ModPlan v1.4 - User Guide

By: `Team AY1920S1-CS2113T-F10-1` Since: `Aug 2019` License: `MIT`

1. Introduction

ModPlan

Your one stop solution to module planning through out your university tenure!

ModPlan is a module planning system that expands on NUSMODS to help NUS Computer Engineering students to plan their modules for all four years of their bachelor's degree.

ModPlan also helps NUS Computer Engineering students to customise the modules they want to take and keep track of the current and future modules to be taken.

As such, NUS Computer Engineering students can graduate in time with the planning provided by the ModPlan!

Developed and maintained by the AY1920S1-CS2113T-F10-1 team.

2. Quick Start

1. Ensure you have Java 11 or above installed in your Computer.
2. Download the latest `[CS2113T-F10-1][ModPlan].jar` here.
3. Copy the file to the folder you want to use as the home folder for your ModPlan application.
4. Open any terminal application, for example `Command Prompt` on Windows or `Terminal` on Mac in the folder directory containing ModPlan.
5. Run the following `java -jar ModPlan-v1.4.0.jar` in the terminal application.
6. A welcome message should be prompted in the terminal.
7. Type the commands in the terminal and press `Enter` to execute it.
8. Some example commands you can try:
    - **`show module`** : shows all modules
    - **`add`**`module CG2028` : searches for a module with module code of CG2028 and adds the module into the module list
    - **`remove`**`module 3` : deletes the 3rd module shown in the module list, if there are at least 3 modules in the list
    - **`bye`** : exits the app
9. Refer to Section 3, "Features" for details of each command.

3. Features

Command Format

- Words in `UPPER_CASE` are the parameters to be supplied by you e.g. in `add module MOD_CODE`, `MOD_CODE` is a parameter which can be used as `add module CG1112` or `add module GES1012`.
- Arguments for time `--begin` and `--end` are supposed to be given in 24 hour format. Due to the use of `Natty` library for relative date parsing, other formats would work but may result in anomalous behavior. The list of known expected behavior can be found under Section 6, "Expected Anomalous Behavior".

3.1. Viewing help `--help`

If you are unsure about any of the commands or what they do, use the `--help` command to view a list of the commands you can input in ModPlan, as well as a brief description of what the command does.

TIP	If you are still unsure on how to use the commands, keep reading the User Guide for specifics on how to use ModPlan!

Format: `--help`

```
--help
usage: ModPlanner [-h]
                  {add,cap,reminder,scheduleCca,grade,show,clear,update,sort,remove,bye}
                  ...

ModPlanner Argument Parser

positional arguments:
  {add,cap,reminder,scheduleCca,grade,show,clear,update,sort,remove,bye}
    add                 Add a module or cca
    cap                 Calculate your CAP from your input or list
    reminder            Setting reminders
    scheduleCca         Add schedule to a CCA
    grade               Enter your grades and let me calculate your GPA
                        for you!
    show                Show infos about your timetable
    clear               Clear your data as specified
    update              Updates local module data file
    sort                Sort your modules and/or ccas in the order you
                        desire
    remove              Remove a module or cca
    bye                 Exit ModPlanner

named arguments:
  -h, --help            show this help message and exit
```

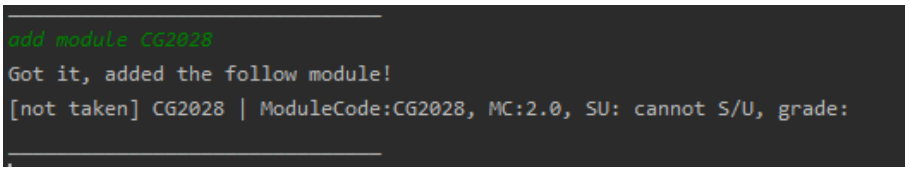
3.2. Adding a module or CCA: `add`

Adds the specified module or CCA to the module list or CCA list respectively.

