it better, your help is very welcome. The contribution should be sent by a *pull request*. Next sections will explain how to implement and submit a new excel function:

- clone the repository
- implement a new function/functionality
- open a pull request

2.2.5.1 Clone the repository

The first step to contribute to **formulas** is to clone the repository:

- Create a personal fork of the formulas repository on Github.
- Clone the fork on your local machine. Your remote repo on Github is called `origin`.
- Add the original repository as a remote called `upstream`, to maintain updated your fork.
- If you created your fork a while ago be sure to pull `upstream` changes into your local repository.
- Create a new branch to work on! Branch from `dev`.

2.2.5.2 How to implement a new function

Before coding, study the Excel function that you want to implement. If there is something similar implemented in **formulas**, try to get inspired by the implemented code (I mean, not reinvent the wheel) and to use `numpy`. Follow the code style of the project, including indentation. Add or change the documentation as needed. Make sure that you have implemented the **full function syntax**, including the array syntax.

Test cases are very important. This library uses a data-driven testing approach. To implement a new function I recommend the test-driven development cycle. Hence, when you implement a new function, you should write new test cases in `test_cell/TestCell.test_output` suite to execute in the *cycle loop*. When you think that the code is ready, add new raw test in `test/test_files/test.xlsx` (please follow the standard used for other functions) and run the `test_excel/TestExcelModel.test_excel_model`. This requires more time but is needed to test the **array syntax** and to check if the Excel documentation respects the reality.

When all test cases are ok (`python setup.py test`), open a pull request.

Do do list:

- Study the excel function syntax and behaviour when used as array formula.
- Check if there is something similar implemented in formulas.
- Implement/fix your feature, comment your code.
- Write/adapt tests and run them!

Tip: Excel functions are categorized by their functionality. If you are implementing a new functionality group, add a new module in `formula/function` and in `formula.function.SUBMODULES` and a new worksheet in `test/test_files/test.xlsx` (please respect the format).

Note: A pull request without new test case will not be taken into consideration.

2.2.5.3 How to open a pull request

Well done! Your contribution is ready to be submitted:

- Squash your commits into a single commit with git's interactive rebase. Create a new branch if necessary. Always write your commit messages in the present tense. Your commit message should describe what the commit, when applied, does to the code – not what you did to the code.
- Push your branch to your fork on Github (i.e., `git push origin dev`).
- From your fork open a *pull request* in the correct branch. Target the project's `dev` branch!
- Once the *pull request* is approved and merged you can pull the changes from `upstream` to your local repo and delete your extra branch(es).

2.2.6 Donate

If you want to support the **formulas** development please donate and add your excel function preferences. The selection of the functions to be implemented is done considering the cumulative donation amount per function collected by the campaign.

Note: The cumulative donation amount per function is calculated as the example:

Function	Donator 1	Donator 2	Donator 3	TOT	Implementation order
•	150€	120€	50€	•	•
SUM	50€	40€	25€	125€	1st
SIN	50€		25€	75€	3rd
TAN	50€	40€		90€	2nd
COS		40€		40€	4th

2.2.7 API Reference

The core of the library is composed from the following modules:

It contains a comprehensive list of all modules and classes within formulas.

Modules:

`parser`	It provides formula parser class.
`builder`	It provides AstBuilder class.
`errors`	Defines the formulas exception.
`tokens`	It provides tokens needed to parse the Excel formulas.
`functions`	It provides functions implementations to compile the Excel functions.
`ranges`	It provides Ranges class.
`cell`	It provides Cell class.
`excel`	It provides Excel model class.

2.2.8 Changelog

2.2.8.1 v1.1.1 (2021-10-13)

Feat

- (excel): Improve performances of *complete* method.
- (setup): Add add python 3.9 in setup.py.
- (functions): Add *SEARCH*, *ISNUMBER*, and *EDATE* functions.
- (travis): Update python version for coveralls.

Fix

- (doc): Correct missing documentation link.
- (doc): Correct typo.
- (operator) #70: Correct % operator preceded by space.

2.2.8.2 v1.1.0 (2021-02-16)

Feat

- (look) #57: Add *SINGLE* function.
- (function) #51: Add google Excel functions.
- (logic) #55, #57: Add IFS function.
- (excel) #65: Add documentation and rename method to load models from ranges.
- (excel) #65: Add method to load sub-models from range.
- (doc): Update Copyright.
- (excel): Improve performances.
- (excel) #64: Read model from outputs.
- (core): Update range definition with path file.
- (excel) #64: Add warning for missing reference.
- (excel) #64: Add warning message when book loading fails.
- (readme) #44: Add example to export and import the model to JSON format.
- (readme) #53: Add instructions to install the development version.
- (excel) #44: Add feature to export and import the model to JSON- able dict.
- (stat, comp) #43: Add *STDEV*, *STDEV.S*, *STDEV.P*, *STDEVA*, *STDEVPA*, *VAR*, *VAR.S*, *VAR.P*, *VARA*, and *VARPA* functions.

