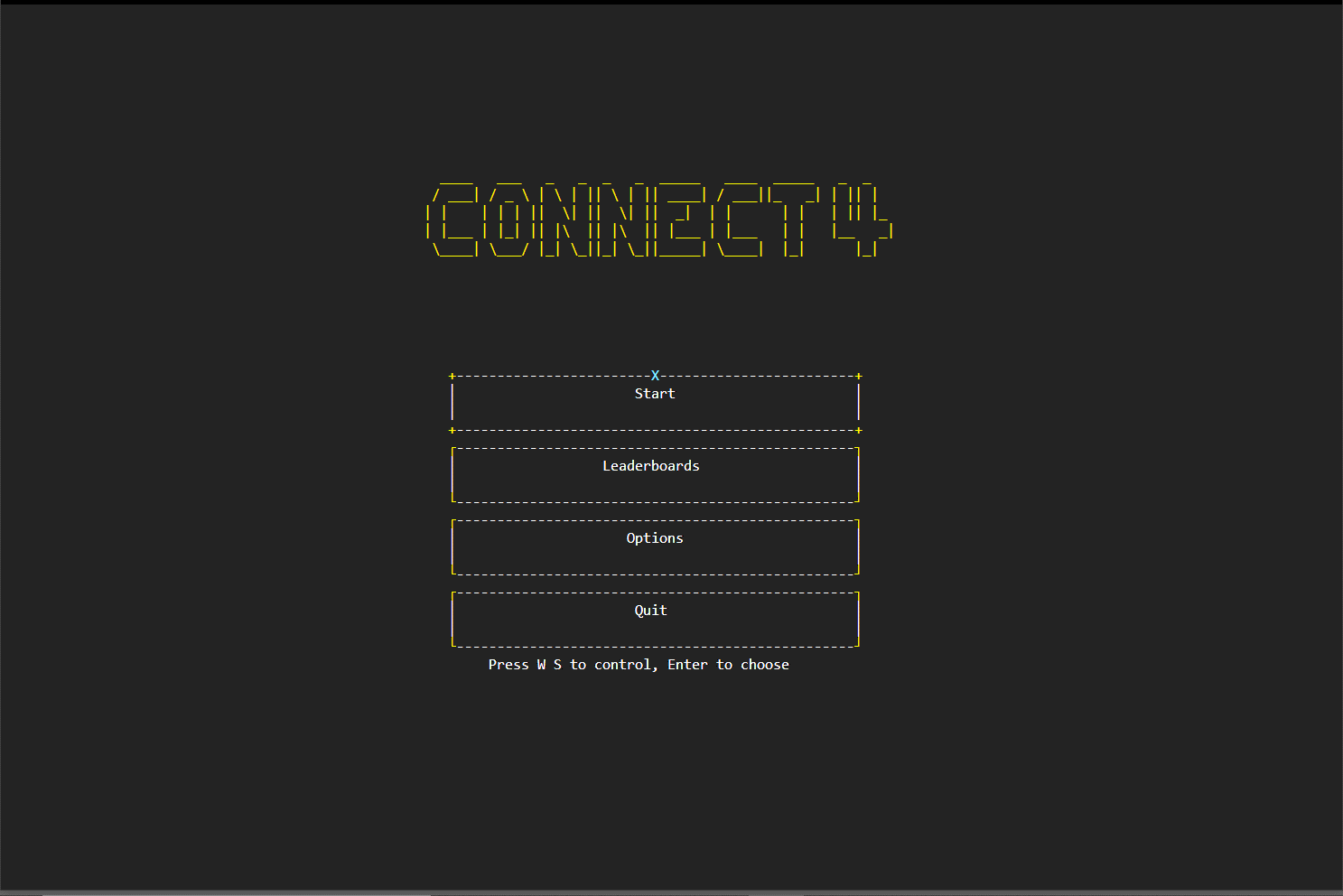
Connect 4

2019 March Programming Principles Assignment Group 4

|  |  |
| --- | --- |
| Task | People in charged |
| GUI | Chang Cheng (1802 6856),  Keanu (1802 0552) |
| Rules | Ke Xin (1710 6097),  Wen Li (1711 3564) |
| AI | Yuen Yuee (1900 0272) |



