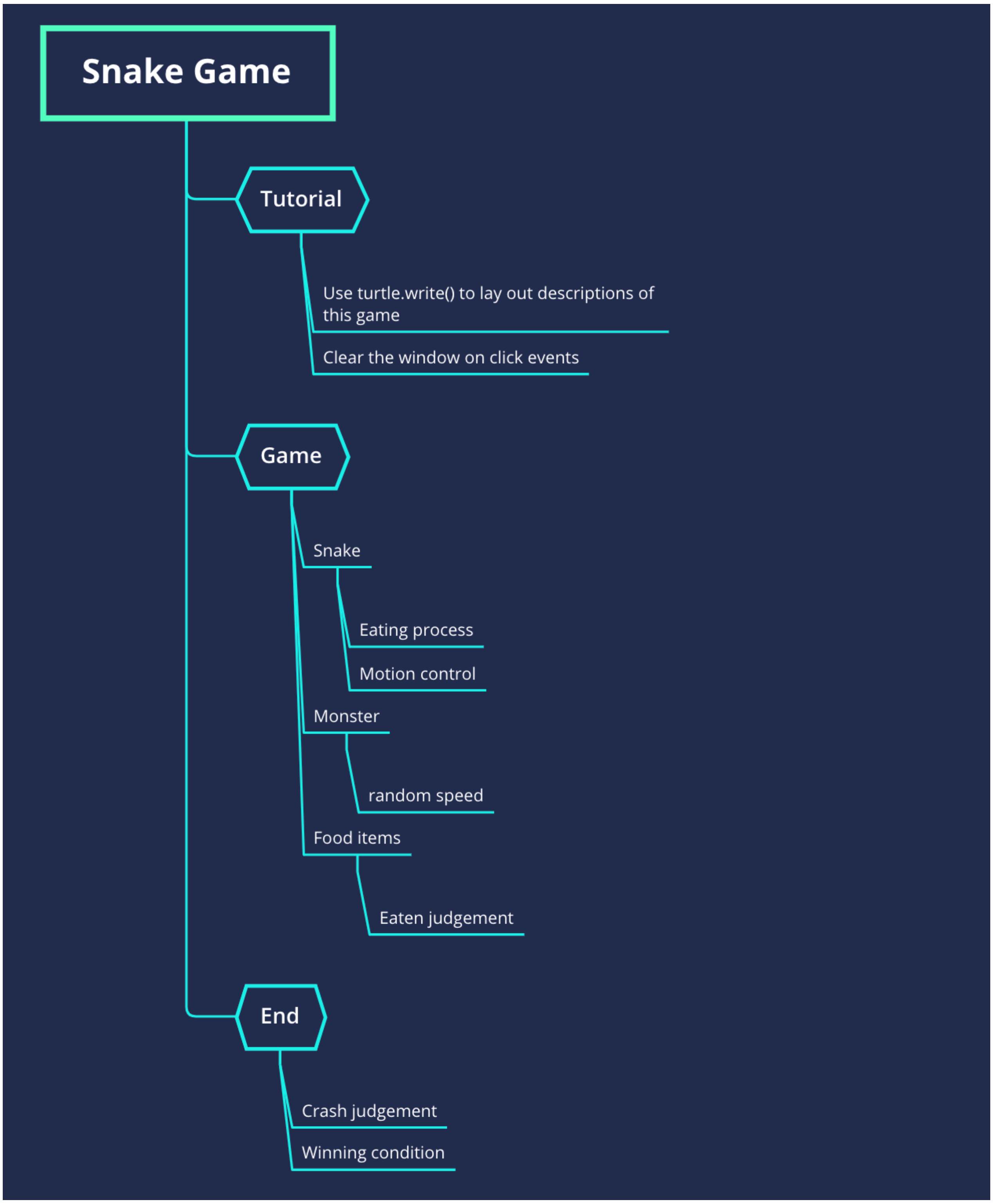
A2_SSE_119010136_Design

1.Design

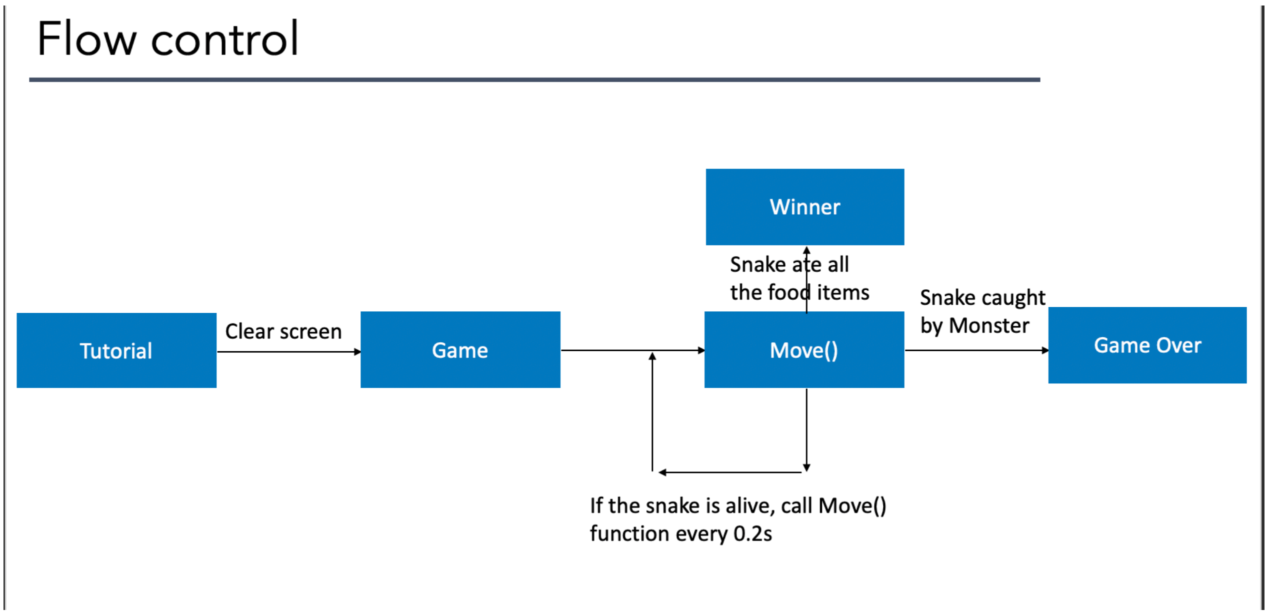
1.1 Environment

- macOS Catalina, Version 10.15.3
- Python 3.8.1 64-bit

1.2 Outline



1.3 Design logic


Flow control
Winner
Snake ate all the food items
Tutorial
Clear screen
Game
Move()
Snake caught by Monster
Game Over
If the snake is alive, call Move() function every 0.2s

We have several useful methods in turtle module : `ontimer`, `update`.

1. `ontimer` can call a function between constant time intervals. So we can use it to call the functions that simulate the motion of the snake and the monster.

   Because `ontimer` only accept one function as its parameter. We should build a function that includes the motion control of both the snake and the monster.
2. `update` can refresh the screen so that the motion of the snake and the monster looks like animation.

1.4 Modules import

```
from turtle import Turtle, Screen # Core function library
from random import randint # Randomly pick up positions for food items and a
monster
from math import sqrt # Use it to calculate the distance
```

1.5 Constants

```
SIZE = 20
```

Because the initial size of turtle.shape('square') is 20. So we set SIZE = 20

1.6 Global variables

```
SIZE = 20
g_dx = 0
g_dy = 0
g_surf = Screen()
g_game_on = 1
g_snake = []
g_snake_pen = artist()
```

```
g_snake_pen.shape('square')