3.2.1. Searching and adding a module: `add module`

Searches for a module and adds it to the module list, if the module is present in the list of NUS modules.
Format: `add module MOD_CODE`

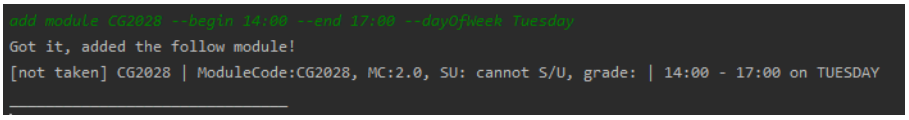
- The `MOD_CODE` must be in the format of the NUS module code name.
- Example: `add module CG2028`



3.2.2. Searching and adding a module with time period of the week: `add module`

Searches for a module and adds it to the module list, if the module is present in the list of NUS modules.
Format: `add module MOD_CODE --begin BEGIN_TIME --end END_TIME --dayOfWeek DAY_OF_WEEK`

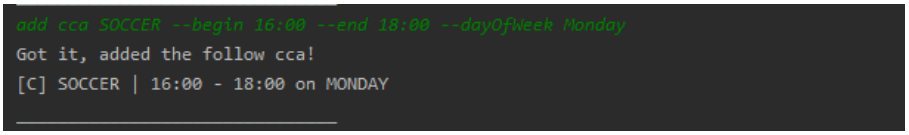
- The `MOD_CODE` must be in the format of the NUS module code name.
- The `BEGIN_TIME` and `END_TIME` must be in the format of HHmm.
- The `DAY_OF_WEEK` must be in the format of the day names such as, "Monday", "Tuesday", etc.
- Example: `add module CG2028 --begin 14:00 --end 17:00 --dayOfWeek Tuesday`



3.2.3. Adding CCA : `add cca`

Adds a CCA task into the CCA list.
Format: `add cca CCA_NAME --begin BEGIN_TIME --end END_TIME --dayOfWeek DAY_OF_WEEK`

- The `BEGIN_TIME` and `END_TIME` must be in the format of HHmm.
- The `END_TIME` can exceed the limit of the current day, and will spill over into the next day.
- The `DAY_OF_WEEK` must be in the format of the day names such as, "Monday", "Tuesday", etc.
- Example: `add cca SOCCER --begin 16:00 --end 18:00 --dayOfWeek Monday`



3.3. Listing modules or cca : `show`

Shows a list of specificed modules or ccas added in the module or cca list respectively.

3.3.1. Listing all modules : `show module`

Shows a list of all modules added in the module list.
Format: `show module`

```
show module
All modules in the list!
1. [not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
2. [not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:
3. [not taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:
4. [taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
5. [not taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:
6. [not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
7. [not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
8. [not taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:
9. [not taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:
10. [not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
11. [not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
12. [not taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:
_____
```

- Shows the module code, the number of MCs of the module and if the module can be S/U'ed.

3.3.2. Giving a report on core modules: `show core`

Prints out a report on all the core modules taken in the semester, together with the number of core modules left to take for graduation .
Format: `show core`

```
Here is your list of core modules being added:
1. [not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
2. [not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:
3. [not taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:
4. [taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
5. [not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:

Number of core modules required to take for graduation:
17
_____
```

3.3.3. Giving a report on General Education modules: `show ge`

Prints out a report on all the General Education(GE) modules taken in the semester, together with the number of GE modules left to take for graduation.
Format: `show ge`

```
show ge
Here is your list of general education modules being added:
1. [not taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:
2. [not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
3. [not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:

Number of general education modules required to take for graduation:
2
_____
```

- If more than one type of GE module is being added, the programme will inform you and prompt you to add only one type of GE module.

```
show ge
Here is your list of general education modules being added:
1. [not taken] GES1012 | ModuleCode:GES1012, MC:4.0, SU: can S/U, grade:
2. [not taken] GET1001 | ModuleCode:GET1001, MC:4.0, SU: can S/U, grade:
3. [not taken] GET1002 | ModuleCode:GET1002, MC:4.0, SU: can S/U, grade:

There are more than one type of GE modules added.
Please add only one type of GE module each.

Number of general education modules required to take for graduation:
3
```

3.3.4. Giving a report on Unrestricted Electives modules: `show ue`

Prints out a report on all the Unrestricted Electives(UE) modules taken in the semester, together with the number of UE modules left to take for graduation.
Format: `show ue`

```
show ue
Here is your list of unrestricted elective modules being added:
1. [not taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:
2. [not taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:
3. [not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
4. [not taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:

Number of unrestricted elective modules required to take for graduation:
4
```

3.3.5. Listing all CCAs: `show cca`

Shows a list of all CCAs added in the CCA list.
Format: `show cca`

```
show cca
All ccas in the list!
1. [C] SOCCER | 16:00 - 18:00 on MONDAY
2. [C] BASKETBALL | 14:00 - 20:00 on FRIDAY
3. [C] NUSSU | 10:00 - 12:00 on TUESDAY
4. [C] Ultimate Frisbee | 22:00 - 00:00 on THURSDAY
```

3.4. Removing a module or CCA : `remove`

Removes the specified module or CCA.

