# README

## Overview

To run the program, run main.py from the command line. There are two main uses of this program: designing a custom maze and running the genetic algorithm to optimize some property of a maze. If the user chooses to design a custom maze, they can input certain specifications that will guide the way the maze is created. If the user chooses to run the genetic algorithm, then the maze can be optimized for difficulty, easiness, or maximum distance from start to end. The GUI is designed to guide the user through the program.

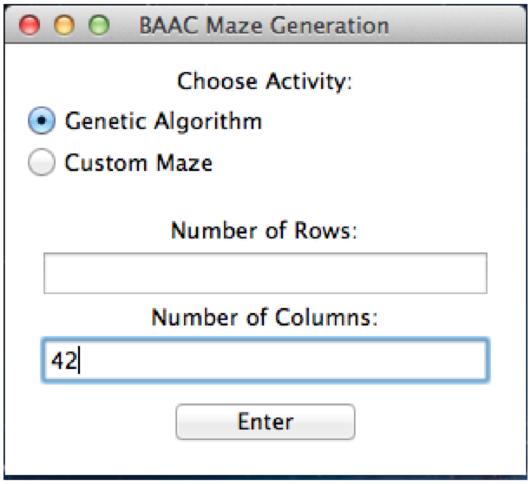
## Requirements for Running

This program requires Python 2.7.3 or Python 2.7.4. All necessary packages are included in the default distribution.

## Walkthrough of GUI

### Activity Window

The first window asks the user to choose the desired activity and to input the number of rows (default = 20) and columns (default = 20) desired. The number of columns and rows must each be greater than or equal to 4. To get a good display, do not exceed about 80 rows and 80 columns in a custom maze. Considerably smaller mazes (max 50 x 50) are advised when running the genetic algorithm to avoid very long runtimes. The user must choose an activity. However, if the rows and/or columns fields are left blank, the default values will be used



BAAC Maze Generation

Choose Activity:

● Genetic Algorithm

○ Custom Maze

Number of Rows:

Number of Columns:

42

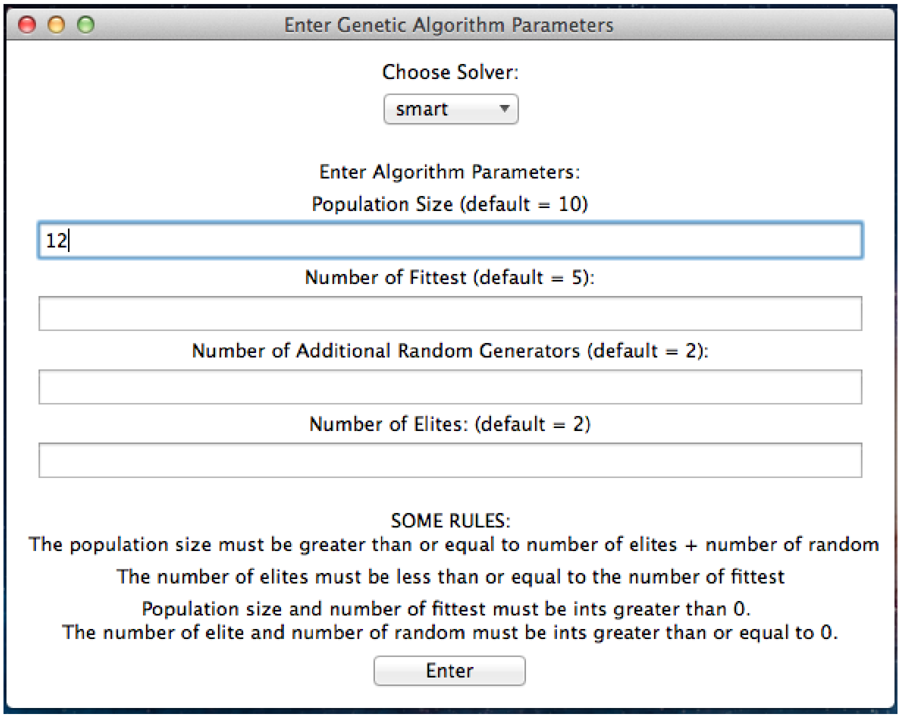
### Genetic Algorithm Window

If the user selects the Genetic Algorithm Activity, the GUI will prompt the user to choose the solver desired and to enter the population size, number of fittest, number of additional random generators, and number of elites for the genetic algorithm. Any space left blank will be taken as the default value. The different maze solver options are briefly described:

