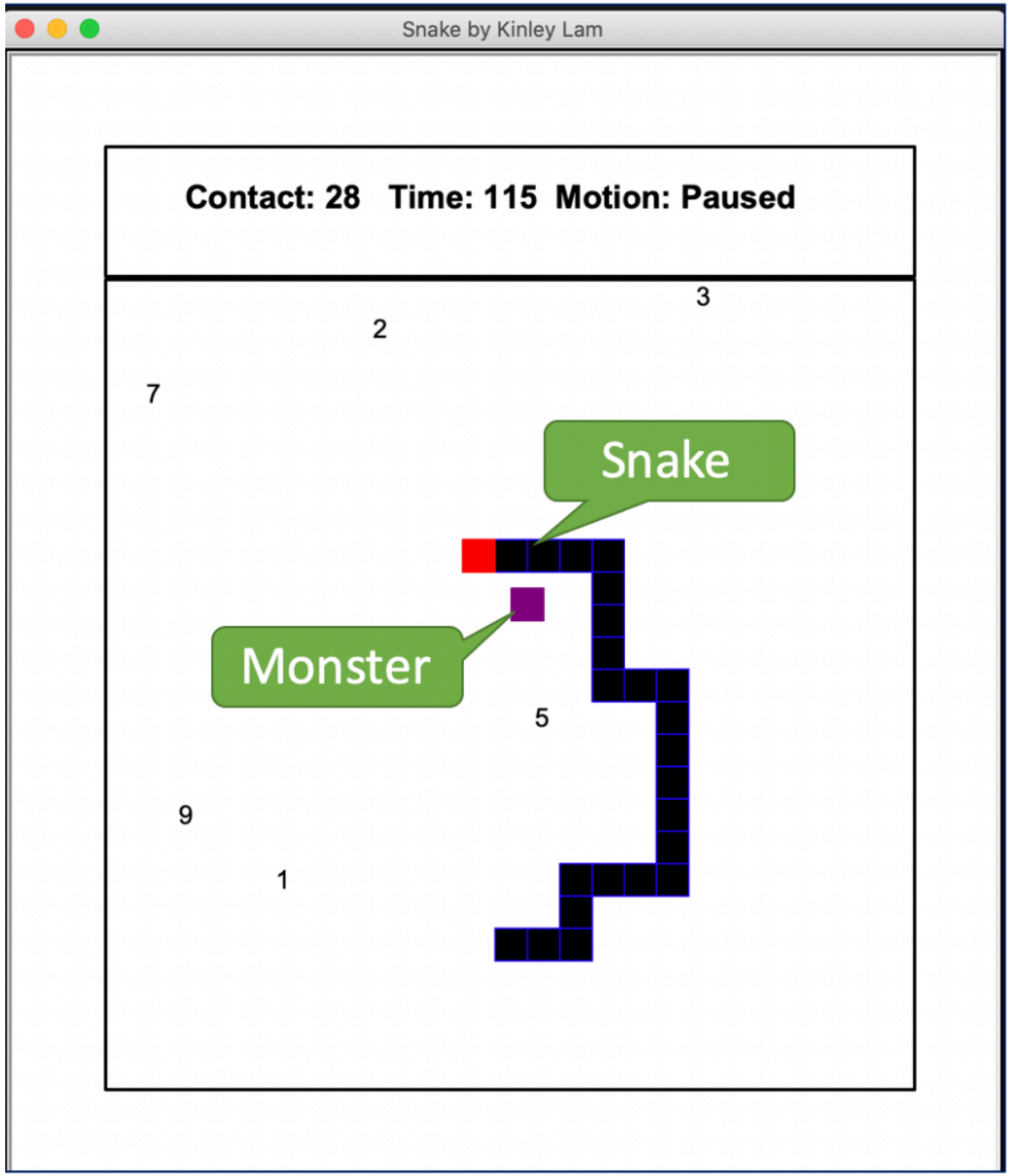
CSC 1002 Week 11 & 12

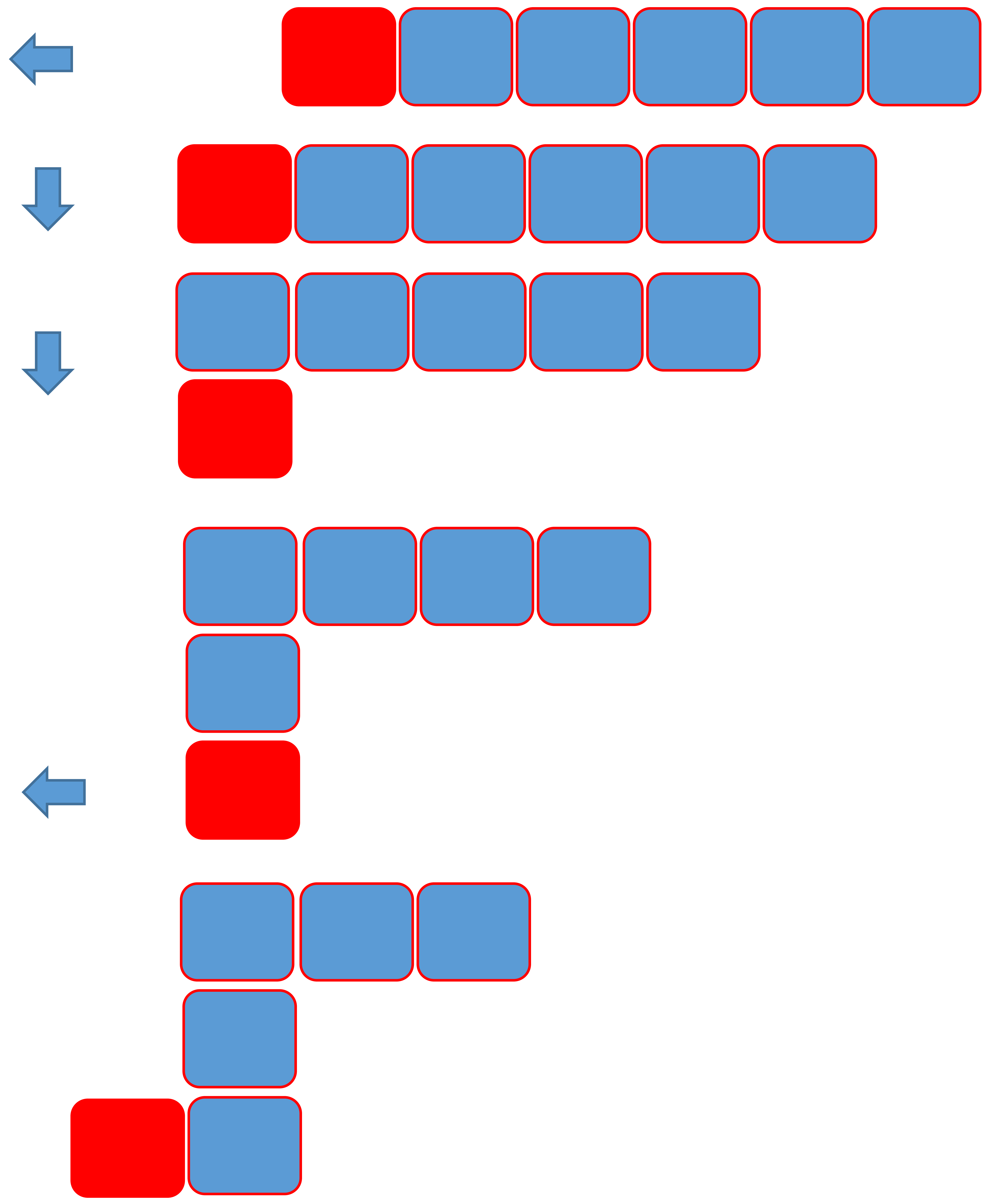
Snake Design

Snake - 2023


Snake by Kinley Lam
Contact: 28   Time: 115 Motion: Paused
3
2
7
Snake
Monster
5
9
1

Snake – design

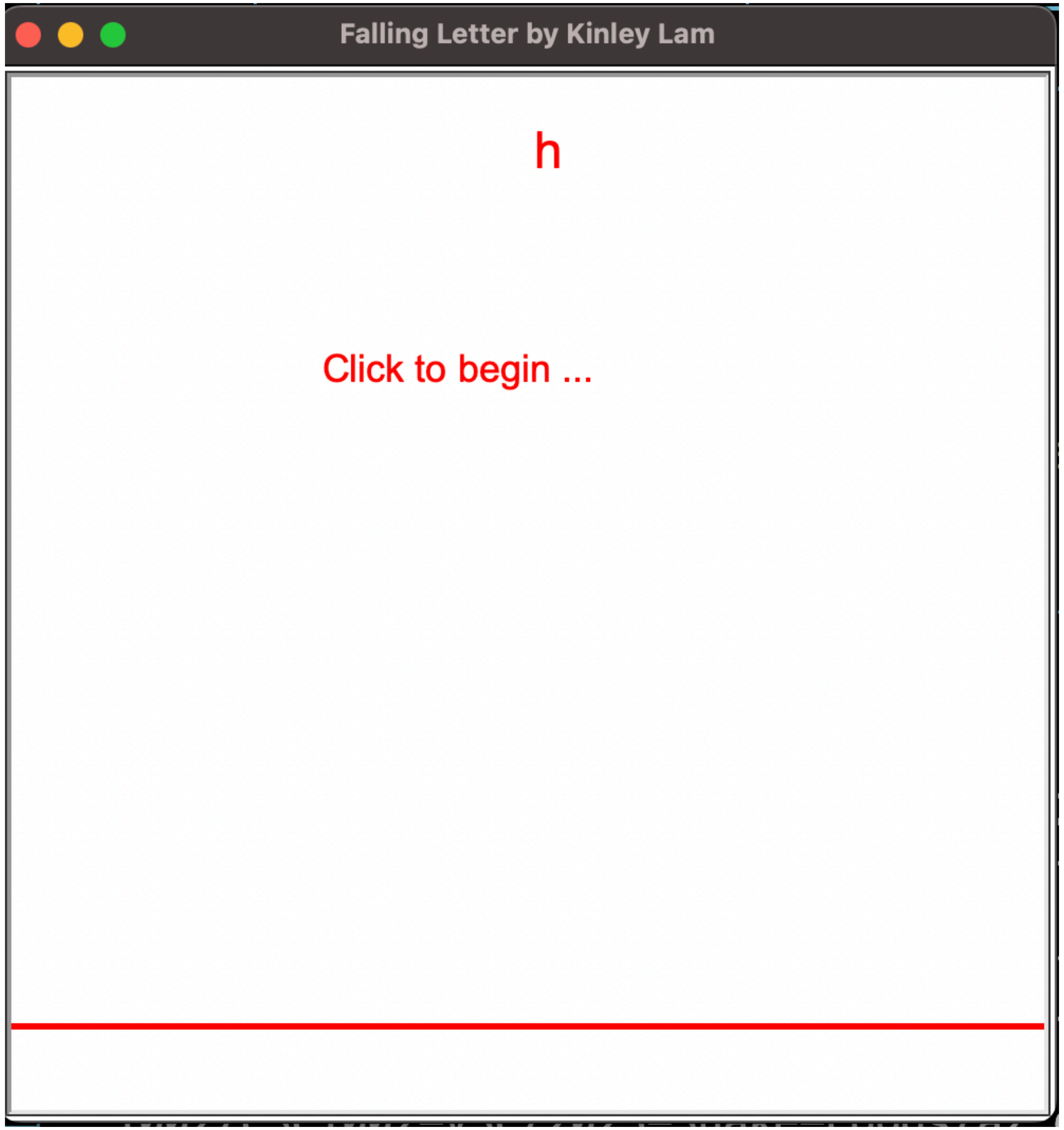
- Screen Layout – Border, Status, Motion Area
  - use a turtle object with transparent body color (blank string “”) and solid border, have it stretched to match the size of the desired dimension, or draw the borders with goto/fd()
- Handling of user interaction – Mouse Click, Keyboard, Turtle Graphics
- Motion Refresh – auto vs manual (sync update via timer)
- Food Items – alignment of the food items with the head of the snake, plus avoiding overlapping among food items
- Monster – motion towards the snake, plus its alignment with the snake
- Snake – motion of the snake + body, as well as its tail extension
- Detection of overlapping and direction between 2 turtle objects
- Game time – tracking total game time via “time” module



