FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Info-F-106 : Computer science project

Project 1 – Tetraminos

Anthony Cnudde Gwenaël Joret Tom Lenaerts Robin Petit
Loan Sens Cédric Simar

Version of the 7th of december 2023

General presentation

A three-step project

The aim of the project is to create a Python 3 implementation of *Tetramino*. It's a one player game on a data grid. The goal is to put all the data pieces, tetraminos, on the grid in order to make them overlap.

Tétramino

The version of the game we want to suggest is inspired of the Two-player game *Gagne ta maman* [1] / *Gagne ton papa* [2] adapted for one player (NB: don't get distracted by the awkwardly put name of the game, it's a well-thought game):

Each colored piece is a tetramino, composed of several unit blocks. In the version we are offering you, one part involves solving a problem described on a given card:
it specifies the tetraminos to be placed, as well as the size of the grid. The goal is to place each tetramino on the grid without overlapping. The larger the grid and the number of pieces, the more challenging the game becomes.

1. https://www.gigamic.com/jeux-pedagogiques/189-gagne-ta-maman-3421271314615.html
2. https://www.gigamic.com/jeux-pedagogiques/27-gagne-ton-papa-3421271314318.html

The project is extensively described later in this document, but before delving into the core of the subject, here are some general instructions for the completion of this project. These instructions will also apply to the second project in the second semester.

General Instructions

- The entire project is to be implemented in Python 3.
- The INFO-F106 module consists of two separate programming projects, one for each quadrisemester.
- In addition to the projects to be submitted, each student must submit a final report in the second semester (Q2).
- The distribution of points is as follows :
  - Project 1 (Q1, Tetromino) : 40 points
  - Project 2 (Q2) : 40 points
  - Report (Q2) : 20 points
  - **Total : 100 points**
- Both projects and the report must be submitted to the UV.
- **There will be no second session for these projects!**

Please also note that the project will receive a zero score without exception if:

- the project cannot be executed correctly using the commands described in the statement.
- The function names (and your scripts) are different from those described in this statement or have different parameters;
- à through the use of specialized automatic tools, we detect clear plagiarism (between projects of multiple students or with elements found on the Internet). We emphasize this last point because experience shows that every year, a handful of students believe that a small copy-paste of a function, followed by a reorganization of the code and some variable renaming, will go unnoticed. This will result in a zero score for the entire project for all individuals involved, as well as possible additional sanctions. To avoid this situation, be especially careful not to share code snippets on forums, Facebook, Discord, etc.

File submission

File submission is done through the section open on the UV.

Automated testing

To assess your code, we will provide you with a test file. This file contains tests that will validate (or not) the functions you have implemented. These tests will help you determine if your code reacts as we expect or if you need to rework it. However, this implies that you must strictly adhere to the instructions given to you; otherwise, these tests will fail.

We will require your code to pass all provided tests; otherwise, we will not grade your project, and your final score will be 0.

Allowed Librairies

Except for those mentioned in the instructions, no libraries are permitted for this project.

Communication with the educational Team

Preliminary Note: There are 566 students enrolled in this course (figures as of November 13, 2023), making it simply impossible for us to respond individually to questions via email. This leads us to the following rule:

Golden Rule: Never send emails!

Question-and-answer sessions will be organized regularly, so you will have numerous opportunities to ask your questions. This way it benefits everyone ! And it saves us from answering the same question 25 times. To emphasize the importance of this rule, here are some examples of situations in which a student might want to contact an assistant or the course instructor. In each of these situations, sending an email (or individual Teams message) is prohibited:

— « I have a question about the project statement » → Ask →the question during the Q&A sessions organized by the assistants. ;

— « I don't understand the grade I received for my project; I would like to discuss it » → Attend the copy review session organized by the responsible assistants.;

— « I received a zero for my project due to non-compliance with instructions because of a typo in my file name; I find it too harsh » → We strictly apply the instructions announced in the statement, without any exceptions. Learning to follow instructions is one of the educational objectives of this year-long project.

