YEAH Hours: Assignment 4

Elina Thadhani

Part 1: Lists

Goal: Create a list of lists!

Here you are given two input lists of the **same** length. The task is to make “mini” lists with the paired contents of each list and append those to a **new** list.

zip2lists(['a', 'b', 'c'], ['d', 'e', 'f'])
should return the following new list (of lists):

[['a', 'd'], ['b', 'e'], ['c', 'f']]

To iterate through a list: use:

For i in range(len(list)):

    Element = list[i]

Part 2: Sunset

GOAL: Animate the setting sun!

1. The sun should start at top of the screen, centered horizontally, and animate down. Each frame the sun should move one pixel down and the animation should pause for 1/50th of a second (DELAY).
2. The sky is **'blue'**.
3. The sun is initially **'yellow'**.
4. When the middle of the sun passes ORANGE_Y it should turn **'orange'**.
5. When the middle of the sun passes RED_Y it should turn **'red'**.
6. Halt your animation loop once the sun goes off the screen.

Part 2: Syntax

Syntax:

canvas.create_rectangle(x1, y1, width, height, fill=color)

canvas.create_oval(x1, y1, width, height, fill = color)

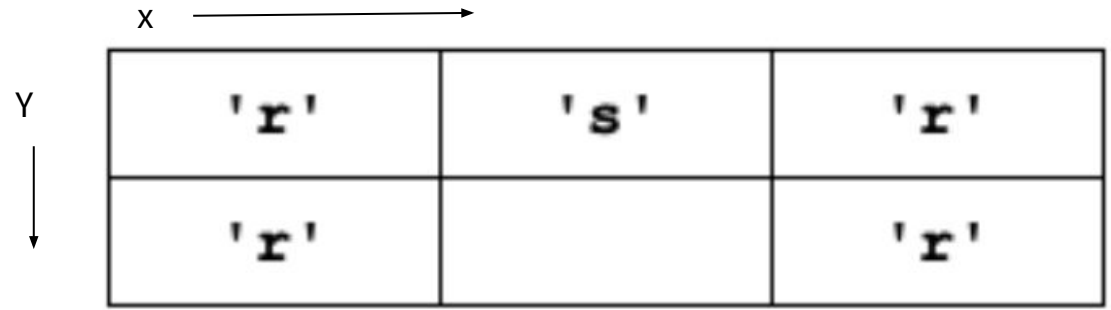
canvas.move(object, x,y)

canvas.update()

time.sleep(DELAY)

Part 3: Sand

How is the world “stored” ?

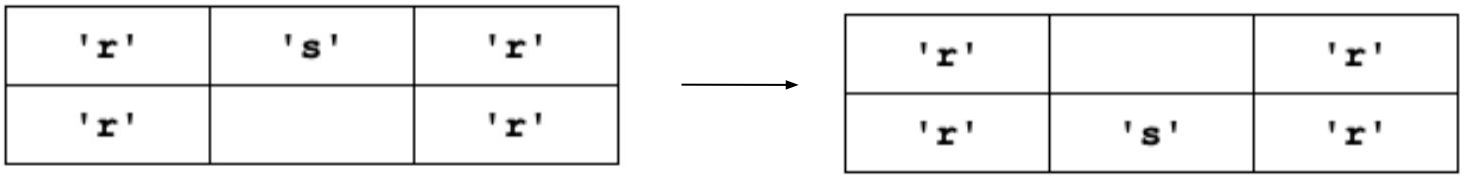


Part 3: Sand: do_move

do_move(grid, x1, y1, x2, y2):

- Goal: move whatever is at x1,y1 to the location x2,y2

do_move(grid, 1,0, 1,1) will result in:

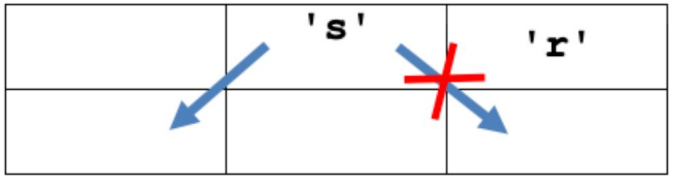


Part 3: check_move

Rule 1: The destination must be within the edges of the grid.

Rule 2: The destination square in the grid must be empty. (think of how to check what is at that location)

Rule 3: For a diagonal down-left or down-right move, the corner square must be empty (that is, it should contain None).



Part 3/4: Gravity and do whole grid

1. If there is not a sand 's' at location (x,y), do nothing, the move is over.

2. down: if the sand can move down, do it, this ends the move.

3. down-left: otherwise if the sand can move down-left, do it, this ends the move.

4. down-right: otherwise if the sand can move down-right, do it, this ends the move.

Write code for the do_whole_grid function which just calls do_gravity once for every (x,y) location in the grid. The function should return the grid when it is done

Part 6: Brownian Motion

1. Check if the square is sand. Proceed to the next steps only if it is sand. If it is not sand, the square does not have Brownian motion.
2. Create a random number in the range 0 to 99. Note that the Python function random.randrange(n) (from the random library) returns a random number uniformly distributed in the range from 0 to n – 1, inclusive. So, you could use the following call:
   ```
   num = random.randrange(100)
   ```
   Proceed to step #3 below only if num < brownian (recall that brownian is a parameter to this function). In this way, for example, if brownian is 50, we'll do the Brownian move about 50% of the time.
3. To try to move left or right, set a "coin" variable like a coin flip with the following line. This line sets coin to either 0 or 1:
   ```
   coin = random.randrange(2)
   ```
4. If coin is 0, try to move the sand at the current (x,y) location one cell to the left. If coin is 1, try to move the sand at the current (x,y) location one cell right. Use your helper functions to check if the move is possible and then move the sand if the move is legal. Don't try both directions. Based on the coin flip, you choose one direction for the sand at this location and see if it can move (and move it if it can).

Good Luck!