Screen Events: Key Press & Mouse-Click

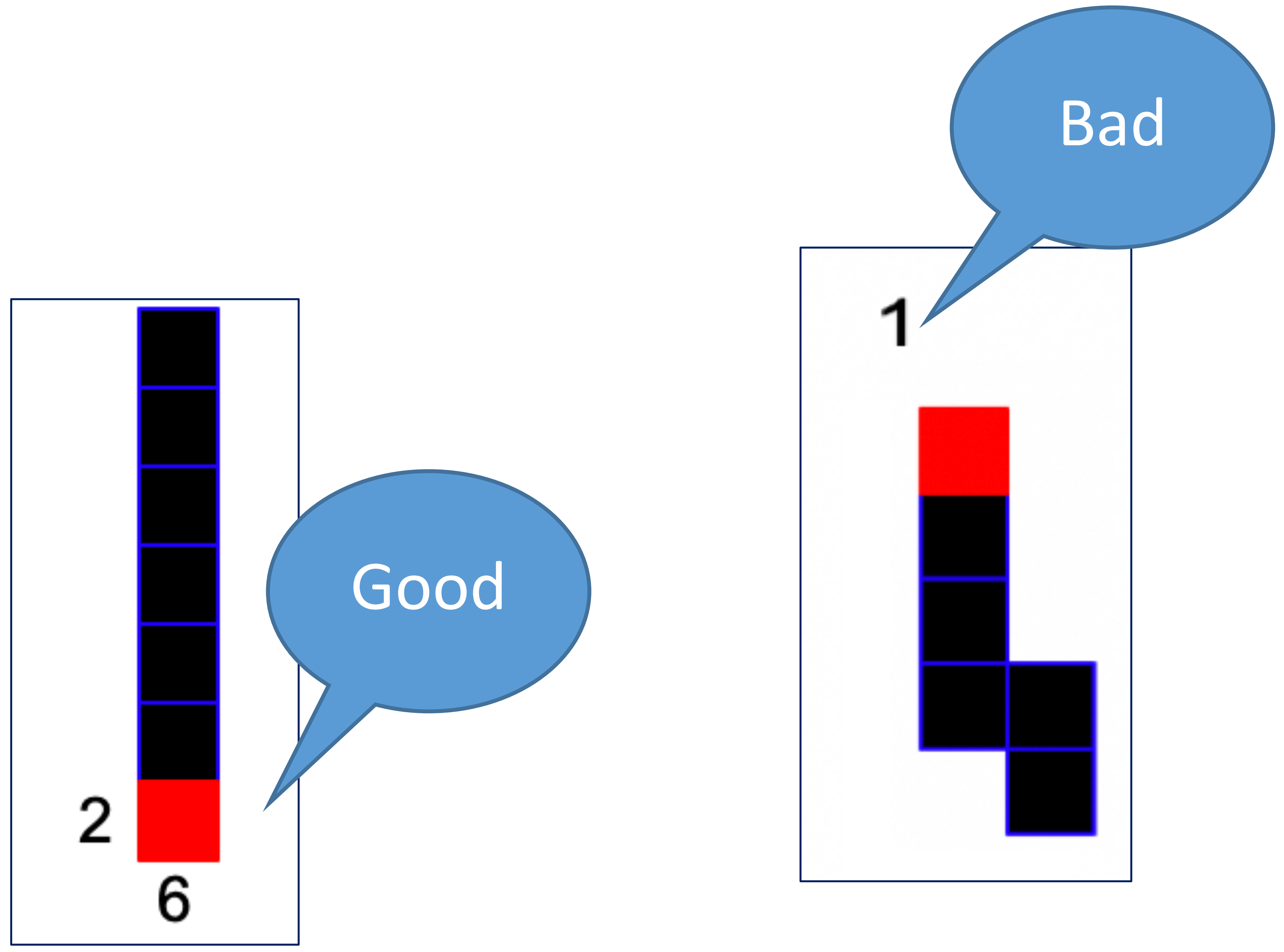
- The onkey(func, key) function of the screen object binds a function to a keyboard key:
  - Non-printable keys: “Return”, “Up”, “Down”, “Left”, “Right”, “space”, “Tab”
  - Printable keys: pass to the function as it is, ex: “a”, “1”, “K”, ….
- The onscreenclick()/onclick() binds/unbinds a function to a mouse-click event against the turtle graphics’ canvas or turtle’s object.
- call listen() to enable key event process onkey()
- place listen() before mainloop()

Key Press Demo

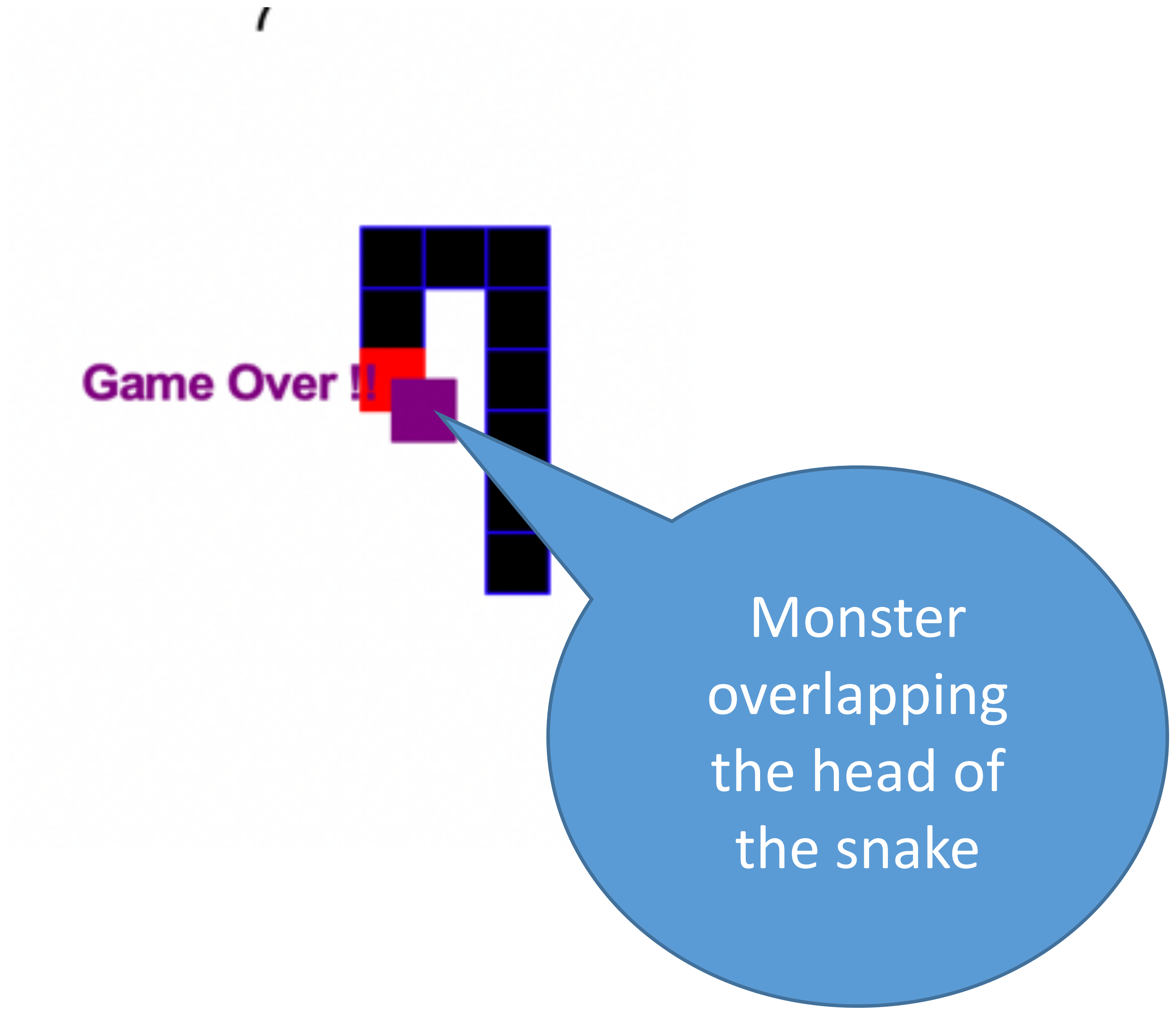


Alignment (Food Items, Snake, Monster)

- Food Items vs Snake



- Snake vs Monster



Displaying text (Turtle)

- turtle graphics doesn’t provide a way to write a text string on the screen freely
- need a turtle object and use write() to “draw” a text string where the turtle is.
  - write(string, move, align, font)
- calling write() again with a blank string at the same position will not erase the text
- either calling undo() to undo the last turtle action or invoking clear() which clears out ALL the drawing content (lines, shapes, string, etc) for that turtle
- use hide() to hide the turtle shape, if needed

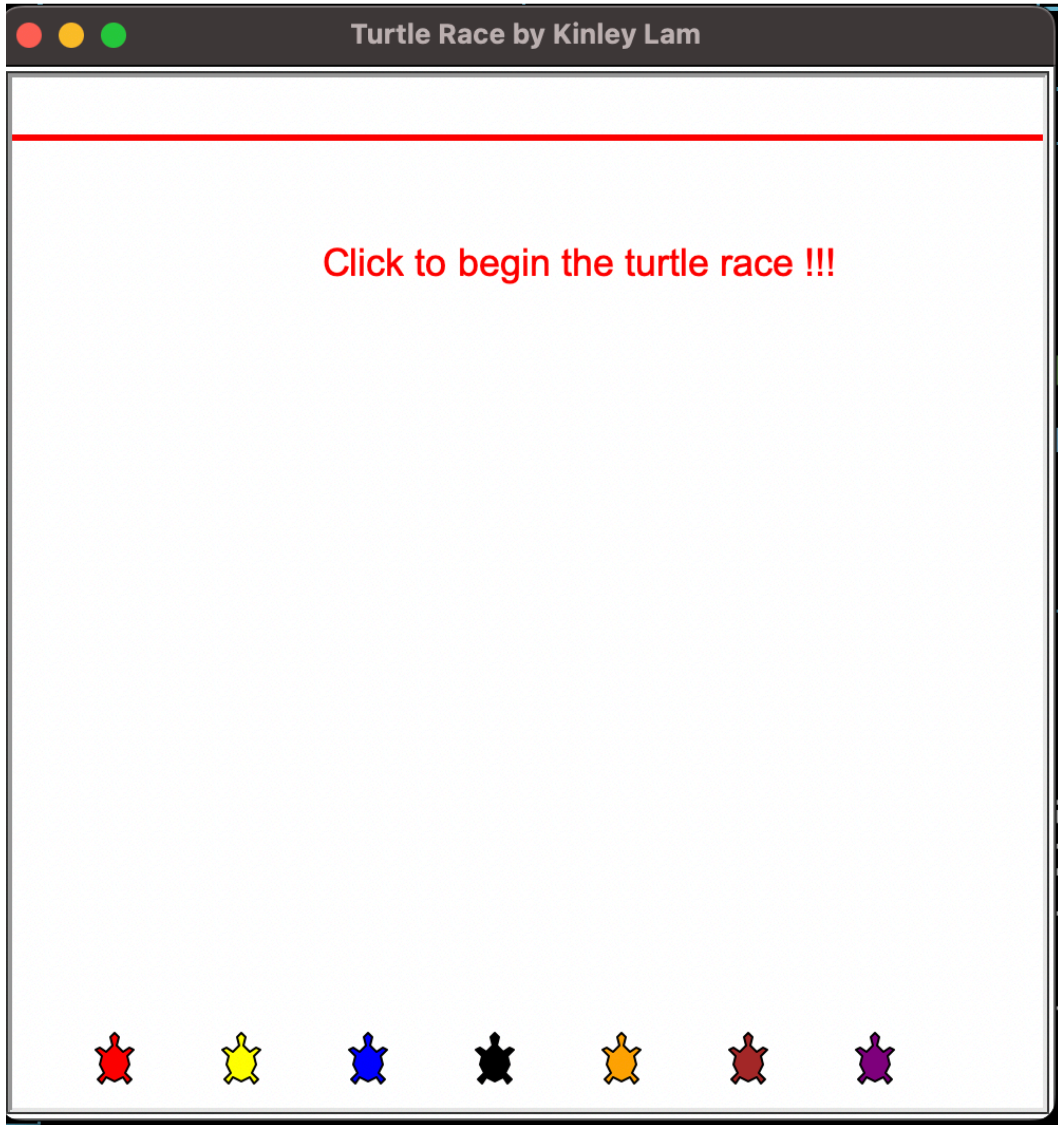
Alignment Demo

0

Timer Event (Screen)