Note that we will simply not respond to emails regarding the year-long project if you happen to send them to us. The only exception to this rule concerns administrative emails addressed to the instructors (sick leave certificates, reorientation, EBS, etc.).

Educational objectives

This interdisciplinary project aims to engage various skills.

— *Knowledge acquired in programming, language, algorithmic, or mathematical courses. The scope of the two projects will require a more rigorous and in-depth analysis than that demanded for the writing of a one-page project, as well as a meticulous application of the various concepts taught in the courses.*

— *Knowledge not covered in the courses. Students will be encouraged to study them on their own, guided by the advice provided by the team overseeing the course.*

— *Communication skills. Students will be required to write a scientific report in LATEX. This will provide students with the opportunity to familiarize themselves with this language, used for writing scientific documents. Correct spelling will be required.*

In summary, you will need to demonstrate that you can apply concepts learned in class, explore new subjects on your own, and finally, communicate the results of your work in a scientific manner.

Good Luck !

1 Project 1 : Tetromino

The objective of the project is to implement Tetramino, a single-player game in Python 3 that requires the player to align given pieces to fill a determined-size grid.

1.1 The Game

The game begins on a grid size $w \times h$. The player receives a set of pre-determined size pieces. These pieces can be rotated clockwise or counterclockwise, but cannot be flipped. The player must place these pieces on the grid in a way that fills every space. Specifically, the player chooses a piece and can freely move it even on an already placed piece or border. However, the player can only drop a piece in an opened area, without border or any other piece. Once the piece is placed, they can choose another piece (or the same), until they find a winning configuration.

For this implementation, the game will take place on a grid size $(3w + 2) * (3\overset{\times}{h} + 2)$ to allow easier movement of the pieces. The playing area to be filled will be the central part, and each piece will initially be positioned in one of the peripheral grids.

The *screenshots* below (Figure 1) depict the initial playing area and the configuration to achieve.

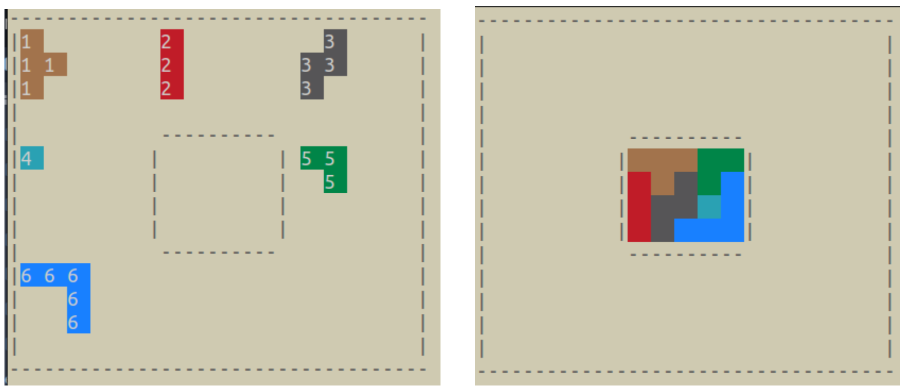


Figure 1 – On the left, the game board at the start of the game. On the right, the game board at the end of the game.

The program will be launched using the following command:

```
python3 tetramino.py carte
```

where `carte` is the name of the text file corresponding to one of the provided maps. Each map encodes a problem for the player to solve: the size of the board and the different pieces offered for the game.

To use command-line parameters in your code, you can use the `sys.argv` list from the `sys` module. For example, if we launch the program using the command

```
python3 tetramino.py carte_1.txt
```

then `sys.argv` will be the list of strings: `[ 'tetramino.py' , 'carte_1.txt' ]`. The first element of the list corresponds to the name of the Python script, and the second contains the name of the text file corresponding to the map.

Ensure that your file can be imported; therefore, place the call to the `main` function in a conditional test: `if__name == ' _main__' :`.

1.2 Program's structure

The game is parameterized by a "carte" file that you must read and process at the program's startup. This file contains:

— on the first line, the dimensions of the game grid.;
— for each subsequent line, the positions and colour of a piece to be placed on the grid.