g_snake_v = 2  # g_snake's initial speed
g_snake_al = 5  # At the start of the game, the length of the tail is set to 5
g_snake_cnt = 0
g_monster_pen = artist()
g_monster_pen.shape('square')
g_monster_pen.color('purple','purple')
g_foods = []
g_contacted = 0
g_ptime = 0
```

Use set_screen to set the parameters for this screen.

```
def set_screen():
    global g_surf
    g_surf.setup(500, 500)
    g_surf.tracer(0)
    g_surf.listen()
    g_surf.onkey(up, "Up")
    g_surf.onkey(down, "Down")
    g_surf.onkey(left, "Left")
    g_surf.onkey(right, "Right")
    g_surf.onkey(stop, 'space')
    g_surf.onkey(close, 'q')
```

2.Functions

2.1 Directions

We use four functions to react to user events

```
def up():
    global g_dx, g_dy
    g_dx = 0
    g_dy = SIZE


def down():
    global g_dx, g_dy
    g_dx = 0
    g_dy = -SIZE


def left():
    global g_dx, g_dy
    g_dx = -SIZE
    g_dy = 0
```

```
def right():
    global g_dx, g_dy
    g_dx = SIZE
    g_dy = 0
```

2.2 Game status

2.2.1 Stop/Continue playing

```
def stop():
    global g_game_on
    g_game_on ^= 1
```

`g_game_on` is a boolean variable. Switching it to `False` menas stopping the game. Switching it to `True` means continuing playing.

In the following functions, before we move the snake and the monster we shall check the value of `g_game_on`. Use it as global variable can help us do this conveniently.