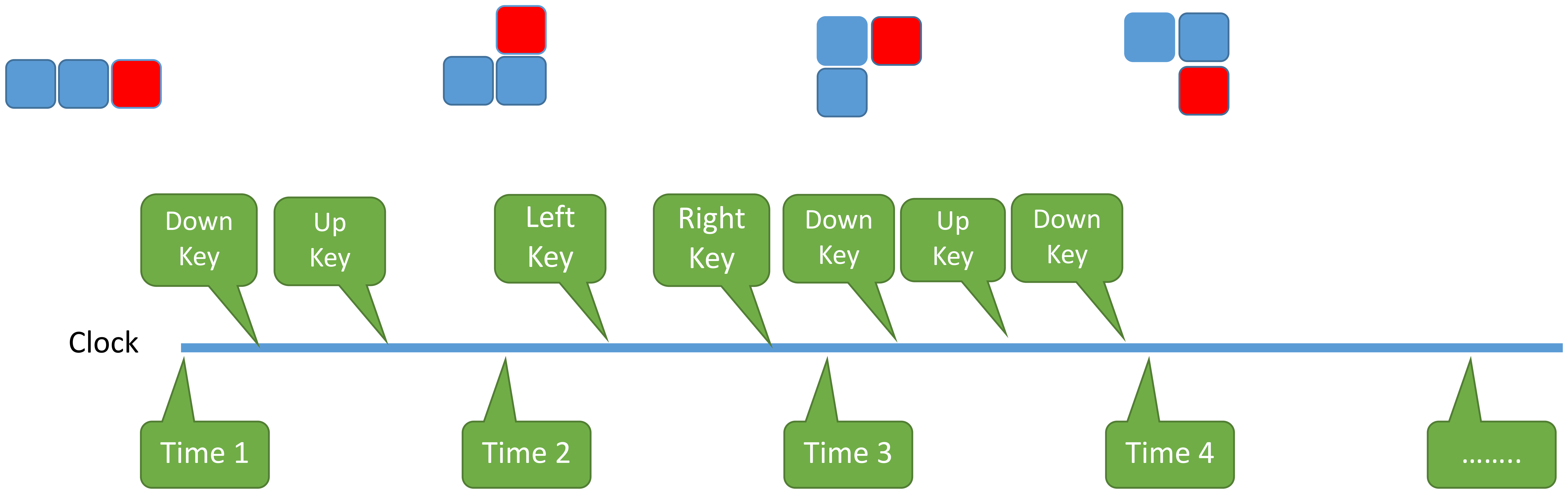
- use ontimer(func, delay) to call a function after a fixed delay (in milliseconds)
- the ontimer() is set for one time call only
- you need to call ontimer() within the function to repeat the event if needed

Timer Demo


Turtle Race by Kinley Lam

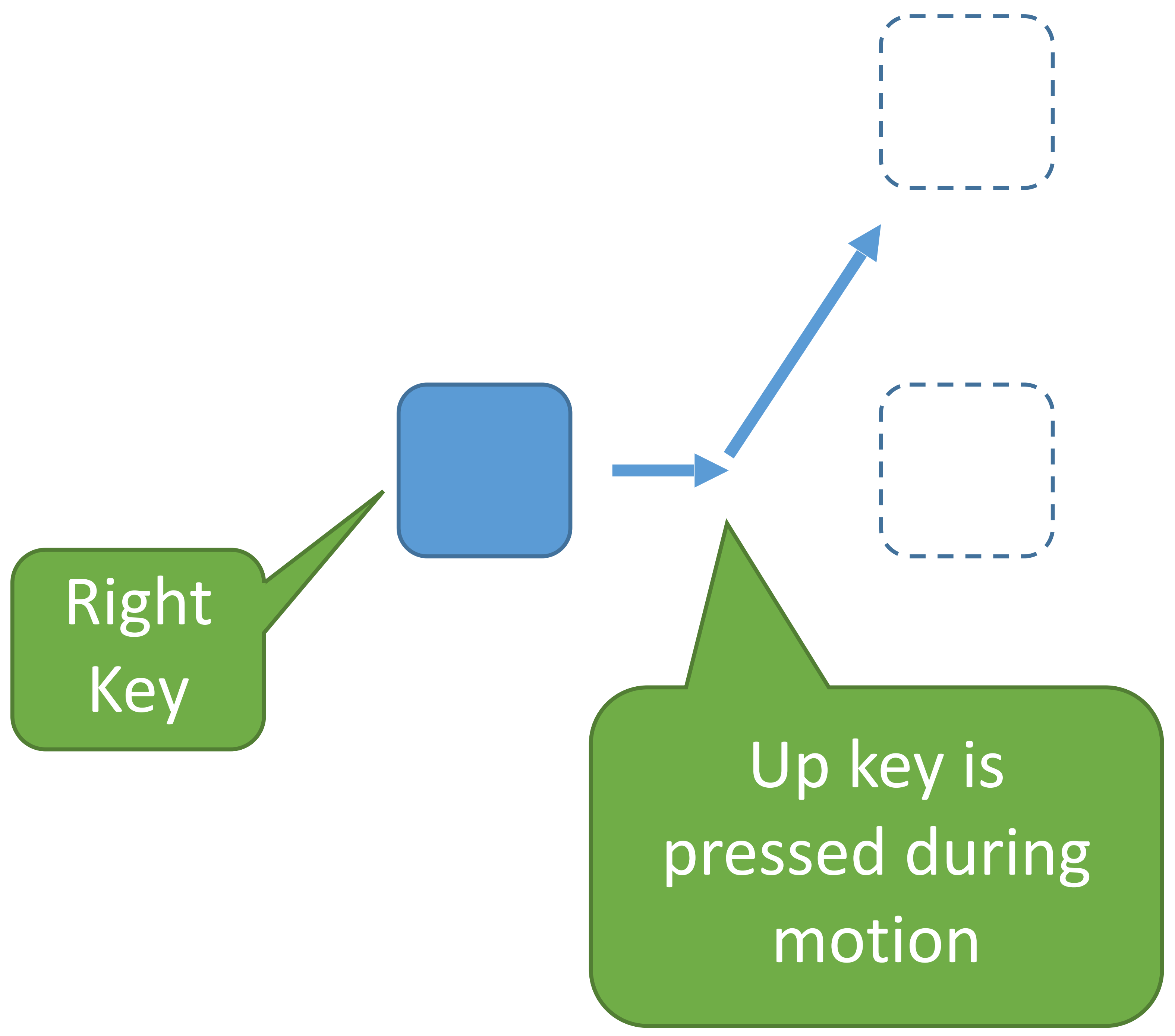
Click to begin the turtle race !!!

Manual Refresh

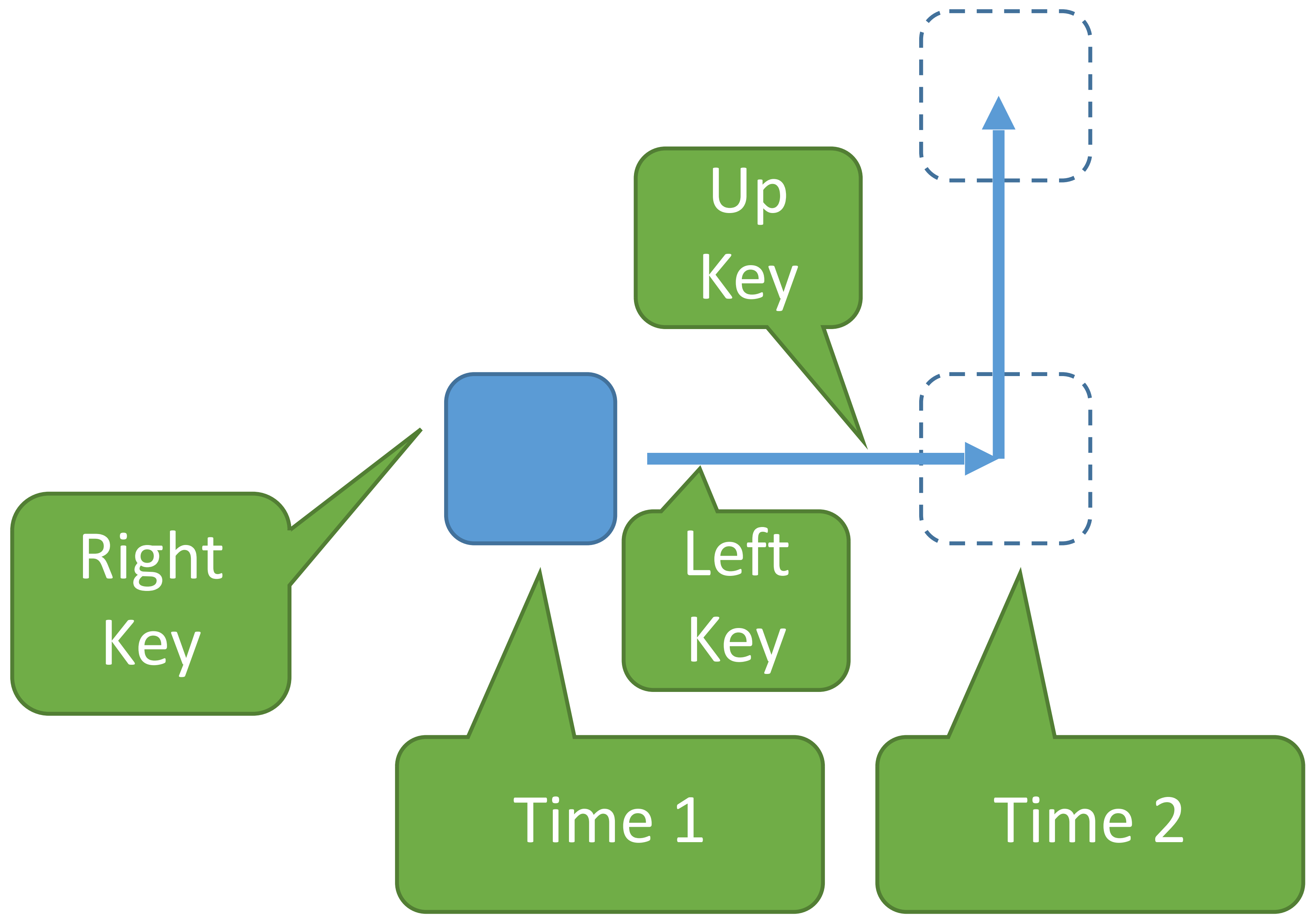


Motion Refresh (right then up)