1.2.1 Reading of the game's file

Each given file will look like this:

```
5, 4
(0, 0);(0, 1);(0, 2);(1, 1);;0;37;43
(0, 0);(0, 1);(0, 2);;0;37;41
(1, 0);(1, 1);(0, 1);(0, 2);;0;37;45
(0, 0);;0;37;46
(0, 0);(1, 0);(1, 1);;0;37;42
(0, 0);(1, 0);(2, 0);(2, 1);(2, 2);;0;37;44
```

— The first line corresponds to the dimensions of the game grid; the first number is the number of columns, and the second is the number of rows. Remember that the actual size of the grid in this version must be tripled and include the central borders (in this case, the game board would be 17 columns and 14 rows).
— The subsequent lines describe the pieces to be placed, one piece per line, described as follows: first, the coordinates of the unit blocks constituting the piece, and then the color of the piece. Specifically:
  — The coordinates correspond to the position in (x, y) of each unit block constituting the piece. For example, a square tetromino consisting of 4 blocks is described as follows:

    ```
    (0, 0);(0, 1);(1, 0);(1, 1)
    ```

  — The color code, separated from the block coordinates by two semicolons, provides a color when integrated into a specific character string in the terminal. This character string is as follows:

    ```
    '\x1b[' + code_couleur + 'm' + texte + '\x1b[0m'
    ```

    where `color code` is a string containing the color code, and `texte`is a string containing the text to be displayed. For example, printing the following character string

    ```
    '\x1b[0;37;44mTetramino\x1b[0m'
    ```

    will display :

Tetramino

Remarque Side note: to display a full coloured box without text, a space (i.e ' ') as a text.

It is up to you to transform the content of these files into interpretable data. Note that the coordinates correspond to those of **an orthonormal coordinate system with the y-axis reversed,** meaning that the position $(0, 0)$ corresponds to the upper-left corner of the matrix, and the position $(w$ 1$, h$ 1$)$ to the bottom-right corner, where w and h are the dimensions of the matrix.

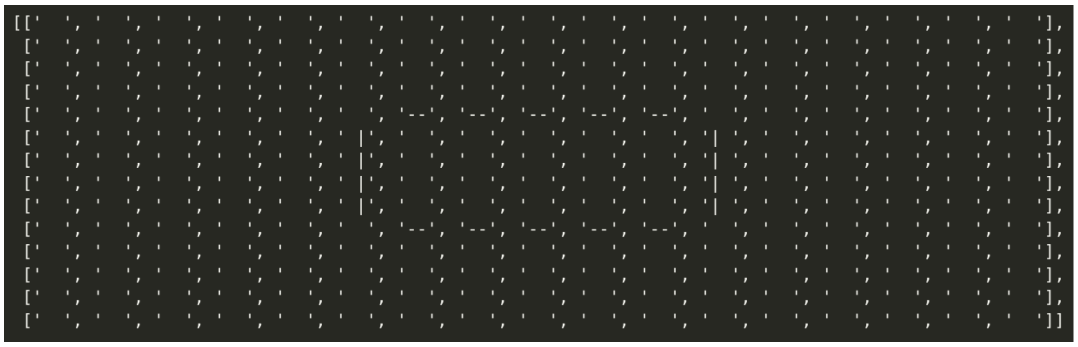
1.2.2 Game Board

As indicated, the game board must be of dimensions $(3w + 2) \times (3h + 2)$ to allow for easier selection and movement of pieces.

Empty areas on the board will be represented by two spaces (i.e ' ' *2), allowing for a more harmonious display of shapes.

