UoL Grades Calculator

Release 0.13.1

Sebastien Lavoie

Jan 16, 2023

CONTENTS

CHAPTER

ONE

AVAILABLE COMMANDS

1.1 check

```
$ ugc check --help

Usage: ugc check [OPTIONS] COMMAND [ARGS]...

Perform sanity checks against the results generated.

Options:
--help  Show this message and exit.

Commands:
score-accuracy  Check for rounding errors when averaging module score.
```

1.2 check score-accuracy

```
$ ugc check score-accuracy --help

Usage: ugc check score-accuracy [OPTIONS]

Check for rounding errors when averaging module score.

Options:
--help  Show this message and exit.
```

Example output 1:

```
❯ ugc check score-accuracy
Algorithms and Data Structures I: 94% actual (expected 95.0%)
Discrete Mathematics: 98% actual (expected 100.0%)
Fundamentals of Computer Science: 78% actual (expected 98.0%)
```

```
❯ ugc check score-accuracy
Algorithms and Data Structures I: 94% actual (expected 95.0%)
Discrete Mathematics: 98% actual (expected 100.0%)
Fundamentals of Computer Science: 78% actual (expected 98.0%)
```

Example output 2:

```
❯ ugc check score-accuracy
All module scores are accurate!
```

```
❯ ugc check score-accuracy
All module scores are accurate!
```

1.3 generate-sample

```
$ ugc generate-sample --help

Usage: ugc generate-sample [OPTIONS]

Generate a sample grades JSON config file.

Options:
-f, --force-overwrite  Overwrite the existing config file, if any.
--help                 Show this message and exit.
```

Example output:

```
Configuration file not found: /Users/sglavoie/.ugc-grades.json
Try `ugc generate-sample --help`
→ Configuration file generated.
```

```
Configuration file not found: /Users/sglavoie/.ugc-grades.json
Try `ugc generate-sample --help`
→ Configuration file generated.
```

1.4 plot

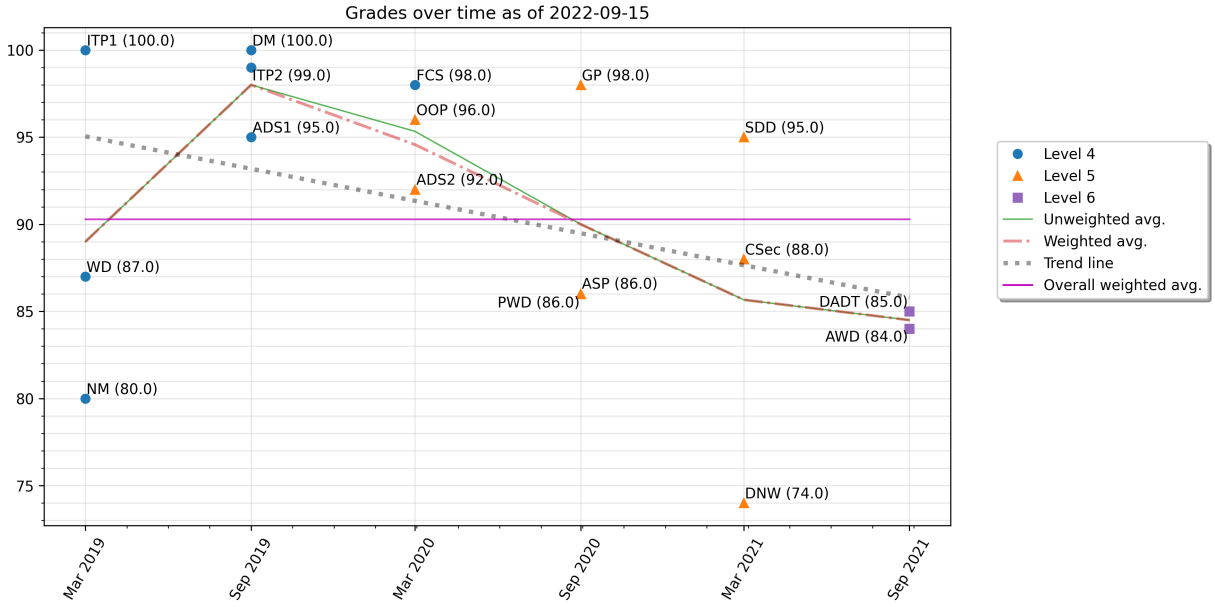
```
$ ugc plot --help

Usage: ugc plot [OPTIONS] COMMAND [ARGS]...

Plot progress made over time.

Options:
--help  Show this message and exit.

Commands:
modules  Produce a scatter plot showing all individual grades.
```