- Auto



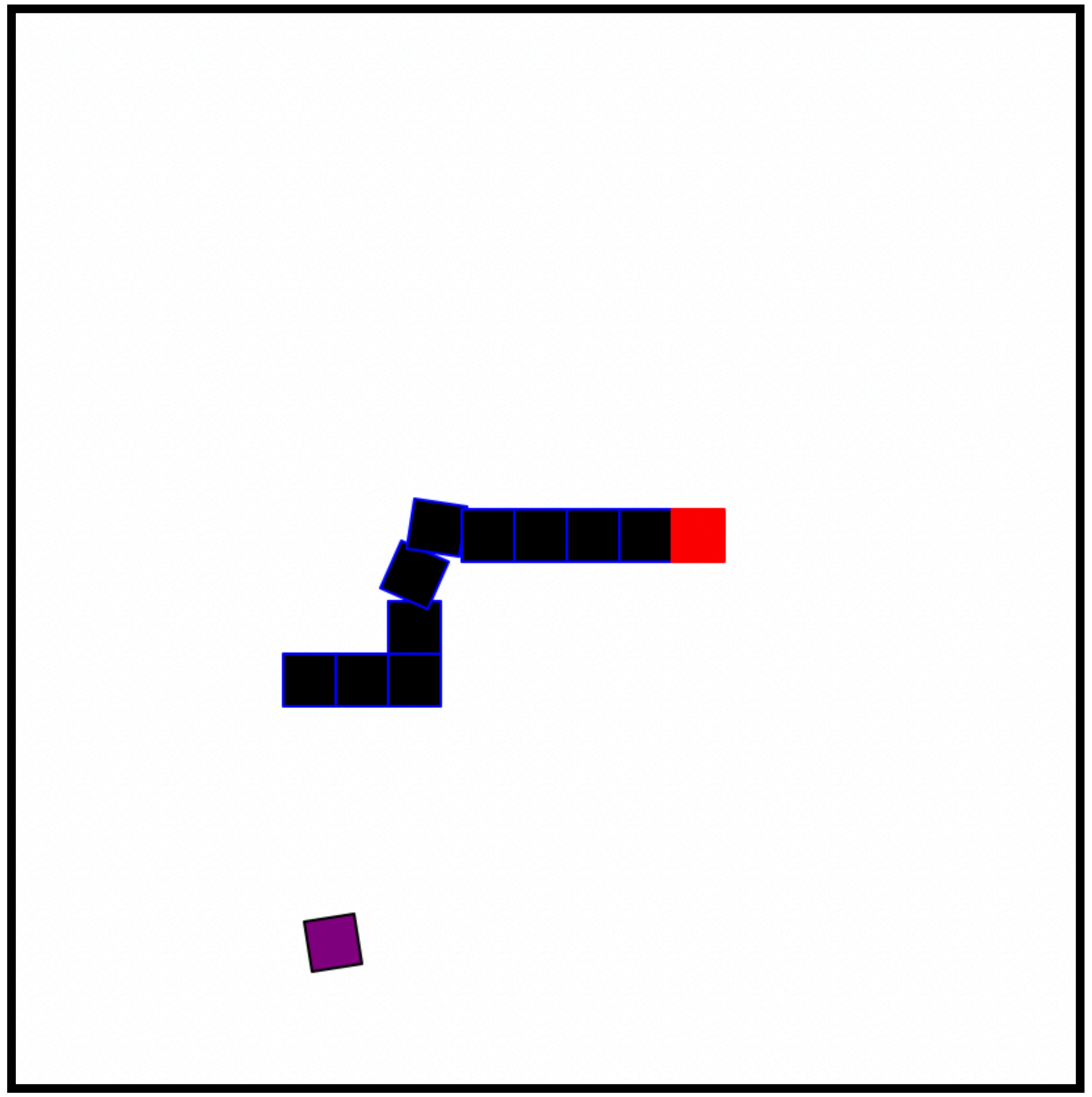
- Manual



Timer Event (Screen)

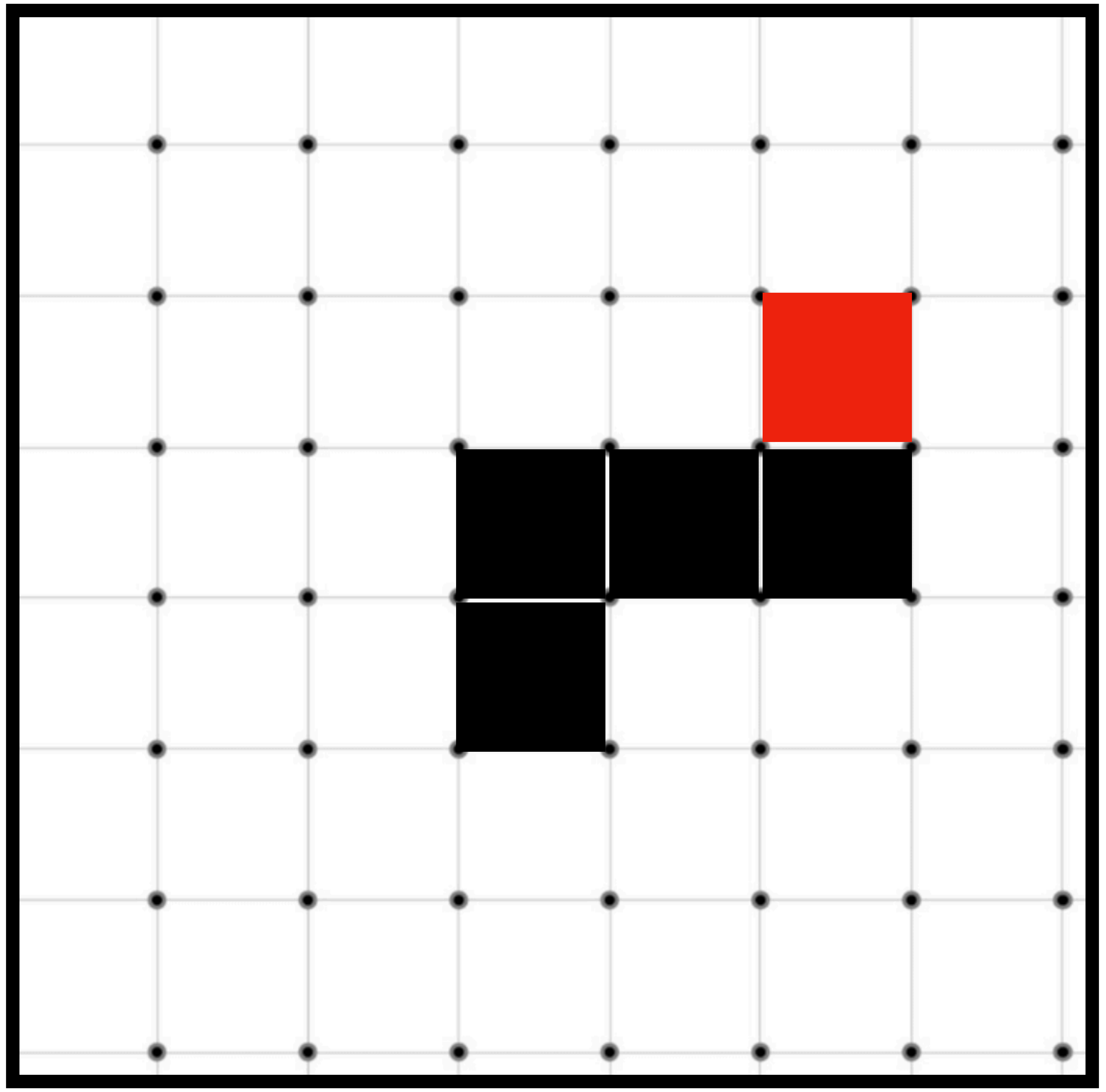
- use ontimer(func, delay) to call a function after a fixed delay (in milliseconds)
- the ontimer() is set for one time call only
- you need to call ontimer() within the function to repeat the event if needed

