i) Clearly define the set of percepts (at least 5 distinct percepts); all the possible moves (at least 4 moves) and the set of actions (at least 5 or more distinct actions).

Percepts (inputs to the neural network):

- Danger straight - boolean indicating danger straight ahead

- Danger right - boolean indicating danger to the right

- Danger left - boolean indicating danger to the left

- Direction left - boolean indicating snake is moving left

- Direction right - boolean indicating snake is moving right

- Direction up - boolean indicating snake is moving up

- Direction down - boolean indicating snake is moving down

- Food left - boolean indicating food is to the left of snake's head

- Food right - boolean indicating food is to the right of snake's head

- Food up - boolean indicating food is above snake's head

- Food down - boolean indicating food is below snake's head

Moves:

- Move straight

- Move right

- Move left

- Move down

Actions (output from the neural network):

- Move straight

- Turn right

- Turn left

**Actions:**

1. **Turn Left**:
   * Instructs the snake to change its direction to the left.
2. **Turn Right**:
   * Instructs the snake to change its direction to the right.
3. **Continue Straight**:
   * Instructs the snake to continue in its current direction.
4. **Move Towards Food**:
   * Instructs the snake to move in the direction of the food.
5. **Avoid Collision**:
   * Instructs the snake to take an action that avoids a potential collision.
6. **Explore**:
   * Encourages the snake to explore new areas of the grid, potentially discovering new food.

So in summary, there are 11 distinct percepts given as input to the neural network, 4 possible moves the snake can make, and 3 actions produced as output from the network. The agent must learn to map percepts to actions in order to play the game successfully.

3)

Based on the code you provided, it looks like this AI agent for playing Snake is implemented using a reinforcement learning technique. Here are the key details:

- The core game logic and UI is defined in snake\_game.py. It uses PyGame to implement the graphics and game loop.

- The Agent class in agent.py contains the RL agent logic. It uses a neural network defined in model.py called Linear\_QNet to approximate the Q-function. This allows it to estimate the long-term reward for taking different actions in different states.

- The Agent uses epsilon-greedy exploration to balance exploiting the best known action with exploring new actions. This encourages discovering new strategies.

- The Agent stores experiences (state, action, reward, next state) in a deque memory buffer. It samples from this to do mini-batch training of the Neural Net.

- It uses a Q-learning update rule to train the network. This iterates the Q-values towards the observed returns.

- The neural net has an input layer with size equal to the state space (11 variables), a hidden layer of 256 nodes, and output layer with size equal to the action space (3 possible actions).

- The agent generates the state representation using information like dangerstraight, dangerleft, food location relative to head. This encodes the key aspects of the situation.

- The training loop involves playing games, storing experiences, and retraining the network. Plotting shows the agent learns to get higher scores over time.

So in summary, this uses a Neural Net to approximate the optimal action value function, combined with episodic training and experience replay. The agent should learn through trial-and-error to play the game well like a human.