Operator `^` is `Xor`. This operator can transform 1 to 0 and 0 to 1. This helps me change to game status.

2.2.2 Quit

```
def close():
    g_surf.bye()
```

Press the "Q" button on your key board to quit the game. The window will automatically close if you press "Q".

2.3 Initialisation

2.3.1 Get pens to draw

```
def artist(color='black', speed=0):
    pen = Turtle(visible=False)
    pen.speed(speed)
    pen.color(color)
    return pen


def sou(pen, x, y):
    pen.up()
    pen.setposition(x, y)
    pen.down()
```

We use function `artist` to create a Turtle instance. Use this function to set up everything well. We don't want to see the turtle so `visible=False`.

We use function `sou` to move the turtle without leaving trails on the screen.

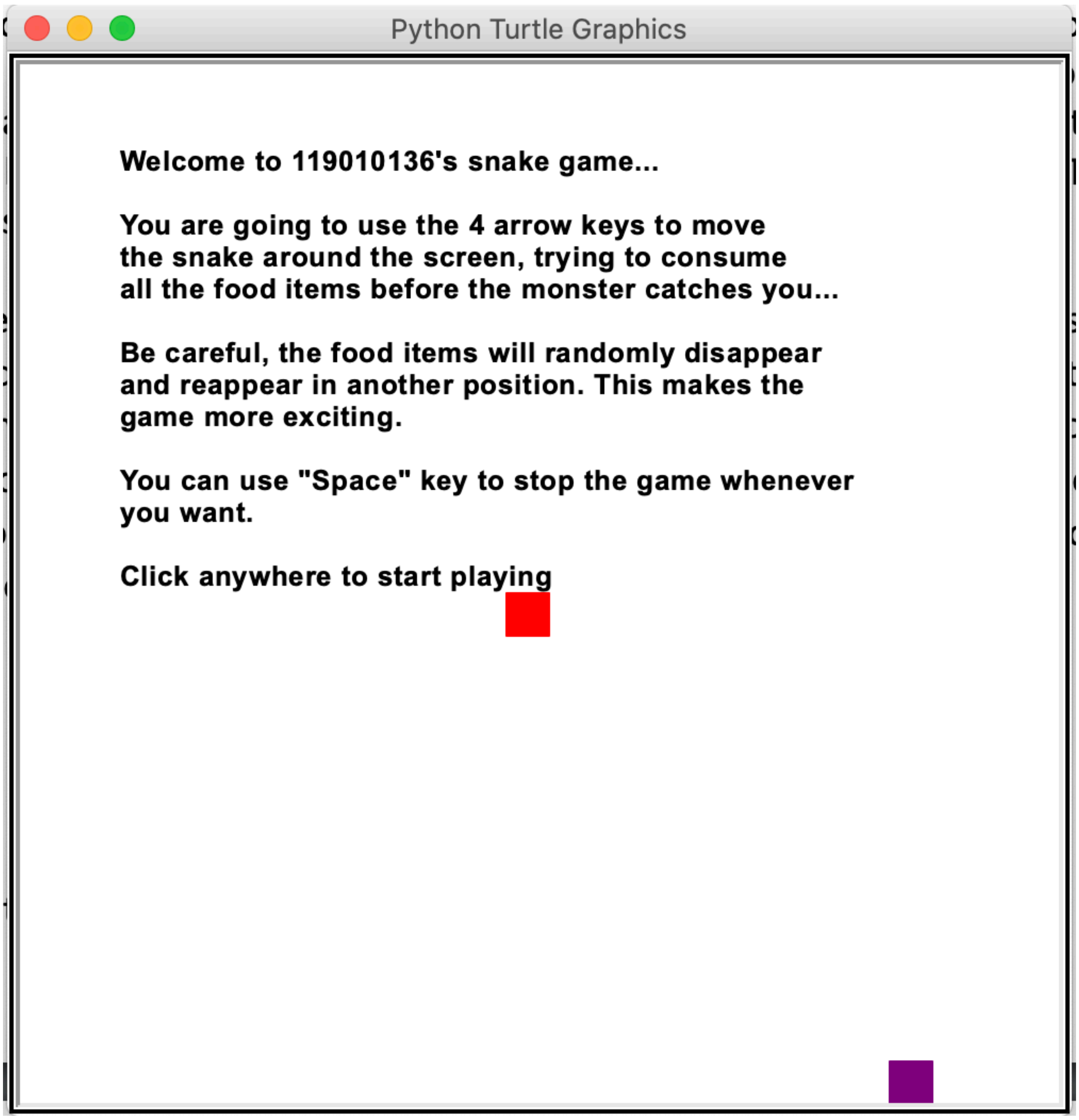
2.3.2 Tutorial interface

```python
def tutorial(pen=artist()):
    show_snake()
    show_mons()
    sou(pen, -200, 0)
    pen.write('Welcome to 119010136\'s snake game...\n\nYou are going to use
the 4 arrow keys to move\nthe snake around the screen, trying to consume\nall
the food items before the monster catches you...\n\nBe careful, the food items
will randomly disappear\nand reappear in another position. This makes
the\ngame more exciting.\n\nYou can use "Space" key to stop the game
whenever\nyou want.\n\nClick anywhere to start playing',
              align='left', font=('Arial', 13, 'bold'))
    def fun(x, y):
        g_surf.onclick(None)  # Event-binding will be removed
        pen.clear()
        game()
    g_surf.onclick(fun)
```