Timer Demo - Snake

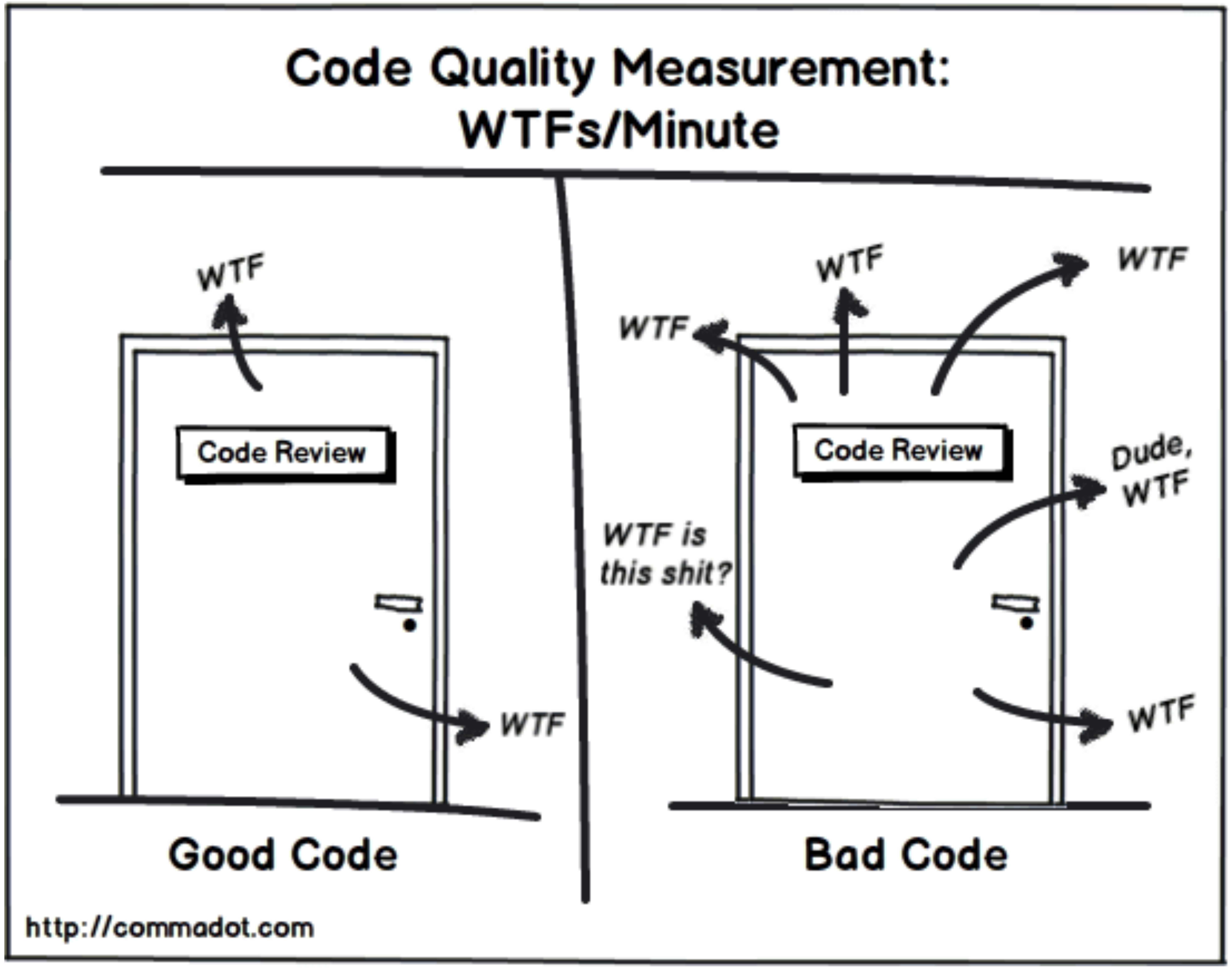


Tail Implementation - Design

Grid Approach - Cell Tracking



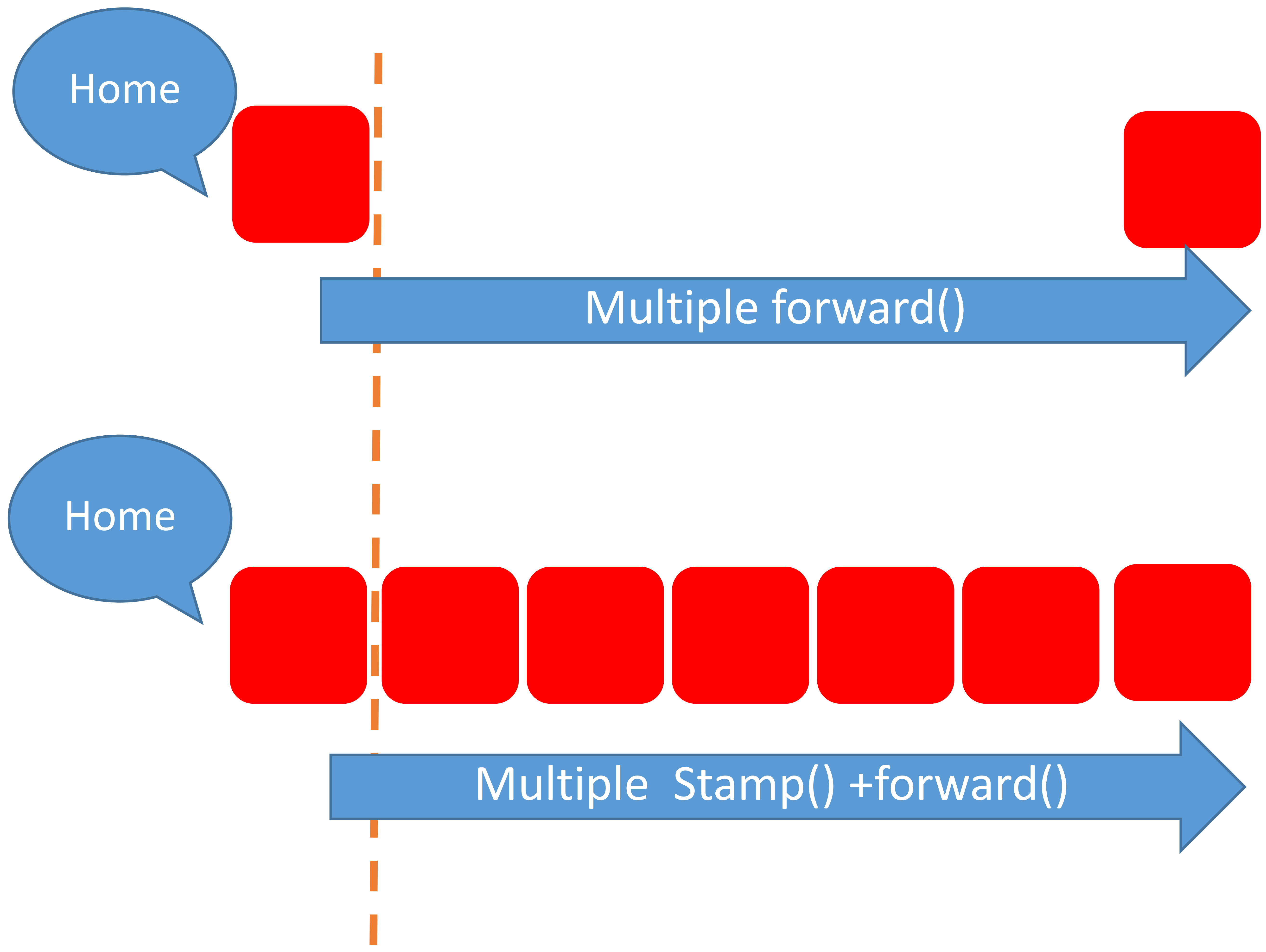
Goal #2 - Clean Code



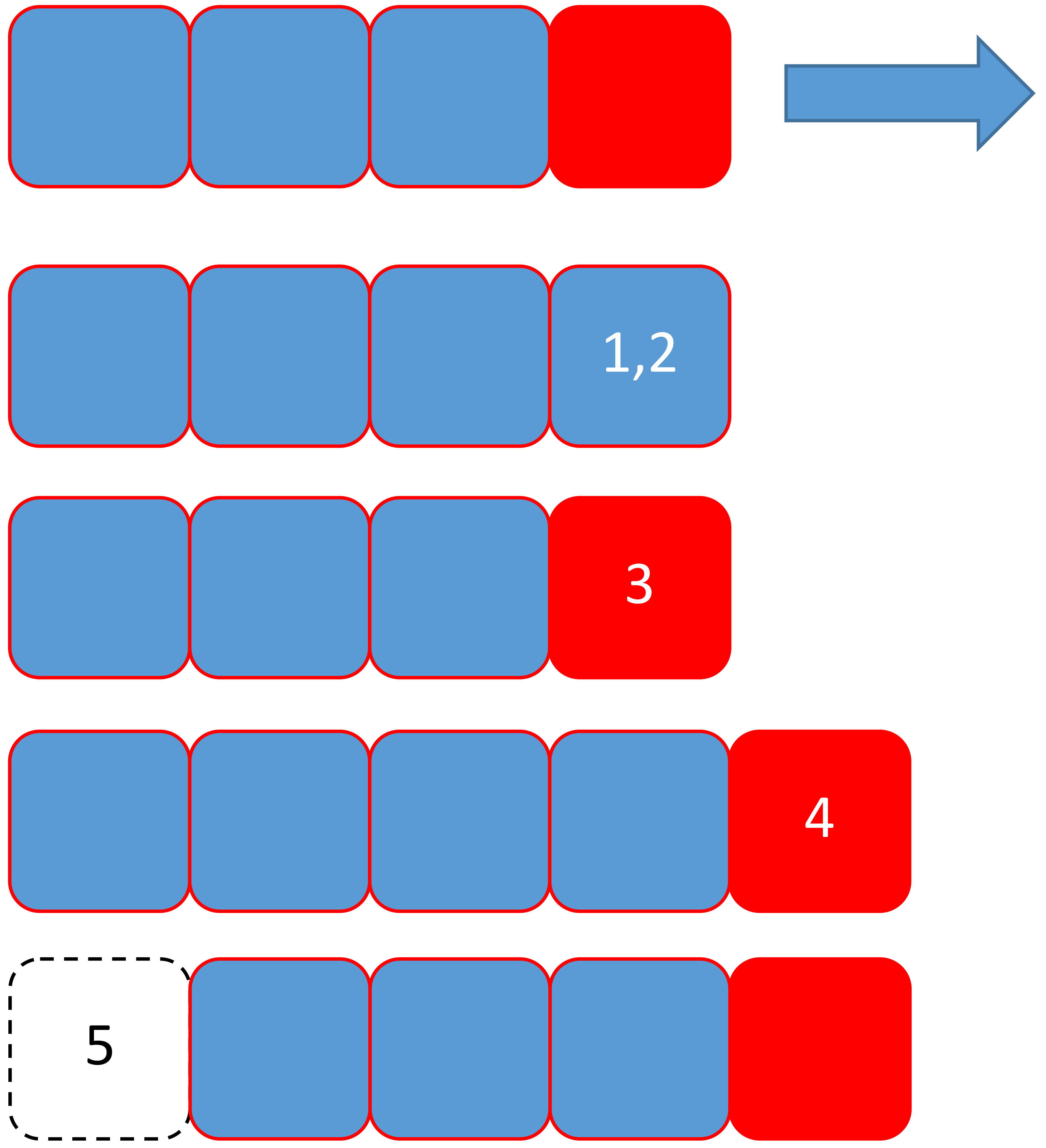
Stamp – copy of turtle shape (Turtle)

- stamp() saves a copy of the turtle shape at the current position
  - the copy is called stamp
  - stamp() will return a stamp id of the saved copy
  - each turtle maintains a list of stamps in the order that the stamp() was invoked
- clearstamp(stamp_id) is used to remove a specific stamp
- clearstamps() is used to remove all stamps
- clearstamps(x) removes first/last x stamps:
  - when x > 0, it removes the first X stamps
  - when x < 0, it removes the last X stamps
- stampItems is a list containing all stamp IDs
- NOTE: clone() duplicates the turtle object including all the properties (size, position, heading, …etc)

Stamp() vs forward()

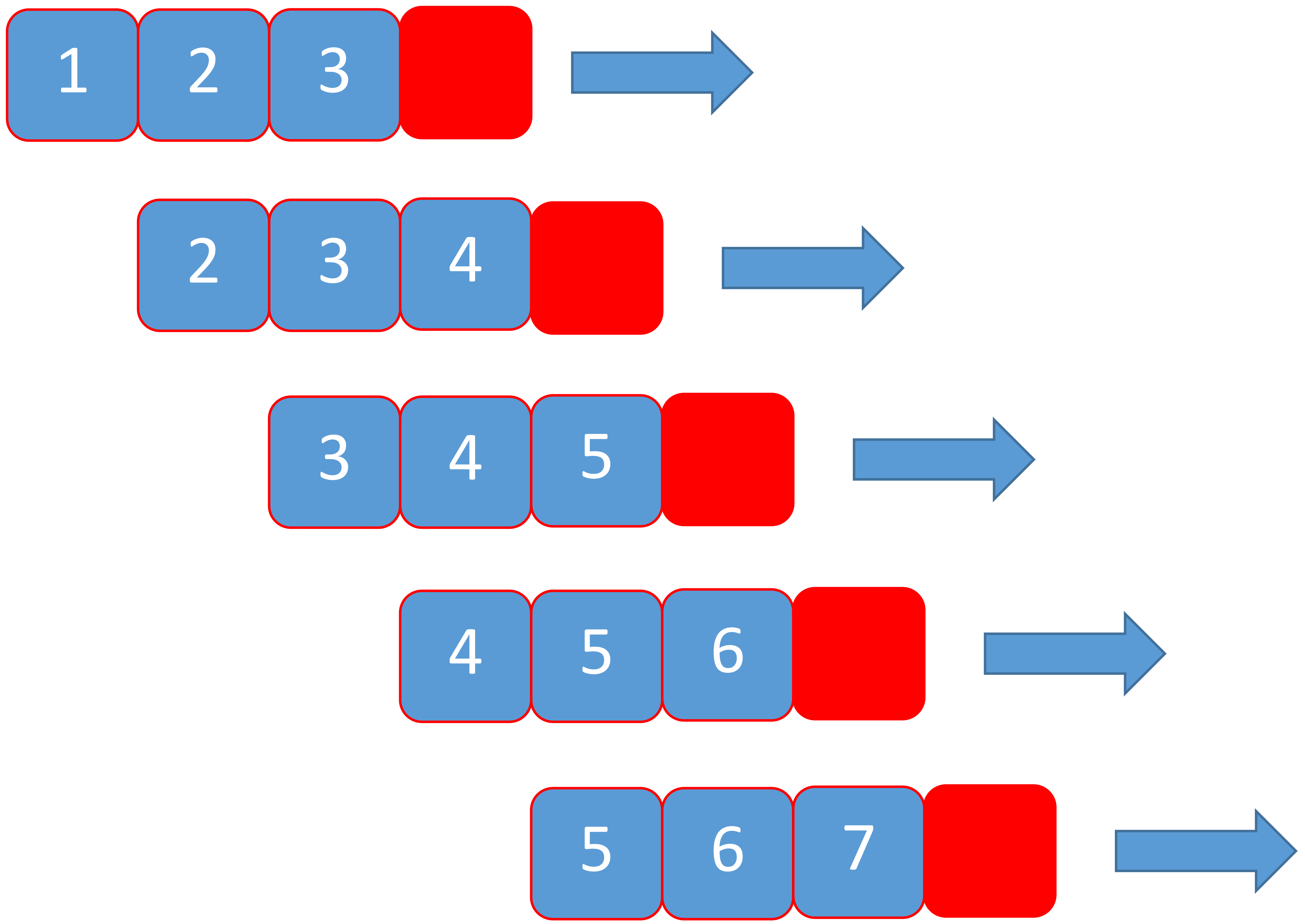


Snake motion via stamp()



