```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
 1  #!/usr/bin/python
 2  # Shadowstone
 3  # Code Angel
 4
 5  import sys
 6  import os
 7  import pygame
 8  from pygame.locals import *
 9  import random
10
11  import people_items
12
13  # Define the colours
14  WHITE = (255, 255, 255)
15  BLACK = (0, 0, 0)
16  DARK_BLUE = (22, 68, 152)
17  GREEN = (40, 180, 40)
18
19  # Define constants
20  SCREEN_WIDTH = 640
21  SCREEN_HEIGHT = 480
22
23  PRINT_LINE_SIZE = 20
24  PRINT_BOX_MARGIN = 48
25
26  CARD_PADDING = 10
27  LEFT_MARGIN = 20
28  CARD_X = 220
29  DICE_X = 270
30  DICE_Y = 210
31  ITEM_X = [406, 482, 555]
32  CHOOSE_CARD_BOTTOM = 200
33
34  TURN_X = 258
35  PLAYER_TURN_Y = 165
36  OPPONENT_TURN_Y = 454
37
38  PAUSE_TIME = 2000
39  DICE_ROLL_TIME = 300
40
41  STRENGTH_ATTACK_MULTIPLIER = 2
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
42  DEXTERITY_ATTACK_MULTIPLIER = 1
43  WEAPON_ATTACK_MULTIPLIER = 3
44
45  STRENGTH_DEFENCE_MULTIPLIER = 0.5
46  DEXTERITY_DEFENCE_MULTIPLIER = 1
47  ARMOUR_DEFENCE_MULTIPLIER = 3
48
49  # Setup
50  os.environ['SDL_VIDEO_CENTERED'] = '1'
51  pygame.init()
52  game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
53  pygame.display.set_caption('Shadowstone')
54  clock = pygame.time.Clock()
55  font = pygame.font.SysFont('Helvetica', 16)
56  small_font = pygame.font.SysFont('Helvetica', 10)
57  large_font = pygame.font.SysFont('Helvetica', 32)
58
59  # Load images
60  background_image = people_items.load_image('general', 'background')
61  stats_box_image = people_items.load_image('general', 'stats_box')
62  message_box_image = people_items.load_image('general', 'message_box')
63  lge_message_box_image = people_items.load_image('general', 'lge_message_box')
64  player_box_image = people_items.load_image('general', 'player_box')
65
66  turn_image = people_items.load_image('general', 'turn')
67
68
69  def main():
70
71      # Initialise variables
72      level = 1
73      opponent_weapon = ''
74      player_weapon = ''
75
76      dice_images = people_items.get_dice_images()
77
78      # Set up player opponents and items dictionary
79      opponents = people_items.set_up_opponents()
80      characters = people_items.set_up_characters()
81
82      # items = people.set_up_items(item_images)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
    items = people_items.set_up_items()

    # Set up items
    player_items = people_items.get_player_items()

    game_screen.blit(background_image, [0, 0])
    display_choose_characters(characters)
    pygame.display.update()

    player_character = ''
    character_chosen = False

    # repeat until character 1-4 chosen
    while character_chosen is False:

        for event in pygame.event.get():
            key_pressed = pygame.key.get_pressed()

            if key_pressed[pygame.K_1]:
                player_character = characters.get('Player A')
                character_chosen = True

            elif key_pressed[pygame.K_2]:
                player_character = characters.get('Player B')
                character_chosen = True

            elif key_pressed[pygame.K_3]:
                player_character = characters.get('Player C')
                character_chosen = True

            elif key_pressed[pygame.K_4]:
                player_character = characters.get('Player D')
                character_chosen = True

            check_for_quit(event)

    # 3 Conquests as long as player has health
    player_health = player_character.get('health')

    while player_health > 0 and level < 4:

    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
124         # Display the scene description and wait for RETURN to be pressed
125         opponent = people_items.get_next_opponent(opponents, level)
126         opponent_items = people_items.get_opponent_items(level)
127
128         game_screen.blit(background_image, [0, 0])
129         display_description(level, opponent.get('name'), opponent.get('type'))
130         pygame.display.update()
131
132         wait_for_return()
133
134         # Display board
135         game_screen.blit(background_image, [0, 0])
136         display_board(opponent, player_character, opponent_items, player_items, items)
137
138         opponent_name = opponent.get('name')
139         board_message_box('You come face to face with ' + opponent_name + '.')