The Screen.onlick() attribute accept one parameter, a function. This function should accept the (x,y) coordinates of the click event on the screen. Here we use a closure structure to solve this problem.

Use `def fun()` we can erase all the tutorial information as soon as the user click on the screen.

It looks like this.



Welcome to 119010136's snake game...

You are going to use the 4 arrow keys to move
the snake around the screen, trying to consume
all the food items before the monster catches you...

Be careful, the food items will randomly disappear
and reappear in another position. This makes the
game more exciting.

You can use "Space" key to stop the game whenever
you want.

Click anywhere to start playing

2.3.3 Make food

```
def make_food():
    global g_foods
    x_pool = sample([i for i in range(-20,20)], 10)
    y_pool = sample([i for i in range(-20,20)], 10)
    for i in range(9):
        f = [x_pool[i]*10, y_pool[i]*10, i+1, artist(speed=8)]
        g_foods.append(f)
        sou(f[3], f[0], f[1])
        f[3].write(str(f[2]), font=('Arial', 14, 'normal'))
```

`foods` is a global list. We store the (x,y) coordinates and the value of each food item.

After that, we create a turtle to draw each food on the screen.

We use font in size 14. Arial style.

Use x_pool and y_pool to make sure no item would be placed on the same position.

Every food item will be randomly placed inside the 400 * 400 inside the screen. (The screen size is 500 * 500)

2.3.4 Create a snake and a monster

```
def creature():
    global g_monster_pen
    show_head()
    x, y = randint(-11,11)*20, randint(-11,11)*20
    while sqrt(x**2+y**2) < 20:
        x, y = randint(-11,11)*20, randint(-11,11)*20  # To ensure the
g_monster is far from the g_snake at first
    sou(g_monster_pen, x, y)
    g_monster_pen.stamp()
```

Because the monster and the snake both have 20*20 size. So we set the monster's coordinates with randint(-11,11) * 20.

Also, we use a while loop to make sure we generate a monster that is far enough from the snake at first.

2.4 Game core functions

2.4.1 Flow control

```
def game():
    global g_surf, g_snake_pen, g_monster_pen, g_game_on, g_monster_v
    make_food()
    g_surf.ontimer(snake_move, g_snake_v)
    g_surf.ontimer(monster_move, g_monster_v)
```

This function is triggered by funtion `tutorial`. As soon as the user clicks, the program will jump into this function.

We use `game()` to control the overall update speed of the screen. And check if the game reach an end.

If the snake consumed all the food, we call function `end()` to print out the winning message.

If the snake crash with the monster, we call function `end()` to print out the losing message.

Because we have to count the current state and time and print these informatin on the screen title. So we have to call the move() and counter() at the same time, at the same frequency. So put them all in function `play()`. Trigger them together by calling `ontimer(play(), 200)`. So the screen state updates every 0.2 second.

2.4.2 Move

Everytime check first if the snake ate some food. Use `eat_food` to check it out.

Use snake_cnt to count the time. `g_snake_cnt` is always smaller than snake_v because `g_snake_cnt=(g_snake_cnt + 1) mod g_snake_v`

We set snake_v = 2

So when snake_v =1, we do not move the snake.

But when snake_v = 0, we move it.

So generally the snake moves every 0.4s because we call `move()` every 0.2s.

$$snake\ normal\ speed = 5\ unit \cdot s^{-1}$$
$$snake\ eating\ food\ speed = \frac{5}{3} \approx 1.6\ unit \cdot s^{-1}$$

We use `randint(0,2)` to generate a random number in interval [0,2]. So we know the monster have $\frac{1}{3}$ possibility to move everytime.

And its moving frequency expectation is

$$\begin{aligned} E(monster\ speed) &= 1/\sum 0.2s\ \cdot\ P \\ &= 1/(0.2s\ \cdot\ \frac{1}{3} + 0.4s\ \cdot\ \frac{1}{9} + 0.6s\ \cdot\ \frac{1}{27} + \ldots) \\ &= 1/0.15\ unit \cdot s^{-1} \\ &\approx 6.67\ unit \cdot s^{-1} \end{aligned}$$

It seems little faster than the snake.