Contents

[Introduction 3](#_Toc12525975)

[Getting Started 5](#_Toc12525976)

[Prerequisites 5](#_Toc12525977)

[Issues 5](#_Toc12525978)

[Classes and functions 6](#_Toc12525979)

[Flowchart of game flow 18](#_Toc12525980)

[Flowcharts of file 19](#_Toc12525981)

[Flowcharts of functions 28](#_Toc12525982)

[Built with 54](#_Toc12525983)

[Acknowledgement 54](#_Toc12525984)

[Contact 54](#_Toc12525985)

# Introduction

The following program is modelled after the popular board game "Connect Four". This variant of the game utilizes the Python language, as well as the curses module to design and recreate the game.

The *features* of this game are as follows:

1. Colorful graphics

2. Customizable game settings

3. Navigation menu

4. Saving game states

1. **Colorful graphics**

Via usage of the curses module, menus and game boards are created with colors, which can interact with the player. This is done by initializing color pairs to be used when printing strings or objects in the program.

2. **Customizable game settings**

Before players start a Connect Four game, they are not only presented with a choice of game difficulty, but they are also able to modify player preferences, such as symbol color, and background music of choice. This allows for a more varied and enticing player experience.

3. **Navigation menu**

Unlike most variants of the Connect Four game which utilize terminals as well as manual input, this variant presents a Graphical User Interface (GUI) which can be interacted with by the player via the Arrow and Enter keys, allowing for easier navigation between menus.

4. **Saving game states**

In between each turn between the player and the computer, the program automatically saves the game state and stores the current game board status in a file. This allows players to pause and save games, to be continued at a later time.

As a result, the above features overall enhance the player experience and reduce player inconvenience.

Next, the *major strengths* of our program are as follows:

1. Easy to navigate

2. Challenging AI

3. Visually appealing

1. **Easy to navigate**

As stated earlier, the introduction of a navigation menu drastically reduces time spent navigating between game menus and screens, compared to that of a terminal program. This allows players to start the game without wasting any time.

2. **Challenging AI**

Unlike most variants of the Connect Four game which utilize Python's randint() function, this variant utilizes the Minimax-alike algorithm to accurately predict upcoming moves by the player, allowing the computer to optimize choices to prevent the player from winning. Overall, this makes the game more challenging and fun to play.

3. **Visually appealing**

Lastly, the usage of colors and symbols creates a familiar experience pertaining to 8-bit style retro games, and visually attracts the player's attention. With this, players are more likely to continue playing.

As such, this variant of the Connect Four game has strengths and features which outweigh that of other variants.

(422 words)

# Getting Started



Double click app.py to launch the program

# Prerequisites

This program required python 3 version in order to run, if you do not have python 3, please visit to <https://www.python.org> to download and install.

# Issues

These are multiple issues found (solved) :

1. If you are facing “\_curses\_error addwstr() returned ERR”:



Go to setting -> system -> Display -> change the scale and layout to 125% or below

1. Python/pip is not in system path:



Please refer to: <https://geek-university.com/python/add-python-to-the-windows-path/>

If you had found any new issue, please email to [18026856@imail.sunway.edu.my](mailto:18026856@imail.sunway.edu.my) for further helps.