3.4.1. Removing a module : `remove module`

Removes the specified module from the module list.
Format: `remove module INDEX`

```
show module
All modules in the list!
1. [not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
2. [not taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:
____________________________
remove module 2
Got it, module will be deleted
[not taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:
____________________________
```

- Removes the module at the specified `INDEX`.
- The index refers to the index number shown in the displayed module list.
- The index **must be a positive integer** 1, 2, 3, …

Examples:

- `show module`
  `remove module 2`
  Removes the 2nd module in the module list.

3.4.2. Removing a CCA: `remove cca`

Removes a CCA which is added.
Format: `remove cca INDEX`

- Removes the CCA at the specified `INDEX`.
- The index refers to the index number shown in the displayed CCA list.
- The index **must be a positive integer** 1, 2, 3, …

Examples:

- `show cca`
  `remove cca 2`
  Removes the 2nd CCA in the CCA list.

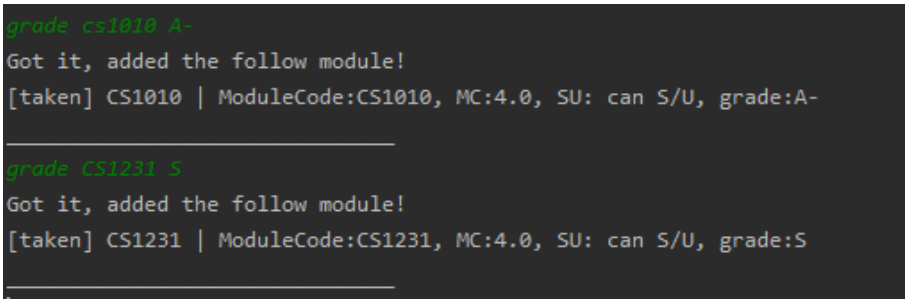
```
show cca
All ccas in the list!
1. [C] SOCCER | 16:00 - 18:00 on MONDAY
2. [C] BASKETBALL | 14:00 - 20:00 on FRIDAY
____________________________________
remove cca 2
Got it, cca will be deleted
[C] BASKETBALL | 14:00 - 20:00 on FRIDAY
____________________________________
```

3.5. Grading your modules: `grade`

Allows you to input your letter grade received for the modules you have taken.
Format: `grade MOD_CODE LETTER_GRADE`

- Type `grade MOD_CODE LETTER_GRADE` into the command line, replacing `MOD_CODE` with an actual module code, and `LETTER_GRADE` with the grade you received for that module.
- ModPlan will either update the grade of the module if it is in your list, or add the module with the letter grade included if it is not in your list.
- ModPlan will also check if the module is S/U-able, and will allow the user to input S and U grades accordingly.
  - If the module is not S/U-able, ModPlan will inform the user if they try to input a S or U grade.

Example:
`grade CS1010 A-`
`grade CS1231 S`



3.6. Calculating the CAP : `cap`

Calculates your overall CAP or predicted CAP in different ways.

3.6.1. Calculating CAP from user input.

Calculates your CAP according to your custom input of modules and grades.
Format: `cap overall`

- Typing `cap overall` into the command line shows a CAP calculation message.
- Type the module taken, along with it's letter grade.
  Keep typing all the module names in the module list and their respective grades with the format shown below.
- Format: `MOD_CODE GRADE_LETTER`
- Type `done` when you are ready to calculate the CAP.
- ModPlan then shows your current cumulative or predicated CAP.

Example:
`cap overall`
`CG2028 A`
`CG2027 B-`
`done`

```
cap overall
Start typing the module you have taken, along with it's letter grade
Type 'done' when you are ready to calculate your CAP
CG2028 A
CG2027 B-
done

_____________________________________
Here is your current cumulative/predicted CAP
4.00
_____________________________________
```

3.6.2. Calculating CAP from the module list.

Calculates your CAP from the `taken` modules in your list. Format: `cap list`

- Type `cap list` into the command line.
- ModPlan will show you your list of modules and grades to calculate CAP from.
- ModPlan will then calculate your CAP based on the completed modules in your module list.
    - Note that S/U'ed modules or modules without a grade will not be used in the calculation.