- You must also consider placing the boundaries of the central playing area, which must be completely filled to win the game. To facilitate board management, these boundaries will also be part of the board encoding and represented as follows:
  - The character string' |' (i.e | preceded by a space) for the left vertical boundaries, the character string'| ' (i.e | followed by a space) for the right side;
  - The character string' --' for the horizontal boundaries

Thus, the encoding of the final board, including boundaries of the central area, will be of dimensions $(3w + 2) \times (3h + 2)$. An example is given in Figure 2.



```
[['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '--', '--', '--', '--', '--', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', ' |', '  ', '  ', '  ', '  ', '  ', '| ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', ' |', '  ', '  ', '  ', '  ', '  ', '| ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', ' |', '  ', '  ', '  ', '  ', '  ', '| ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', ' |', '  ', '  ', '  ', '  ', '  ', '| ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '--', '--', '--', '--', '--', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  '],
 ['  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ', '  ']]
```

FIGURE 2 – Game board 5×4 as it should be represented in your code

For the representation of your board during a game in the terminal, we ask you to follow the following instructions:

- Pieces are displayed in the color indicated in the imported file.
- During the selection of the piece **only** (not when the player has chosen their piece and is using it), the piece number is indicated in the left part of each unit block constituting the piece.

— When a piece is positioned in a non-valid area (above another piece or a border), it is displayed with the text 'XX' on the problematic block(s) to indicate that placing the piece is not possible due to this/these block(s). An illustration is provided in Figure 3.

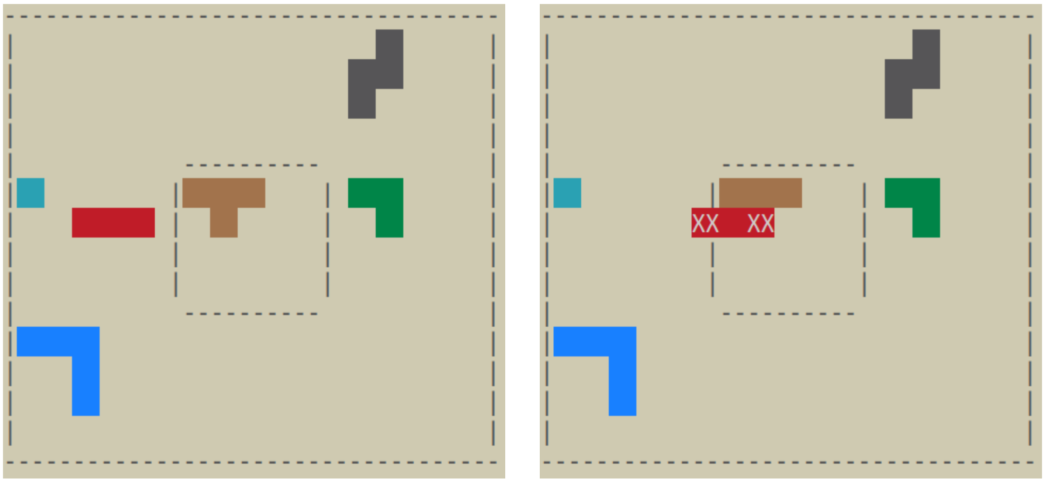


Figure 3 – F How an invalid placement should be indicated in the terminal. On the right, you can see that the blocks positioned over the border and the other piece are displayed with the text 'XX'.

1.2.3 Movement of tetraminos

A tetramino is a set of blocks aligned to compose a specific shape of a given color. As explained earlier, this information is provided in the given maps. To enable efficient and practical management of the movements and rotations of these pieces during the game, tetraminos will be encoded as a list with:

— at position 0: l the **initial coordinates** of the piece, in the form of a list of tuples. Each tuple corresponds to the position of one of the blocks of the piece. **These coordinates will only be modified in one case: when a rotation of the piece takes place**. They will not change when the piece is moved vertically or horizontally.
— at position 1 : **the color** of the tetramino, as indicated in the previous sections.
— at position 2 : u **an offset, which corresponds to the cumulative movements of the piece**. The offset is a tuple (x, y) tallying the total movements made on the piece from its initial coordinates. For example, a block with initial coordinates at $(0, 0)$ and moved 2 times to the right and 3 times down will have an offset of $(2, 3)$.

Any mention of tetraminos in your code from this point forward will refer to this structure.

