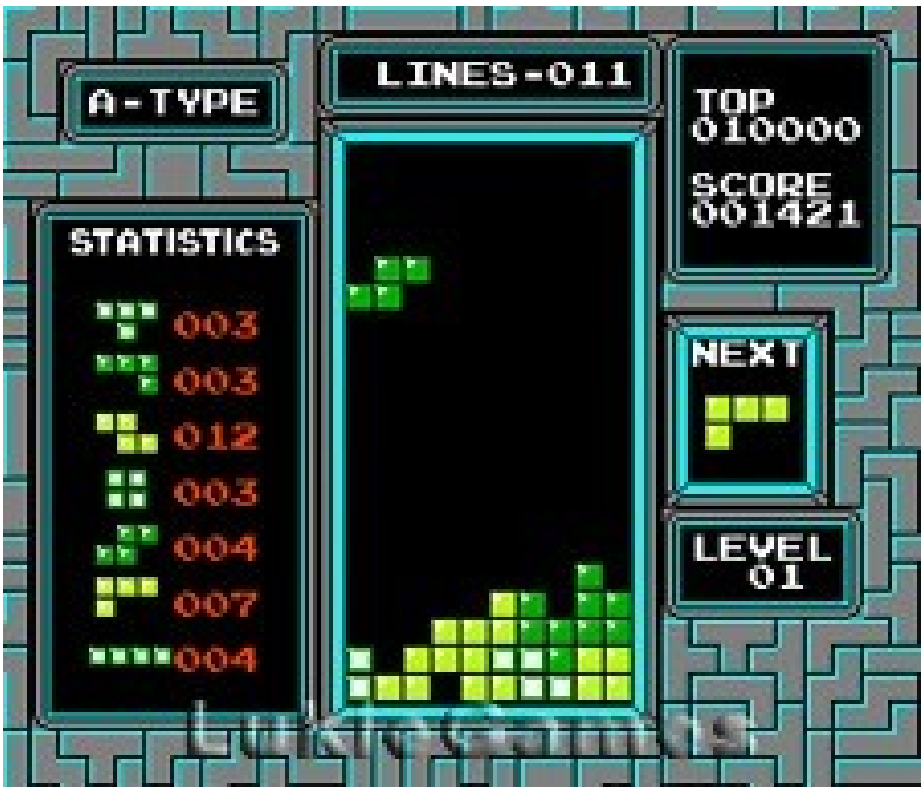
SSAD & Project – Monsoon 2016
ASSIGNMENT 1
Object Oriented Programming in Python

BUILDING BLOCKS

Welcome to your first assignment for SSAD! You will be building a variant of the classic brick video game you might have played in your childhood.

Remember this?



THE ARENA : This game is played on a board which is a matrix of white spaces 30 rows by 32 lines. The falling blocks are specific configurations of four squares. These blocks fall, one at a time, from the top of a board until they either reach the bottom of the board or reach a square on the board that is already filled. When a piece stops falling, it fills in the squares of the board that it occupies, and then the next piece begins falling.

GAMEPLAY: Using keys on the keyboard, the player can control the block by rotating it or move it left /right. Whenever an entire row of the board is filled, that row is removed and all the rows above it are moved down.

OBJECTIVE : The objective is to keep playing as long as possible, which is accomplished by filling in rows in order to remove them. When there is no longer room on the board to drop another piece, the game ends.

SCORING : Each piece dropped scores 10 points. Each row removed scores an extra 100 points.

IMPLEMENTATION DIRECTIONS:

You have to implement the game in **Python** using **OOP principles [Inheritance, Polymorphism, Encapsulation and Modularity** are must]. Rather than write long methods, try to break up the functionality into more short methods, so that when you create subclasses, it will be easier to override specific parts of the functionality of the class. The better code you write, more marks you get. Do not use Pygame.