However, because we use random functions. So somtimes the monster will move really slow so that it is slower than the snake.

Be careful when snake head is near the monster because the monster might suddenly speed up and kill the snake !

snake_move()

```
def snake_move():
    global g_snake, g_snake_al, g_dx, g_dy, g_snake_v, g_snake_pen, g_surf,
g_game_on, g_end
    if g_end:
        return
    if g_game_on:
        counter()
        eat_food()
        go_on()
        x, y = g_snake_pen.position()[0], g_snake_pen.position()[1]
        if -240 <= x + g_dx <= 220 and -220 <= y + g_dy <= 240:
            g_snake_pen.clearstamp(g_snake[-1][2])
            g_snake[-1] = [x, y, g_snake_pen.stamp()]
            sou(g_snake_pen, x + g_dx, y + g_dy)
            show_head()
            if g_snake_al:
                g_snake_v = 400
                g_snake_al -= 1
            else:
                g_snake_v = 200
                g_snake_pen.clearstamp(g_snake[0][2])
                del g_snake[0]
            g_surf.update()
        go_on()
    if not g_end:
        g_surf.ontimer(snake_move, g_snake_v)
```

Check if the current i is the head of this snake. Use red color to fill it.

monster move()

```
def monster_move():
    global g_monster_pen, g_snake, g_surf, g_game_on, g_monster_v, g_end
    if g_end:
        return
    if g_game_on:
        encounter()
        g_monster_pen.clearstamps()
        x, y = g_monster_pen.position()[0], g_monster_pen.position()[1]
        if abs(g_snake[-1][0]-x) > abs(g_snake[-1][1]-y):
            if g_snake[-1][0] > x:
                x += SIZE
            elif g_snake[-1][0] < x:
                x -= SIZE
        else:
            if g_snake[-1][1] > y:
                y += SIZE
            elif g_snake[-1][1] < y:
                y -= SIZE
        sou(g_monster_pen, x, y)
        g_monster_pen.stamp()
        g_monster_v = randint(200, 600)
        g_surf.update()
        go_on()
    if not g_end:
        g_surf.ontimer(monster_move, g_monster_v)
```

The monster should chase after the snake. But it can only move one step each time. So we use if elif sentenses.

