Advanced Python
Lesson 7: Two Dimensional Arrays

A **two-dimensional array** or 2D array is an array that contains more arrays. You have probably used these already, just not realized what they are.

A 1D array looks like this:

```
[1, 2, 3]
```

A 2D array looks like this:

`[array1, array2, array3]` or `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]`

If you remember doing the Hangman lesson, then we used a 2D array to hold our answer and hints. `[[“apple”, “food”], [“pizza”, “food”]]` is a 2D array.

PART 1: Creating 2D Arrays

There are many ways to create a 2D array. The simplest way is to make as many arrays as you need, and then make an array to hold those arrays.

```
array1 = [1, 2, 3]

array2 = [4, 5, 6]

array_2D = [array1, array2]
```

However this may be a lot of coding depending on the number of arrays.

Let’s say we want a 2D array with 5 arrays that contain 5 zeroes in each array.

Instead of typing 5 lines of array1=[0,0,0,0,0], array2=[0,0,0,0,0],…, we can do:

```
array_2D = [[0]*5 for i in range(5)]
```

You may not make sense of it, but it is making a list of 5 zeroes 5 times in a list.

Another way to do this with a little more code is:

```
array_2D = []

for i in range(5):

    array2D.append([])

    for i in range(5):

        array2D[i].append(0)
```

We call this style the double for loop because we use two for loops to make a 2D array.

PART 2: Printing a 2D Array as a Grid

Open a new Python program and name it **array_2D.py**

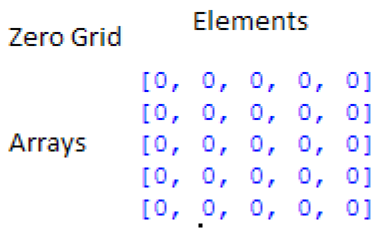
Try typing the following code:

```
array_2D = [[0]*5 for i in range(5)]

for array in array_2D:

     print (array)
```

Now run the code. It should print 5 arrays containing 5 zeroes making a big square of zeroes. We will call this square a grid, like a chess board or graph paper.



```
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
```

Let’s say we wanted to change the middle zero to a 1. We would need to access the middle array’s middle element.

We can access a single element in a 2D array with this code:

```
array_2D[array_number][element_number]
```

Add this line of code to your program after making the 2D array:

```
array_2D = [[0]*5 for i in range(5)]

array_2D[2][2] = 1

for array in array_2D:
```

Now run your code. You should see the middle element is now a 1. Notice we use [2,2] because the array and element numbers start at 0 and end at 4.

The top left zero is at `array_2D[0][0]` and the bottom right zero is at `array_2D[4][4]`.

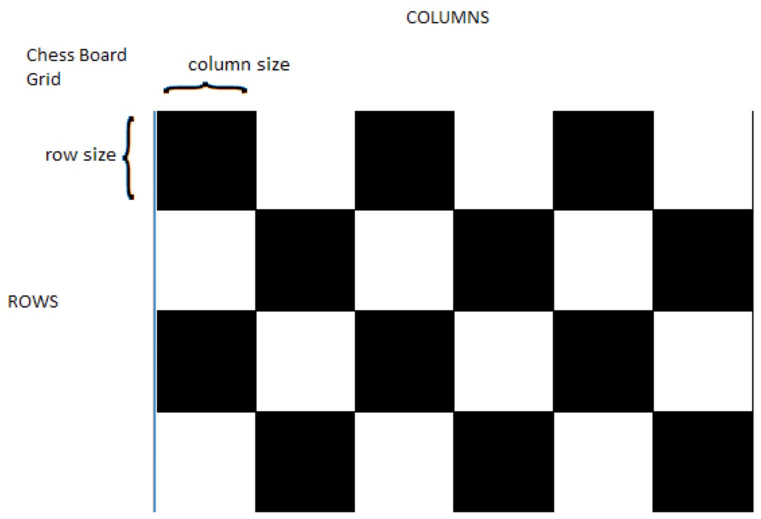
Test yourself and see if you can change a certain zero to a one.