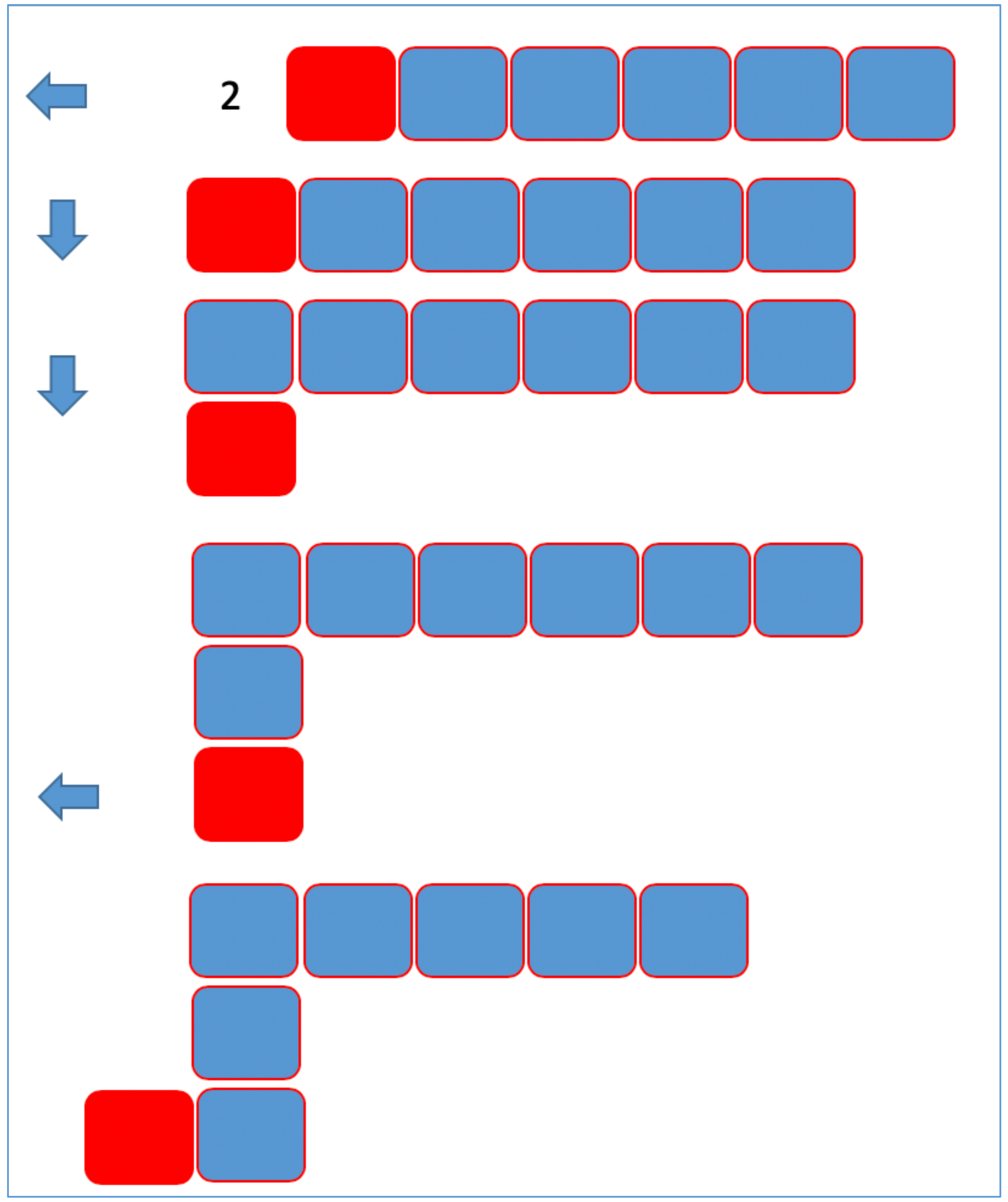
1. Convert turtle to body color
2. Call stamp() to duplicate the turtle shape
3. Restore color
4. Advance turtle
5. Delete the oldest stamp

Snake motion via stamp()

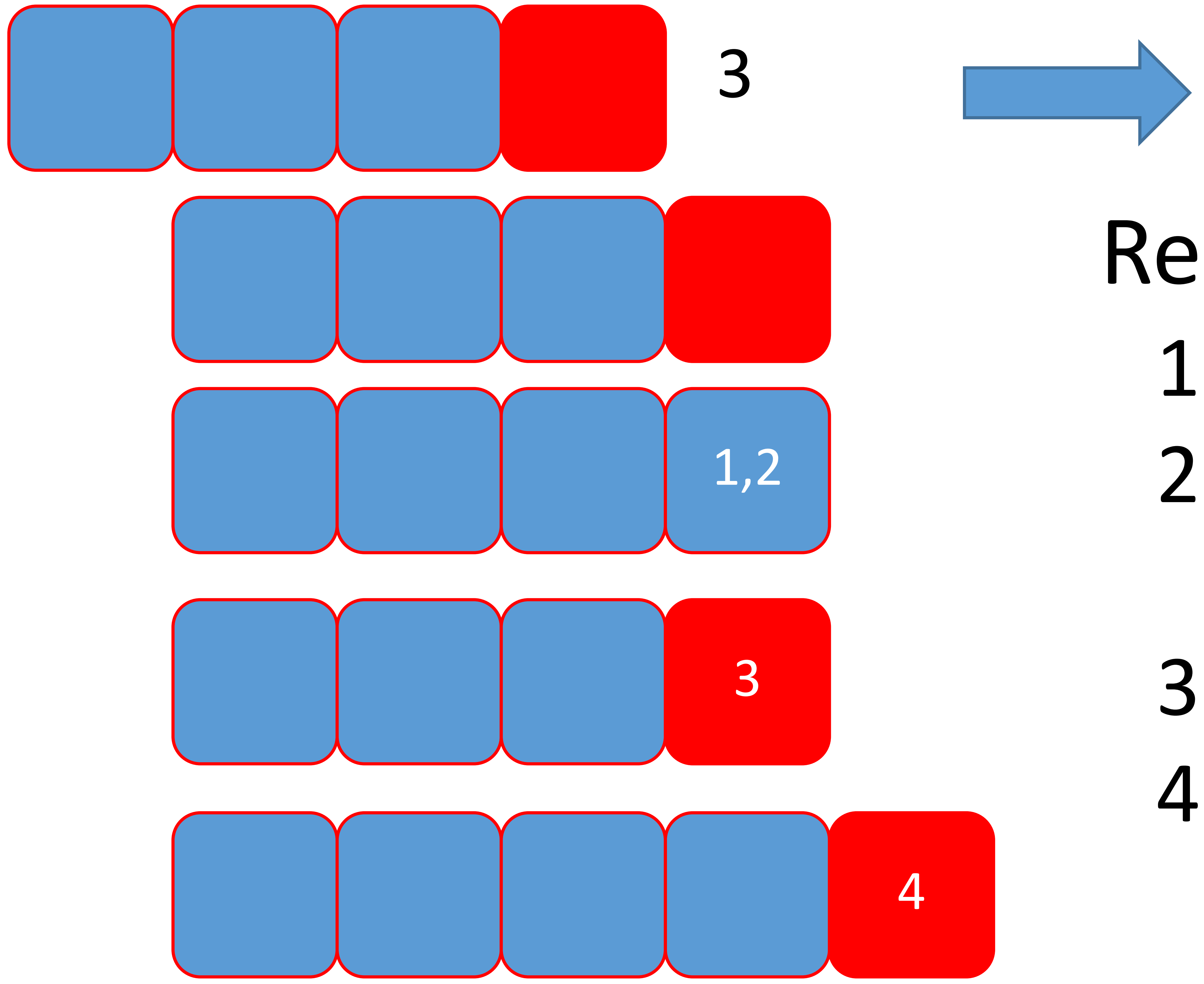


Snake Motion Demo

Snake extension



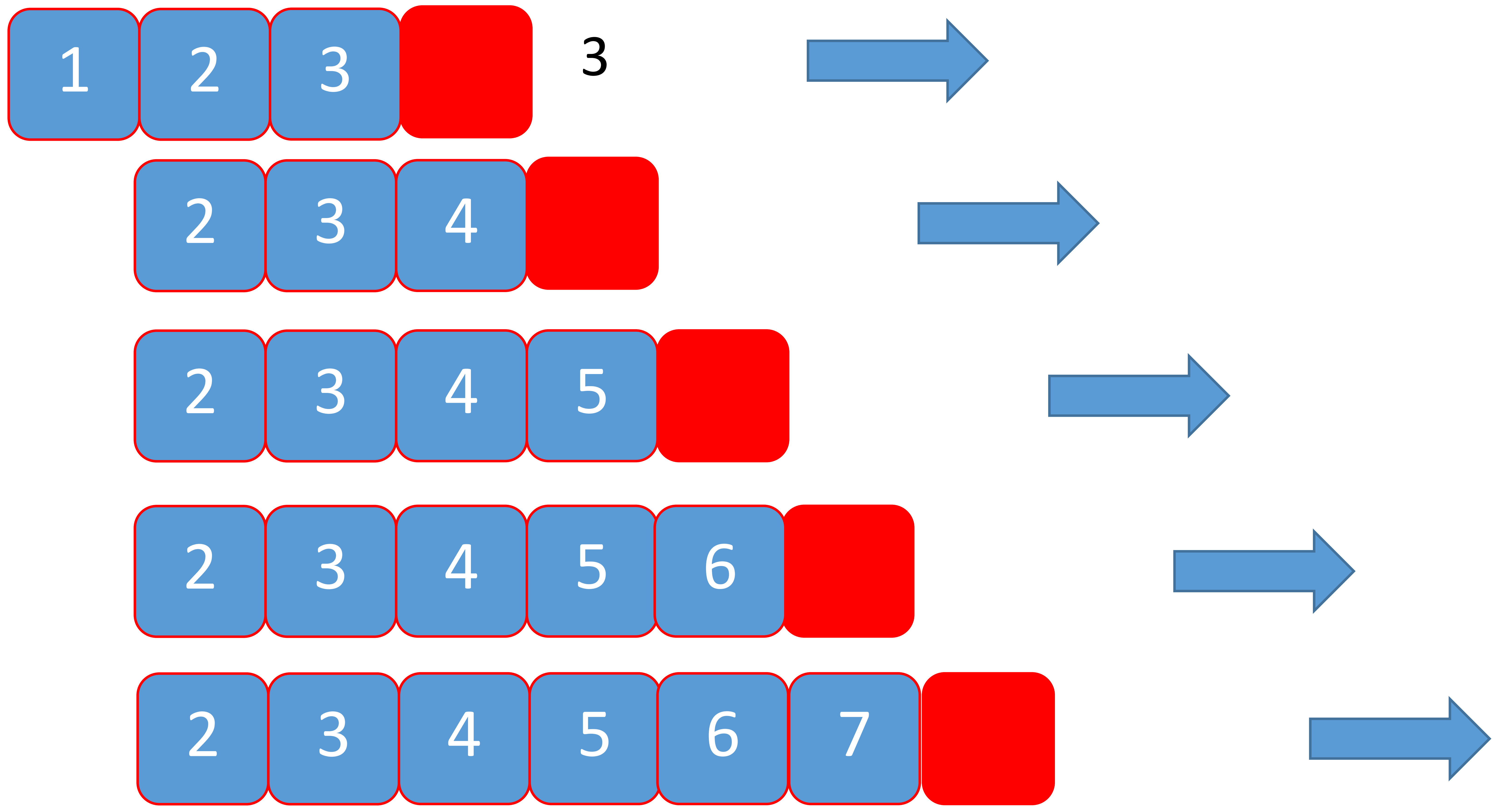
Tail extension via stamp()



