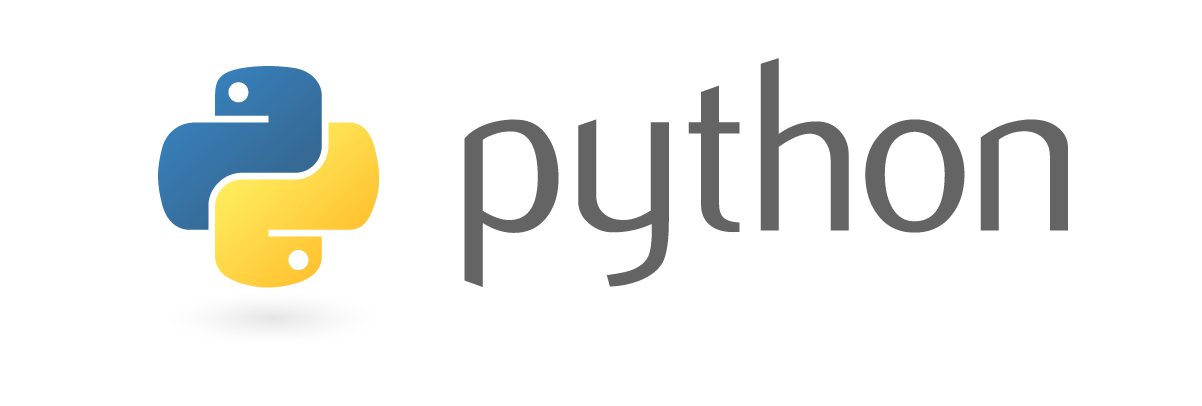
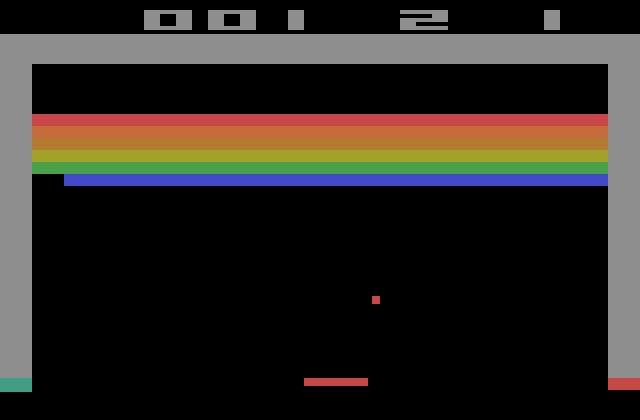
Analysis of Program: Breakout



By: Eric Bachmeier

# Analysis of Program: Olympics

By: Eric Bachmeier

## Table of Contents

**Overview pg.1**

**Graphics pg.1**

**Looping pg.1**

**Decision Making pg.2**

**User-Defined Functions pg.2**

**Strings pg.3**

**Lists pg.3**

**Text Files pg.4**

**Conclusion pg.4**

# Analyzing the Program: Olympics

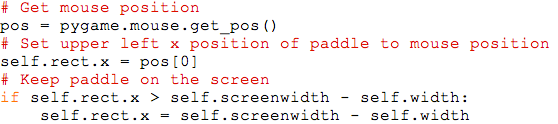
By: Eric Bachmeier

### Overview

My program posed a challenge but turned out as a successful version of the old arcade game Breakout. The final program allows users to play the game trying to bounce the ball to destroy all blocks and keeps a top highscores text file updated. The most difficult part was getting the player paddle, blocks, and ball to update at the same time so the game look fluent and smooth. The easy part was implementing all of the required concepts learned throughout the year because the code was very strenuous.

### Graphics

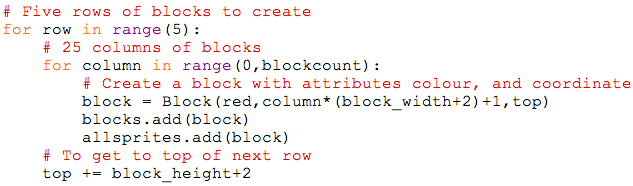
I used the PyGame module to perform graphics and present my game with a visually appealing interface. PyGame was something I touched on earlier this year, but I had to learn a lot more in order to make a game. PyGame certainly has its limitations but it worked out well in the end to make a simple arcade game. I used pre-defined functions such as ‘.render()’ and ‘.get\_rect()’ to achieve various information about the users options. The second functioned mentioned is used to get the rectangle around the defined object so it can be used for collisions. Here is an example of how I used PyGame functions to get the mouse position for the paddle.