1.5 plot modules

```
$ ugc plot modules --help

Usage: ugc plot modules [OPTIONS]

Produce a scatter plot showing all individual grades.

Options:
-d, --dpi INTEGER RANGE  Specify the output quality in dots per inch.
                         [default: 300;100<=x<=1000]
--filename TEXT          Change the output file name.
--long-module-names      Display the full name of each module.
--no-avg-overall         Remove the weighted average obtained across the
                         degree.
--no-avg-unweighted      Remove the unweighted average per semester.
--no-avg-weighted        Remove the weighted average per semester.
--no-avgs                Remove all unweighted and weighted average lines.
--no-grades              Do not display the grade for any module.
--no-module-names        Remove the display of module names entirely.
--no-trend               Remove the trend line.
--path TEXT              Set the output path to save the generated plot.
--title TEXT             Print a custom title for the graph.
--title-keep-date        Append today's date to the title when used with
                         `--title`.
--title-no-date          Remove the part `as of YYYY-MM-DD` in the title of
                         the graph.
--help                   Show this message and exit.
```

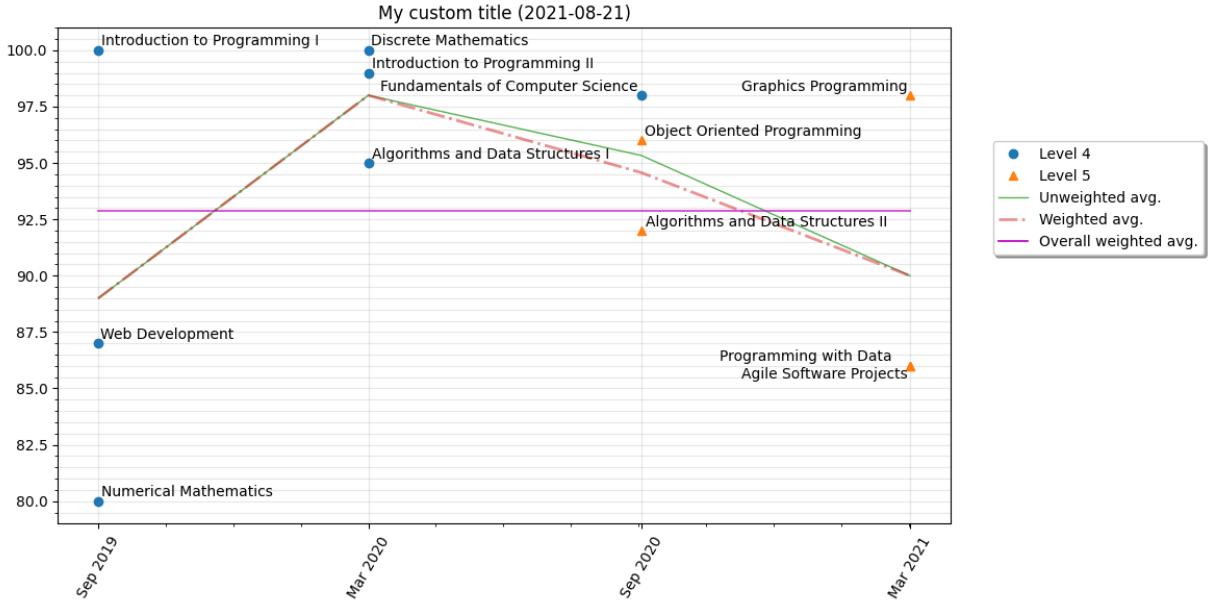
Example output 1:

```
$ ugc plot modules -d 100

Plot saved to /home/user/Downloads/2021-08-21_grades_over_time.png
```



Example output 2:

```
$ ugc plot modules --dpi 100 --filename new_name --long-module-names \
    --no-grades --path ~ --title "My custom title" --title-keep-date --no-trend

Plot saved to /home/user/new_name.png
```