# Classes and functions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class | Function | Function Description | Value Returned | Parameters | Parameters Description |
| -  (app.py) | main() | Main function for the program, handling setup, updates and resize the window screen automatically | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | music() | A private function that play menu page background music, its should be used with a daemon thread to play the music. | NA | - | - |
| Rectangle  (GUI-> Component-> low\_level\_component.py) | \_\_init\_\_() | Initialize the rectangle class | NA | Window **(curses object)**,  Init\_content\* **(string)**,  Top\_row\* **(boolean)**,  Top\_sym\* **(string)**,  Color\* **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **init\_content** is the content that will be at the middle of the rectangle (i.e. "start"),  **top\_row** is a boolean parameter (True = there are string that will be displayed at the top of the rectangle (i.e. +---1---+),  **top\_sym** is a string parameter for top\_row (i.e. 1/Score Board/X),  **color** is the color of the content, use curses defined color for this parameter (i.e. curses.COLOR\_YELLOW) |
|  | draw\_rectangle() | draw rectangle at the y and x given, it get the top left corner position, and bottom right corner position to draw the rectangle on the window initialized | NA | up\_left\_y **(integer)**, up\_left\_x **(integer)**, low\_right\_y **(integer)**, low\_right\_x **(integer)**, default\_corn\_sym = True | **up\_left\_y** is the top left corner's y value,  **up\_left\_x** is the top left corner's x value, and same to **low\_right\_y** and **low\_right x** |
|  | refresh\_rectangle() | Refresh rectangle from the virtual window | NA | - | **-** |
|  | property function: content, color | functions that with @property, so that it can be changed in future, content is the content that is in the rectangle, color is the color of the content | NA | - | - |
| LoadingAnimation  (GUI-> Component-> low\_level\_component.py) | \_\_init\_\_() | Initialize the loading animation class | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | draw\_loading() | Draw the loading animation at the y and x given | NA | Y **(integer)**,  X **(integer)** | **y** is the y point,  **x** is the x point |
| GameBoard  (GUI-> Component-> game\_board.py) | \_\_init\_\_() | Initialize the board | NA | Window **(curses object)**,  Box\_size **(integer)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Box\_size** is the size for every tile in the game board |
|  | draw\_board() | Draw the game board at the window parsed with the row and column that required (i.e. 6,7 or 6,9) | NA | Row\_amount **(integer)**,  Column\_amount **(integer)** | **Row\_amount** is the amount of row required (i.e. 6)  **Column\_amount** is the amount of column required (i.e. 9) |
|  | refresh\_board() | Refresh the game board from the virtual window of curses and draw to the window parsed | NA |  |  |
|  | data() | Return the data of the board in a two-dimensional list | A Two-dimension-al list which contain the board data in column by row format | - | - |
|  | data\_reset() | It clears every data inside the game board list to empty string | NA | - | - |
|  | draw\_column() | A private function to draw the rectangle (game board tile) column by column in game\_board\_list | NA | - (inherit from game board constructor) | - |
| ScoreBoard  (GUI-> Component-> score\_board.py) | \_\_init\_\_() | Initialize score board | NA | Window **(curses object)**,  Nlines **(integer)**,  Ncols **(integer)**,  Game\_mode **(string)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Nlines** is the height of the score board  **Ncols** is the width of the score board  **Game\_mode** is the difficulty (“6:7”/”6:9”) |
|  | draw\_score\_board() | A private function that draws the score board at the window’s (0,0) | NA | - | - |
|  | show\_scores() | A private function that used by draw\_score\_board() to retrieve the scores from file and add into score board | NA | - | - |
| - (main\_menu.py) | main() | Main program of main\_menu.py, the reason of not making this into a class is to prevent it create an instance which will kill the window created originally | NA | Window **(curses object)** | **-** |
|  | navigation() | A private function for navigation of the page to another page from the parameter passed | NA | Window**(curses object)**  Current\_button**(integer)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Current\_button** is an integer that’s indicate which button user choose (i.e. 1 = first button, 2 = second button) |
|  | clicking() | A private function to play the clicking sound effect, recommended to use with a daemon thread | NA | - | **-** |