PART 3: Drawing a 2D Array as a Grid

Now that we are starting to get a grasp on what a 2D array is and what it looks like, let's draw a chess board using tkinter and a double for loop.

Instead of printing zeros for square of the grid, we will draw a rectangle for a grid square.

We will be referring to the lines of arrays as rows and the lines of elements as columns.



Erase you previous code and type the following code:

```
from tkinter import *
myTk = Tk()
c = Canvas(myTk, width = 600, height = 400)
c.pack()
rowsize = columnsize = 100  #rowsize=100 and columnsize=100
rows, columns = 4, 6  #row=4 and column=6
for i in range(rows):
     for j in range(columns):
          x, y = j*columnsize, i*rowsize
          rect = c.create_rectangle(x, y, x + columnsize, y + rowsize)
myTk.update()
```

Run your code. It should draw a 4 x 6 grid of empty rectangles.

You may notice that we don't use a 2D array in the code before, but it made a grid like our previous 2D array. We can add a 2D array to maintain our rectangles really easily.

Add the following lines of code into your program:

```
rows, columns = 4, 6  #row=4 and column=6
array_2D = []
for i in range(rows):
     array_2D.append([])
     for j in range(columns):
          x, y = j*columnsize, i*rowsize
          rect = c.create_rectangle(x, y, x + columnsize, y + rowsize)
          array_2D[i].append(rect)
c.itemconfig(array_2D[2][2], fill = "green")
myTk.update()
```

Run your code. You should now have a green square at the [2][2] grid square.

We now have a 2D array that contains a rectangle variable at every element.

Let's now colorize our grid to look like a chess board.

Add the following lines of code into your program:

```
~~c.itemconfig(array_2D[2][2], fill = "green"~~) #Remove this
for i in range(rows):
     for j in range(columns):
          if (i+j)%2 == 0:
               c.itemconfig(array_2D[i][j], fill = "black")
          else:
               c.itemconfig(array_2D[i][j], fill = "white")
myTk.update()
```

Run your code. We should now have a grid that looks like a chess board pattern.

`(i+j)%2` is a simple way of making this pattern. It switches from 1 to 0 every grid square. It is checking if `(i+j)` is an even or odd number.

Two dimensional arrays are very useful as you can see. They can hold many list to one array. They can also draw grids and maps for games.

PART 4: Bonus: Additional 2D Array Exercises

Now that you're pretty good at making 2D arrays, let's try to put your skills to the test.

1) Draw a grid like our chess board but give it a random color for each square.

Hint: You'll need a color list like `colors = ["white", "blue", "magenta", …]` and import random module for the `choice()` function.

2) Draw a grid like our chess board, but now it changes random colors every second like a disco dance floor.

Hint: You'll need to put the double for loop that colors the 2D array randomly into a while loop and import the time module for the `sleep()` function.

3) Draw a grid like our chess board. When the user clicks the mouse button it will destroy the grid and make a new one based on where the mouse is located. Also, when the space bar is clicked, it will randomly colorize the grid.

Hint: May sound really difficult, but is not too hard. Start with your code from Bonus 1.

The code will start the same until we get to the for loop. We will need to make three functions: the first is make a new grid, the second is mouse click, and then space click.

The program should look like:

```
...
def draw_chessboard(rowsize, columnsize):
        #code to erase old grid and draw a new one given rowsize and columnsize
def mouse_click(event):  #mouse click location determines new grid size
        width, height = event.x, event.y
        rowsize, columnsize = int(height/rows), int(width/columns)
        draw_chessboard(rowsize, columnsize)
def space_click(event):       #randomly color the grid
#bind keys to functions and draw initial chessboard
myTk.mainloop()  #similar to myTk.update() but waits for key presses
```