# CHECK NUMBERS GRID GAME

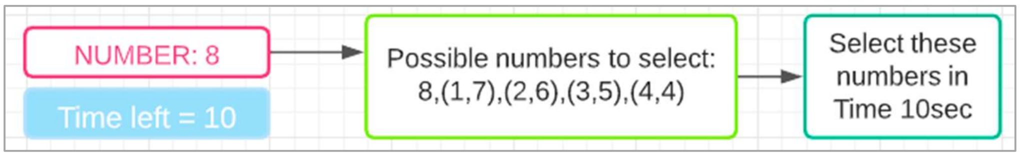
## ABSTRACT

The **Check Number Grid Game** is based on the **mathematical addition** of numbers which will be presented in a form of grid providing a user-friendly interface. It will allow us to select either the **numbers which represents the sum equal to given number** or **a single number that will be equal to the given number**. This provides an interactive way to deals with **addition principle and numbers identification** in a due time. The paper attempts to build the game using python programming by going deeper into the functions of the game. The identified problem here needs to be solved followed by **Computational thinking** consisting of **Decomposition, Abstraction, Algorithm** and **Pattern recognition**. The **python program** is made to provide all the necessary details using Graphical user interface and perform the **operations** according to the requirements.

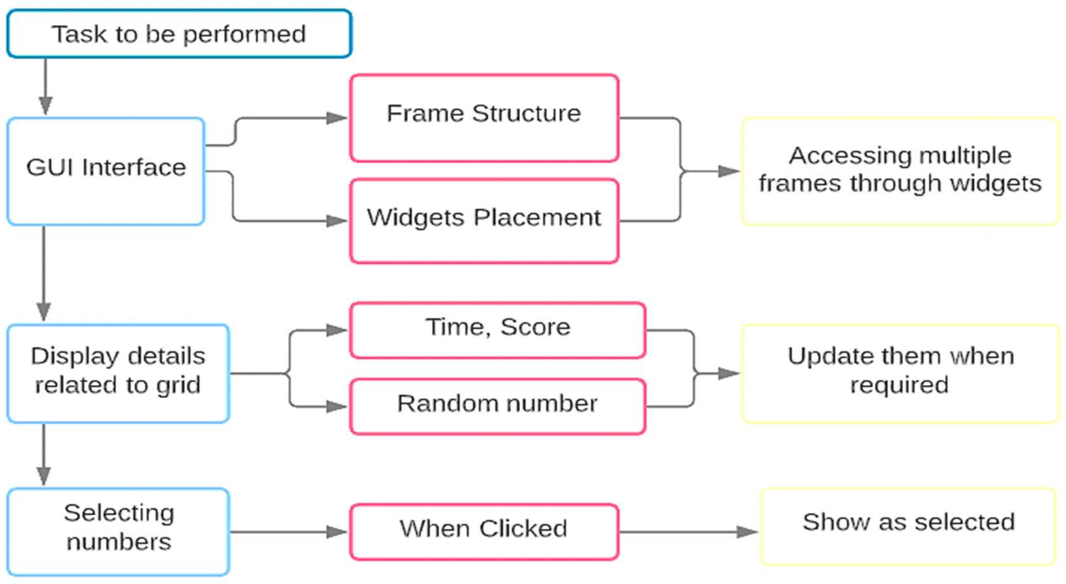
## INTRODUCTION

The Check Number Grid Game has the basic objective to select the pair of tiles whose sum will be equal to given random number or a single tile that will be equal to given number. The numbers cannot be repeated and when selected its colour should be changed to represent the selected tiles which will record the score as well. For the new user game instructions should be there to go through and then start the game.