Fix

- (financial): Correct requirements for *irr* function.
- (excel) #48: Correct reference pointing to different workbooks.
- (function) #67: Correct compilation of impure functions (e.g., *rand*, *now*, etc.).
- (look) #66: Correct *check* function did not return value.
- (test): Remove *temp* dir.
- (excel): Correct external link reading.
- (operator) #63: Correct operator parser when starts with spaces.
- (text) #61: Convert float as int when stringify if it is an integer.
- (math) #59: Convert string to number in math operations.
- (functions): Correct *_xfilter* operating range type.
- (parser) #61: Skip *n* in formula expression.
- (operator) #58: Correct operator parser for composed operators.
- (excel): Correct invalid range definition and missing sheet or files.
- (operand) #52: Correct range parser.
- (operand) #50: Correct sheet name parser with space.
- (tokens): Correct closure parenthesis parser.
- (excel): Skip function compilation for string cells.
- (tokens): Correct error parsing when sheet name is defined.

2.2.8.3 v1.0.0 (2020-03-12)

Feat

- (core): Add *CODE_OF_CONDUCT.md*.
- (function) #39: Transform *NotImplementedError* into *#NAME?*.
- (text) #39: Add *CONCAT* and *CONCATENATE* functions.
- (logic) #38: Add TRUE/FALSE functions.
- (excel) #42: Save missing nodes.
- (excel) #42: Update logic for *RangesAssembler*.
- (excel): Improve performance of *finish* method.
- (core): Update build script.
- (core): Add support for python 3.8 and drop python 3.5 and drop *appveyor*.
- (core): Improve memory performance.
- (refact): Update copyright.
- (operand): Add *fast_range2parts_v4* for named ranges.

Fix

- (math) #37: Match excel default rounding algorithm of round half up.
- (cell): Correct reference in *push* method.
- (readme): Correct doctest.
- (token): Correct separator parser.
- (excel) #35: Update logic to parse named ranges.
- (operand): Associate *excel_id==0* to current excel.
- (array): Ensure correct deepcopy of *Array* attributes.
- (operand) #39: Correct range parser for named ranges.
- (operand) #41: Correct named ranges parser.

2.2.8.4 v0.4.0 (2019-08-31)

Feat

- (doc): Add binder.
- (setup): Add env *ENABLE_SETUP_LONG_DESCRIPTION*.
- (core): Add useful constants.
- (excel): Add option to write all calculate books inside a folder.
- (stat) #21: Add *COUNTBLANK*, *LARGE*, *SMALL* functions.
- (date) #35: Add *NPV*, *XNPV*, *IRR*, *XIRR* functions.
- (stat) #21: Add *AVERAGEIF*, *COUNT*, *COUNTA*, *COUNTIF* functions.
- (math) #21: Add *SUMIF* function.
- (date) #21, #35, #36: Add *date* functions *DATE*, *DATEVALUE*, *DAY*, *MONTH*, *YEAR*, *TODAY*, *TIME*, *TIMEVALUE*, *SECOND*, *MINUTE*, *HOUR*, *NOW*, *YEARFRAC*.
- (info) #21: Add *NA* function.
- (date) #21, #35, #36: Add *date* functions *DATE*, *DATEVALUE*, *DAY*, *MONTH*, *YEAR*, *TODAY*, *TIME*, *TIMEVALUE*, *SECOND*, *MINUTE*, *HOUR*, *NOW*, *YEARFRAC*.
- (stat) #35: Add *MINA*, *AVERAGEA*, *MAXA* functions.

Fix

- (setup): Update tests requirements.
- (setup): Correct setup dependency (*beautifulsoup4*).
- (stat): Correct round indices.
- (setup) #34: Build universal wheels.
- (test): Correct import error.
- (date) #35: Correct behaviour of *LOOKUP* function when dealing with errors.

- (excel) #35: Improve cycle detection.
- (excel,date) #21, #35: Add custom Excel Reader to parse raw datetime.
- (excel) #35: Correct when definedName is relative *#REF!*.