Example:
`cap list`

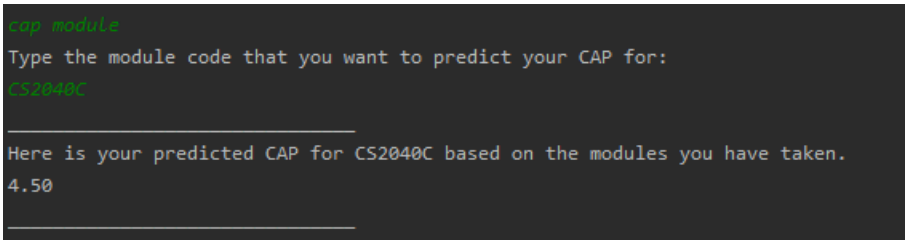
```
cap list
Here is your list of modules to calculate CAP from
1 [not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
2 [taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+
3 [taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
4 [taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
5 [taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:CS
6 [not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
7 [not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
8 [taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
9 [taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
10 [not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
11 [not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
12 [taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
13 [taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
14 [taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
_____________________________
Here is your current cumulative/predicted CAP
4.16
_____________________________
```

3.6.3. Calculating predicted CAP of a module from it's prerequisites.

Calculates the predicted CAP of a module based on the prerequisites of the inputted module.
Format: `cap module`

- Type `cap module` into the command line.
- ModPlan will then prompt you for the module to calculate CAP for.
- Type the module code of the module you wish to predict your CAP for.
- ModPlan will automatically sort the prerequisites of that module and check for your grades in them.
  - Note that these prerequisites have to be added and graded in your module list.
  - If any prerequisites are not completed, ModPlan will print a list of the prerequisites you have yet to complete/give a grade for.
  - If you encounter any issues with this command, please refer to Section 6, "Expected Anomalous Behavior".

Example:
`cap module`
`CS2040C`



3.7. Sorting and printing the modules and/or ccas : `sort`

Sorts out modules and/or ccas accordingly. For all the sorting methods listed below, enter the optional flag `--r` to sort in the reverse order.
Example:
`sort cca --r`
`sort module code --r`

3.7.1. Sorting and printing the CCAs : `sort cca`

Sorts the cca list according to alphabetical order and prints the cca list.
Format: `sort cca`

```
sort cca
Here are your sorted cca:

_____________________________________
[C] BASKETBALL | 14:00 - 20:00 on FRIDAY
[C] NUSSU | 10:00 - 12:00 on TUESDAY
[C] SOCCER | 16:00 - 18:00 on MONDAY
[C] Ultimate Frisbee | 22:00 - 00:00 on THURSDAY
_____________________________________
```

3.7.2. Sorting and printing the CCAs and modules of a certain day of the week : `sort time`

Sorts the cca and modules together list according to alphabetical order and prints the cca list.
Format: `sort time DAY_OF_WEEK`, replace DAY_OF_WEEK by any of `monday` `tuesday` `wednesday` `thursday` `friday` `saturday` and `sunday`.

- Note that if two ccas have the same name, only the first one will be considered for sorting.

Example:
`sort time tuesday`

```
sort time tuesday
Here are your sorted time:

_____________________________________
NUSSU[10:00 - 12:00 on TUESDAY]
CG2028[14:00 - 17:00 on TUESDAY]

_____________________________________
```

3.7.3. Sorting and printing the modules : `sort module code`

Sorts the module list according to alphabetical order and prints the module list.
Format: `sort module code`

```
sort module code
Here are your sorted module:
_____________________
[taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
[not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+ | 09:00 - 12:00 on MONDAY
[taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
[taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
[taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
[taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
[taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
[taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
[not taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
[not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
[not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
[taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:CS
[not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
[not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
_____________________
```

3.7.4. Sorting and printing the modules : `sort module grade`

Sorts the module by the grade entered and prints the module list.
Format: `sort module grade`

```
sort module grade
Here are your sorted module:
_____________________________
[taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
[taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
[taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
[not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+ | 09:00 - 12:00 on MONDAY
[taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
[taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
[taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
[taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
[taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:S
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
[not taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
[not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
[not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
[not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
[not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
_____________________________
```