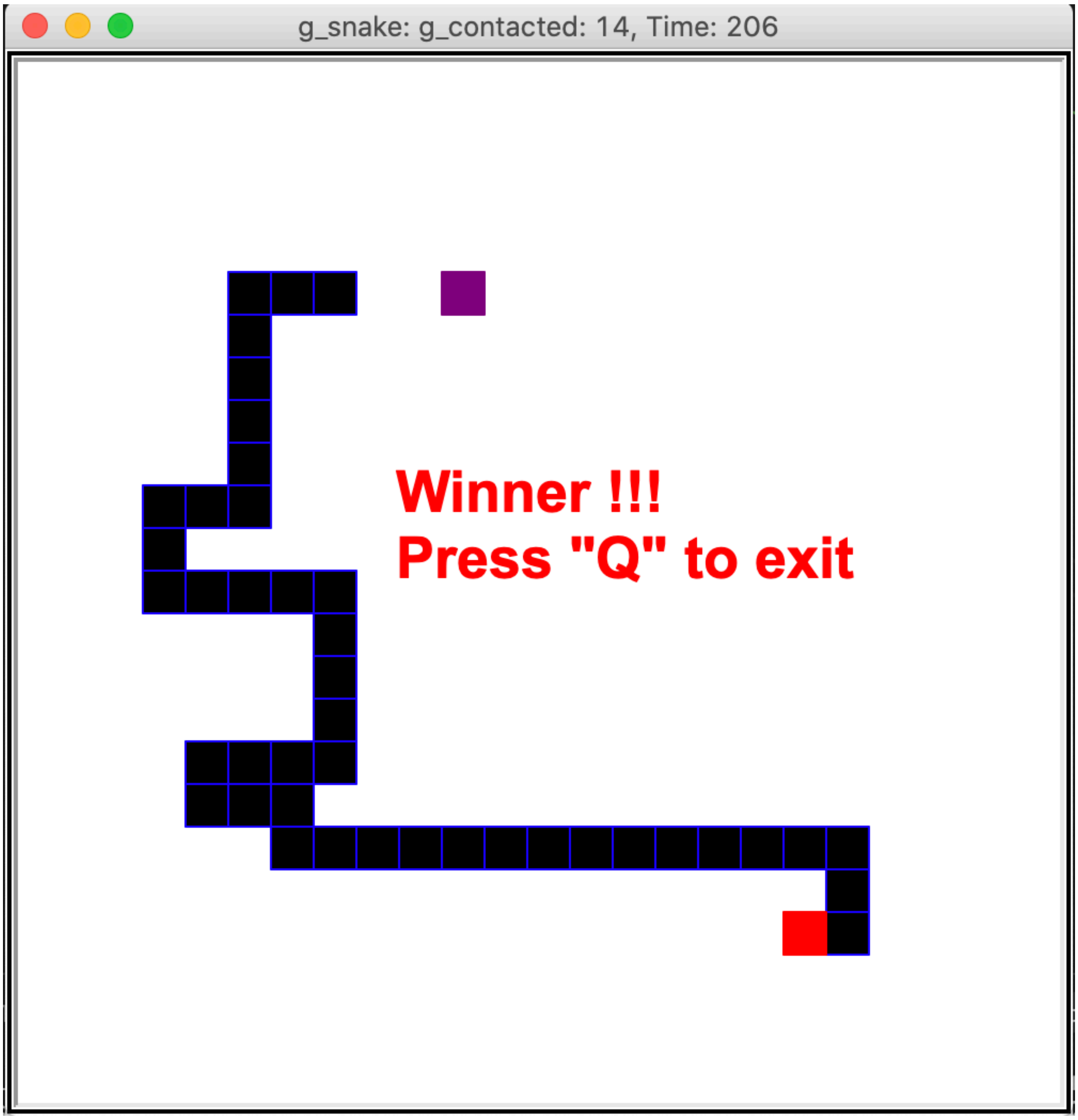
2.4.3 Time

Print the playing time on the screen title

```
def counter():
    global g_contacted, g_ptime, g_surf
    g_surf.title('g_snake: g_contacted: '+str(g_contacted)+', Time:
'+str(int(time() - g_ptime)))
```