1.6 summarize

```
$ ugc summarize --help

Usage: ugc summarize [OPTIONS] COMMAND [ARGS]...

    Print a summary of the progress made so far.

Options:
    --help  Show this message and exit.

Commands:
    all       Output includes modules done as well as those in progress.
    done      Output includes only modules that are done and dusted.
    progress  Output includes only modules that are in progress.
```

Example output:

```
❯ ugc summarize all
Modules completed
```

Progress made — Modules done

Completion date	Level	Module name	Score	ECTS	US
2018-09	4	How Computers Work	N/A	N/A	N/A
2019-03	4	Introduction to Programming I	100	A	A
2019-03	4	Numerical Mathematics	80	A	B-
2019-03	4	Web Development	87	A	B+
2019-09	4	Algorithms and Data Structures I	95	A	A
2019-09	4	Discrete Mathematics	100	A	A
2019-09	4	Introduction to Programming II	99	A	A
2020-03	4	Fundamentals of Computer Science	98	A	A
2020-03	5	Algorithms and Data Structures II	92	A	A-
2020-03	5	Object Oriented Programming	96	A	A
2020-09	5	Agile Software Projects	86	A	B
2020-09	5	Graphics Programming	98	A	A
2020-09	5	Programming with Data	86	A	B
2021-03	5	Computer Security	88	A	B+
2021-03	5	Databases Networks and the Web	74	A	C
2021-03	5	Software Design and Development	95	A	A
2021-09	6	Advanced Web Development	84	A	B
2021-09	6	Data Science	N/A	N/A	N/A
2021-09	6	Databases and Advanced Data Techniques	85	A	B
2021-09	6	Machine Learning and Neural Networks	N/A	N/A	N/A

```
Weighted average: 89.0 (ECTS: A, US: B+)
Unweighted average: 90.76 (ECTS: A, US: A-)
Classification (weighted): First Class Honours
GPA (weighted): 3.3 US — 4 UK
Total credits done: 300 / 360 (83.33%)

Modules in progress
```

Work in progress — Modules with pending grades

Module name	Level	Midterm	ECTS	US
3D Graphics Animation	6	84	A	B
Mobile Development	6	94	A	A

```
Weighted average (including modules in progress): 89.0 (ECTS: A, US: B+)
Unweighted average (including modules in progress): 90.58 (ECTS: A, US: A-)
```

```
❯ ugc summarize all
Modules completed
```

Progress made — Modules done

Completion date	Level	Module name	Score	ECTS	US
2018-09	4	How Computers Work	N/A	N/A	N/A
2019-03	4	Introduction to Programming I	100	A	A
2019-03	4	Numerical Mathematics	80	A	B-
2019-03	4	Web Development	87	A	B+
2019-09	4	Algorithms and Data Structures I	95	A	A
2019-09	4	Discrete Mathematics	100	A	A
2019-09	4	Introduction to Programming II	99	A	A
2020-03	4	Fundamentals of Computer Science	98	A	A
2020-03	5	Algorithms and Data Structures II	92	A	A-
2020-03	5	Object Oriented Programming	96	A	A
2020-09	5	Agile Software Projects	86	A	B
2020-09	5	Graphics Programming	98	A	A
2020-09	5	Programming with Data	86	A	B
2021-03	5	Computer Security	88	A	B+
2021-03	5	Databases Networks and the Web	74	A	C
2021-03	5	Software Design and Development	95	A	A
2021-09	6	Advanced Web Development	84	A	B
2021-09	6	Data Science	N/A	N/A	N/A
2021-09	6	Databases and Advanced Data Techniques	85	A	B
2021-09	6	Machine Learning and Neural Networks	N/A	N/A	N/A

```
Weighted average: 89.0 (ECTS: A, US: B+)
Unweighted average: 90.76 (ECTS: A, US: A-)
Classification (weighted): First Class Honours
GPA (weighted): 3.3 US — 4 UK
Total credits done: 300 / 360 (83.33%)

Modules in progress
```

Work in progress — Modules with pending grades

Module name	Level	Midterm	ECTS	US
3D Graphics Animation	6	84	A	B
Mobile Development	6	94	A	A

```
Weighted average (including modules in progress): 89.0 (ECTS: A, US: B+)
Unweighted average (including modules in progress): 90.58 (ECTS: A, US: A-)
```

CHAPTER

TWO

FOR DEVELOPERS

2.1 Getting a copy of the source code

Clone this repository.