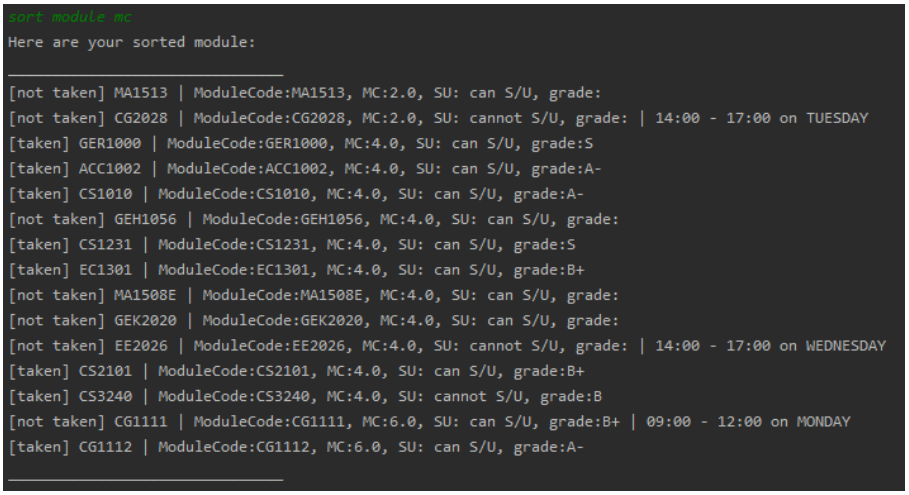
3.7.5. Sorting and printing the modules : `sort module level`

Sorts the module list by the numerical order and prints the module list.
Format: `sort module level`

```
sort module level
Here are your sorted module:
_____________________________
[taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:S
[taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
[taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
[not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
[not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+ | 09:00 - 12:00 on MONDAY
[taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
[taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
[taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
[not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
[not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
[not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
[not taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
[taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
[taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
_____________________________
```

3.7.6. Sorting and printing the modules : `sort module mc`

Sorts the module list according to the number of mcs and prints the module list.
Format: `sort module mc`



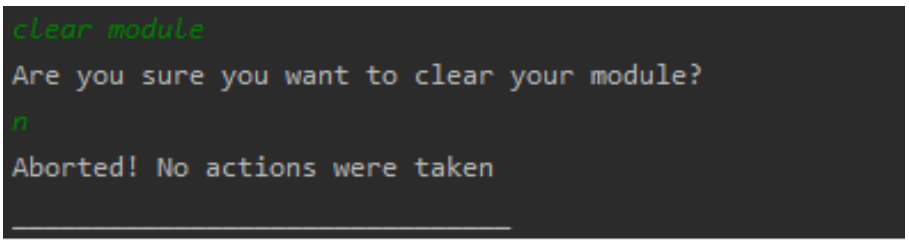
3.8. Clearing the data : `clear`

Clears the specified data. After inputting the parameter that you want to clear, ModPlan will prompt you again to **reconfirm** that you want to clear your data.

TIP	You should type either `y` or `n` to confirm or deny ModPlan's request to clear your data. ModPlan also allows other common forms of `yes` and `no`.

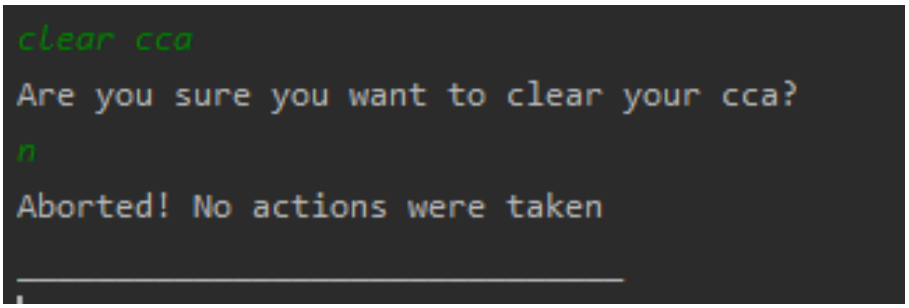
3.8.1. Clearing the modules data : `clear module`

Clears and empties the list of modules being added.
Format: `clear module`



3.8.2. Clearing the CCA data : `clear cca`

Clears and empties the list of CCAs being added.
Format: `clear cca`



3.8.3. Clearing the password : `clear password`

Remove the current password.
ModPlan will ask for your current password if you have previously setup one. If the current password cannot be provided, clearing password will fail.
Note that the password protection feature is currently disabled to facilitate feature testing and will only be reactivated in ModPlan 2.0
Format: `clear password`

```
clear password
Are you sure you want to clear your password?
y
No active password found!
```