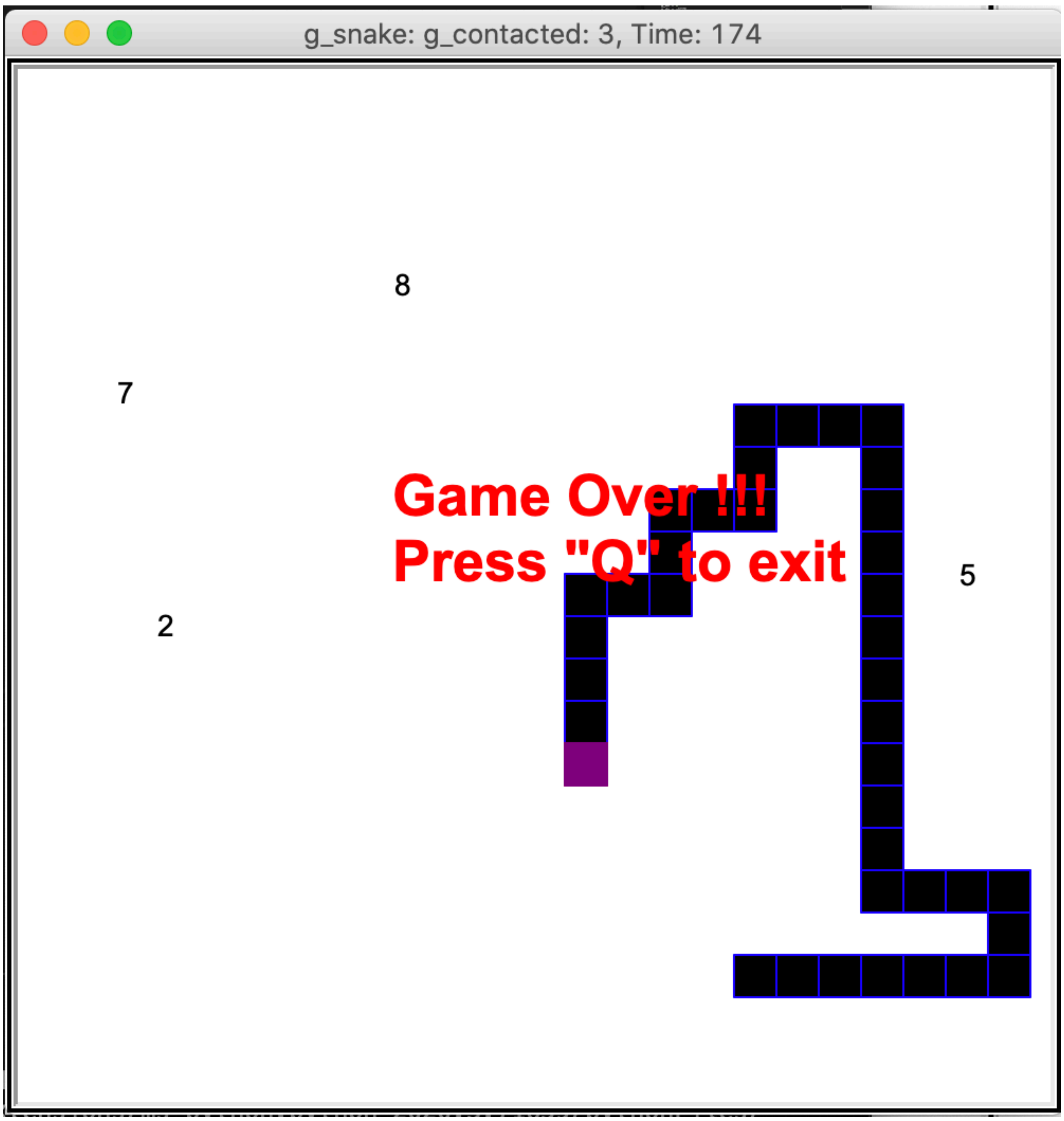
3.Output

3.1 Winner


g_snake: g_contacted: 14, Time: 206

Winner !!!
Press "Q" to exit

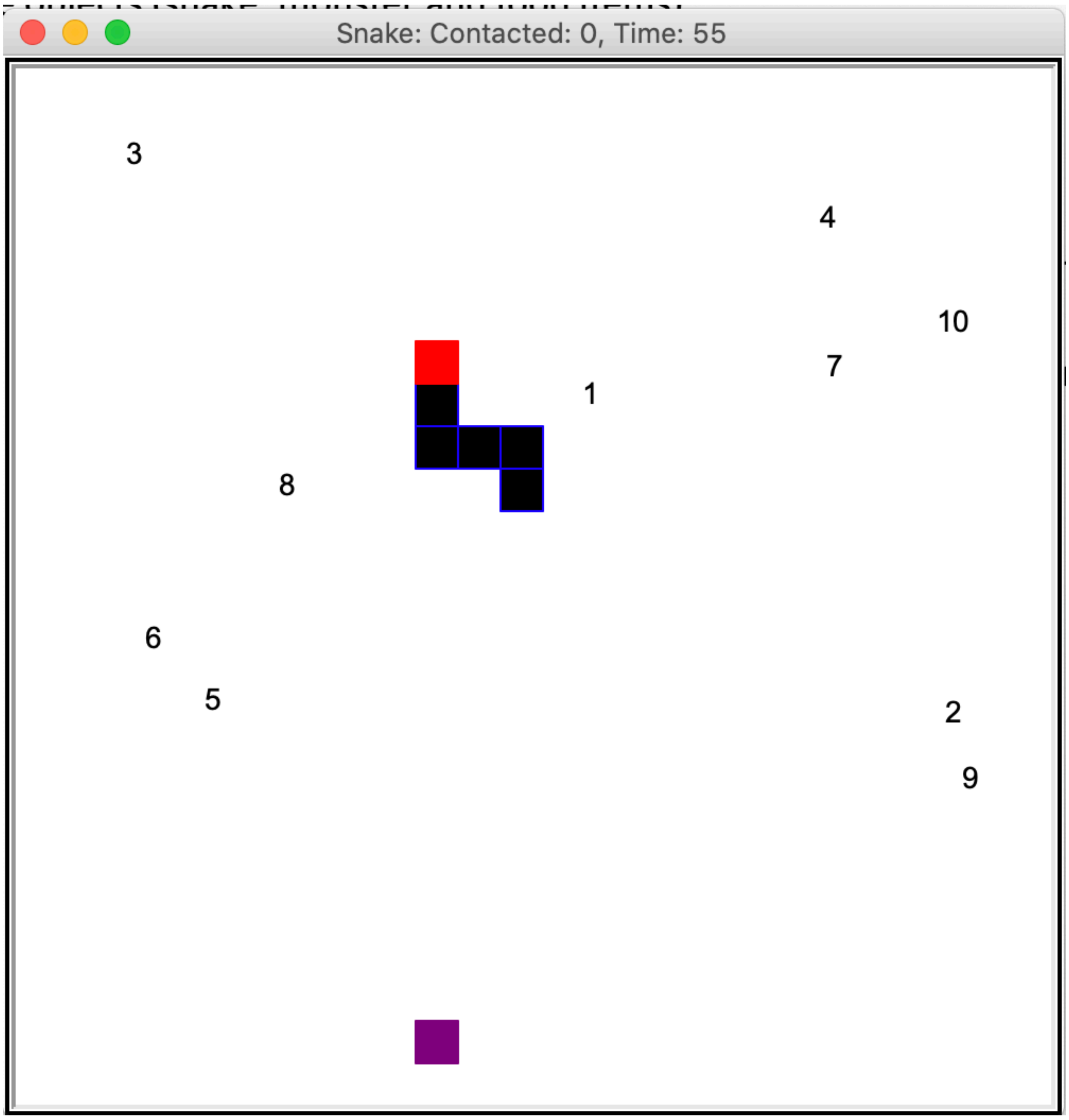
3.2 Game over


g_snake: g_contacted: 3, Time: 174