Repeat until fully extended

1. Convert turtle to body color
2. Call stamp() to duplicate the turtle shape
3. Restore color
4. Advance turtle

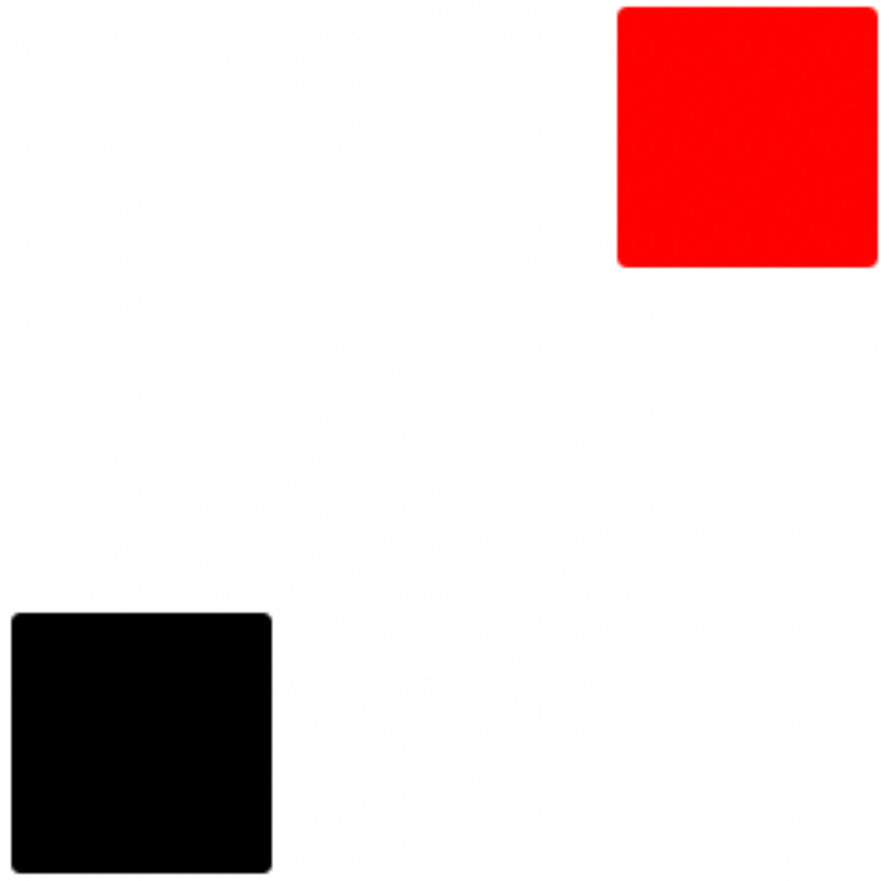
Tail extension

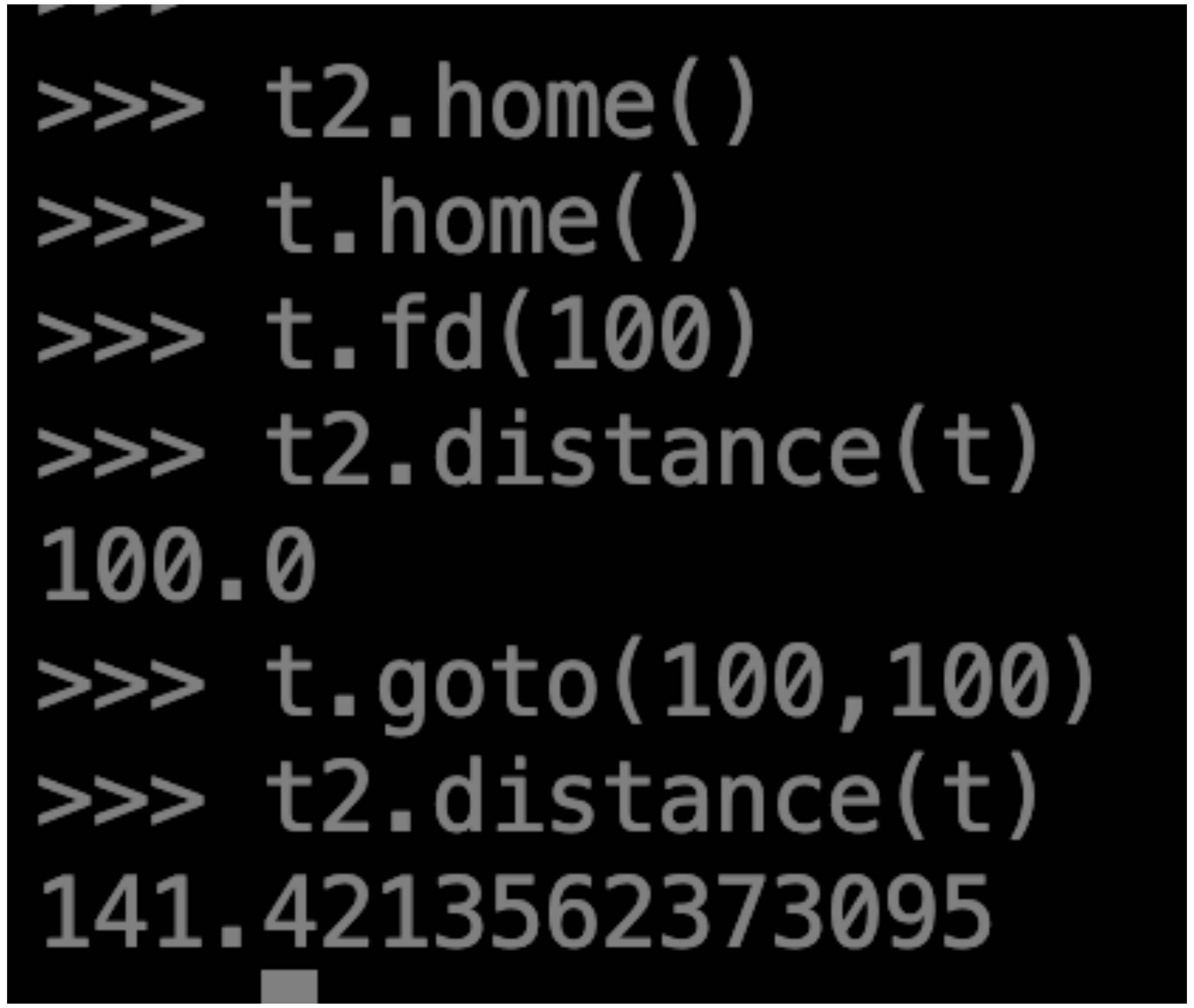


Tail Extension Demo

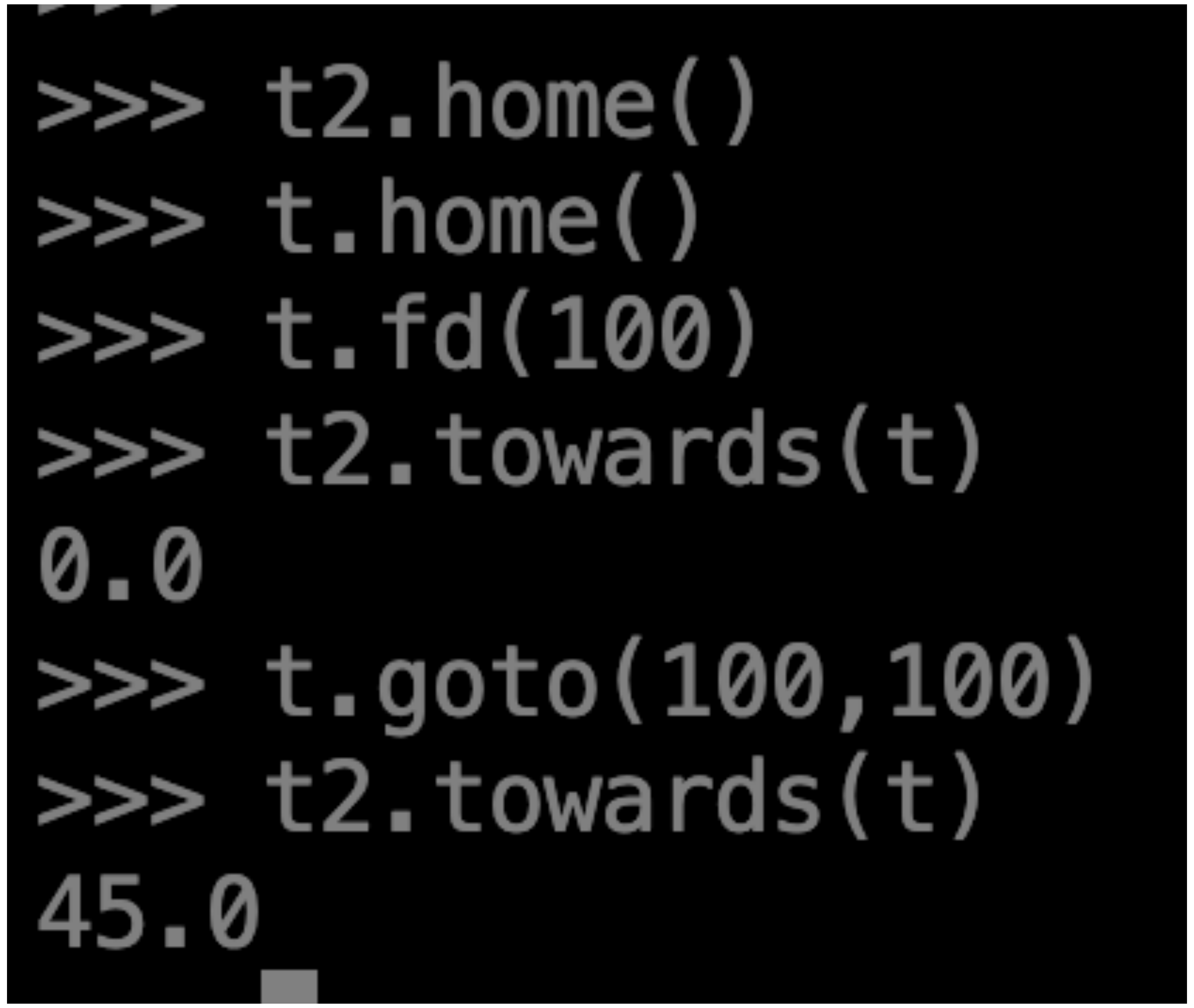
Others

Distance and Towards between 2 turtles



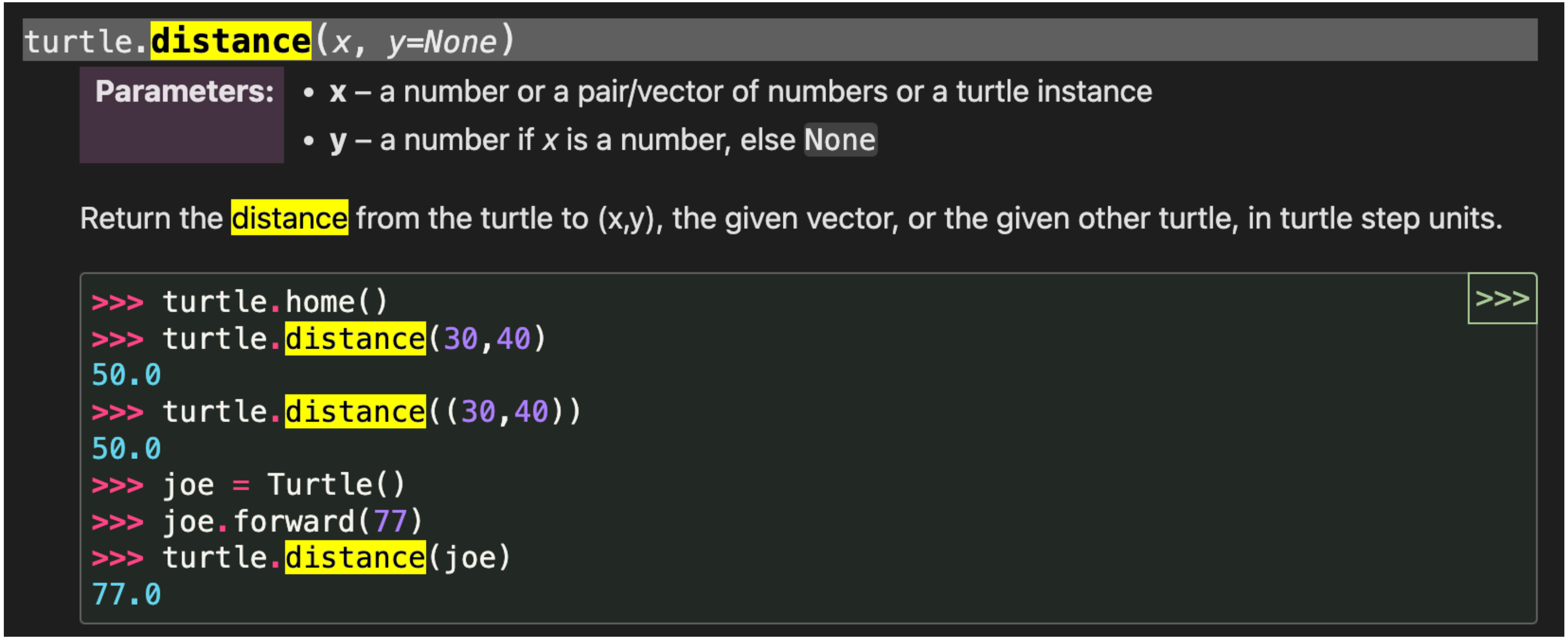


```
>>> t2.home()
>>> t.home()
>>> t.fd(100)
>>> t2.distance(t)
100.0
>>> t.goto(100,100)
>>> t2.distance(t)
141.4213562373095
```



```
>>> t2.home()
>>> t.home()
>>> t.fd(100)
>>> t2.towards(t)
0.0
>>> t.goto(100,100)
>>> t2.towards(t)
45.0
```

distance(x, y=None)