3.8.4. Clearing all the data : `clear data`

Remove all current user data.
Format: `clear data`

```
clear data
Are you sure you want to clear your data?
n
Aborted! No actions were taken
```

3.9. Adding another schedule to a CCA : `scheduleCca`

Adds another schedule to a CCA which is already added, as the CCA may have multiple slots.
Format: `scheduleCca INDEX --begin BEGIN_TIME --end END_TIME --dayOfWeek DAY_OF_WEEK`

```
scheduleCca 1 --begin 13:00 --end 15:00 --dayOfWeek Tuesday
Got it, added the follow cca!
[C] BASKETBALL | 14:00 - 20:00 on FRIDAY, 13:00 - 15:00 on TUESDAY
```

- The `BEGIN_TIME` and `END_TIME` must be in the format of HH:mm.
- The `DAY_OF_WEEK` must be in the format of the day names such as, "Monday", "Tuesday", etc.
- Example: `scheduleCca 1 --begin 13:00 --end 15:00 --dayOfWeek Tuesday`

3.10. Updating module data : `update`

Allows the user to directly update the module data.
Format: `update module`

```
update module
Your module data files has been updated!
_____________________________________
```

- This command requires Internet connection to download data from NUSMOD API.
- If either their server is down or there is no stable Internet connection, a `ModBadRequestStatus` is thrown.

3.11. Enabling and disabling of reminders to update the module data: `reminder`

Allows you the start and stop the reminder to update the module data for a specified period of time. Format: `reminder`

3.11.1. Shows the list of the different specified time interval of reminder: `reminder list`

Gives four options to determine how often you want to set the reminder. Format: `reminder list`

```
reminder list
_____________________________________
Would you like to set your reminder to every:
1) for 10 seconds
2) for 30 seconds
3) for 1 minutes
4) for 2 minutes
*helpline* : for 1), enter 'reminder one'
_____________________________________
```

3.11.2. Choosing the desired time interval of reminder: `reminder NUMBER`

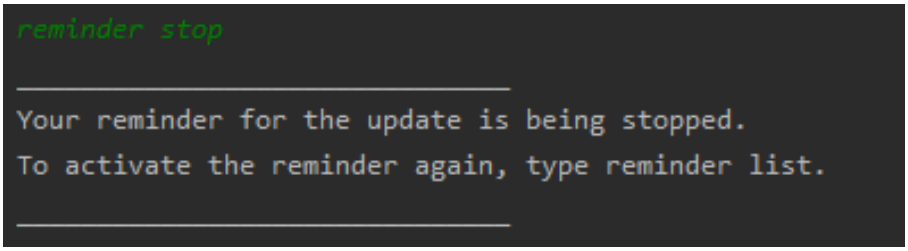
Allows you to choose the desired period of time for the reminder to appear, which ranges from 10 seconds to 2 minutes. Format: `reminder NUMBER`

```
reminder two
_____________________________
Please remember to update your module information!
To do so, you can input the update command in the following format:
update module
_____________________________
```

- There are currently four supported time intervals for the reminder.
- Example: `reminder one`

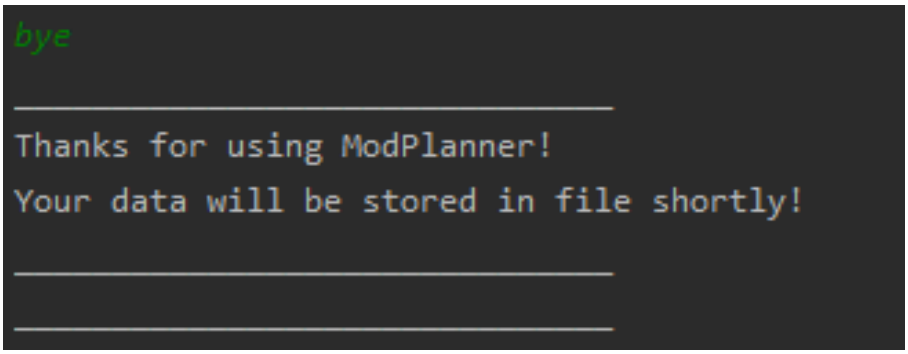
3.11.3. Stopping the reminder: `reminder stop`

Allows you to stop the reminder and the reminder message will stop appearing according to the selected time interval.



3.12. Exiting the program : `bye`

Exits the program.
Format: `bye`



- Typing `bye` into the command line shows a goodbye message, saves the module list, and closes the program.