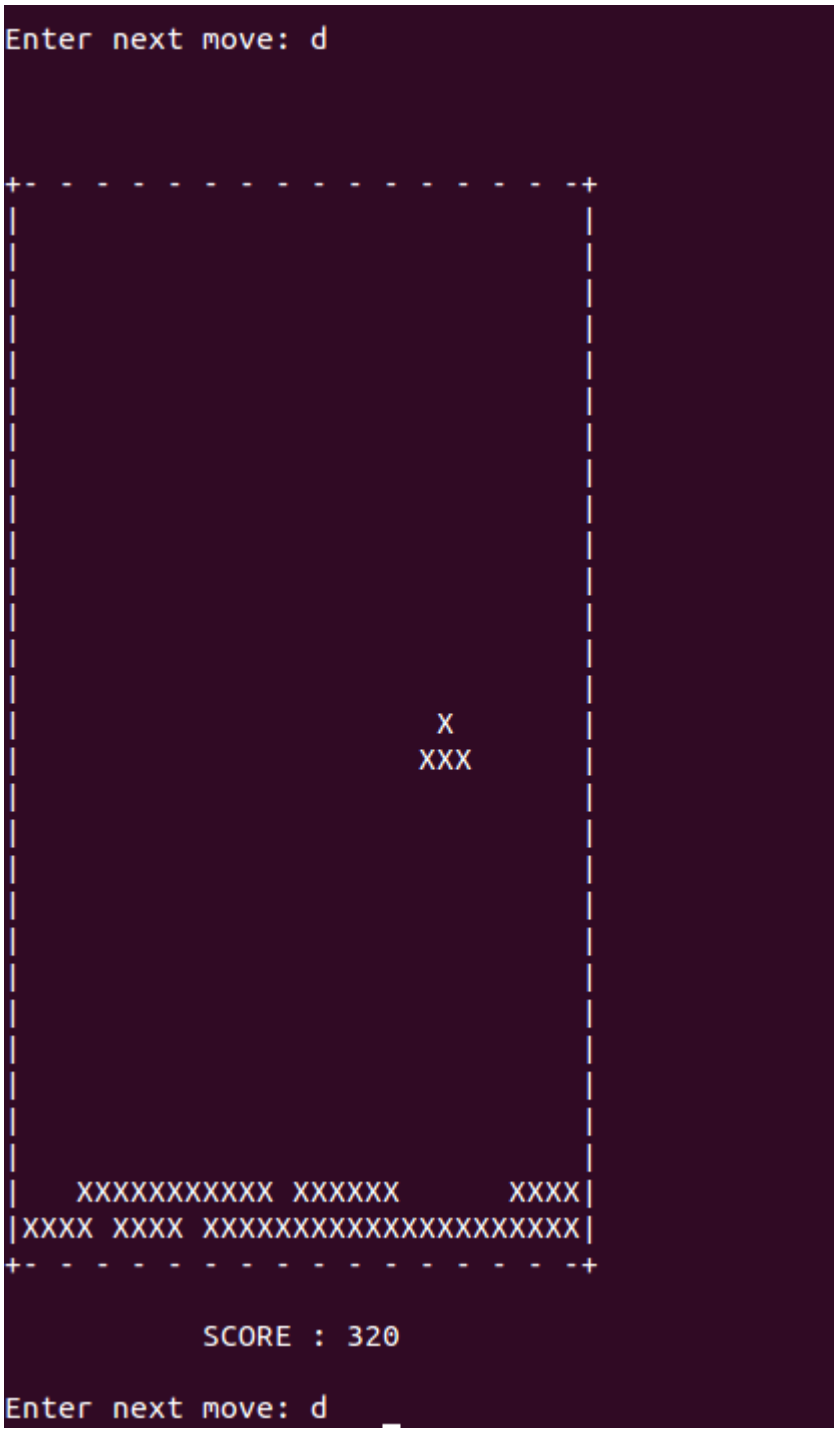
(For some fun python philosophy : Open your terminal. Run the python interpreter and type the statement “import this”)

Preferred size of the board is **30 x 32**. Filled spaces on the board and the blocks are represented by **'X'**. The board should have a boundary and the score should be dispayed with every new move.

Implement the game such that the board is refreshed and displayed every 1 second and the blocks fall down a line a second. Use the keys **'a'** & **'d'** to move the falling block left or right. Using key **'s'** the blocks can rotate clockwise. By pressing the **spacebar** key, the block should instantly fall below to its place on the board.

Once a block reaches the bottom of the board, a new block should appear at the **top center** of the board and begin falling. The type of the block should be selected at **random**.

You are required to implement minimum **any 4** of the following types of blocks :


Enter next move: d
X
XXX
SCORE : 320
Enter next move: d

Your code must have the following classes and methods however you are free to add more:

1) **Gameplay** Class : Which draws the board array and has the following methods

a) **checkRowFull()** : to check if a row is full (has all Xs).

b) **checkRowEmpty()** : to check if row is empty (has no Xs and only spaces).

c) **updateScore()**

d) **selectPiece()**

2) **Board** Class which the following methods :

a) **checkPiecePos()** : It should take a row and column index and a 2D array as parameters. It should check to see that the whitespaces of the given array correspond to empty positions on the grid, assuming that the given array's upper left were positioned at the given row and column. Use this to determine if a piece can move into a given position.

b) **fillPiecePos()** : To fill in the board square covered by a piece, it would be helpful to have a method similar to the one above that copies the X's from the given array into the corresponding board squares.

3) **Block** Class which has the following methods :

a) rotate()

b) moveLeft()

c) moveRight()

d) draw()

The Block and Board classes inherit from Gameplay. The different types of blocks implemented should exhibit inheritance and polymorphism.

Update the score with each new block and with each cleared row. With each cleared row, the semifilled rows above fall down eliminating the filled row. **End the game** when the board is filled to the top and there is no space for a new block to fall.

BONUS : Bonus marks will be given for additional features like (More than 4 types of block, special type of block which helps clearing the board faster, additional levels with random walls existing on the board, levels where a semifilled row appears at the bottom after a few new pieces and pushes up the existing semi-filled rows, increase in block falling speed in different levels etc)

RUBRICS FOR EVALUATION:

Assesment Criteria	Marks
OOP : Inheritance	10
OOP : Polymorphism	10
OOP : Encapsulation	10
OOP : Modularity	10
Functionality of the Game	40
Bonus	20