Laboratory 12

Random Movement of a particle

We want to demonstrate random movement of a particle. The particle starts out a the origin of a plain and each second moves to a position up, down, left, or right by one step. We want to visualize the movement of a particle over a certain time.

(1) Write a class Particle. A particle has two instance fields, an x value and a y value. Beside the dunder __str__, write a method move that updates the x and y value by randomly picking one and randomly incrementing or decrementing it.

(2) Add an instance method random_trail of one additional variable *n* that returns a list of *n* locations of a particle resulting from *n*-1 moves. This is called a random path.

Displaying in two dimensions

In Python, a two-dimensional matrix is implemented most commonly as a list of list. As an example, consider this matrix

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}.$$

We can write it as a list of list in Python

```
my_matrix = [ [1,  2,  3,  4],
              [5,  6,  7,  8],
              [9, 10, 11, 12]
            ]
```

To obtain the element in the second row and third column, we first select the second row as `my_matrix[1]` and then the third element in this row as `my_matrix[1][2]`. Notice that this is not different from Mathematics, where the corresponding element of M is denoted by $M_{1,2}$ (if Mathematicians would start counting from zero, which however they are not doing, so they would call this element $M_{2,3}$.)

We now want to use ASCII art in order to display points (with x and y coordinates). The basic idea is to create a 'view' as a matrix of single characters, initially equal to the white space `' '`.

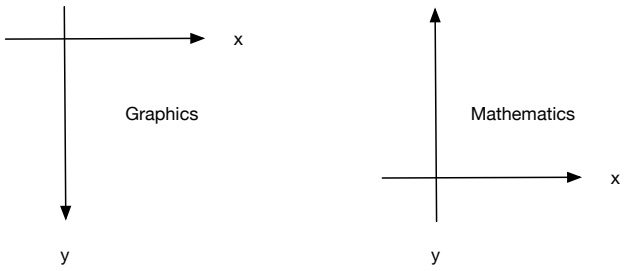
First, we notice that when we specify points, we usually first give the x-coordinate and then the y-coordinate. However, the x-coordinate corresponds to the row and the y-coordinate to the column. When we address elements in a matrix, we however pick the row first and the column later, which is just the reverse, and very, very confusing.

(3) Create a class View. View has three fields, width, height, and display. Display is a height by width double list with rows of length width, there being height rows:

```
self.display = [ [' ' for x in range(self.width)] for y in
                      range(self.height) ]
```

Hint: When you create an object of class View, you should use specific variable names as in `my_view = View(height = 20, width = 50)` to keep confusion to a minimum.

When you print out a display, we print the top line first, whereas mathematical convention has us number rows in the opposite direction. We can adopt the graphics coordinate system, where the y-axis goes down or we can continue to use the Mathematical convention for the y-axis to go up. We will choose the latter possibility. To print out a double array like display, you print out all the rows made into a string by the use of `"".join`. Remember to start with the last row and print it out first.



(4) Write a str-dunder that returns a string giving the width and height followed by display on a newline. To do so, first create a string using `''.join` of each row, and then concatenate all of the rows using `'\n'.join()`, but be careful to start with the last row first.

(5) Write a method `set(x, y, symbol)` that sets the symbol in position x and y with the symbol, usual a single character. For debugging purposes, put the assignment into a try, intercept IndexError, and handle an index error by printing out the offending coordinates and the symbol and then raising an IndexError again.

(6) In class View, write a method initialize that fills in the first and last row of display with minus signs and the first and last column of display with vertical bars. In the corners, put plus signs.

(7) In class View, create a method display_trail(n) that creates an object of type particle, then create a random trail, and finally display the random trail in the view. The random trail will start at (0,0), so you need to translate (0,0) linearly to the middle of the display.