

```
1  #!/usr/bin/env python
2
3  import curses
4  import copy
5  import sys
6  import time
7  import random
8
9
10 #####
11 # INITIALIZATION #
12 #####
13
14
15 def main(argv):
16     if not len(argv) == 2:
17         usage()
18         exit(1)
19
20     width = 0
21     height = 0
22     try:
23         width = int(argv[0])
24         height = int(argv[1])
25     except:
26         usage()
27         exit(2)
28
29     board = createBoard(width, height)
30     if board == None:
31         usage()
32         exit(3)
33
34     try:
35         curses.wrapper(snake, board)
36     except Exception as err:
37         print(err)
38
39     exit(0)
40
41
42 def usage():
```

```
43     print("USAGE: ./snake.py [WIDTH] [HEIGHT]")
44     print("where [HEIGHT] and [WIDTH] are integers between 10 and the bounds of your terminal")
45
46
47 def createBoard(width, height):
48     if width < 10 or height < 10:
49         return None
50
51     board = []
52     for i in range(height + 2):
53         board.append([])
54         for j in range(width + 2):
55             if i == 0 or i == height + 1:
56                 board[i].append("-")
57             elif j == 0 or j == width + 1:
58                 board[i].append("|")
59             else:
60                 board[i].append(" ")
61
62     board[0][0] = "+"
63     board[height + 1][0] = "+"
64     board[0][width + 1] = "+"
65     board[height + 1][width + 1] = "+"
66
67     return board
68
69
70 #####
71 # GAME FUNCTIONS #
72 #####
73
74
75 def snake(stdscr, board):
76     height = len(board)
77     width = len(board[0])
78
79     maxHeight = curses.LINES - 3
80     maxWidth = curses.COLS - 1
81
82     if height >= maxHeight or width >= maxWidth:
83         raise Exception("Bounds too large, the max width is " + str(maxWidth - 3) + " and the max height is " +
84 str(maxHeight - 3))
```

```
84
85     curses.start_color()
86     curses.init_pair(1, curses.COLOR_BLACK, curses.COLOR_GREEN)
87     curses.init_pair(2, curses.COLOR_BLACK, curses.COLOR_RED)
88
89     curses.curs_set(False)
90
91     stdscr.clear()
92     stdscr.refresh()
93
94     direction = "right"
95
96     snake = []
97     for i in range(3):
98         snake.append([height // 2, (width // 4) - i])
99
100    fruit = [(height // 2), width - (width // 4)]
101
102    drawScreen(board, snake, fruit, stdscr)
103
104    stdscr.addstr(height + 2, 0, "Press any key to start")
105    stdscr.refresh()
106    stdscr.getch()
107    stdscr.addstr(height + 2, 0, "")
108
109    drawScreen(board, snake, fruit, stdscr)
110
111    while True:
112        time.sleep(0.1)
113
114        direction = getDirection(direction, stdscr)
115        drawScreen(board, snake, fruit, stdscr)
116        if updateGame(board, snake, fruit, direction) == False:
117            break
118
119        if snake[0][0] == 0 or snake[0][1] == 0 or snake[0][0] == len(board) - 1 or snake[0][1] == len(board[0])
- 1:
120            break
121
122    stdscr.nodelay(0)
123    stdscr.addstr(height + 2, 0, "GAME OVER - SCORE: " + str(len(snake) - 3), curses.A_BOLD | curses.A_UNDERLINE)
124    stdscr.addstr(height + 3, 0, "Press any key to exit")
```

```
125     stdscr.refresh()
126     time.sleep(0.25)
127
128     stdscr.getch()
129
130
131 def drawScreen(board, snake, fruit, stdscr):
132     stdscr.clear()
133
134     for i in range(len(board)):
135         for j in range(len(board[i])):
136             stdscr.addstr(i, j, board[i][j])
137
138     for i, pos in enumerate(snake):
139         stdscr.addstr(pos[0], pos[1], " ", curses.color_pair(1) | curses.A_DIM)
140         if i == 0:
141             stdscr.addstr(pos[0], pos[1], "X", curses.color_pair(1) | curses.A_BOLD)
142
143     stdscr.addstr(fruit[0], fruit[1], " ", curses.color_pair(2) | curses.A_BOLD)
144
145     stdscr.addstr(len(board) + 2, 0, "SCORE: " + str(len(snake) - 3))
146
147     stdscr.refresh()
148
149
150 def updateGame(board, snake, fruit, direction):
151     oldSnake = copy.deepcopy(snake)
152
153     if direction == "right":
154         snake[0][1] += 1
155     elif direction == "left":
156         snake[0][1] -= 1
157     elif direction == "up":
158         snake[0][0] -= 1
159     elif direction == "down":
160         snake[0][0] += 1
161
162     for i in range(1, len(snake)):
163         snake[i] = oldSnake[i - 1]
164
165     if snake[0] == fruit:
166         snake.append(oldSnake[-1])
```

```
167         fruit[:] = list(newFruit(snake, fruit, board))
168
169     for i in range(1, len(snake)):
170         if snake[0] == snake[i]:
171             return False
172
173     return True
174
175
176 def getDirection(direction, stdscr):
177     stdscr.nodelay(1)
178     key = stdscr.getch()
179
180     if key == curses.KEY_RIGHT and not direction == "left":
181         return "right"
182     elif key == curses.KEY_LEFT and not direction == "right":
183         return "left"
184     elif key == curses.KEY_UP and not direction == "down":
185         return "up"
186     elif key == curses.KEY_DOWN and not direction == "up":
187         return "down"
188     else:
189         return direction
190
191 def newFruit(snake, fruit, board):
192     height = len(board)
193     width = len(board[0])
194     newFruitY = random.randrange(1, height - 1)
195     newFruitX = random.randrange(1, width - 1)
196
197     if [newFruitY, newFruitX] == fruit:
198         return newFruit(snake, fruit, board)
199
200     for i in snake:
201         if [newFruitY, newFruitX] == i:
202             return newFruit(snake, fruit, board)
203
204     return [newFruitY, newFruitX]
205
206
207 #####
208 # STARTUP #
```

```
209 #####
210
211
212 if __name__ == "__main__":
213     main(sys.argv[1:])
214
215
216 # ex: set nu relativenumber list :
```