| OptionPage  (GUI-> option\_page.py) | \_\_init\_\_() | Initialize option page, a page to set all the game preferences i.e. music, color) | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | main() | Load the option page | NA | - | **-** |
|  | clicking() | A private function to play the clicking sound effect, recommended to use with a daemon thread | NA | - | **-** |
|  | disrupt\_music() | A private function to disrupt the music that’s being play (previewing) | NA | - | **-** |
|  | save\_configuration() | A private function to save the setting of the option page to a json file | NA | Color**(curses object)**  Music**(string)** | **Color** is a curses object that is predefined, use curses.COLOR\_YELLOW, etc.  **Music** is a string that contain the music name (i.e. music 1, music 2, music 3) |
| LeaderBoardsPage  (GUI -> leaderboards\_page.py | \_\_init\_\_() | Initialize leaderboards page, a page to view the leaderboards | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | main() | Load the leaderboards page | NA | - | **-** |
| GameOptionsPage  (GUI -> game\_option.py) | \_\_init\_\_() | Initialize the game options page, a page to set the game setting, i.e. new game/ continue | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | main() | Load the game option page | NA | - | **-** |
|  | draw\_menu() | A private function that will draw the page layout and button on the window parsed | NA | Current\_button**(integer)** | **Current\_button** is an integer that indicate where the user’s cursor is currently pointing at. (initialized with 1) |
|  | navigation() | A private function for navigation of the page to another page from the parameter passed | NA | Window**(curses object)**  Current\_button**(integer)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Current\_button** is an integer that’s indicate which button user choose (i.e. 1 = first button, 2 = second button) |
|  | clicking() | A private function to play the clicking sound effect, recommended to use with a daemon thread | NA | - | **-** |
| NewGameOptions  (GUI -> game\_newgame.py) | \_\_init\_\_() | Initialize the new game page, a page after user choose for new game and for user to choose difficulty, i.e. normal/advance | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | main() | Load the new game page | NA | - | **-** |
|  | draw\_menu() | A private function that will draw the page layout and button on the window parsed | NA | Current\_button**(integer)** | **Current\_button** is an integer that indicate where the user’s cursor is currently pointing at. (initialized with 1) |
|  | navigation() | A private function for navigation of the page to another page from the parameter passed | NA | Window**(curses object)**  Current\_button**(integer)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Current\_button** is an integer that’s indicate which button user choose (i.e. 1 = first button, 2 = second button) |
|  | clicking() | A private function to play the clicking sound effect, recommended to use with a daemon thread | NA | - | **-** |
| ContinueGameOptions  (GUI -> game\_continue.py) | \_\_init\_\_() | Initialize the continue game page | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | main() | Load the continue game page | NA | - | **-** |
|  | draw\_menu() | A private function that will draw the page layout and button on the window parsed | NA | Current\_button**(integer)** | **Current\_button** is an integer that indicate where the user’s cursor is currently pointing at. (initialized with 1) |
|  | navigation() | A private function for navigation of the page to another page from the parameter passed | NA | Window**(curses object)**  Current\_button**(integer)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Current\_button** is an integer that’s indicate which button user choose (i.e. 1 = first button, 2 = second button) |
|  | clicking() | A private function to play the clicking sound effect, recommended to use with a daemon thread | NA | - | **-** |
| GameBoardPage  (GUI -> game\_board\_page.py) | \_\_init\_\_() | Initialize game board page | NA | Window **(curses object)**,  row\_size **(integer)**,  col\_size **(integer)**, game\_mode **(string)**, load\_saved = False | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Row\_size** is the size of row of the board  **Col\_size** is the size of column of the board  **Game\_mode** is the difficulty (“6:7”/”6:9”)  **Note: this is difference with the GameBoard class, this is the game board page (or we say, game play page)** |
|  | main() | Load the game board page | NA | - | **-** |