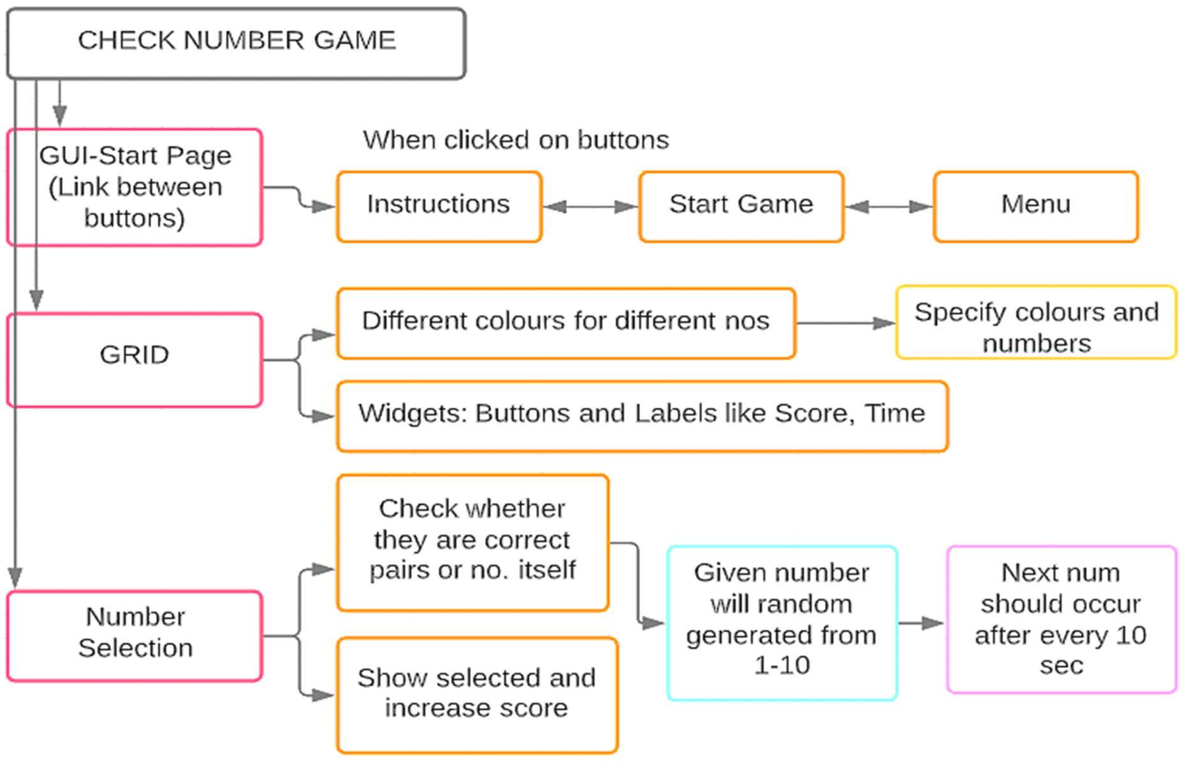
Functionality of the game is dependent on that generated given number and time left for that. The number will be randomly generated from 0 to 10, user will get 10 seconds for each random number to identify the pairs or number itself in the grid. This will be like:


NUMBER: 8
Time left = 10
Possible numbers to select: 8,(1,7),(2,6),(3,5),(4,4)
Select these numbers in Time 10sec

At last every tile in the grid need to be selected. All of these operations need to performed in an organised way to reach the required output. To get more deeper knowledge of task need to performed the time chart is created to divide the work and perform the same.