3.13. Saving the data

Task list data are saved in the hard disk automatically after any command that changes the data. There is no need to save manually.

4. Errors

Error Handling When you input commands or parameters in a way in which the program does not understand, errors will be thrown, informing the user what was causing the error.

TIP

If you follow what the errors tell you to fix in your command, you can get the program to work as intended!

Or even better, simply type or add `-h` to the end of the command you intend to input and ModPlanner will output a detailed guildline for you!

4.1. Parser Errors

If you encountered an error message starting with `ModPlanner: error:`, then this section is for you!

There are 4 common types of Parser Errors:

4.1.1. `ModPlanner: error: invalid choice …`

This error appears when you input an invalid command or argument to ModPlanner. However, the error message will display the valid options for you. In some cases, ModPlanner may even suggest a possible command that it thinks you intended to write!

Example of input that can cause this error: `clean`
Example error message:

```
clean
usage: ModPlanner [-h]
                  {add,cap,reminder,scheduleCca,grade,show,clear,update,sort,remove,bye}
                  ...
ModPlanner: error: invalid choice: 'clean' (choose from 'add', 'cap',
'reminder', 'scheduleCca', 'grade', 'show', 'clear', 'update', 'sort',
'remove', 'bye')

Did you mean:
    clear
```

> ***Solving the error:***
> Select one from the provided legal options. ModPlanner even noticed that you probably meant `clear` which is a valid command, and suggested it.

4.1.2. `ModPlanner: error: too few arguments`

This error appears when you do not supply enough arguments for a specific command.

Example of input that can cause this error: `add module`
Example error message:

```
add module
usage: ModPlanner add module [-h] moduleCode
ModPlanner: error: too few arguments
```

> ***Solving the error:***
> Look for the missing arguments as provided in the error message. In this case, it is `moduleCode`. If you are unsure what to input for `moduleCode`, try `add module -h`.

```
add module -h
usage: ModPlanner add module [-h] moduleCode

positional arguments:
  moduleCode            Codename of module to add

named arguments:
  -h, --help            show this help message and exit
```

4.1.3. `ModPlanner: error: unrecognized arguments: …`

This error appears when the name of a named argument is specified incorrectly.

Example of input that can cause this error: `add cca test cca --beginTime 15:00 --end 5pm --dayOfWeek MONDAY`
Example error message:

```
add cca test cca --beginTime 15:00 --endTime 5pm --dayOfWeek MONDAY
usage: ModPlanner add cca [-h] --begin BEGIN [BEGIN ...]
                  --end END [END ...] --dayOfWeek DAYOFWEEK name [name ...]
ModPlanner: error: unrecognized arguments: '--beginTime'
```

> ***Solving the error:***
> Look for the correct argument name as provided in the error message! In this case, `--beginTime` should be changed to `--begin`.

4.1.4. `ModPlanner: error: argument index: could not convert …`

Certain arguments should be parsed in the correct format in order for the value to be evaluated correctly. If you encounter this error, chances are you tried to parse a non-integer value to an integer-type argument.

Example of input that can cause this error: `remove cca notANumber`
Example error message:

```
remove cca notANumber
usage: ModPlanner remove [-h] {module,cca} index
ModPlanner: error: argument index: could not convert 'notANumber' to
integer (32 bits)
```

> ***Solving the error:***
> Look for the correct type of the argument from the error message and change your argument to match the type. In this case, `index` should be an `int` but the ModPlanner could not convert the input value `notANumber` to an `int`. An example of a correct command is `remove cca 1` (provided your CCA list is not empty!).

4.2. `ModBadRequestStatus`

This error appears when there is poor or unstable Internet connection. The information from the nusMods V2 API is not fully fetched.

Example of error message: `Error: Bad Status Connection!`

> ***Solving the error:***
> Reconnect to a stronger and more stable wifi connection.

4.3. `ModFailedJsonException`

This error appears when the file from the nusMods V2 API is not correctly converted for Java usage.

Example of error message: `Error: Failed to parse data file!`

> ***Solving the error:***
> Reconnect to a stronger and more stable wifi connection.

4.4. `ModNotFoundException`

This error appears when you search for a module code that is not found in the nusMod list.

```
Error: Module not found :(
```

> ***Solving the error:***
>
> - Input another module code which exists in the nusMod list.
> - Input the correct module code into the command line.