|  | \_board() | A private function to draw the game board to the page | NA | Board\_window **(curses object)**,  Box\_size **(integer)** | **Board\_window** is the window that’s for game board, it is separated with the original window to avoid curses.clear() to erase the board  **Box\_size**  is the size of the tile of the game board, 5 by default. |
|  | \_AI\_move() | A private function to calculate the appropriate AI response to the game state | A tuple which consist (column index, row index) | - | **-** |
|  | \_clicking\_music() | A private function to play the clicking sound effect, recommended to use with a daemon thread | NA | - | **-** |
|  | \_play\_background() | A private function to play the background music, recommended to use with a daemon thread | NA | - | **-** |
|  | \_loading() | A private function to draw the loading animation to the appropriate place of the page | NA | Window **(curses object)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper() |
|  | \_score\_board() | A private function to draw the score board on the page | NA | - | **-** |
|  | \_game\_over\_page() | A private function to call the game over page and draw in appropriate place of the page | NA | Game\_state **(string)** | **Game\_state** is a string that will be pass to game over page, “O”/”X”/”draw” |
| -  (rules.py) | winning\_check() | This is to check if there is required pattern exists in the game board, its been reused by ai.py again in order for AI to recognize pattern. | A tuple (column,row),  A string(type of the pattern), boolean | Win\_connect **(integer)**,  Filename **(string)**,  Game\_mode **(string)**,  Ai\_mode=False,  Specific\_check = “”,  Specific\_sym = “” | **win\_connect** is the required amount of connected symbol ( i.e. 4 for 6:7, 5 for 6:9),  **filename** is the name of the file which stored the board data  **Game\_mode** is the difficulty (“6:7”/”6:9”)  **ai\_mode** is for ai’s usage to check for specific pattern in order to perform the algorithm, False means it’s a normal winning\_check  **specific\_check** is the specific pattern that’s required: “hori” for horizontal “verti” for vertical, “pdiag” for positive diagonal (left to right), “ndiag” for negative diagonal (right to left). |
| GameLogic  (GUI -> Game Logic -> game\_logic.py) | slot\_check() | This is to check if the move entered is valid | Boolean,  Integer (the index of available slot) | Game\_list **(list)**,  Col\_key **(integer)**,  Ai\_mode = False | **Game\_list** is the list that contains the board data,  **Col\_key** is the index of column entered,  **Ai\_mode** true if its being used by ai |
|  | save\_data() | This is to save the game | NA | Game\_list**(list)**,  Game\_mode**(string)**,  Total\_attempt**(integer)** | **Game\_list** is the list that contains the board data,  **Game\_mode** is the difficulty (“6:7”/”6:9”) |
|  | load\_saved\_data() | This is to load the saved game | A list that contains the board data | Game\_mode**(string)** | **Game\_mode** is the difficulty (“6:7”/”6:9”) |
|  | reset\_data() | It reset the data in the data file, and the data in the GameBoard class | NA | Game\_mode**(string)** | **Game\_mode** is the difficulty (“6:7”/”6:9”) |
|  | dropping\_animation() | It takes in the column index and perform dropping animation on the game board | NA | Game\_list**(Array)**  Col\_key**(integer)**  Move\_index**(integer)**  Char**(string)** | **Game\_list** is the list that contains the board data,  **Col\_key** is the column index of the game board  **Move\_index** is the row index of the game board  **Char** is the string that represent the symbol of player/AI. |
| -  (ai.py) | ai() | It decides the move for AI based on the algorithm designed | A tuple: column, index of the row | Game\_mode**(string)** | **Game\_mode** is the difficulty (“6:7”/”6:9”) |
| GameOverPage  (GUI -> gameover\_page.py) | \_\_init\_\_() | Initialize game over page | NA | Window **(curses object)**,  Orig\_window **(curses object)**,  Status **(char)**  Total\_attempt **(integer)**,  Game\_mode **(string)** | **Window** is a curses object that’s created by using curses.newwin(), or curses.wrapper()  **Orig\_window**  is the window of the game board page  **Status** is “O”/”X”/”draw”, O means player wins, X means AI wins.  **Total\_attempt** is the total move of player taken. |
|  | main() | Load game over page | NA | - | **-** |
|  | save\_score() | A private function to save player’s name, date and score | NA | Name **(string)**  Score **(integer)** | **Name** is the input name of the player,  **Score** is the score of the game |
|  | play\_background() | A private function to play the background music for game over page | NA | State **(string)** | **State** is the state of the game, function will play the corresponding music for state input. |