140
141         pygame.display.update()
142         wait_for_return()
143
144         # Play Turn
145         turn = random.choice(['Player', 'Opponent'])
146
147         opponent_health = opponent.get('health')
148
149         # Keep playing while both still have health
150         while opponent_health > 0 and player_health > 0:
151
152             if turn == 'Opponent':
153                 board_message_box(opponent_name + ' strikes...')
154                 game_screen.blit(turn_image, [TURN_X, OPPONENT_TURN_Y])
155             else:
156                 board_message_box('You attack ' + opponent_name + '...')
157                 game_screen.blit(turn_image, [TURN_X, PLAYER_TURN_Y])
158
159             pygame.display.update()
160             wait_for_return()
161
162             # Opponent attack
163             if turn == 'Opponent':
164
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
                opponent_weapon = get_opponent_weapon(opponent_items, items)
                player_shield = get_specific_item(player_items, items, 'Shield')
                attack_dice = get_attack_dice(opponent, opponent_weapon, player_character, player_shield)

                board_message_box(opponent.get('name') +
                                  ' attacks you with a ' +
                                  opponent_weapon.get('name') +
                                  ', needs to roll a '
                                  + str(attack_dice) +
                                  ' or above.')

            else:

                # Player attack
                board_message_box('Choose a weapon to attack (1-5)')
                pygame.display.update()

                # Get a valid weapon number (1-5)
                weapon_no = 0

                while weapon_no == 0:
                    for event in pygame.event.get():
                        key_pressed = pygame.key.get_pressed()

                        if key_pressed[pygame.K_1]:
                            weapon_no = 1
                        elif key_pressed[pygame.K_2]:
                            weapon_no = 2
                        elif key_pressed[pygame.K_3]:
                            weapon_no = 3
                        elif key_pressed[pygame.K_4]:
                            weapon_no = 4
                        elif key_pressed[pygame.K_5]:
                            weapon_no = 5

                        if weapon_no > 0:
                            if player_items[weapon_no - 1] == 'empty':
                                weapon_no = 0
                                board_message_box('No weapon in that slot. Choose a weapon to attack (1-5)')
                                pygame.display.update()

    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
206                         check_for_quit(event)
207
208                 player_weapon_chosen = player_items[weapon_no - 1]
209                 player_weapon = items.get(player_weapon_chosen)
210                 opponent_shield = get_specific_item(opponent_items, items, 'Shield')
211
212                 attack_dice = get_attack_dice(player_character, player_weapon, opponent, opponent_shield)
213
214                 board_message_box('You attack with a ' +
215                                   player_weapon.get('name') +
216                                   ' and need to roll a ' +
217                                   str(attack_dice) +
218                                   ' or above.')
219
220             pygame.display.update()
221             wait_for_return()
222
223             # Roll Attack Chance
224             board_message_box('')
225
226             chance_roll = random.randint(1, 20)
227             game_screen.blit(dice_images[chance_roll - 1], [DICE_X, DICE_Y])
228             pygame.display.update()
229             wait_for_return()
230
231             # Attack is successful
232             if chance_roll >= attack_dice:
233                 board_message_box('Attack succeeded, rolling for health damage...')
234                 pygame.display.update()
235                 wait_for_return()
236
237                 if turn == 'Opponent':
238                     max_damage = opponent_weapon.get('attack')
239                 else:
240                     max_damage = player_weapon.get('attack')
241
242                 board_message_box('')
243
244                 # Roll for damage
245                 damage_roll = random.randint(1, max_damage)
246                 game_screen.blit(dice_images[damage_roll - 1], [DICE_X, DICE_Y])
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
247
248                 pygame.display.update()
249                 wait_for_return()
250
251                 # Reduce health points
252                 if turn == 'Opponent':
253                     player_health -= damage_roll
254                     player_character['health'] = player_health
255                     board_message_box('Your health takes a damage of ' + str(damage_roll) + '.')
256
257                 else:
258                     opponent_health -= damage_roll
259                     opponent['health'] = opponent_health
260                     board_message_box(opponent_name + ' takes a damage of ' + str(damage_roll) + ' to health.')
261
262                 pygame.display.update()
263                 wait_for_return()
264
265                 # If both players still alive, random test for weapon or armour damage
266                 if opponent_health > 0 and player_health > 0:
267
268                     board_message_box('Checking for any weapon or armour damage...')
269                     pygame.display.update()
270                     wait_for_return()
271
272                     if turn == 'Opponent':
273                         damage_item = random.choice(player_items)
274                     else:
275                         damage_item = random.choice(opponent_items)
276
277                     if damage_item == 'hands' or damage_item == 'empty':
278
279                         # If empty slot or hands slot selected, no damage
280                         board_message_box('No weapon or armour damage inflicted.')