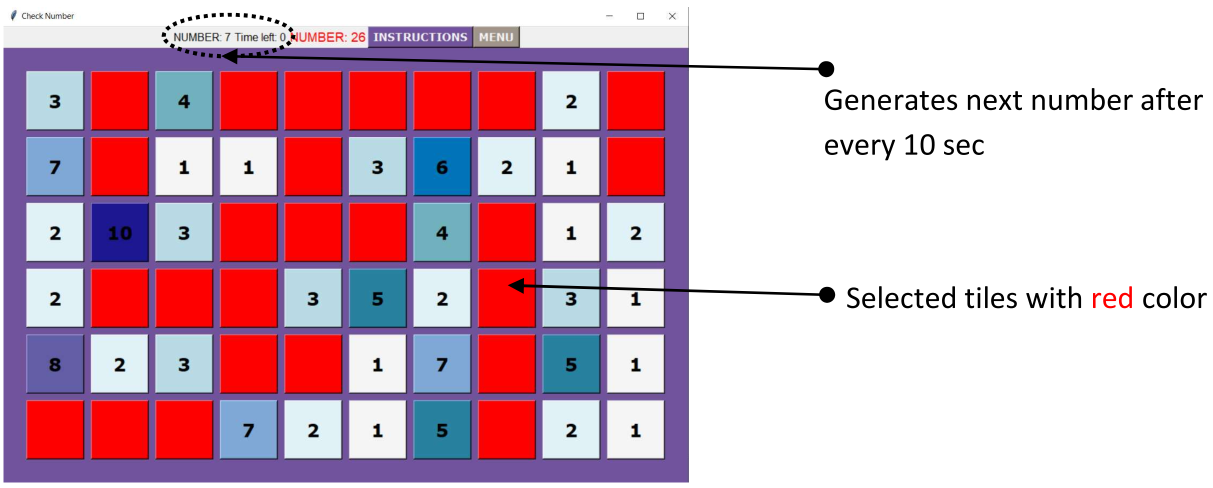


The basic problem which we need to solve is to select the grid tiles on the basis of the numbers. The game provides a user-friendly interface to work and perform task the problem needs to be go deeper inside to reach the solution. On the basis of problem solving and computational thinking aspects the solution is reached. Decomposition of the problem is represented through issue tree as follows:


CHECK NUMBER GAME
GUI-Start Page (Link between buttons)
When clicked on buttons
Instructions
Start Game
Menu
GRID
Different colours for different nos
Specify colours and numbers
Widgets: Buttons and Labels like Score, Time
Number Selection
Check whether they are correct pairs or no. itself
Show selected and increase score
Given number will random generated from 1-10
Next num should occur after every 10 sec

The patterns identified for the functioning of the game are:

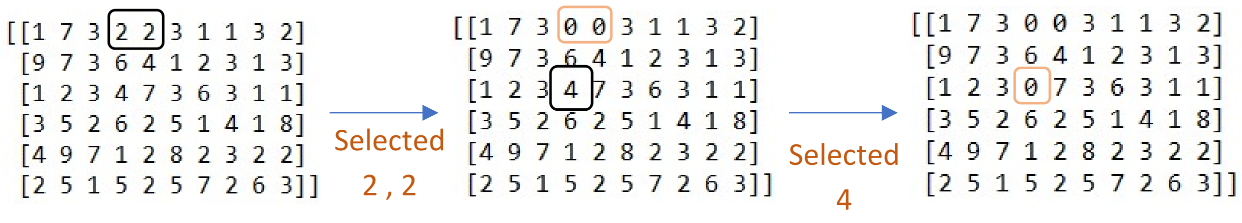
- For the pair of numbers whose sum will be equal to that generated number, numbers will be selected 2 at a time. To check for a pair the current and last element will be checked for the addition.
- Colouring of the selected tiles will be red with no text. Score will add this into and increase the same for all tiles
- Only 10 chances will be given to select the number of tiles for that random number. After 10 sec a new number will be generated. This will occur till time gets over.


Check Number
NUMBER: 7 Time left: 0
NUMBER: 26
INSTRUCTIONS
MENU
Generates next number after every 10 sec
Selected tiles with red color

## LOGIC

The logic behind the functioning of the game is based on the accessing the elements of the matrix. The matrix is of 6 x 10 with elements at randomly located positions. Taking row and column as the input will gave value of number. There are 2 list named as list_num and sub_num which will be initialized as empty.

- If the number is not in the list_num than it will check whether the number is equal to that random number or not. If yes its value in matrix will be changed to 0 and if not it will be stored in sub_num
- Taking next number, if not equal it will check the sum of current number and last number in sub_num. If the sum is equal than its value in the matrix will be 0.



```
[[1 7 3 2 2 3 1 1 3 2]
 [9 7 3 6 4 1 2 3 1 3]
 [1 2 3 4 7 3 6 3 1 1]
 [3 5 2 6 2 5 1 4 1 8]
 [4 9 7 1 2 8 2 3 2 2]
 [2 5 1 5 2 5 7 2 6 3]]
```

```
[[1 7 3 0 0 3 1 1 3 2]
 [9 7 3 6 4 1 2 3 1 3]
 [1 2 3 4 7 3 6 3 1 1]
 [3 5 2 6 2 5 1 4 1 8]
 [4 9 7 1 2 8 2 3 2 2]
 [2 5 1 5 2 5 7 2 6 3]]
```

```
[[1 7 3 0 0 3 1 1 3 2]
 [9 7 3 6 4 1 2 3 1 3]
 [1 2 3 0 7 3 6 3 1 1]
 [3 5 2 6 2 5 1 4 1 8]
 [4 9 7 1 2 8 2 3 2 2]
 [2 5 1 5 2 5 7 2 6 3]]
```