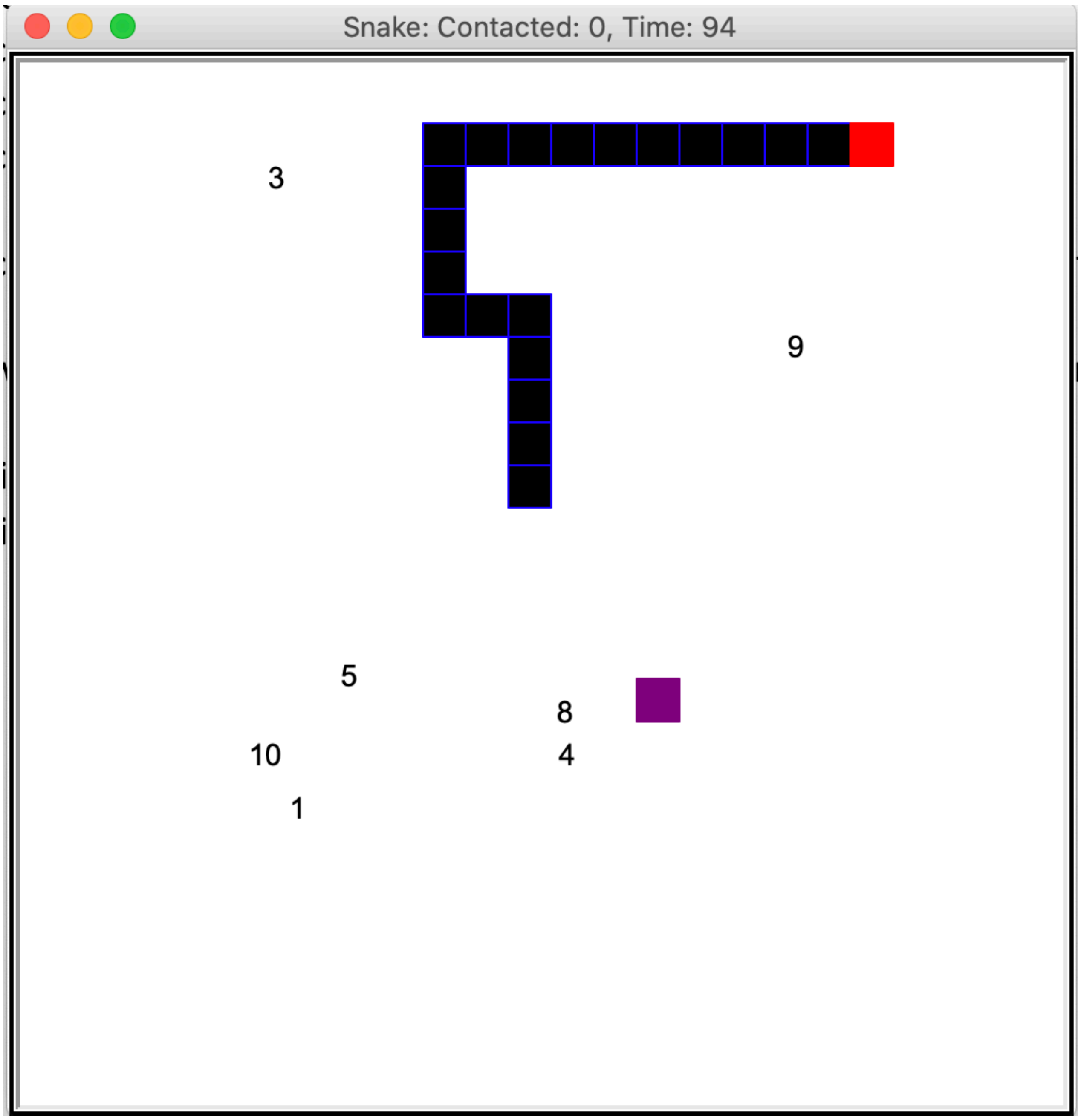
Game Over !!!
Press "Q" to exit

3.3 Food left

3.3.1 None consumed


Snake: Contacted: 0, Time: 55

3.3.2 3 items consumed


Snake: Contacted: 0, Time: 94
3
9
5
8
10
4
1