**All function names are works in progress*

https://github.com/aoursler/JustSnekThings

Snek.py: A python program which creates and manages a matrix array representing a game of Snek. Accessed by just_snek_things.erl via erlport.

Moved all information to a class MySnek:

- init(Node,PID): Takes in the Node and PID of the calling erlang server. Creates the state relevant to a game of Snek.
    - Self.server – a tuple.
    - Self.board – a 2d array of single characters.
    - Self.players – a Dict keyed on player (Node,PID) as a tuple, which is faster accessed than a list. Dict values are a list including:
        - current token location, token character for the given snake, player score, last token location, player energy and a sublist of all locations occupied by the player tail.
- Remove_player(Node,PID, head): Removes the current player from the game. Called internally on player death as well as via erlport on quit.
- _move_check:(Node, PID, oldLocation, newLocation): Internal function called by move, which checks for how to handle a given move attempt – death on collision, extra energy on powerup, otherwise, bookkeeping on tail and head locations.
- move(Node,PID,direc): attempts to move a given snake on the board. Invoked by erlport.
- get_board(): outputs the current board state as a 2d list, converted to a tuple for fast translation across erlport.
- Add_player(Node, PID): adds a new player to the board, seeded at a random, free location. Accessed via erlport.
- find_empty_slot(): Internal function to find free spaces on the board for seeding of new players/powerups

Just_Snek_Things.erl: an erlang gen_server which contains the code to take in moves from clients as well as for processing client messages as a server.

- Start(HostName, GameName, UserName): Starts a game and joins it
- Join_game(HostName, GameName, UserName): client function to subscribe to the given Game
- move(Hostname, GameName, UserName, character): passes a move from the client to the server
- quit(HostName, GameName, UserName): Leaves the given game.

- start_link(GameName): Creates an instance of the game via erlport

- stop(GameName): closes out the python VM and closes the given game
- handle_cast({subscribe, UserName}, {PlayerData,PythonPID}): server function to handle subscription requests. Passes to python game via erlport
- handle_cast({unsubscribe,UserName}, {PlayerData, PythonPID}): server function to handle quit requests from players
- handle_call({move, UserName, Direction}, {PlaterData, PythonPID}): server function to handle move requests. Sent via erlport to the python VM at PythonPID
- terminate(Reason, Data): invokes internal bookkeeping regarding server termination and client unsubscription

Tkinter Client: Keypresses mapped via internal function for output to erlport/Just_Snek_things.erl clients and tuples taken in via erlport and represented on screen.