```
 1 # import pygame library
 2 import pygame
 3
 4 # initialise the pygame font
 5 pygame.font.init()
 6
 7 # Total window
 8 screen = pygame.display.set_mode((550, 650))
 9
10 # Title and Icon
11 pygame.display.set_caption("SUDOKU SOLVER USING
   BACKTRACKING")
12
13
14 x = 0
15 y = 0
16 dif = 500 / 9
17 val = 0
18 # Default Sudoku Board. Could be loaded in reverse,
   still troubleshooting that
19 grid = [
20         [7, 8, 0, 4, 0, 0, 1, 2, 0],
21         [6, 0, 0, 0, 7, 5, 0, 0, 9],
22         [0, 0, 0, 6, 0, 1, 0, 7, 8],
23         [0, 0, 7, 0, 4, 0, 2, 6, 0],
24         [0, 0, 1, 0, 5, 0, 9, 3, 0],
25         [9, 0, 4, 0, 6, 0, 0, 0, 5],
26         [0, 7, 0, 3, 0, 0, 0, 1, 2],
27         [1, 2, 0, 0, 0, 7, 4, 0, 0],
28         [0, 4, 9, 2, 0, 6, 0, 0, 7]
29        ]
30 # If grid gets loaded in reverse
31 '''[7, 6, 0, 0, 0, 9, 0, 1, 0],
32 [8, 0, 0, 0, 0, 0, 7, 2, 4],
33 [0, 0, 0, 7, 1, 4, 0, 0, 9],
34 [4, 0, 6, 0, 0, 0, 3, 0, 2],
35 [0, 7, 0, 4, 5, 6, 0, 0, 0],
36 [0, 5, 1, 0, 0, 0, 0, 7, 6],
37 [1, 0, 0, 2, 9, 0, 0, 4, 0],
38 [2, 0, 7, 6, 3, 0, 1, 0, 0],
39 [0, 9, 8, 0, 0, 5, 2, 0, 7]'''
```

```
40
41 # Load test fonts for future use
42 font1 = pygame.font.SysFont("comicsans", 40)
43 font2 = pygame.font.SysFont("comicsans", 20)
44
45
46 def get_cord(pos):
47     global x
48     x = pos[0] // dif
49     global y
50     y = pos[1] // dif
51
52
53 # Highlight the cell selected
54 def draw_box():
55     for i in range(2):
56         pygame.draw.line(screen, (255, 0, 0), (x *
   dif - 3, (y + i) * dif), (x * dif + dif + 3, (y + i
   ) * dif), 7)
57         pygame.draw.line(screen, (255, 0, 0), ((x + i
   ) * dif, y * dif), ((x + i) * dif, y * dif + dif), 7)
58
59
60 # Function to draw required lines for making Sudoku
   grid
61 def draw():
62     # Draw the lines
63     for i in range(9):
64         for j in range(9):
65             if grid[i][j] != 0:
66                 # Fill blue color in already numbered
    grid
67                 pygame.draw.rect(screen, (0, 153, 153
   ), (i * dif, j * dif, dif + 1, dif + 1))
68
69                 # Fill grid with default numbers
   specified
70                 text1 = font1.render(str(grid[i][j
   ]), 1, (0, 0, 0))
71                 screen.blit(text1, (i * dif + 15, j
    * dif + 5))
```

```
 72      # Draw lines horizontally and vertically to form
     grid
 73      for i in range(10):
 74          if i % 3 == 0:
 75              thick = 7
 76          else:
 77              thick = 1
 78          pygame.draw.line(screen, (0, 0, 0), (0, i *
    dif), (500, i * dif), thick)
 79          pygame.draw.line(screen, (0, 0, 0), (i * dif
    , 0), (i * dif, 500), thick)
 80
 81 # Fill value entered in cell
 82 def draw_val(val):
 83      text1 = font1.render(str(val), 1, (0, 0, 0))
 84      screen.blit(text1, (x * dif + 15, y * dif + 15))
 85
 86
 87 # Raise error when wrong value entered
 88 def raise_error1():
 89      text1 = font1.render("WRONG !!!", 1, (0, 0, 0))
 90      screen.blit(text1, (20, 570))
 91
 92
 93 def raise_error2():
 94      text1 = font1.render("Wrong !!! Not a valid Key"
    , 1, (0, 0, 0))
 95      screen.blit(text1, (20, 570))
 96
 97
 98 # Check if the value entered in board is valid
 99 def valid(m, i, j, val):
100      for it in range(9):
101          if m[i][it] == val:
102              return False
103          if m[it][j] == val:
104              return False
105      it = i // 3
106      jt = j // 3
107      for i in range(it * 3, it * 3 + 3):
108          for j in range(jt * 3, jt * 3 + 3):
```

```
109             if m[i][j] == val:
110                 return False
111     return True
112 
113 
114 # Solves the sudoku board using Backtracking
    Algorithm
115 def solve(grid, i, j):
116     while grid[i][j] != 0:
117         if i < 8:
118             i += 1
119         elif i == 8 and j < 8:
120             i = 0
121             j += 1
122         elif i == 8 and j == 8:
123             return True
124     pygame.event.pump()
125     for it in range(1, 10):
126         if valid(grid, i, j, it) == True:
127             grid[i][j] = it
128             global x, y
129             x = i
130             y = j
131             # white color background\
132             screen.fill((255, 255, 255))
133             draw()
134             draw_box()
135             pygame.display.update()
136             pygame.time.delay(20)
137             if solve(grid, i, j) == 1:
138                 return True
139             else:
140                 grid[i][j] = 0
141             # white color background\
142             screen.fill((255, 255, 255))
143 
144             draw()
145             draw_box()
146             pygame.display.update()
147             pygame.time.delay(50)
148     return False
```

```
149
150
151 # Display instruction for the game