- smart solver: best approximates human behavior and runtime
- inverse solver: inverts the process and determines the easiest maze
- pythagorean solver:  maximizes the distance between the start point and the end point

The algorithm parameters inputted should be relatively close to the default. Large numbers will drastically increase the runtime.  The algorithm parameters are described as follows:

- population size (default = 10): the number of generators in a generation
- number of fittest (default = 5): the number of generators that will be used for the breeding of the next generation
- number of random (default = 2): number of new random generators to introduce in the next generation
- number of elites (default = 2): number of the fittest generators to keep in the next generation



Enter Genetic Algorithm Parameters

Choose Solver: smart

Enter Algorithm Parameters:

Population Size (default = 10): 12

Number of Fittest (default = 5):

Number of Additional Random Generators (default = 2):

Number of Elites: (default = 2)

SOME RULES:
The population size must be greater than or equal to number of elites + number of random
The number of elites must be less than or equal to the number of fittest
Population size and number of fittest must be ints greater than 0.
The number of elite and number of random must be ints greater than or equal to 0.

With respect to the genetic algorithm, the user must respect the rules given.

- All inputted parameters must be integers
- The population size and number of fittest must be greater than 0
- The number of elite and the number of random must be greater than or equal to 0
- The population size must be greater than or equal to the sum of the number of elites and the number of random
- The number of elites must be less than or equal to the number of fittest

When the user enters the parameters, that’s it! The genetic algorithm will run, breeding a number of generations until the added difficulty is negligible for three consecutive generations (typically between 8-16 generations). Then, the very best maze of the generation will be outputted in the terminal and browser

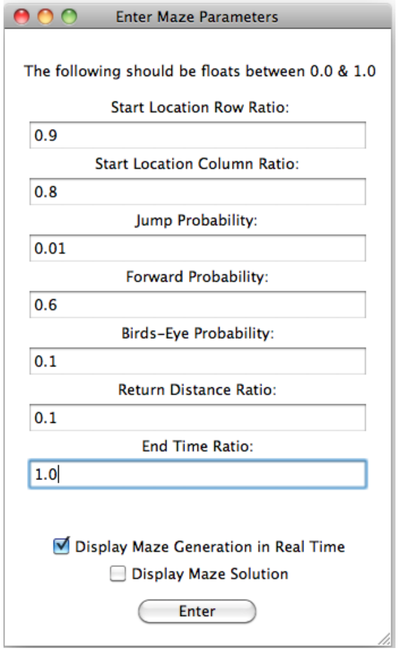
Custom Maze Window

If the user selects the Custom Maze Activity, the GUI will prompt the user to enter the parameters for the custom maze. All of these parameters are floats and most can be thought of as probabilities or ratios. Any space left blank will be taken as the default value.

The parameters given are:

- start location row (default = 0.0): starting location as ratio of row/maze_num_rows
- start location column (default = 0.0): starting location as ratio of column/maze_num_cols
- jump probability (default = 0.2) : the probability of jumping to a different spot and branching off a new path when adding a next square
- forward probability (default = 0.7): the probability of moving forward - ie. in the same direction as before - as opposed to turning
- birds-eye probability (default = 0.8): if jumping, probability of doing so through a "birds-eye" calculation as opposed to picking a random square
- return distance ratio (default = 0.7): if doing a "birds-eye" jump, return_dist is the ratio of the desired distance from start (in the square jumped to) to the current distance from the start in the square jumped to
- end time ratio (default = 1.0): rough approximation of what percentage of maze will be completed when end is placed
- display maze generation in real time (default = unchecked): checking this box will show the step-by-step creation of the maze graphically in the terminal window
- display maze solution (default = unchecked): checking this box will display the solution