4.5. `ModCommandException`

This error appears when you do not input a valid command name into the command line.

```
Error: Must be a valid command!
```

> ***Solving the error:***
>
> - Input a valid command name into the command line.
> - If unsure of the available command names, refer to the Section 8, “Command Summary”.

4.6. `ModCcaScheduleException`

This error appears when you input a CCA whose time period clashes with another CCA.

Example of error message: `Error: This CCA clashes with existing CCA!`

> ***Solving the error:***
> Input another CCA with a timing that does not clashes with the exisiting CCAs.

4.7. `ModBadGradeException`

This error appears when you input an invalid letter grade.

Example of input that can cause this error: `grade CS1010 O`
Example of error message:

```
Error: Please enter a valid letter grade!
```

> ***Solving the error:***
> Input one of the following grades: "A+, A, A-, B+, B, B-, C+, C, D+, D, F, S or U".

4.8. `ModBadSuException`

This errors appears when you input an S or U grade for a module that does not have an S/U option.

Example of input that can cause this error: `grade CG2028 S`
Example of error message:

```
Error: S/U option is not allowed for this module!
```

Solving the error:

- Use `show module` to check whether the module is S/U'able.
- Only modules with the "SU: true" indicates that the module is S/U'able and an S or U grade can be input.

5. Coming in ModPlan v2.0

5.1. Password management

5.1.1. Add or update an existing password: `passwd`

Add or update an existing password.
If a password has been previously set (updating password mode), a prompt will show up asking for old password before a new one can setup. In that case, kindly enter your current password as prompted to setup a new one.
Format:
`passwd myPassword`

5.2. GUI

ModPlan v2.0 will support GUI (Graphical User Interface). We have not decided whether the CLI (Command Line Interface) will be fully replaced by GUI or will be an additional feature. However, the new GUI will hopefully improve ModPlan's user-friendliness and user's productivity.

6. Expected Anomalous Behavior

- Writing shortform commands for some of the commands may work, due to the program identifying the shortform as a unique input for the command.
  - Example `sh module` will return the same output as `show module` as `sh module` is unique enough for the program to identify its intended command.
- Inputting an invalid time beyond `00:00` such as `25:00` will cause the time to automatically be set to `00:00`
- Some of the prerequisites/preclusions in NUSMODS may not be updated for the `cap module`, or is not checked against each other in the prerequisite tree.
  - If you have completed all the necessary prerequisites but it is still showing you have not completed some of them, please add those modules into your list with the same grade you obtained as its preclusion and try the command again.

7. FAQ

Q: How do I transfer my data to another Computer?
A: Install the app in the other computer and overwrite the empty `userProfile.json` file it creates with the file from your previous Data folder.

8. Command Summary

- **Help**:
  - `--help`
- **Add** :
  - `add module MOD_CODE`
    e.g. `add CG2028`
  - `add cca CCA_NAME --begin BEGIN_TIME --end END_TIME --dayOfWeek DAY_OF_WEEK`
    e.g. `add cca SOCCER --begin 16:00 --end 18:00 --dayOfWeek Monday`
- **Remove** :
  - `remove module INDEX`
    e.g. `remove module 3`
  - `remove cca INDEX`
    e.g. `remove cca 2`
- **Show** :
  - `show module`
  - `show core`
  - `show ge`
  - `show ue`
  - `show cca`
- **CAP** :
  - `cap overall`, `MOD_CODE GRADE_LETTER`, `done`
    e.g. `cap overall`
    `CG2027 B-`
    `CG2028 A`
    `done`
  - `cap list`
  - `cap module`, `MOD_CODE`
    eg. `cap module`
    `CS2040C`
- **Grade** :
  - `grade MOD_CODE LETTER_GRADE`
    e.g `grade CS1010 A-`

grade CS1231 S

- **Sort** :
  - sort cca
  - sort cca --r
  - sort time, DAY_OF_WEEK
    eg. sort time monday
  - sort module code
  - sort module grade
  - sort module level
  - sort module mc
- **Clear** :
  - clear modules
  - clear ccas
  - clear data
- **Schedule CCA** :
  - scheduleCca INDEX --begin BEGIN_TIME --end END_TIME --dayOfWeek DAY_OF_WEEK
    e.g scheduleCca 1 --begin 13:00 --end 15:00 --dayOfWeek Tuesday
- **Update** :
  - update module
- **Reminder** :
  - reminder list
  - reminder NUMBER
    eg. reminder one
  - reminder stop
- **Exit** :
  - bye