Random number = 4

## DESIGN AND IMPLEMENTATION

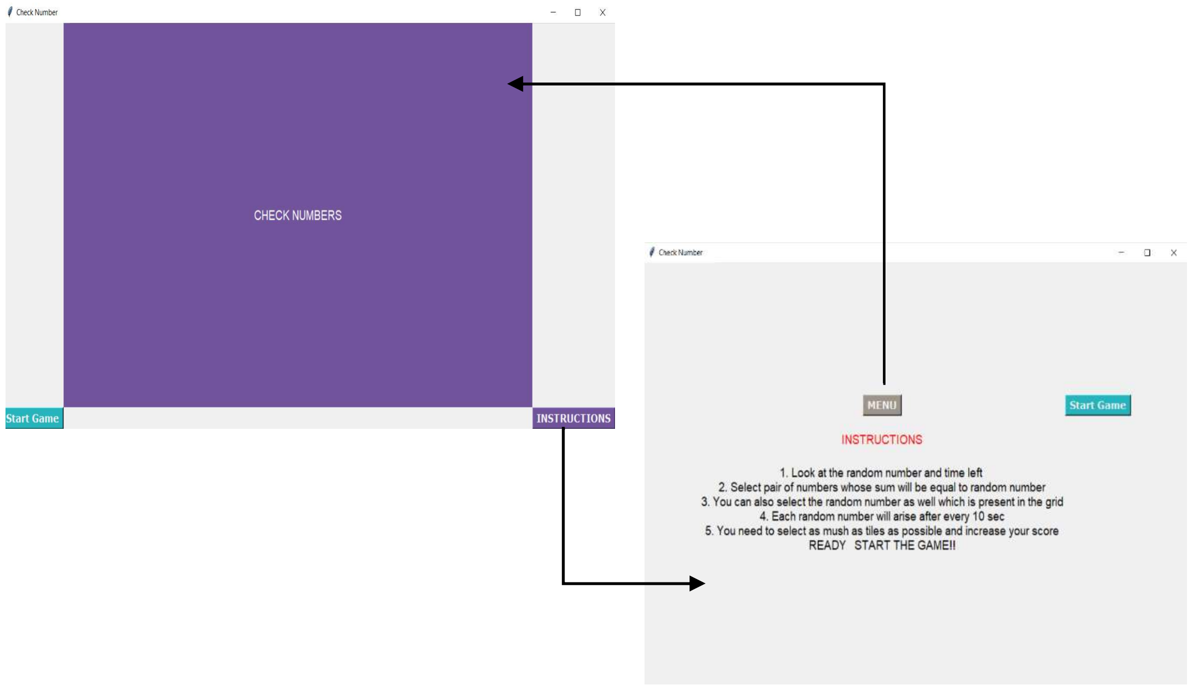
- **Functions and Modules**
  To develop a Graphical user interface for the game the python program imports tkinter module and for performing different operations random and numpy modules are imported. The functions included in the program are divided to perform their specific task. All the operations related to the game is combined in the form of a class Grid_num inside which the following functions are created.