2.2.8.5 v0.3.0 (2019-04-24)

Feat

- (logic) #27: Add *OR*, *XOR*, *AND*, *NOT* functions.
- (look) #27: Add *INDEX* function.
- (look) #24: Improve performances of *look* functions.
- (functions) #26: Add *SWITCH*.
- (functions) #30: Add *GCD* and *LCM*.
- (chore): Improve performances avoiding *combine_dicts*.
- (chore): Improve performances checking intersection.

Fix

- (tokens): Correct string nodes ids format adding *"*.
- (ranges): Correct behaviour union of ranges.
- (import): Enable PyCharm autocomplete.
- (import): Save imports.
- (test): Add repo path to system path.
- (parser): Parse empty args for functions.
- (functions) #30: Correct implementation of *GCD* and *LCM*.
- (ranges) #24: Enable full column and row reference.
- (excel): Correct bugs due to new *openpyxl*.

2.2.8.6 v0.2.0 (2018-12-11)

Feat

- (doc) #23: Enhance *ExcelModel* documentation.

Fix

- (core): Add python 3.7 and drop python 3.4.
- (excel): Make *ExcelModel* dillable and pickable.
- (builder): Avoid FormulaError exception during formulas compilation.
- (excel): Correct bug when compiling excel with circular references.

2.2.8.7 v0.1.4 (2018-10-19)

Fix

- (tokens) #20: Improve Number regex.

2.2.8.8 v0.1.3 (2018-10-09)

Feat

- (excel) #16: Solve circular references.
- (setup): Add donate url.

Fix

- (functions) #18: Enable *check_error* in *IF* function just for the first argument.
- (functions) #18: Disable *input_parser* in *IF* function to return any type of values.
- (rtd): Define *fpath* from *prj_dir* for rtd.
- (rtd): Add missing requirements *openpyxl* for rtd.
- (setup): Patch to use *sphinxcontrib.restbuilder* in setup *long_description*.

Other

- Update documentation.
- Replace *excel* with *Excel*.
- Create PULL_REQUEST_TEMPLATE.md.
- Update issue templates.
- Update copyright.
- (doc): Update author mail.

2.2.8.9 v0.1.2 (2018-09-12)

Feat

- (functions) #14: Add *ROW* and *COLUMN*.
- (cell): Pass cell reference when compiling cell + new function struct with dict to add inputs like CELL.

Fix

- (ranges): Replace system max size with excel max row and col.
- (tokens): Correct number regex.

2.2.8.10 v0.1.1 (2018-09-11)

Feat

- (contrib): Add contribution instructions.
- (setup): Add additional project_urls.
- (setup): Update *Development Status* to *4 - Beta*.

Fix

- (init) #15: Replace *FUNCTIONS* and *OPERATORS* objs with *get_functions*, *SUBMODULES*.
- (doc): Correct link docs_status.

2.2.8.11 v0.1.0 (2018-07-20)

Feat

- (readme) #6, #7: Add examples.
- (doc): Add changelog.
- (test): Add info of executed test of *test_excel_model*.
- (functions) #11: Add *HEX2OCT*, *HEX2BIN*, *HEX2DEC*, *OCT2HEX*, *OCT2BIN*, *OCT2DEC*, *BIN2HEX*, *BIN2OCT*, *BIN2DEC*, *DEC2HEX*, *DEC2OCT*, and *DEC2BIN* functions.
- (setup) #13: Add extras_require to setup file.

Fix

- (excel): Use DispatchPipe to compile a sub model of excel workbook.
- (range) #11: Correct range regex to avoid parsing of function like ranges (e.g., HEX2DEC).

2.2.8.12 v0.0.10 (2018-06-05)

Feat

- (look): Simplify *_get_type_id* function.

Fix

- (functions): Correct ImportError for FUNCTIONS.
- (operations): Correct behaviour of the basic operations.

2.2.8.13 v0.0.9 (2018-05-28)

Feat

- (excel): Improve performances pre-calculating the range format.
- (core): Improve performances using *DispatchPipe* instead *SubDispatchPipe* when compiling formulas.
- (function): Improve performances setting *errstate* outside vectorization.
- (core): Improve performances of range2parts function (overall 50% faster).

Fix

- (ranges): Minimize conversion str to int and vice versa.
- (functions) #10: Avoid returning shapeless array.

2.2.8.14 v0.0.8 (2018-05-23)

Feat

- (functions): Add *MATCH*, *LOOKUP*, *HLOOKUP*, *VLOOKUP* functions.
- (excel): Add method to compile *ExcelModel*.
- (travis): Run coveralls in python 3.6.
- (functions): Add *FIND*,`LEFT`,`LEN`,`LOWER`,`MID`,`REPLACE`,`RIGHT`,`TRIM`, and`UPPER` functions.
- (functions): Add *IRR* function.
- (formulas): Custom reshape to Array class.
- (functions): Add *ISO.CEILING*, *SQRTPI*, *TRUNC* functions.