*One suggested configuration*



Enter Maze Parameters

The following should be floats between 0.0 & 1.0

Start Location Row Ratio: 0.9

Start Location Column Ratio: 0.8

Jump Probability: 0.01

Forward Probability: 0.6

Birds-Eye Probability: 0.1

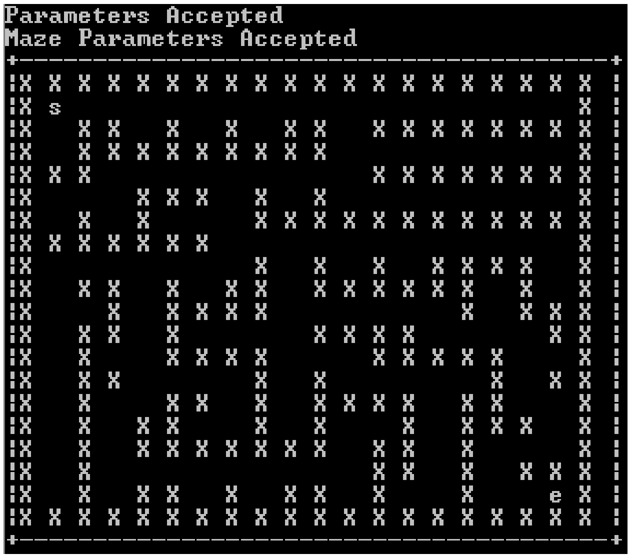
Return Distance Ratio: 0.1

End Time Ratio: 1.0

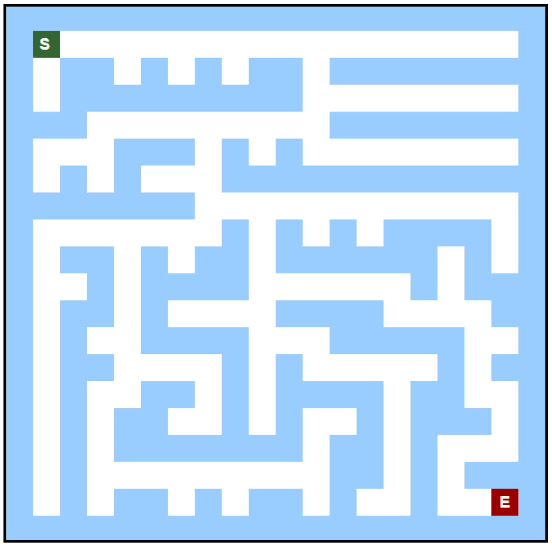
☑ Display Maze Generation in Real Time

□ Display Maze Solution

When the user enters the parameters, that's it! The custom maze will be outputted in the terminal (ASCII) and default browser (HTML). It should look something like this:

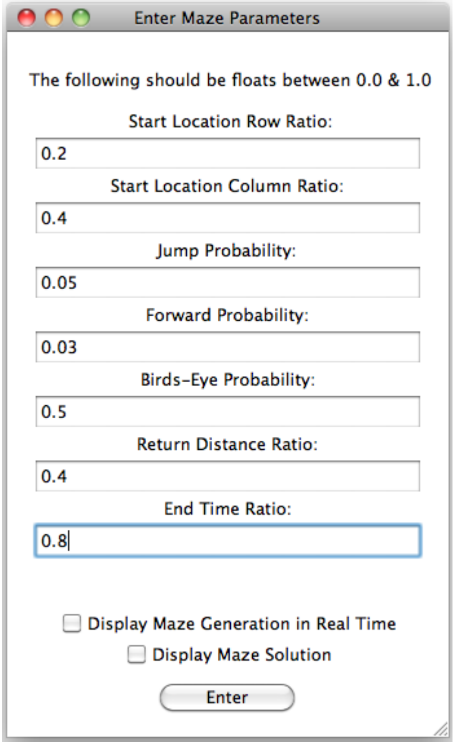


```
Parameters Accepted
Maze Parameters Accepted
+----------------------------------------+
!X X X X X X X X X X X X X X X X X X X X !
!X s                                   X !
!X   X X   X   X   X X   X X X X X X X X !
!X   X X X X X X X X X                 X !
!X X X                   X X X X X X X X !
!X       X X X   X   X                 X !
!X   X   X       X X X X X X X X X X X X !
!X X X X X X X                         X !
!X               X   X   X   X X X X   X !
!X   X X   X   X X   X X X X X X   X   X !
!X     X   X X X X             X   X X X !
!X   X X   X         X X X X         X X !
!X   X     X X X X       X X X X X     X !
!X   X X         X   X           X   X X !
!X   X     X X   X   X X X X   X X     X !
!X   X   X X     X   X     X   X X X   X !
!X   X   X X X X X X X   X X   X       X !
!X   X                   X X   X   X X X !
!X   X   X X   X   X X   X     X     e X !
!X X X X X X X X X X X X X X X X X X X X !
+----------------------------------------+
```