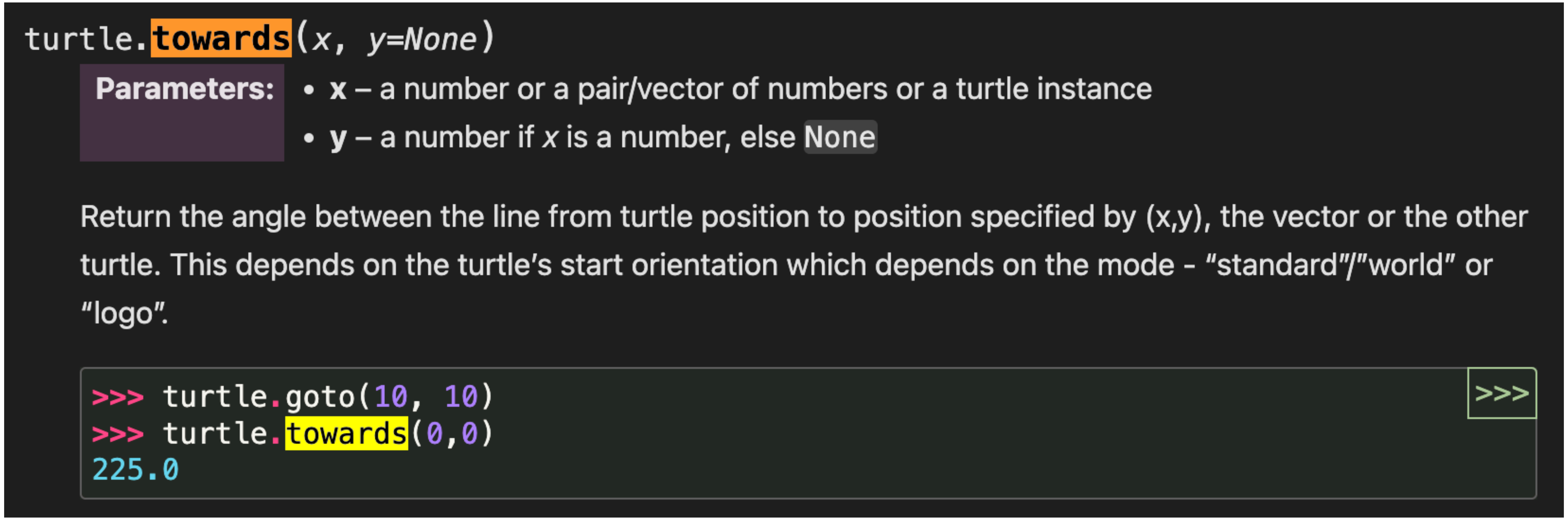
```
turtle.distance(x, y=None)
```

Parameters:
- **x** – a number or a pair/vector of numbers or a turtle instance
- **y** – a number if *x* is a number, else `None`

Return the distance from the turtle to (x,y), the given vector, or the given other turtle, in turtle step units.

```
>>> turtle.home()
>>> turtle.distance(30,40)
50.0
>>> turtle.distance((30,40))
50.0
>>> joe = Turtle()
>>> joe.forward(77)
>>> turtle.distance(joe)
77.0
```

towards(x, y=None)



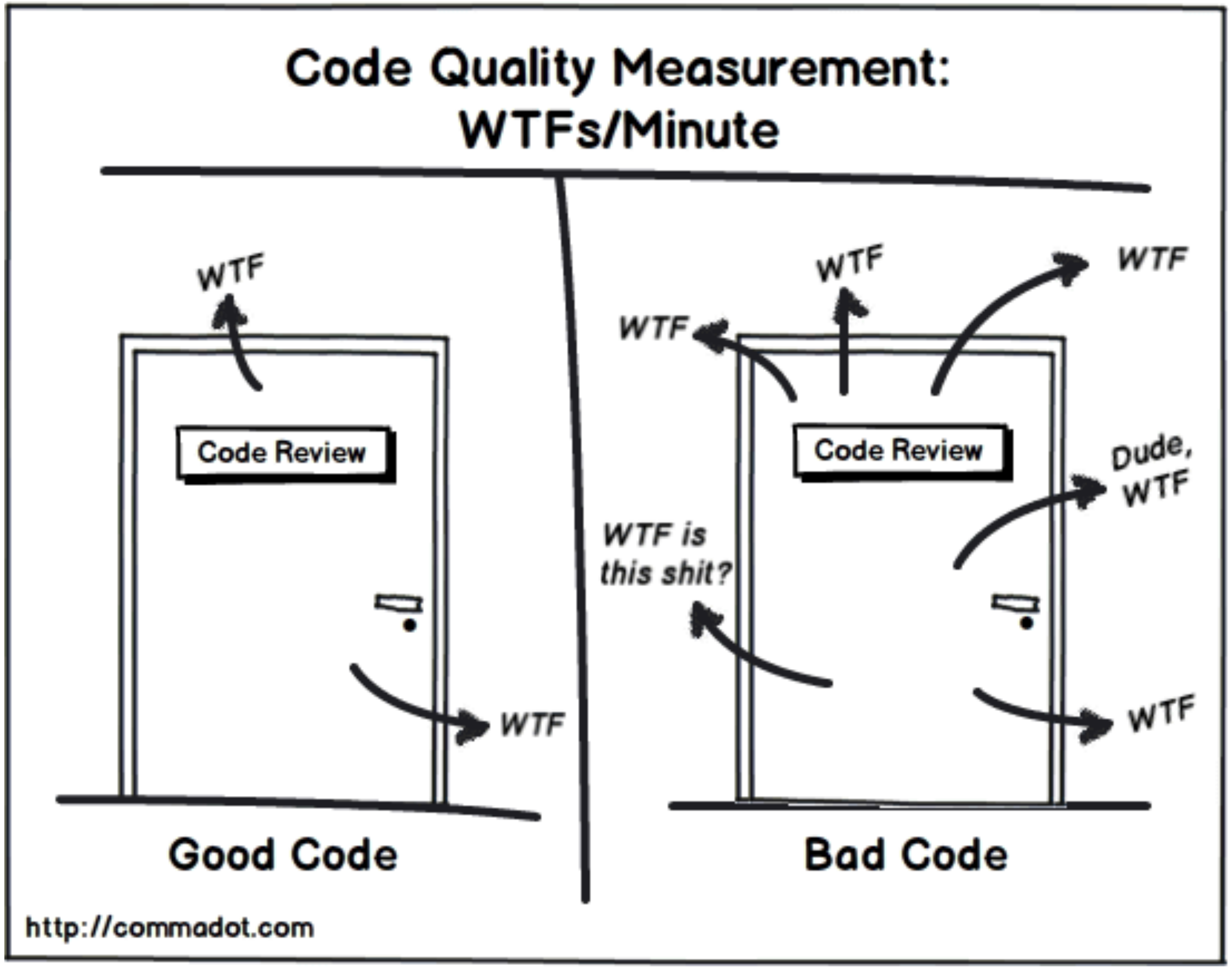
turtle.towards(*x*, *y=None*)

Parameters:
- **x** – a number or a pair/vector of numbers or a turtle instance
- **y** – a number if *x* is a number, else None

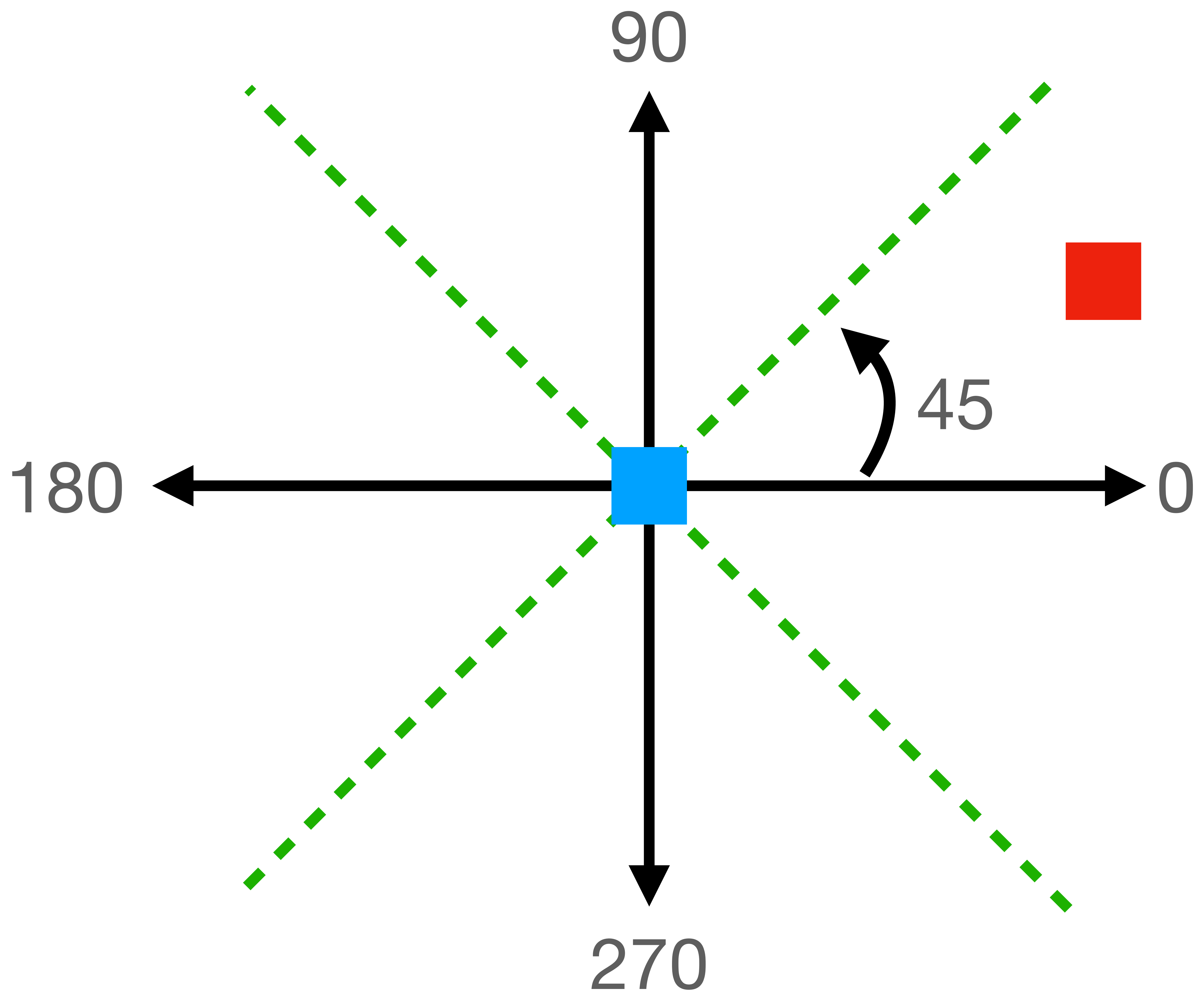
Return the angle between the line from turtle position to position specified by (x,y), the vector or the other turtle. This depends on the turtle’s start orientation which depends on the mode - “standard”/”world” or “logo”.

```
>>> turtle.goto(10, 10)
>>> turtle.towards(0,0)
225.0
```

Goal #2 - Clean Code



heading vs Angle



heading	Quadrant	Angle (A)
0	0,7	A <= 45 or A > 315
90	1,2	45 < A <= 135
180	3,4	135 < A <= 225
270	5,6	225 < A <= 315

The End – Thank You