1. Design：

1)thoughts and overall approach

Turtle: We will use 12 turtles: g\_monster, g\_snake, and t0 to t9. g\_monster represents the monster. g\_snake represents the snake. t0 is used to write the description. t1 to t9 are used to write the food number 1-9.

Screen: We use g\_screen to represent the screen. Its height and width are 500.

Process: First, we write the description and show the turtle and monster to the player in the screen. After the play click the screen, we start the game. We clear t0(description). At the same time, t0-t9 will start writing the food number on the screen. If the distance of the food and the snake is less than 60, then the food will be eaten. And we clear that number. If the distance of the monster and the body of snake is less than 15, then the number of contacts plus one. If the distance of the monster and the head of snake is less than 15, then game over. If no food on the screen, the player wins.

1. The motion:

We use ontimer function to move the snake and monster forward continuously. If the snake is eating(extending), the speed is 0.35s 20 forward. If it is not eating, the speed is 0.27s 20 forward.The speed of the monster depends on that of snake. For a 20 forward. The time will be set randomly, between the time needed for the snake+100 and -10. We use onkey funstion. If we push ‘up’, then if will move up. Other direction keys are just like that. If we push the space, then the state of the snake will be changed. If the state is True, then the snake can move. Else it can’t move. The monster still can move at any time.

1. Date structure.

Snake: We use list\_snake to record the id of every part of the snake(such as 65). We use list\_snake\_pos to record the position of every part of the snake(such as (100,90)). the size of both lists equal to the length of the snake. And state\_running is used to record the state of the snake.(True means running, False means stop). extend\_length records the length that the snake should extend. Direction means the direction of the movement of the snake(‘left’,’right’).

Monster: pos means the position of the snake. Pos2 means the position of the monster. If the distance between the snake and the monster is less than 15, then the monster won’t move any more.

Food: the listFood is used to record the number and the position of the food.(such as (100,-134,8), the number 8 is at 100,-134) And t1 write 1, t2 write 2 and so on.

1. Tail extending:

Every time the snake moves, we append the id of the head into list\_snake and the position of the head into list\_snake\_pos. We also use stamp method copy a square at the position of the head. If extend\_length is 0, that means the snake don’t extend. We delete the first element in list\_snake and list\_snake\_pos. We also clear the stamp whose id is the first element in list\_snake. If extend\_length is not 0, that means the snake should extend. We just don’t clear and delete them. At first, list\_snake will be (that means the initial length of the snake). If the snake eat food, we just add the number to extend\_length.

1. Contact between snake and monster.

If the distance between the monster and every part of the snake is less than 15, that means contact.(if that’s the head, game over.)

1. **Functions**
2. def origin\_of\_monster():

It will return the initial position of the monster. It will choose a random position in the screen, but the distance between it and the initial position of the snake(0,0) should be larger than 90.

For example (-80,100)

2)def configureScreen(w=500,h=500):

It was used to design the screen. The title, the width and the height.

1. def configureTurtle(shape='square',color='red',x=0,y=0):

It is used to design the head of the snake.

3)def contacted():

If the distance between the snake and the monster is less than 15, it returns True. Else, it returns False.

4)def moveUp():

The direction change into up.

Setheading(90)

5)def moveDown():

The direction change into down.

Setheading(270)

6)def moveLeft():

The direction change into left.

Setheading(180)

7)def moveRight():

The direction change into right.

Setheading(0)

8)def movesnake(d=20):

First, we check if the state\_running is True. Then we check if the monster contact the body of the snake. If yes, then the contacting\_times+1. We check if the monster contact the head of the snake. If yes, it means the player fails. We check if the listFood is empty. If yes, it means the player wins. We check if the snake hits the boundary. We also check if the snake is eating.

After all checks pass, we move the snake forward. And we change the contacted times and time in the title.

9)def movemonster(d=20):

If the game ends, then the monster can’t move. If the transverse distance is larger than the  longitudinal distance, the monster will move breadthwise toward the head of the snake. The same check method with the transverse distance is smaller than the  longitudinal distance.

The speed is above.

1. def checkBoundary():

We use it to check if the movement of the snake will hit the boundary(500\*500). If the snake is right over the boundary and the direction is still right, the fun return False. The same method with other three direction.

1. def fail():

It was used when the player fails, we write ‘game over’.

1. def win():

It is used when the player wins. We write ‘you win’

13)def pause():

The state\_running will change.

1. def configureKey(s):

Player can use the direction key to change the directions with it.