| Modules | Usage |
|---|---|
| tkinter<br>`from tkinter import *`<br>`import tkinter as tk` | Provide a GUI consisting widgets, functions and inputs. Creates different frames and main grid that takes input of user and select the tile |
| Random<br>`import random` | Generate random number for which we need to select tile through shuffle function: `random.shuffle(self.l)` |
| numpy<br>`import numpy as np` | Create matrix and insert numbers between 0 to 10 at random position with each number specific probability (times) |

| Functions | Specifications |
|---|---|
| **`tileselect( )`** | Check the number at clicked position row and column. If number is equal to that random number than change the colour of tile in GUI and if not store the same into a list. Check next number with same if not equal than will sum the both the numbers for equality. |
| **`init_grid(),`<br>`init_matrix(),`<br>`start_grid()`** | Init_grid( ) will create the frame of cells and grid following which the init_matrix will generate matrix with numbers. Start_grid will represent the init_matrix in grid with different numbers colours |
| **`Game_widgets()`** | Place the buttons and labels at respective location |
| **`Countdown()`** | Take the time limit input and decrement it by 1 for each 1 sec. Update the same in the grid GUI frame as well |
| **`Random_num()`** | Generate random number from 1 to 10 after every 10 sec and following this numbers the tiles are selected |
| **`Start()`** | Start the game with initially with plane frame of 2 options instructions and start game |
| **`Instructions()`** | Display the instructions for the game and place the Menu and Start game button |

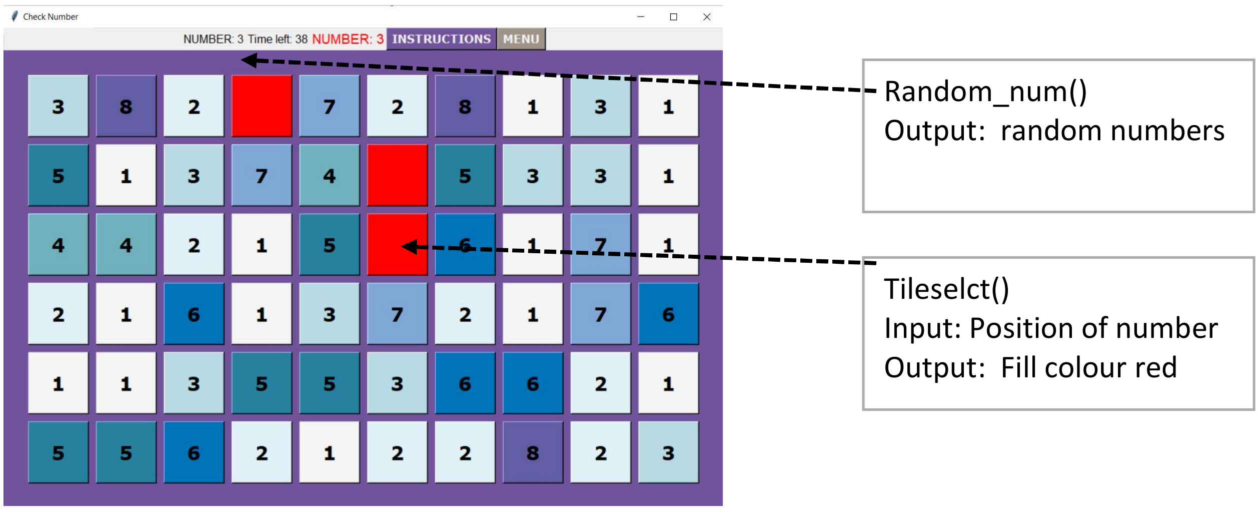
| | |
|---|---|
| **`Startgame()`** | Start the Game with grid and widgets displayed and starts timer with number |

- **Architecture**
  The functions inside the class are interrelated to each for performing their task. They are dependent on output of each and every function of the class. We access the class attributes and methods using self keyword which binds the attribute with the given arguments. The architecture followed to connect above mentioned functions is:



INSTRUCTIONS

1. Look at the random number and time left
2. Select pair of numbers whose sum will be equal to random number
3. You can also select the random number as well which is present in the grid
4. Each random number will arise after every 10 sec
5. You need to select as mush as tiles as possible and increase your score
READY   START THE GAME!!