1.2.4 Rotation of tetraminos

P To enable the rotation of tetraminos, you will need to use the following formula, given the x and y coordinates at time t denoted as x_t, y_t :
— Clockwise rotation :

  — $x_{t+1} = - y_t$
  — $y_{t+1} = x_t$
— Counterclockwise rotation:
  — $x_{t+1} = y_t$
  — $y_{t+1} = - x_t$

This simple formula allows you to calculate the rotation of the piece based on its initial coordinates. In other words, you will update the **initial coordinates** of the piece using this formula. (If you apply this formula to the actual position of the piece, i.e., initial coordinates incremented by the offset, your rotation will not work as expected).

1.2.5 Initial position of tetraminos

As mentioned earlier, tetraminos should, at the start of the game, be distributed on the 8 grids surrounding the central grid. The filling order to follow is indicated in the first image of Figure 4. This order corresponds to the order in which the pieces appear in the text file. The piece placed in its sub-matrix will be aligned with the upper-left corner of it, as shown in the second image of Figure 4 .

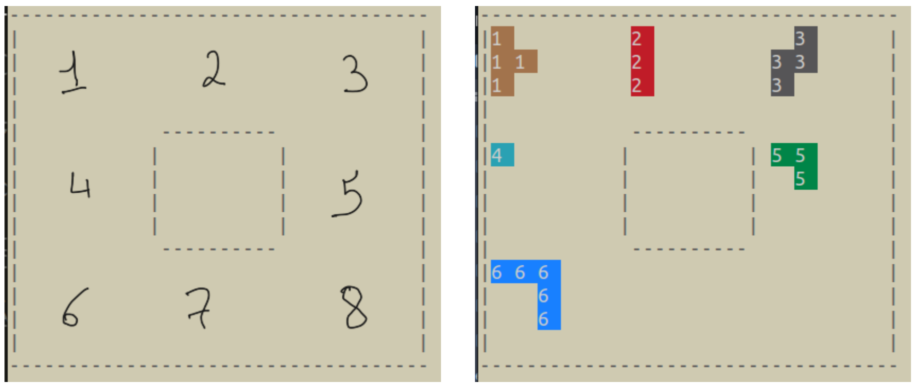


Figure 4 – Left image: Tetraminos' position at the start of the game. Right image: Tetraminos' position at the beginning of the game for the card' `carte_1.txt`'

1.2.6 Handling of the movements

Player input actions will be read through the `getkey` function from the `getkey.py` file provided to you on the UV. f This function will allow you to manage user inputs easily without requiring them to validate each movement with the enter key. Specifically, the `getkey` function waits for the user to press a key on the keyboard and then returns the corresponding character (compared to the `input`, l function, the advantage here is that you don't need to press enter).

To use the `getkey`, function, place the `getkey.py` file in the same directory as your `tetramino.py` file and add the following line at the beginning of the latter :

```
from getkey import getkey
```

The inputs that the user should be able to enter are :

— When choosing the piece to play: numbers 1 to 8 to choose the corresponding piece.
— `i` to move the piece up, `k` for down, `j` for left et `l` for right.
— `o` allows the piece to rotate clockwise; `u` counterclockwise
— `v` allows validating the position of the piece (only if the current position is valid). If the position is not valid, nothing happens, and the player must enter a new movement.

1.3 Clearing Previous Outputs

To clear the outputs from previous rounds and make the console clearer during the game, you can use the 'os' library. The command to use is different for Windows and linux/mac, so use the following snippet to ensure that the function is usable regardless of the operating system used :

```
if(os.name == 'posix'):
    os.system('clear')
else:
    os.system('cls')
```

1.4 Implementation

We ask you to implement the following functions. This instruction must be strictly followed; otherwise, the automatic tests will not pass, and your code will not be corrected. Of course, as long as all these functions are implemented, you can certainly add others that would allow you to write cleaner and more readable code; this is strongly encouraged.