14)def extend\_no\_eating(heading=0,dist=20):

It is used when the snake isn’t extending. It delete the tail and copy another square just in the position of the head.

15)def extend\_eating():

It is used when the snake is extending. It copy another square just in the position of the head.

16)def check\_eat\_food():

It was used to check if the food is eaten.

We check every element in listFood. If the distance between the food and snake is less than 15, that means the food is eaten. Don’t forget that one turtle write one food. We just clear that number and the element after the food is eaten.

17)def distance(a,b):

It return the distance between two point a and b.

If a=(3,0) b=(4,0), then it returns 5.0

18)def write():

It is used to write the description.

19)def setfood():

It is used to write the food on the screen. Vaild is used to check. The distance among each food should be over 60. The distance between the food and the origin should be over 15, or the food may be in the snake’s head initially. And the food can’t be out of the screen. The vaild will be True if the position satisfied those conditions. So the food can be written.

20)def start(a,b):

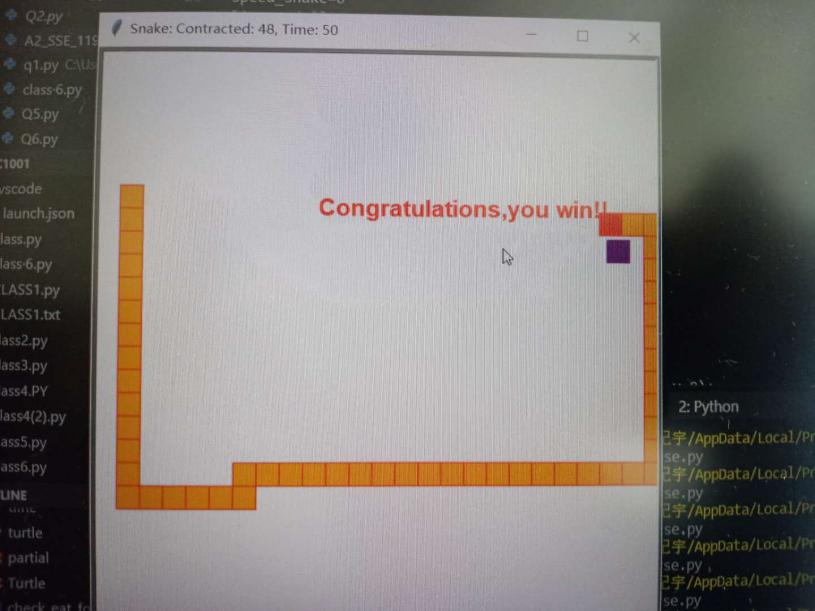
It was used to start the game. We record the start time. We clear the description. We set the food on the screen. We move the snake and the monster. After the game start, click won’t make sense. We also use the configurekey function to start the keyboard control.

21)if \_\_name\_\_ =='\_\_main\_\_':

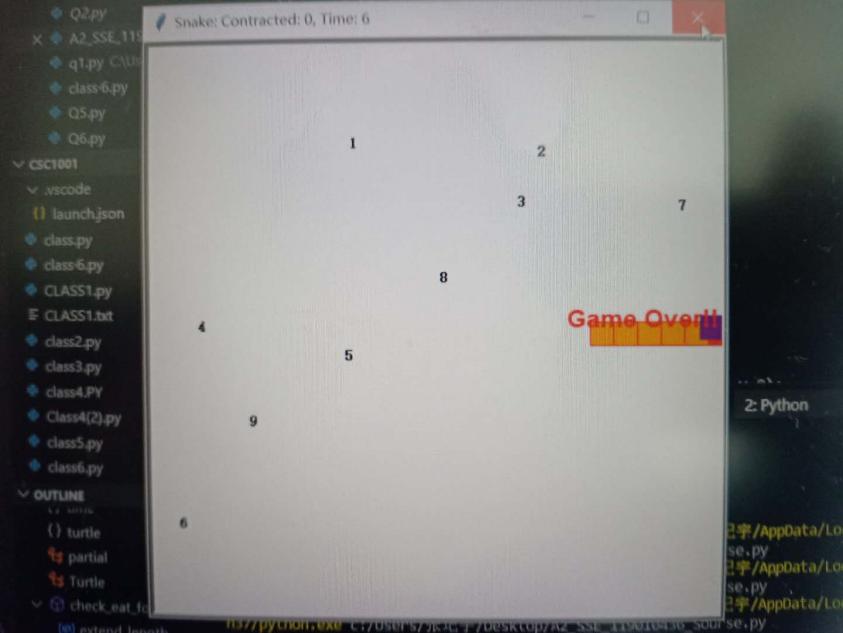
It does some preparations for the game.