152 def instruction():
153     text1 = font2.render("PRESS D TO RESET TO
    DEFAULT / R TO EMPTY", 1, (0, 0, 0))
154     text2 = font2.render("ENTER VALUES AND PRESS
    ENTER TO VISUALIZE", 1, (0, 0, 0))
155     screen.blit(text1, (12, 520))
156     screen.blit(text2, (10, 540))
157
158
159 # Display options when solved
160 def result():
161     text1 = font1.render("FINISHED PRESS R or D", 1
    , (0, 0, 0))
162     screen.blit(text1, (20, 570))
163
164
165 run = True
166 flag1 = 0
167 flag2 = 0
168 rs = 0
169 error = 0
170 # The loop thats keep the window running
171 while run:
172
173     # White color background
174     screen.fill((255, 255, 255))
175     # Loop through the events stored in event.get()
176     for event in pygame.event.get():
177         # Quit the game window
178         if event.type == pygame.QUIT:
179             run = False
180             # Get the mouse position to insert
    number
181         if event.type == pygame.MOUSEBUTTONDOWN:
182             flag1 = 1
183             pos = pygame.mouse.get_pos()
184             get_cord(pos)
185         # Get the number to be inserted if key
```

```
185 pressed
186         if event.type == pygame.KEYDOWN:
187             if event.key == pygame.K_LEFT:
188                 x -= 1
189                 flag1 = 1
190             if event.key == pygame.K_RIGHT:
191                 x += 1
192                 flag1 = 1
193             if event.key == pygame.K_UP:
194                 y -= 1
195                 flag1 = 1
196             if event.key == pygame.K_DOWN:
197                 y += 1
198                 flag1 = 1
199             if event.key == pygame.K_1:
200                 val = 1
201             if event.key == pygame.K_2:
202                 val = 2
203             if event.key == pygame.K_3:
204                 val = 3
205             if event.key == pygame.K_4:
206                 val = 4
207             if event.key == pygame.K_5:
208                 val = 5
209             if event.key == pygame.K_6:
210                 val = 6
211             if event.key == pygame.K_7:
212                 val = 7
213             if event.key == pygame.K_8:
214                 val = 8
215             if event.key == pygame.K_9:
216                 val = 9
217             if event.key == pygame.K_RETURN:
218                 flag2 = 1
219                 # If R pressed clear the sudoku
    board
220             if event.key == pygame.K_r:
221                 rs = 0
222                 error = 0
223                 flag2 = 0
224                 grid = [
```

```
225                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
226                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
227                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
228                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
229                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
230                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
231                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
232                     [0, 0, 0, 0, 0, 0, 0, 0, 0],
233                     [0, 0, 0, 0, 0, 0, 0, 0, 0]
234                 ]
235             # If D is pressed reset the board to
    default
236             if event.key == pygame.K_d:
237                 rs = 0
238                 error = 0
239                 flag2 = 0
240                 grid = [
241                         [0, 2, 0, 7, 0, 1, 0, 4, 0],
242                         [0, 5, 1, 0, 0, 9, 0, 0, 6],
243                         [0, 0, 3, 0, 0, 0, 8, 0, 0],
244                         [5, 0, 8, 2, 4, 0, 0, 1, 3],
245                         [3, 0, 2, 0, 9, 0, 0, 0, 0],
246                         [0, 0, 0, 0, 6, 0, 2, 0, 4],
247                         [6, 0, 0, 0, 1, 0, 0, 0, 9],
248                         [0, 0, 0, 9, 0, 8, 0, 0, 0],
249                         [0, 0, 0, 0, 7, 0, 5, 3, 0]
250                     ]
251     if flag2 == 1:
252         if solve(grid, 0, 0) == False:
253             error = 1
254         else:
255             rs = 1
256         flag2 = 0
257     if val != 0:
258         draw_val(val)
259         # print(x)
260         # print(y)
261         if valid(grid, int(x), int(y), val) == True:
262             grid[int(x)][int(y)] = val
263             flag1 = 0
264         else:
```

```
265                 grid[int(x)][int(y)] = 0
266                 raise_error2()
267             val = 0
268 
269         if error == 1:
270             raise_error1()
271         if rs == 1:
272             result()
273         draw()
274         if flag1 == 1:
275             draw_box()
276         instruction()
277 
278         # Update window
279         pygame.display.update()
280 
281 # Quit pygame window
282 pygame.quit()
```