## Some Cool Configurations

*1. Diagonals*



Enter Maze Parameters

The following should be floats between 0.0 & 1.0

Start Location Row Ratio: 0.2

Start Location Column Ratio: 0.4

Jump Probability: 0.05

Forward Probability: 0.03

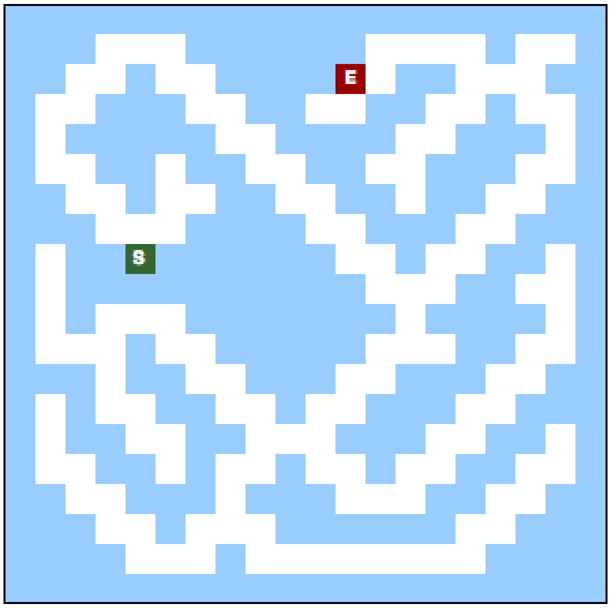
Birds-Eye Probability: 0.5

Return Distance Ratio: 0.4

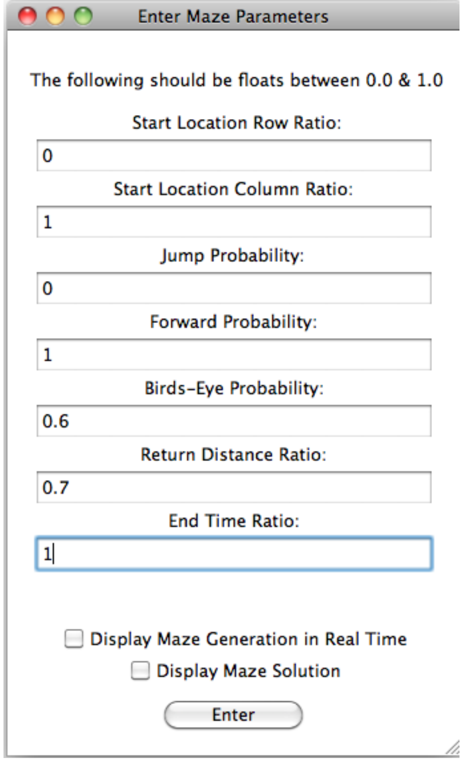
End Time Ratio: 0.8

□ Display Maze Generation in Real Time

□ Display Maze Solution



## *2. Spirals*



Enter Maze Parameters

The following should be floats between 0.0 & 1.0

Start Location Row Ratio: 0

Start Location Column Ratio: 1

Jump Probability: 0

Forward Probability: 1

Birds-Eye Probability: 0.6

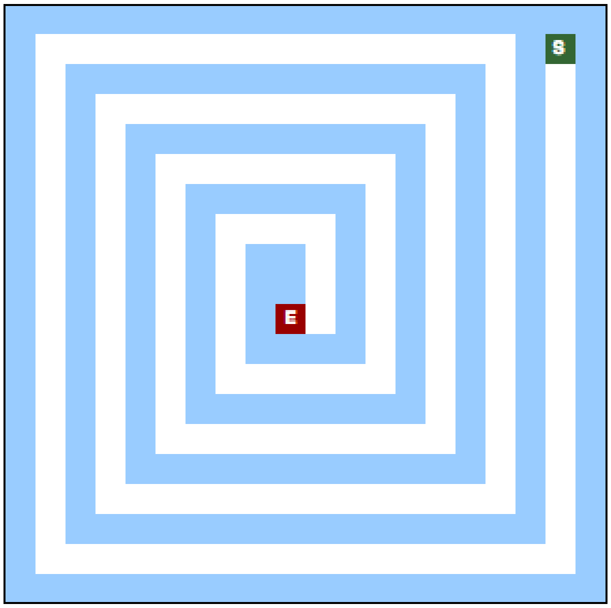
Return Distance Ratio: 0.7

End Time Ratio: 1

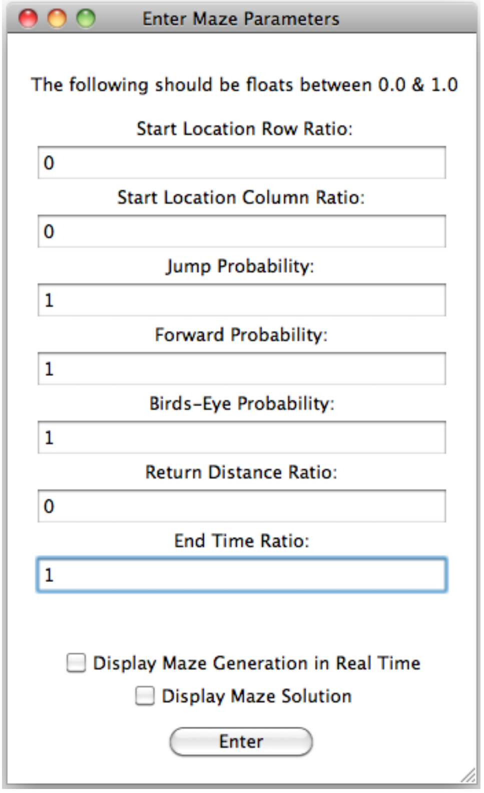
□ Display Maze Generation in Real Time

□ Display Maze Solution

Enter



## *3. Leaf*



Enter Maze Parameters

The following should be floats between 0.0 & 1.0