Note : From here, to simplify the definition of functions, the term "tetramino" will be used to define a list containing the information of a given tetramino, namely (see section 1.2.3) :

`[initial coordinates: list(tuple(x, y)), color:string, offset: tuple(x,y)]`

Here are the instructions that must be implemented :

Fonction 1: `create_grid(w: int, h: int)`

Description:
Create a grid of size $(3w+2) \times (3h+2)$, including the boudaries of the central zone.

Return:
list(list) : Matrix of size $(3w + 2) \times (3h + 2)$ rfilled as indicated in the instructions of point 1.2.2. (Note: In the context of this project, a matrix in Python is encoded simply as a list of lists).

Fonction 2: `import_card(file_path: string)`

Description:
Pe Allows the import of a game file. This function opens the file, extracts its content, and processes it to generate the defined tetraminos as well as the dimensions of the board.

Return:

(tuple`(x, y)`, list(tetramino)) → Tuple containing the dimentions of the board (under the shape of a `tuple(x, y)`), and a list of tetraminos.

Fonction 3: `setup_tetraminos(tetraminos: list(tetramino), grid: list(list))`

Description:
Sets up the tetraminos imported from the file, each in its sub-matrix on the main matrix. This function is called only once at the beginning of the game to initialize the starting grid.

Return:
(list(list), list(tetramino)) → New grid with the tetraminos placed, and the list of tetraminos with their new position updated

---

Fonction 4: `place_tetraminos(tetraminos: list(tetramino), grid: list(list))`

Description:
Places the tetraminos on the given grid. This function is used to make a move at any point in the gamelace.

Return:
list(list) → New grid with the tetraminos placed

Fonction 5: `rotate_tetramino(tetramino: tetramino, clockwise: bool (default=True))`

Description:
Allows the rotation of the tetramino. To perform the rotation, only the first element of the tetramino, its initial coordinates, is modified. Be careful not to use the position incremented by the offset.

Return:
tetramino → Tetramino with the initial coordinates modified by the rotation.

Fonction 6: `check_move(tetramino: tetramino, grid: list(list))`

Description:
Checks if the current position of the tetramino is valid (it does not overlap with another piece or a border).

Return:
bool → `True` if the current position is valid, `False` otherwise.

Fonction 7: `check_win(grid: list(list))`

Description:
Checks if the current position of the pieces corresponds to a winning configuration.

Return:
bool → `True` if it's a winning configuration, `False` otherwise.

Fonction 8: `main()`

Description:
Manages the game flow. The main function is the main function that will be called when the program is launched and will end at the end of the game.

Return:
bool → `True` once the game is won.

Fonction 9: `print_grid(grid: list(list), no_number: bool)`

Description:
P Prints the game board. To improve playability, the boundaries of the board should be displayed using the characters '|' (vertical borders) and (-) (horizontal borders). Refer to the figures in the previous sections for an example (e.g., Figure 1).
The `no_number` parameter indicates whether the pieces should be displayed (if its value is `False`) or without (if its value is `True`) the corresponding numbers.

Return:
None

As mentioned earlier, you can (and are encouraged) to implement other functions to achieve a coherent and readable structure for your code.

The entirety of your code should be placed in a file called `tetramino. py`

1.5 Submission instructions

Your `tetramino.py` file (and only this file, do not include `getkey.py` or specific maps) is to be submitted on the UV in the corresponding section, before **Sunday, December 17th at 10 PM.** Be sure to respect the file name : `tetramino.py`, all in lowercase. (The file must be submitted as is, do not create a `.zip` file).

Important note: **No extensions are to be allowed.** (Note : do not send your code by email if you are late).

Notice to the "last-minute" specialists: We strongly advise you to submit an initial version of your project on the UV as soon as you have code that passes all tests, and then update it on the UV after each working session. This way, if you encounter a last-minute technical issue, the previous version of your code will still be on the UV.

As mentioned earlier, submission is contingent on the success of **ALL** the automated tests that will be provided to you. Without passing these tests, your code will not be graded. So, make sure your code passes the tests.

Good Luck!