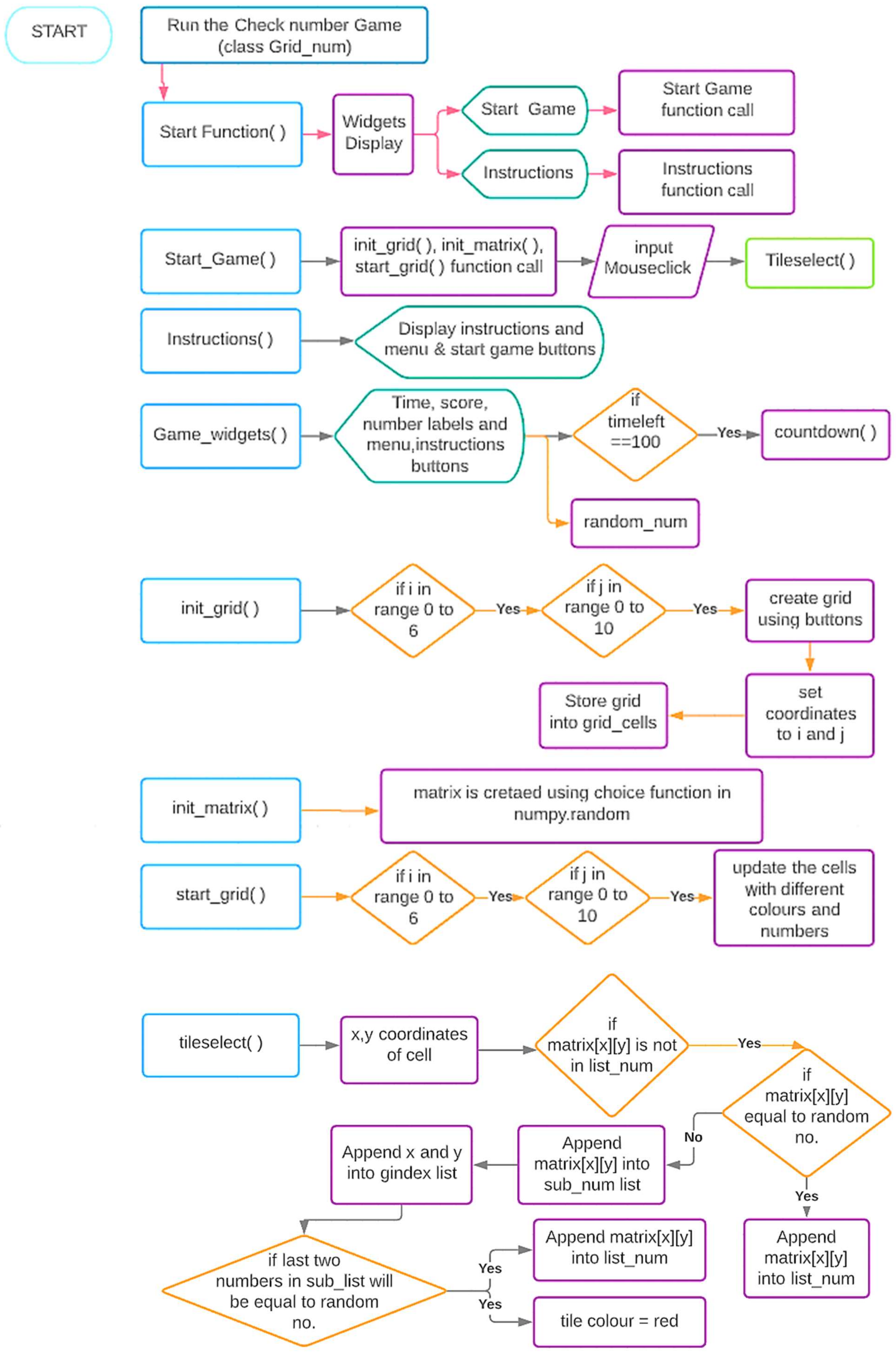
NUMBER: 3 Time left: 38 NUMBER: 3

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 8 | 2 | 🟥 | 7 | 2 | 8 | 1 | 3 | 1 |
| 5 | 1 | 3 | 7 | 4 | 🟥 | 5 | 3 | 3 | 1 |
| 4 | 4 | 2 | 1 | 5 | 🟥 | 6 | 1 | 7 | 1 |
| 2 | 1 | 6 | 1 | 3 | 7 | 2 | 1 | 7 | 6 |
| 1 | 1 | 3 | 5 | 5 | 3 | 6 | 6 | 2 | 1 |
| 5 | 5 | 6 | 2 | 1 | 2 | 2 | 8 | 2 | 3 |

- **Algorithm**
  To perform different operations and make an interactive interface we followed the following algorithm depending upon applied logics and identified functions.