2.2 Developing locally as a package

Installing the necessary requirements:

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

Building the application once (no need to rebuild to test changes on the source code):

```
$ python setup.py develop
```

Then the command `ugc` (short for `uol grades calculator`) becomes available on the command-line. Type `ugc --help` for more information.

The tool can then be uninstalled using the following command:

```
$ python setup.py develop --uninstall
```

2.3 Running the test suite

Default settings are defined in `pytest.ini`. Then, it's just a matter of typing:

```
$ pytest
```

2.4 Managing dependencies

- Requirements to test and develop the application should go into `requirements-dev.txt`. None of these are required to run `ugc` as a user.
- User requirements should go into `requirements.txt`.
- The section `install_requires` in `setup.cfg` should be kept up-to-date when new releases are to be published.
- Non-Python files (e.g. JSON) used by `ugc` should be explicitly included in `MANIFEST.in` to be distributed with the package.

2.5 Publishing the package to PyPI

```
# Remove existing distribution packages
rm -rf dist build

# The following commands would be run preferably
# from a virtual environment

# Generate the distribution packages for PyPI
python setup.py sdist bdist_wheel --universal

# Upload to the test instance of PyPI
python -m twine upload --repository testpypi dist/*

# Upload to the production instance of PyPI
python -m twine upload dist/*
```

2.6 Installing package from PyPI

Install from test.pypi.org:

- Activate a virtual environment, then:

```
# Latest version
pip install -i https://test.pypi.org/simple/ uol-grades-calculator

# Specific version
pip install -i https://test.pypi.org/simple/ uol-grades-calculator==x.y.z
```

Test as a module:

```
python -m ugc
```

Install from pypi.org:

```
# Latest version
pip install uol-grades-calculator
```

(continues on next page)

(continued from previous page)

```
# Specific version
pip install uol-grades-calculator==x.y.z
```

2.7 Adding ugc as a command

To avoid having to activate a virtual environment and calling the program as a module via `python -m ugc`, one can create an alias or put a symbolic link in the `$PATH` to make the command `ugc` accessible.

2.7.1 Creating an alias

As a quick and dirty way to access `ugc` with an alias, a virtual environment can be activated and the Python interpreter can be called from that environment. Adding an alias like the following would do the trick:

```
# Add to `~/.bash_aliases` or equivalent on your system
alias ugc=". /tmp/.venv/bin/activate && python -m ugc"
```

2.7.2 Adding to the $PATH

When developing locally and assuming all dependencies were installed inside a virtual environment:

```
# Make sure the `ugc` package was installed to allow editing source code
# on the fly:
python setup.py develop

# Create a symbolic link from your virtual environment to a directory
# in your path. You can print it to see what it looks like:
echo $PATH

# For instance, if ~/.local/bin is in $PATH, something as follows would
# work, assuming the virtual environment is named `.venv`:
ln -s /path/to/uol_grades_calculator/.venv/bin/ugc ~/.local/bin/ugc

# Then `ugc` can be called as a regular program:
ugc
```

2.8 Documentation

2.8.1 Generating modules documentation

```
$ cd docs/
$ make docs
```

Table 1: Current options passed to build the docs

Flag	Description
`-f`	overwrite existing files
`-M`	put module documentation before submodule
`-P`	include "_private" modules
`-o`	output directory (`docs/source/`)
`-d`	maximum depth of submodules to show in the TOC (set to 1)
`-T`	do not add a TOC for the modules

Rebuilding documentation

```
$ cd docs/
$ make html
```

If something is not rendered even after a force-refresh (such as when editing the config file or adding custom CSS), try running `make clean html` instead: there can be instances where changes are not applied due to the local cache.

2.8.2 Viewing the documentation locally

Once the documentation has been generated, it can be viewed with a simple server. A built-in option could be as follows:

```
$ cd docs/_build/html  # from the root
$ python -m http.server
```

This will set up a server at http://localhost:8000/ by default.