This code allows me to move the paddle with the mouse at the same position and ensure the paddle stays within the dimensions of the screen window.

### Looping

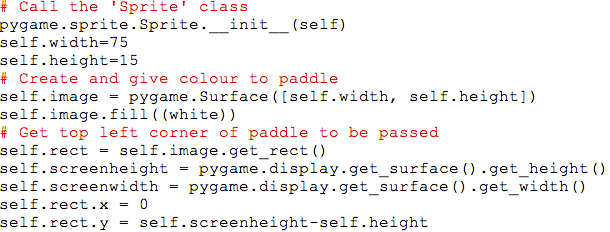
I used looping all throughout my program to accomplish tasks that needed to be executed more than once, to condense the code. ‘For’ loops were used to create the blocks in the specified amount of blocks and rows of blocks. ‘While’ loops were used to write to the Highscores.txt text file and ‘blit’ the ‘GAME OVER’ and high score notifications. This ‘while’ loop is executed only while the exit game variable is False. Looping was essential to making a program that is not hundreds of unessential lines long. Here is an example of looping to make the blocks.



This code above shows the ‘for’ loop I used to make the desired 25 blocks in 5 rows. It runs 25 times for each of the 5 rows, each time passing the colour and position of the block.

### Decision Making

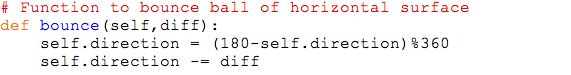
The user doesn’t really have any options in my game, but the decision making part is covered by the user controlling the paddle throughout the course of the game. Getting the mouse position is easy, but having the paddle update to follow that position and stay on the screen gets confusing. I chose to use the top left corner of the paddle as the motion coordinate and that was the point that would mirror the mouse. The following code is an example of decision making of where to move the paddle.



This code calls on the parent class for the paddle to create it and give it colour. It then passed the top left corner after getting the rectangular shape of it. This is how it is able to maintain position with the mouse and be at the users control.

### User-Defined Functions

I used functions that I created all throughout this program to be called upon like the bounce, update, and basic construction functions that I use to check collision and tell the new path of the ball. I also used ‘Sprite’ classes that I refer to later in the code to keep track of the blocks and ball by grouping. Here is an example of the bounce() function;



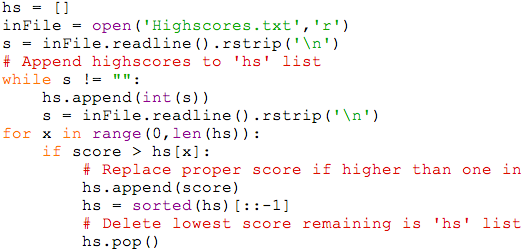
This bounce() function passed in variables for the ball(self) and the motion variable(diff). When I call upon this function the ‘diff’ variable allows the user to bounce the ball left or right depending on the location of the collision with the paddle.

### Strings

I use strings frequently in my program for the titles, score, and text file code. Without strings I couldn’t use text features or be able to write characters to my text file for high scores. A difficult part of using strings was when I wanted to put the score as an updating counter on the actual screen. This was difficult because in order to use the score with Python math function I needed it as an integer but to print it to the screen I needed the string value.

### Lists

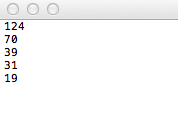
Lists weren’t necessary in the development of this program but definitely were faster for keeping track and sorting high scores. The code I used to do so seems easy, but posed a challenge for me in the long run.



The tricky part was to understand where to put the highscore in the text file so it was in order. With some help I came across a sort() method so I could just append to the list and sort the list. It was important I used lists for this because there are numerous list pre-defined functions that made it easier. After appending the score if it is worthy, I use .pop() to drop the last item in the sorted list(the lowest score).

### Text Files

After I was done the main part of my code, I worked on saving and updating the Highscores.txt text file. I used previous knowledge from the MadLibSentenceMaintenance.py program to delete the contents after reading in and making a high scores list. The easy part was writing the information from the list back to the text file. Here is an example of what the text file contents look like.



### Conclusion

Overall I am very happy with the outcome of my program and am proud to hand it in. If I had more time there are some changes I would have made like using images instead of shapes, having play again buttons, and maybe allowing the user to enter their name along with the high score being saved. I have tested my program and allowed friends to play it, and it has seemed quite bug free. The biggest problem I overcame was the score would only update by one if the player knocked out two blocks in one hit. The way I fixed this was using the ‘deadblocks’ list as the counter instead of adding one to the score per collision. I enjoyed this class and I hope to have fun continuing in a future of Computer Science.