281                         pygame.display.update()
282                         wait_for_return()
283
284                     else:
285
286                         # Item is damaged so remove it
287                         damage_item_name = items.get(damage_item).get('name')
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
                            board_message_box(damage_item_name + ' is destroyed!')

                            if turn == 'Opponent':
                                position = player_items.index(damage_item)
                                player_items[position] = 'empty'
                            else:
                                position = opponent_items.index(damage_item)
                                opponent_items[position] = 'empty'

                            pygame.display.update()
                            wait_for_return()

                else:

                    # Attack has failed
                    board_message_box('Attack failed!')

                    pygame.display.update()
                    wait_for_return()

                # Is player or opponent health 0 or less?
                if player_health <= 0:
                    board_message_box('You have been defeated by ' + opponent.get('name') + '!!!!')
                elif opponent_health <= 0:

                    # Boost player health with random value
                    health_boost = random.randint(1, 3) * level

                    board_message_box('You have defeated ' +
                                      opponent.get('name') +
                                      '. Your health is boosted by ' +
                                      str(health_boost) + '.')

                    player_character['health'] = player_health + health_boost

                    pygame.display.update()
                    wait_for_return()

                    # Give player any gold the opponent had
                    gold = opponent.get('gold')
                    player_character['gold'] = player_character.get('gold') + gold
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
329
330                 reward_item = people_items.win_new_item(level, items)
331                 reward_item_name = items.get(reward_item).get('name')
332                 reward_item_type = items.get(reward_item).get('type')
333
334                 board_message_box('You have won ' +
335                                   str(gold) +
336                                   ' gold pieces and also receive a ' +
337                                   reward_item_name + '.')
338
339                 # If reward is a shield, replace existing shield
340                 if reward_item_type == 'shield':
341                     player_items[len(player_items) - 1] = reward_item
342                 else:
343
344                     # If reward is not a shield, find first empty slot
345                     for item_counter, inventory_item in enumerate(player_items):
346                         if inventory_item == 'empty':
347                             player_items[item_counter] = reward_item
348                             break
349
350                 pygame.display.update()
351                 wait_for_return()
352
353                 level += 1
354
355             else:
356                 # Reverse turns
357                 if turn == 'Opponent':
358                     turn = 'Player'
359                 else:
360                     turn = 'Opponent'
361
362             game_screen.blit(background_image, [0, 0])
363             display_board(opponent, player_character, opponent_items, player_items, items)
364             board_message_box('')
365             pygame.display.update()
366
367     # End of Game
368     game_screen.blit(background_image, [0, 0])
369
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
    if player_health > 0:
        final_gold = player_character.get('gold')
        overlay_message(['In battle you are victorious.'
                         'You have been crowned the Champion of Shadowstone.',
                         '',
                         'You have earned ' + str(final_gold) + ' gold pieces as your reward.'])
    else:
        overlay_message(['Like many who came before, you have fallen in the ancient city of Shadowstone.',
                         '',
                         'There will be many others who foolishly follow in your steps.'])

    pygame.display.update()
    wait_for_return()


# Check if the user has tried to quit
def check_for_quit(event):
    if event.type == QUIT:
        pygame.quit()
        sys.exit()


# Wait until return key is pressed
def wait_for_return():

    return_pressed = False
    while return_pressed is False:

        for event in pygame.event.get():
            key_pressed = pygame.key.get_pressed()

            if key_pressed[pygame.K_RETURN]:
                return_pressed = True

            check_for_quit(event)


# Display the 4 possible player characters
def display_choose_characters(characters):

    message_box_y = 360
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

 

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
411
412         # set up cards and names
413         player_a_card_image = characters.get('Player A').get('image')