# Flowchart of game flow

Game Flow

A close up of a map

Description generated with high confidence

# **Flowcharts of programs**

App.py

A close up of a piece of paper

Description generated with very high confidence

Game\_board\_page.py

A close up of a piece of paper

Description generated with very high confidence

Game\_continue\_options.py

A screenshot of a social media post

Description generated with very high confidence

Game\_option.py

A close up of a sign

Description generated with very high confidence

Gameover\_page.py

A close up of a map

Description generated with high confidence

Leaderboards\_page.py

A close up of a logo

Description generated with very high confidence

Main\_menu.py

A close up of a sign

Description generated with very high confidence

Option\_page.py

A close up of a piece of paper

Description generated with high confidence

Game\_newgame.py

A close up of a sign

Description generated with very high confidence

# Flowcharts of functions

Draw\_loading() [low\_level\_component.py]

A close up of a map

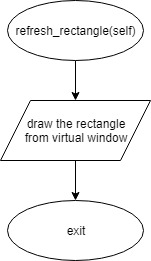
Description generated with very high confidence

Draw\_rectangle [low\_level\_component.py]

A close up of a map

Description generated with high confidence

Refresh\_rectangle() [low\_level\_component.py]



Save\_data() [game\_logic.py]

A close up of a sign

Description generated with very high confidence

Reset\_data() [game\_logic.py]

A close up of a map

Description generated with high confidence

Slot\_check() [game\_logic.py]

A close up of a map

Description generated with high confidence

Winning\_check() [rules.py]

A close up of a map

Description generated with very high confidence

Data() [game\_board.py]

A close up of text on a white background

Description generated with high confidence

Data\_reset() [game\_board.py]

A close up of a map

Description generated with very high confidence

Data\_set() [game\_board.py]

A close up of a map

Description generated with very high confidence

Draw\_board() [game\_board.py]

A close up of a map

Description generated with very high confidence

Refresh\_board() [game\_board.py]

A close up of a logo

Description generated with very high confidence

Draw\_menu [main\_menu.py]

A close up of a map

Description generated with high confidence

Draw\_options [option\_page.py]

A close up of a map

Description generated with high confidence

Dropping\_animation() [game\_logic.py]

A close up of a map

Description generated with high confidence

Load\_save\_data() [game\_logic.py]

A close up of a logo

Description generated with very high confidence

Ai() [ai.py]

A close up of a map

Description generated with very high confidence

Main() [app.py]

A close up of text on a black background

Description generated with very high confidence

Main() [game\_board\_page.py]

A picture containing clock

Description generated with high confidence

Main() [game\_continue\_option.py]

A close up of a map

Description generated with very high confidence

Main() [game\_option.py]

A close up of a map

Description generated with very high confidence

Main() [gameover.py]

A close up of a map

Description generated with high confidence

Main() [leaderboards\_page.py]

A screenshot of a cell phone

Description generated with high confidence

Main() [main\_menu.py]

A close up of a map

Description generated with high confidence

Main() [option\_page.py]

A close up of a map

Description generated with high confidence

# Built with

Python Curses ( <https://docs.python.org/2/library/curses.html> ) – A terminal-based GUI framework

# Acknowledgement

* This program is for 2019 Sunway University Programming Principles Assignment uses
* This program is an initial work
* Open source, free to fork

# Contact

Please contact to [18026856@imail.sunway.edu.my](mailto:18026856@imail.sunway.edu.my) or <https://github.com/pupubird>.