Start Location Row Ratio: 0

Start Location Column Ratio: 0

Jump Probability: 1

Forward Probability: 1

Birds-Eye Probability: 1

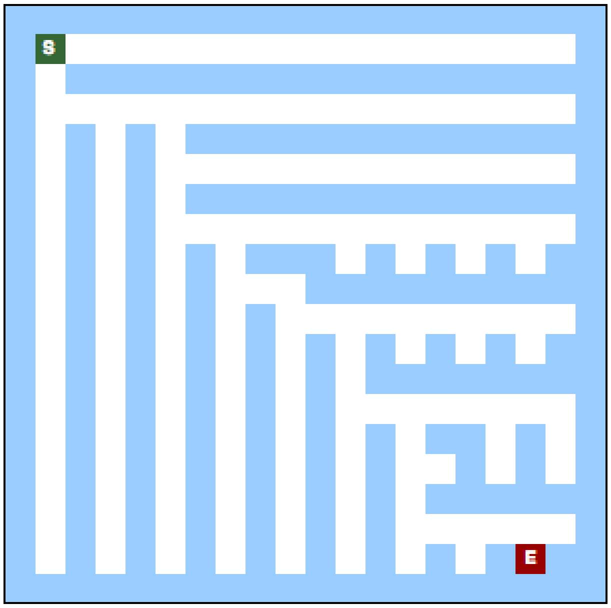
Return Distance Ratio: 0

End Time Ratio: 1

□ Display Maze Generation in Real Time

□ Display Maze Solution

Enter



List of Files
main.py - run this file to execute program

*Helper Files*
activity_gui.py - makes the first “choose activity” prompt window
custom_gui.py - makes the window prompting users for custom maze parameters
display.py - implements ASCII display as class
generation.py - class that consists of one generation of the genetic algorithm
generator.py - implements the generator class
genetic_gui.py - makes the window prompting users for genetic algorithm parametrs
htmldisp.py - implements HTML display as class
genetic_alg.py - runs the genetic algorithm
maze.py - the maze class
path_finder.py - implements the path_finder solver
settings.py - globall settings for the files