- (functions): Add *ROUND*, *ROUNDDOWN*, *ROUNDUP*, *SEC*, *SECH*, *SIGN* functions.
- (functions): Add *DECIMAL*, *EVEN*, *MROUND*, *ODD*, *RAND*, *RANDBETWEEN* functions.
- (functions): Add *FACT* and *FACTDOUBLE* functions.
- (functions): Add *ARABIC* and *ROMAN* functions.
- (functions): Parametrize function *wrap_ufunc*.
- (functions): Split function *raise_errors* adding *get_error* function.
- (ranges): Add custom default and error value for defining ranges Arrays.
- (functions): Add *LOG10* function + fix *LOG*.
- (functions): Add *CSC* and *CSCH* functions.
- (functions): Add *COT* and *COTH* functions.
- (functions): Add *FLOOR*, *FLOOR.MATH*, and *FLOOR.PRECISE* functions.
- (test): Improve log message of test cell.

Fix

- (rtd): Update installation file for read the docs.
- (functions): Remove unused functions.
- (formulas): Avoid too broad exception.
- (functions.math): Drop scipy dependency for calculate factorial2.
- (functions.logic): Correct error behaviour of *if* and *iferror* functions + add BroadcastError.
- (functions.info): Correct behaviour of *iserr* function.
- (functions): Correct error behaviour of average function.
- (functions): Correct *iserror* and *iserr* returning a custom Array.
- (functions): Now *xceiling* function returns np.nan instead Error.errors['#NUM!'].
- (functions): Correct *is_number* function, now returns False when number is a bool.
- (test): Ensure same order of workbook comparisons.
- (functions): Correct behaviour of *min max* and *int* function.
- (ranges): Ensure to have a value with correct shape.
- (parser): Change order of parsing to avoid TRUE and FALSE parsed as ranges or errors as strings.
- (function):Remove unused kwargs n_out.
- (parser): Parse error string as formulas.
- (readme): Remove *downloads_count* because it is no longer available.

Other

- Refact: Update Copyright + minor pep.
- Excel returns 1-indexed string positions???
- Added common string functions.
- Merge pull request #9 from ecatkins/irr.
- Implemented IRR function using numpy.

2.2.8.15 v0.0.7 (2017-07-20)

Feat

- (appveyor): Add python 3.6.
- (functions) #4: Add *sumproduct* function.

Fix

- (install): Force update setuptools>=36.0.1.
- (functions): Correct *iserror iserr* functions.
- (ranges): Replace '#N/A' with '' as empty value when assemble values.
- (functions) #4: Remove check in ufunc when inputs have different size.
- (functions) #4: Correct *power*, *arctan2*, and *mod* error results.
- (functions) #4: Simplify ufunc code.
- (test) #4: Check that all results are in the output.
- (functions) #4: Correct *atan2* argument order.
- (range) #5: Avoid parsing function name as range when it is followed by *(*.
- (operator) #3: Replace *strip* with *replace*.
- (operator) #3: Correct valid operators like ^- or *+.

Other

- Made the ufunc wrapper work with multi input functions, e.g., power, mod, and atan2.
- Created a workbook comparison method in TestExcelModel.
- Added MIN and MAX to the test.xlsx.
- Cleaned up the ufunc wrapper and added min and max to the functions list.
- Relaxed equality in TestExcelModel and made some small fixes to functions.py.
- Added a wrapper for numpy ufuncs, mapped some Excel functions to ufuncs and provided tests.

2.2.8.16 v0.0.6 (2017-05-31)

Fix

- (plot): Update schedula to 0.1.12.
- (range): Sheet name without commas has this [^Wd][w.] format.

2.2.8.17 v0.0.5 (2017-05-04)

Fix

- (doc): Update schedula to 0.1.11.

2.2.8.18 v0.0.4 (2017-02-10)

Fix

- (regex): Remove deprecation warnings.

2.2.8.19 v0.0.3 (2017-02-09)

Fix

- (appveyor): Setup of lxml.
- (excel): Remove deprecation warning openpyxl.
- (requirements): Update schedula requirement 0.1.9.

2.2.8.20 v0.0.2 (2017-02-08)

Fix

- (setup): setup fails due to long description.
- (excel): Remove deprecation warning *remove_sheet* –> *remove*.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

formulas, 15

INDEX

F

M