Pseudo code

```
1. Dictionary of cell colours
   Background_colour_cells = { 2: "#e0f2f8"}

2. (Initialising variables & call function)
   Frame.__init__(self)
   Initailizinf different variables and a frame
   Calling functions like init_grid( )

3. (Background colours)
   Background = Frame(colour, width, height)
   For i in 0 to 6
       Initializing grid_row to empty
       For j in 0 to 10
           Creating cells in the form of 6x10
           grid with initializing the colours and
           font
           The cells will be the buttons (which
           will be selected)
     Appending grid_row to self.grid_cell

4. (Generation of 6x10 matrix)
       num=np.random.choice(10,60,p=[0.181,
       0.169,0.15,0.134,0.117,0.084,0.066,0.05
       ,0.033,0.016])
       matrix=np.reshape(self.num,(6,10))
5. (Updating colour and number)
   For i in 0 to 5 and For j in 0 to 10
     Set new_num as matrix[ I ][ j ]
         Set colour from dictionary

7. (Selection of tiles)

3 list list_num, sub_num and gindex to store
the values of the matrix

Getting x and y coordinates of tile
If matrix[row][col] is not in list_num
  If Matrix[row][col]==random_num
     Setting the colour of grid cell as red
  Else:
     Adding the matrix[row][col] into sub_num
     Adding the row and col into gindex
     If last and 2nd last values in sub_num adds
       to become random number
           Fill the colour of tile to red

8. (Countdown starts to show time left)

Time left = 100
If time_left > 0
    Decrement time left with 1
    Update the frame to display time left
    Repeat the function again & again till
    time left is 0

9. (Generate random number)
Shuffle the existing list
Random num = List[0]
Removing list[0] item from list
Returning random num
```



START
Run the Check number Game (class Grid_num)
Start Function( )
Widgets Display
Start Game
Instructions
Start Game function call
Instructions function call
Start_Game( )
init_grid( ), init_matrix( ), start_grid( ) function call
input Mouseclick
Tileselect( )
Instructions( )
Display instructions and menu & start game buttons
Game_widgets( )
Time, score, number labels and menu,instructions buttons
if timeleft ==100
Yes
countdown( )
random_num
init_grid( )
if i in range 0 to 6
Yes
if j in range 0 to 10
Yes
create grid using buttons
set coordinates to i and j
Store grid into grid_cells
init_matrix( )
matrix is cretaed using choice function in numpy.random
start_grid( )
if i in range 0 to 6
Yes
if j in range 0 to 10
Yes
update the cells with different colours and numbers
tileselect( )
x,y coordinates of cell
if matrix[x][y] is not in list_num
Yes
if matrix[x][y] equal to random no.
No
Append matrix[x][y] into sub_num list
Append x and y into gindex list
Yes
Append matrix[x][y] into list_num
if last two numbers in sub_list will be equal to random no.
Yes
Append matrix[x][y] into list_num
Yes
tile colour = red

Data types included in the program are:

| Data types | Usage |
|---|---|
| Sequence, list | `self.grid_cells = [ ]` |
| String | `self.timeLabel.config(text = "Time left: "+str(self.timeleft))` |
| Integer - int | `self.matrix[x][y]=0` |
| Dictionary | `BACKGROUND_COLOR_CELLS = {2: "#e0f2f8"` |

**CONCLUSION**

The Check Number grid game has a main objective of finding the tiles with same number or pair of tiles whose sum will be equal to given number. We need to implement the operations and mechanism of the game with an interactive interface into the python program which need to be done following computing principles i.e. Decomposition, Pattern recognition, Algorithm and Abstraction. The document highlights a python program including a Grid_num class that consist of functions to perform operations like generating random number, selecting tiles, setting time limit and providing a GUI grid. In the output it generates multiples frames for instructions and start game to provide structured and entertaining game.