Assignment 3

Objective: To explore the functionality of `for` loops, to extract letters as elements of strings and to introduce escape characters.
Note: Include DocStrings in each script you submit analogous to the following:

```
"""This script makes images of checkerboards


Submitted by Mauricio Arias, NetID ma6918
[Include a short explanation here of how the script does that.]
"""
```

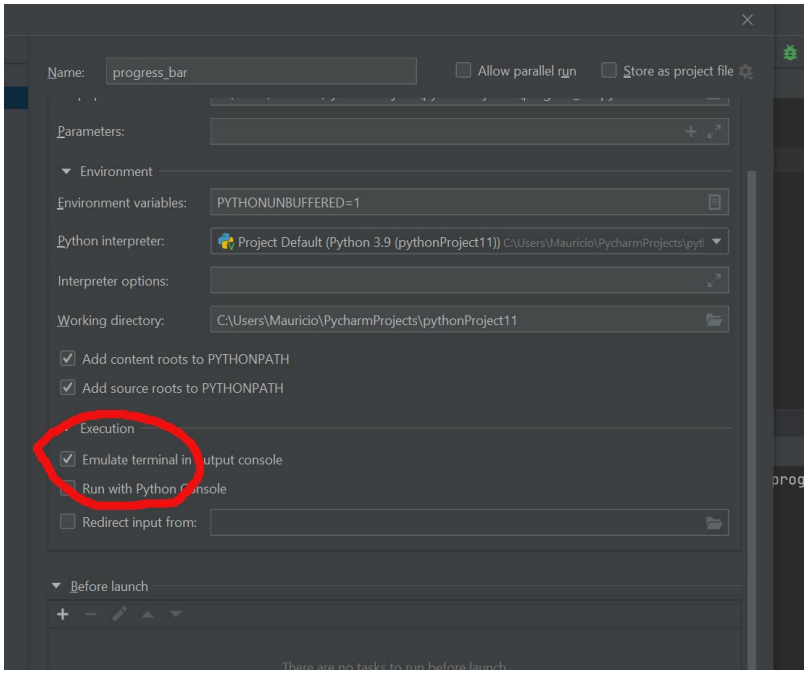
Part 1. Make a functioning progress bar.
Worth 4 points.
Let's start by running some code. Follow these instructions carefully.

Start PyCharm.
Start a new Project.
(Note the suggested location at the top and change it if desired.)
(Change the location for the Virtual Environment accordingly.)
At the end of the top location add progress_bar.
Select the progress_bar folder at the top of the navigation pane on the left
Create a new Python File and name it progress_bar.
Type the following code:

```
import time

print("This is the first line\r", end = '')
time.sleep(5)
print("No! This is the first line.")
```

Run this code.



If nothing seems to happen for a while and the second line above appears after some time, go to `Run → Edit Configurations` and make sure that "Emulate Terminal in output console" is checked. Try again.

This code shows some unexpected behavior for most people. This is caused by the character "`\r`." It is also known as the return carriage character. The terminology dates back to typewriters but it simply means "return the cursor to the beginning of the line." Notice additionally that the print statement has `end = ''`. Usually `print()` adds an extra character at the end: the newline character aka the "`\n`" character. This character moves the cursor to the beginning of the line and feeds one more line, effectively moving the cursor to the line below. (Experiment by inserting this character in some texts using the print statement.) Adding `end = ''` as an extra argument causes the `print()` statement to add nothing at the end of the line. Hence, it allows us to keep writing on the same line. The combination of these two effects allows us to overwrite the same line over and over.

Use this new knowledge to make a progress bar that does not change lines. Ask the user for the "total tasks" to be performed and the number of updates to the bar. Remember that the characteristics of the bar need to be parameterized including the length of the progress bar, the characters to be used, the (fake) total tasks and the completed tasks. A sample implementation for a snapshot has been included in the files.

Make a `for` loop that has `completed_tasks` go from 0 to `total_tasks` exclusive while printing the progress bar. Add the instruction `time.sleep(3/total_tasks)` right after the `print()` statement for the progress bar: this will make the whole process take about 5 to 10 seconds. Make sure the numbers are aligned while the bar changes. A final update should be made when the for loop ends. For this, use the `else` functionality for the loop: the else section is executed after the for loop ends (as long as it ends normally). Watch the accompanying video to observe the expected behavior for the progress bar. (The whole process can be run inside of PyCharm. There is no need to use a terminal.)

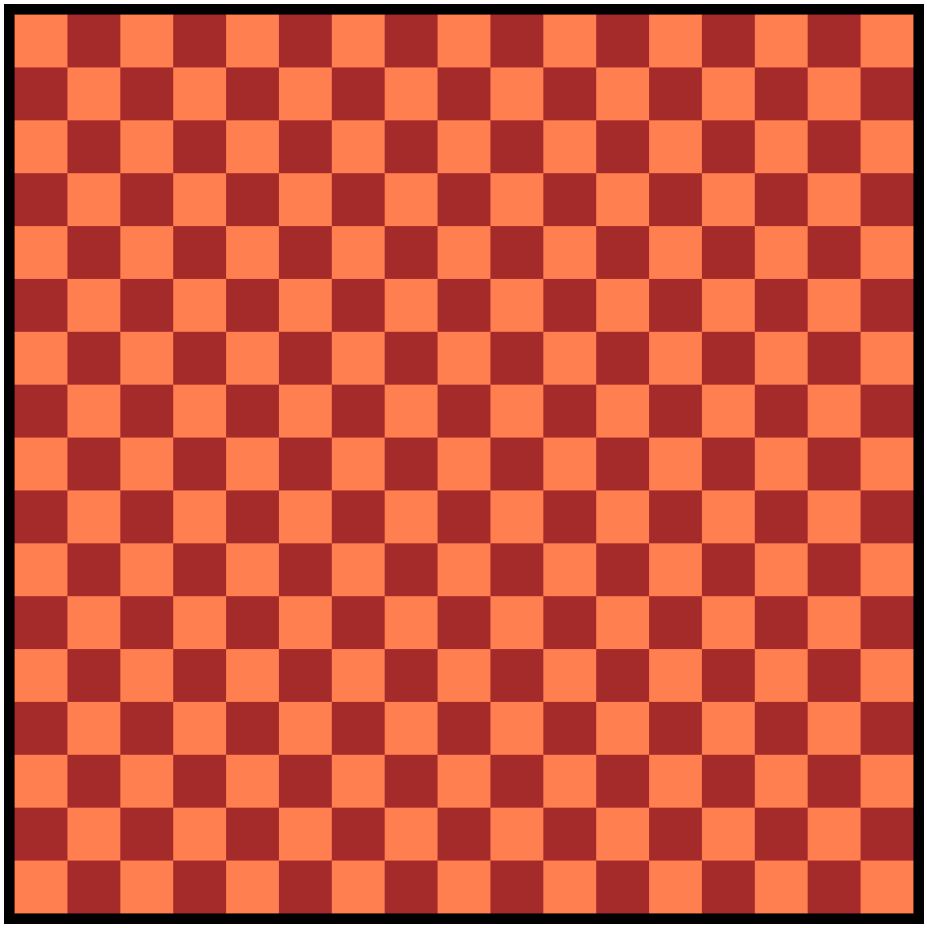
Name your script `[NetID]_live_progress_bar.py` and submit it.

Part 2. Drawing checkerboards.
Worth 4 points.
Your task is to make an image that resembles a checkerboard: see image below. Start a project for this.

The user should provide the length of the cells (x < 200 pixels, verify this), which are squares. The default is 50, if the user does not provide an answer and simply presses enter.

The number of cells per side should be requested (y < 200). The default should be 8. If x * y is greater than 2000, print a message indicating that the values are too high and exit the script: `exit()`.

The edge, shown in black and measured in pixels, should be x * y divided by 80. You need to calculate the length of the full image and provide it to the header function in the utilities file. (See assignment 1 for reference on how to use header(). Notice that the image is a square.)



Use for loops and arithmetic. **Your code should have at least a set of nested `for` loops that is 3 levels deep.** For the colors use brown and coral for the squares. Use black for the edges. For padding use the formula: `-length_of_horizontal_line_in_pixels * bytes_per_pixel % 4`. Use the utilities files we used for assignment 1. Place it in the Lib folder that PyCharm creates for the project: import it properly. Test the script using different values to make sure it works correctly.

Name your file `[NetID]_checkerboard.py` and submit it.

Problem 3. Playing with strings.
Worth 2 points
In Python the characters in a string can be extracted in order by using the `for character in string` setup. Notice that `character` is simply a name; you can choose a different one. Request a string from the user and generate a vertical version of the string: each character is printed in its own line. Generate by concatenation an inverted version of the string and print it last in a new line. Name your file `[NetID]_strings.py` and submit it.