Requirements

Imports:

* Pygame
* Sys
* Random

Screen requirements:

* size
* clock object to manage framerate,
* a way to refresh/update the screen,
* a way to close the screen,

Snake Game requirements:

* Coordinate system, which requires:
  + A grid which fills the screen
  + Size of grid squares
  + Specified number of grid rows/columns
    - V1.1: Grid is visible (bicolor grid squares)
    - V2: size of grid (and therefore screen) customizable by player
      * Small, medium, large, XL
* The snake class, which requires:
  + Instantiation of a snake as an object
  + Position where the snake exists to be designated as the snake
    - V1.1: Updated graphics model
  + Ability to move automatically, which requires:
    - A timer
    - A specified distance/time ratio for movement
  + A way to move the snake, which requires:
    - Player input
    - Means of converting player input to visual output
      * V2: Customizable movement speed and therefore difficulty
  + A way to prevent backwards movement or only allow forward and left/right movement based on snake’s current orientation
  + A method which kills the snake via contact with itself or screen border
  + A way for the snake to grow when points are scored
  + A way to reset the snake to its original size if the game is lost or started over
    - V2: “endless mode” where snake length is capped at specified value and therefore will allow theoretically infinite score to be achieved
* The fruit class, which requires:
  + Instantiation of the fruit as an object
  + Specified shape (square), size (one grid square), color (red)
  + A way for it to spawn at random coordinates
  + A way for it to be “eaten”
  + A way for it to immediately respawn once eaten
  + A way to increase score upon being eaten
  + A way to exclude its spawn coordinates from the coordinates occupied by the snake object
    - V1.1: Updated graphics model
    - V2:
      * Multiple fruit objects spawn at once, quantity customizable by the player
      * 1/10 chance for a moving fruit to spawn which grants an increased score reward when eaten
* Scoring, which requires:
  + Visible display
  + Dedicated surface outside of the play area surface
  + A counter which increments when fruit is eaten and resets upon death

Pseudocode

Screen setup:

* Pygame.init:
  + Grid:
    - gridSize = 50, gridNumber = 40
  + Screen:
    - Make screen: Screen(x,y) = (gridSize\*gridNumber, gridSize\* gridNumber)
    - Close screen: pygame.quit, sys.exit
  + Clock:
    - Pygame.time.clock

Snake class:

* Instantiate snake
  + Pygame.math Vectors(x,y)
  + Starting snake body = vector(x,y),vector(x+1,y), vector(x+2,y)
* Draw snake
  + Body = Pygame.rect(x,y,w,h)
    - ( BodyX\*gridSize, bodyY\*gridSize , gridSize, gridSize)
  + Pygame.draw.rect:
    - (screen, color, body)

Fruit class:

* Instantiate fruit:
  + Random Position (x,y):
    - FruitPosition = (randint(0,gridNumber)),(randint(0,gridNumber))
    - fruitPostition /= snakeBody (x,y)
* Draw fruit:
  + Fruit body: Pygame.rect(x,y,w,h)
    - fruitBody = (fruitPosition, gridSize, gridSize)
    - draw.rect (screen,color,fruitBody)

Snake movement method:

* Instantiate timer
* figure out how to apply user inputs to vectors

Fruit Eaten method:

* If:
  + snakeBody vector(x,y) == fruitPosition(x,y)
  + FruitPosition = (randint(0,gridNumber)),(randint(0,gridNumber))
  + Score = Score + 1
* For:
  + Score > 0
  + snakeBody = snakeBody + vector(((x+2)+score),y)

Snake collides with snake method:

* If:
  + snakeBody vector(x,y) == snakeBody
    - Game over

Snake collides with boundary Method:

* If:
  + snakeBody vector(x,y) >= (gridSize\*gridNumber, gridSize\* gridNumber)
    - Game over

Main Method:

* Snake class
* Fruit class
* Snake movement method
* Fruit eaten method
* Snake collides with snake method
* Snake collides with boundary method