CHAPTER

THREE

UGC PACKAGE

3.1 Subpackages

3.1.1 ugc.utils package

Submodules

ugc.utils.commands_helpers module

ugc.utils.commands_helpers.**dataframe_get_weighted_average**(*df*, *data_col*, *weight_col*, *by_col*) → float

Calculate the weighted average in a dataframe from a numerical column and an integer column (weight) where the results are grouped by the column *by_col*.

Parameters

- **df** (*DataFrame*) – Pandas dataframe, used to temporarily store new columns.
- **data_col** (*number*) – int or float column from a dataframe.
- **weight_col** (*number*) – int or float column from a dataframe.
- **by_col** (*[type]*) – A dataframe column from which weights should be grouped.

Returns

Weighted average calculated from *data_col* and *weight_col*
and grouped by *by_col*.

Return type
float

ugc.utils.commands_helpers.**dataframe_map_module_to_weight**(*row*) → int

Return the weight of a given module from a dataframe row based on the module level and the module name (since the final project is worth more).

Parameters
row (*dataframe row*) – A row from a dataframe containing at least two columns, *Level* and *Module name*.

Returns
Integer value corresponding to the weight of a module.

Return type
int

ugc.utils.commands_helpers.**dataframe_parse_datetime_as_month_year**(*row*) → str

Take in a dataframe row, get a timestamp from a column and return a formatted string in the form MMM YYYY, where MMM is the abbreviation of a month's name.

Parameters
row – A dataframe row.

Returns
A formatted string of the form "MMM YYYY".

Return type
str

ugc.utils.commands_helpers.**generate_sample_copy_config_file_and_print_message**(*config_path: str*) → dict

ugc.utils.commands_helpers.**get_module_score_rounded_up**(*module*) → float

ugc.utils.commands_helpers.**get_modules_done_dataframe**(*grades: Grades*, *finished_modules: list*) → DataFrame

ugc.utils.commands_helpers.**get_modules_in_progress_dataframe**(*grades: Grades*) → tuple

ugc.utils.commands_helpers.**get_template**() → dict

Return the default grades template used for the initial configuration as a dict.

ugc.utils.commands_helpers.**get_template_location**() → Path

ugc.utils.commands_helpers.**pprint_dataframe_done**(*dataframe: DataFrame*, *title: str*) → None

ugc.utils.commands_helpers.**pprint_dataframe_in_progress**(*dataframe: DataFrame*, *title: str*) → None

ugc.utils.commands_helpers.**print_modules_in_progress**(*pretty_printer*, *grades*)

ugc.utils.commands_helpers.**print_unweighted_average_in_progress**(*uavg*, *only_in_progress=False*) → None

ugc.utils.commands_helpers.**print_weighted_average_in_progress**(*wavg*, *only_in_progress=False*) → None

ugc.utils.commands_helpers.**there_are_no_modules_in_progress**(*grades*) → bool

ugc.utils.grades_helpers module

ugc.utils.grades_helpers.**get_classification**(*average*) → str

Return a string containing the classification of the student according to the Programme Specification.

ugc.utils.grades_helpers.**get_ects_equivalent_score**(*score: float*) → str

Return the grade in the ECTS equivalent form. Range from A to E/F.

ugc.utils.grades_helpers.**get_grades_list_as_list_of_dicts**(*grades: list*) → list

ugc.utils.grades_helpers.**get_module_score**(*module*) → float

ugc.utils.grades_helpers.**get_score_of_module_in_progress**(*module: dict*) → float

ugc.utils.grades_helpers.**get_total_score_modules_finished**(*modules: list*) → float

ugc.utils.grades_helpers.**get_total_weight_modules_finished**(*modules: list*) → float

ugc.utils.grades_helpers.**get_total_weight_modules_in_progress**(*modules: list*) → float

ugc.utils.grades_helpers.**get_uk_gpa**(*average*) → float

Return the GPA as calculated in the UK.

ugc.utils.grades_helpers.**get_unweighted_total_score_modules_in_progress**(*modules: list*) → float

ugc.utils.grades_helpers.**get_us_gpa**(*average*) → float

Return the GPA as calculated in the US.

ugc.utils.grades_helpers.**get_us_letter_equivalent_score**(*score: float*) → str

Get the letter equivalent in the US grading system for a given score.

ugc.utils.grades_helpers.**get_weight_of**(*level: int*) → int

Return the weight of a given *level*. The ratio is 1:3:5 for modules of L4:L5:L6 respectively.

ugc.utils.grades_helpers.**get_weighted_total_score_modules_in_progress**(*modules: list*) → float

ugc.utils.grades_helpers.**load_short_module_names**()

ugc.utils.grades_helpers.**score_is_valid**(*module_score: float*) → bool

Check whether a given score is a valid numeric value. Return a Boolean value.

ugc.utils.mathtools module

Math helper functions.

ugc.utils.mathtools.**round_half_up**(*num: float*, *decimals: int = 0*)

Round a float up and return it.