We set the screen, the figure of the snake and the monster. We write description. We use onclick function. If the player click the screen, the start function will start.

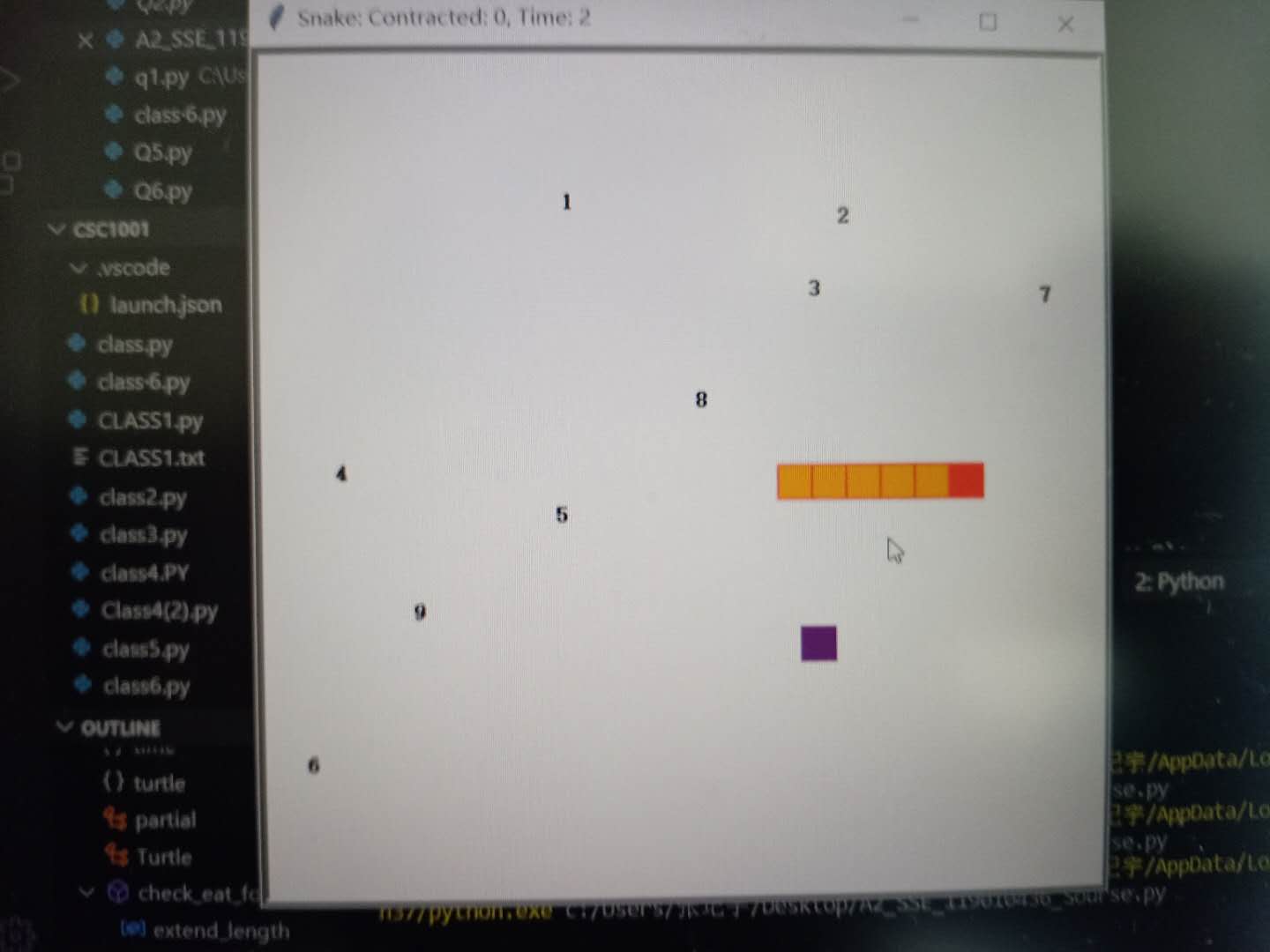
1. **Output**
2. **Winner:**

****

1. Game over:



1. No food is consumed:



1. Three foods are consumed:

