The goal of this second Python programming assignment is to write code inside a provided program framework involving graphics. You will write your own code for the first time.

We will be exploring several graphics functions provided by JES (makeEmptyPicture, show, repaint, and addLine), as well as some color constants used by JES (red, green, blue, yellow, etc.). You may wish to look at the documentation for these functions in the Guzdial book and using the JES help system before attempting this assignment.

In the JES environment, type in the following program code exactly as you see it here and save the program as Lab2.py in your Python folder. Where you see my name in the comment code, replace it with your own name. As always, be very careful about indentation and capitalization. Where you see the | symbols open up several blank lines; you will be writing your own code here at a later time. In the lines between the two sets of triple-double quotes, you will insert your answers from the questions at the end of this assignment.

```python
# William T. Verts - Lab #2

Answers = ""

```

```

def GetANumber (PromptMessage):
    ""

```

```

def MakeImage(NewWidth,NewHeight,NewSeparation):
    MyCanvas = makeEmptyPicture(NewWidth,NewHeight,yellow)
    show(MyCanvas)

```

```

def Run():
    MyName = raw_input("Enter your name: ")
    W = GetANumber("Enter Width, " + MyName + ": ")
    H = GetANumber("Enter Height, " + MyName + ": ")
    N = GetANumber("Enter Steps, " + MyName + ": ")
    MakeImage(W,H,N)
    return
```
**Purpose of the Program**

When the program runs, it will ask first for your name, then for numbers representing the width and height of an image, and a third number representing the spacing between lines. If you enter 320 for the first number, 240 for the second number, and 25 for the third number, you should see a 320×240 pixel yellow box appear on screen with red and blue lines spaced every 25 pixels, as shown below:

![Image](image.png)

In order to accomplish this goal, you will need to fill in the `GetANumber` function and the `MakeImage` functions with your own code, as described next.

**The GetANumber Function**

The purpose of the `GetANumber` function is to ask the user for a positive number, and return that number as the result of the function.

In place of the section of `GetANumber` marked with `|` symbols, *write new code*. Use the Python `input` function to prompt the user with the message string that was passed in as the parameter to `GetANumber`, and assign the result of the function to a variable. Your function must check the numeric value of the entered variable and continue to ask until the user enters a number that is strictly greater than zero. That is, it refuses to accept zero or any negative numbers. Return the number from that variable as the result from the `GetANumber` function.

**The MakeImage Function**

The `MakeImage` function as shown creates a yellow rectangle with the size determined by the values passed in through the `NewWidth` and `NewHeight` variables in the parameter list. In the blank space you opened up in the `MakeImage` function, *write new code* to paint a set of horizontal blue lines, then write additional code to paint a set of vertical red lines, with the separation between lines determined by the `NewSeparation` parameter. For example, if you passed in 320 for the width, 240 for the height, and 25 for the separation, your program would create the image shown earlier on this page.
To paint lines, you will need to use the JES `addLine` function, which has the following format:

\[
\text{addLine}(\text{picture}, \text{startX}, \text{startY}, \text{endX}, \text{endY}, \text{color})
\]

Your code must paint the blue horizontal lines first, starting at Y=0 (the top row of pixels on the screen), and then the red vertical lines next, starting at X=0 (the left column of pixels on the screen). The red lines must cross over top of the blue lines.

**Your code must be general enough** to allow a user to pass in any legal values for image width, image height, and number of steps, and have the program create the correct image. Think about loops! Finally, at the end of your `MakeImage` function, before the return, insert a call to the JES `repaint` function to update the image on the screen. The format is:

\[
\text{repaint}(\text{picture})
\]

**Testing**

In JES click the Load Program button. At the `>>>` prompt, type `Run()` with the parentheses and press Enter→. The program should run. Fix any syntax errors or other mistakes.

Test your program with several different values for the width, height, and separation (the graders will, so make sure it all works).

**Final Questions**

Finally, answer the following questions. Please type them in to your program between the two sets of triple-double quotes at the start of the code. (Your program should still run.)

1. What is the meaning and purpose of the `+` sign in the calls to `GetANumber`?

2. What is the relationship between variable `W` in `Run`, and parameter `NewWidth` in `MakeImage`?

3. Why don’t functions `MakeImage` and `Run` return a value, but function `GetANumber` does?

4. What is the difference between calling `Run()` and answering all the questions, and calling `MakeImage` directly (as in `MakeImage(320, 240, 25)`, for example)?

**Submitting the Code**

When you are finished and everything runs correctly, launch a Web browser and point it at the class site. Find the link to “Form to submit lab assignments” and open it up. Follow the directions to submit your code, and questions, to the TAs for grading.

When you are finished and everything runs correctly, submit your program using the on-line form at: [http://people.cs.umass.edu/~verts/cmpscill9/Lab_Submitter.html](http://people.cs.umass.edu/~verts/cmpscill9/Lab_Submitter.html) (please make certain to select Lab 2 from the drop-down list).