Assumes 10 decimals is enough precision. Also assumes we won't be rounding anything beyond 1,000,000 as that's far outside the range of expected values and would lead to overflow errors in this implementation.

3.2 Submodules

3.3 ugc.__main__ module

Allows calling *python -m ugc* from the root of the project.

3.4 ugc.cli module

Describes the commands available from the terminal when running this tool.

ugc.cli.**print_error**(*context*)

ugc.cli.**print_version**(*context*, *param*, *value*)

Print the program version and exit.

ugc.cli.**run_if_config_exists**(*f*)

3.5 ugc.commands module

List the commands available from the CLI: one per function.

ugc.commands.**check_score_accuracy**(*grades*) → dict

ugc.commands.**generate_sample**(*config*) → dict

Generate a sample grades JSON config file.

ugc.commands.**generate_sample_overwrite**(*config*) → dict

Generate a sample grades JSON config file: overwrite if it exists.

ugc.commands.**plot_modules**(*grades: Grades*, *api=False*, *options: dict = {}*) → dict

Plot modules over time with additional information and save the generated plot to *path*. It might be a good idea to refactor this gigantic function some day.

Parameters

grades (Grades) – ugc grades object.

ugc.commands.**summarize_all**(*grades: Grades*, *symbol: str = '='*, *repeat: int = 80*) → dict

Print a summary of modules done and in progress.

ugc.commands.**summarize_done**(*grades*) → dict

Print a summary of the progress made so far for modules that are done and dusted.

ugc.commands.**summarize_progress**(*grades*) → dict

Print a summary of only the modules that are currently in progress.

ugc.commands.**summarize_progress_avg_progress_only**(*grades*) → dict

3.6 ugc.config module

Manage the configuration file.

class ugc.config.**Config**(*json_str=None*, *config_path=None*)

Bases: object

Loads the configuration where grades are stored.

If *json_str* is passed, load from a JSON string. Instead, if *config_path* is passed, load from a path. Else, try loading from a default configuration file.

static _check_total_weight_sums_up_100_for_module(*module*, *module_name*) → bool

all_modules_are_found_with_valid_names() → bool

all_modules_are_set_to_correct_level()

all_modules_have_valid_float_scores_and_weights() → bool

check_config_is_not_empty() → bool

check_score_accuracy_raises_error_on_RPLed_module_with_scores() → bool

check_total_weight_sums_up_100_in_all_modules() → bool

config_is_a_dict() → bool

load() → dict

Load grades from JSON (string or file).

verify() → None

Check that the config file contains valid data. One of the functions will throw an error if the config is not valid.

exception ugc.config.**ConfigValidationError**(*custom_msg*)

Bases: Exception

Raised when there is an error in the config file.

3.7 ugc.grades module

Command-line application to get information about progress made in a BSc Computer Science at the University of London (calculations are specific to this particular degree).

class ugc.grades.**Grades**(*json_str=None*, *config_path=None*, *verified=True*, *error=None*)

Bases: object

_get_unweighted_data_of_modules_in_progress() → tuple

_get_weighted_data_of_modules_in_progress() → tuple

get_list_of_finished_modules() → list

Return a list of dicts containing information about all the modules that have a valid score (either -1 or 0 <= x <= 100).

get_list_of_modules_in_progress() → list

Return a list of dict containing all the non-empty values of the modules in progress.

get_module_scores_of_finished_modules() → list

Return a list of floats with the score obtained in each module.

get_module_scores_of_finished_modules_for_system(*system: str = 'US'*) → dict

Return a dictionary containing the converted ECTS score for each module.

get_num_of_finished_modules() → int

Return the number of modules completed with a score greater than or equal to zero as an integer.

static get_percentage_degree_done(*num_credits: int*) → float

From the total number of credits, return the percentage done out of 360 credits.

get_scores_of_modules_in_progress() → list

Return a list of floats with the score obtained in each module in progress.

get_scores_of_modules_in_progress_for_system(*system: str = 'US'*) → dict

Return a dictionary containing the converted ECTS score for each module in progress.

property total_credits: int

Get the total number of credits gotten so far as an integer.

property unweighted_average: float

Return the unweighted average across all completed modules.

property unweighted_average_in_progress_only: float

Return the unweighted average across modules in progress only.

property unweighted_average_including_in_progress: float

Return the unweighted average across all completed modules and those in progress.

property weighted_average: float

property weighted_average_in_progress: float

property weighted_average_in_progress_only: float

CHAPTER

FOUR

GRADES CALCULATOR

This tool is all about getting information and generating insights from the progress made in a BSc Computer Science at the University of London (calculations are specific to this particular degree).

Its source code is available on GitHub.

CHAPTER

FIVE

REQUIREMENTS

Python 3.8 and above. This is it!

CHAPTER

SIX

INSTALL AND UNINSTALL

The most straightforward way to use this tool would be to install it from PyPI by typing the following in a terminal (use of virtual environment recommended!):

```
$ pip install uol-grades-calculator
```

Reversing the process is a matter of typing this:

```
$ pip uninstall uol-grades-calculator
```

CHAPTER

SEVEN

TO RUN THE UTILITY

```
$ ugc
```

By passing no arguments, this will print the default help message.

CHAPTER

EIGHT

GENERATE A SAMPLE CONFIG FILE TO GET STARTED

To generate a sample configuration file, run the following command:

```
$ ugc generate-sample
```

The configuration file will be created in your home directory as a hidden file (i.e. `~/.ugc-grades.json`).

8.1 Specifying a different path for the config file

If you want to create it somewhere else:

```
$ ugc --config /path/to/config/file.json generate-sample
```

Note that you will have to indicate where the config is each time you use this tool in this case (you can always create an alias to avoid the trouble of typing it every time). For example:

```
$ ugc --config /path/to/config/file.json summarize
```

8.2 How to fill the config file (`.ugc-grades.json` by default)

Each module described in the config file should contain information adhering to the following indications:

Table 1: Configuration options

Key	Value	Example(s)	Optional*
`completion_date`	Date as a **string**: *YYYY-MM*	`2020-01`	Yes
`final_score`	**Float**: *range* 0.00–100.00	`50`, `50.5`, `90.56`	Yes
`final_weight`	**Integer** expressing a percentage: *range* 0–100	`0`, `40`, `80`, `100`	Yes
`midterm_score`	**Float**: *range* 0.00–100.00	`50`, `50.5`, `90.56`	Yes
`midterm_weight`	**Integer** expressing a percentage: *range* 0–100	`0`, `40`, `80`, `100`	Yes
`module_score`	**Float**: *range* 0.00–100.00	`50`, `50.5`, `90.56`	**No**
`level`	**Integer**: choose *strictly* from `4`, `5` or `6`	`4`, `5`, `6`	**No**

* If a value is `null` (or the key/value pair is absent in a given module), this will affect how the module is taken into account (average across all modules, summary of modules taken, etc.).

Here is a complete example for one module:

```
"Algorithms and Data Structures I": {
    "completion_date": "2020-03",
    "final_score": 92,
    "final_weight": 50,
    "midterm_score": 98,
    "midterm_weight": 50,
    "module_score": 95,
    "level": 4
}
```

8.2.1 Module taken

This means we define a module score between *0* and *100*, both being inclusive values, for a module for which an official grade was confirmed by the university.

```
"Algorithms and Data Structures I": {
    "module_score": 80.5
}
```

8.2.2 Module recognized (RPL)

In this case, we define a score of *-1* to indicate that this module is done but we didn't get a score for it. This way, we can keep track of the fact that the module is *"done"* but exclude it from calculations when getting an average, for instance.

```
"How Computers Work": {
    "module_score": -1
}
```

CHAPTER

NINE

HOW TO USE THIS TOOL

Please refer to the page *Available commands* to see what `ugc` can do for you.

PYTHON MODULE INDEX

u

INDEX

Symbols

A

C

D

G

L

M

P

R

S

T

U

V

W