414         player_b_card_image = characters.get('Player B').get('image')
415         player_c_card_image = characters.get('Player C').get('image')
416         player_d_card_image = characters.get('Player D').get('image')
417
418         char_card_width = get_image_width(player_a_card_image)
419
420         # display cards
421         spacing = (SCREEN_WIDTH - char_card_width * 4) / 5
422         col_1_x = spacing
423         col_2_x = char_card_width + 2 * spacing
424         col_3_x = 2 * char_card_width + 3 * spacing
425         col_4_x = 3 * char_card_width + 4 * spacing
426
427         player_a_y = CHOOSE_CARD_BOTTOM - get_image_height(player_a_card_image)
428         player_b_y = CHOOSE_CARD_BOTTOM - get_image_height(player_b_card_image)
429         player_c_y = CHOOSE_CARD_BOTTOM - get_image_height(player_c_card_image)
430         player_d_y = CHOOSE_CARD_BOTTOM - get_image_height(player_d_card_image)
431         game_screen.blit(player_a_card_image, [col_1_x, player_a_y])
432         game_screen.blit(player_b_card_image, [col_2_x, player_b_y])
433         game_screen.blit(player_c_card_image, [col_3_x, player_c_y])
434         game_screen.blit(player_d_card_image, [col_4_x, player_d_y])
435
436         # Boxes underneath with numbers
437         player_box_y = CHOOSE_CARD_BOTTOM + CARD_PADDING
438         player_box_text_y = CHOOSE_CARD_BOTTOM + CARD_PADDING + 5
439
440         game_screen.blit(player_box_image, [col_1_x, player_box_y])
441         game_screen.blit(player_box_image, [col_2_x, player_box_y])
442         game_screen.blit(player_box_image, [col_3_x, player_box_y])
443         game_screen.blit(player_box_image, [col_4_x, player_box_y])
444
445         # Display the numbers
446         centre_text_with_object('Press 1', col_1_x, char_card_width, player_box_text_y, BLACK)
447         centre_text_with_object('Press 2', col_2_x, char_card_width, player_box_text_y, BLACK)
448         centre_text_with_object('Press 3', col_3_x, char_card_width, player_box_text_y, BLACK)
449         centre_text_with_object('Press 4', col_4_x, char_card_width, player_box_text_y, BLACK)
450
451         # Main message box
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
    message = 'Choose your character (1-4)'
    message_box_width = SCREEN_WIDTH - 2 * LEFT_MARGIN
    game_screen.blit(message_box_image, [LEFT_MARGIN, message_box_y])
    centre_text_with_object(message, LEFT_MARGIN, message_box_width, message_box_y + 28, BLACK)


# Centre any piece of text with a given object
def centre_text_with_object(display_text, object_x, object_width, text_y_coord, text_color):

    text = font.render(display_text, True, text_color)
    text_rect = text.get_rect()
    text_x_coord = object_x + (object_width - text_rect.width) / 2

    game_screen.blit(text, [text_x_coord, text_y_coord])


# Display a level description
def display_description(level, next_opponent_name, next_opponent_type):

    all_levels = [['You arrive at the Kings Tavern, an old inn at the edge of the city. After a light ',
                   'refreshment, you notice a shadow from behind and when turning around you are ',
                   'met by '],

                  ['You walk out into the town square. The low sun casts shadows all around as ',
                   'you edge your way towards the old cathedral at the far end of the square. As your',
                   'eyes adjust you notice a figure moving in a doorway. It is '],

                  ['The doors to the old cathedral swing open. Slowly you edge your way through the  ',
                   'town square to the entrance. It is cold and gloomy inside but through the darkness ',
                   'you just make out someone waiting in the far corner. It is '],
                  ]

    level_text = all_levels[level - 1]
    last_line = len(level_text) - 1
    level_text[last_line] += next_opponent_name + ', the ' + next_opponent_type + '.'
    overlay_message(level_text)


# Display a message on screen taking in a list of lines of text to be displayed
def overlay_message(text_to_display):

    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

 

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
493     box_top = 180
494     box_left = 50
495
496     game_screen.blit(lge_message_box_image, [box_left, box_top])
497
498     number_of_lines = len(text_to_display)
499
500     for line_number, line in enumerate(text_to_display):
501         display_line = font.render(line, True, BLACK)
502
503         message_loc = [PRINT_BOX_MARGIN + PRINT_LINE_SIZE, box_top + PRINT_LINE_SIZE + line_number *
504 PRINT_LINE_SIZE]
505         game_screen.blit(display_line, message_loc)
506
507     # Add a final line of text to tell the user to hit RETURN
508     final_line = font.render('[Hit RETURN to continue]', True, BLACK)
509     final_line_rect = final_line.get_rect()
510     final line loc = [(SCREEN WIDTH - final line rect.width) / 2,
511                       box_top + 2 * PRINT_LINE_SIZE + number_of_lines * PRINT_LINE_SIZE]
512
513     game_screen.blit(final_line, final_line_loc)
514
515
516 # Display a centred message across the middle band of the board
517 def board_message_box(message):
518     message_box_width = SCREEN_WIDTH - 2 * LEFT_MARGIN
519     game_screen.blit(message_box_image, [LEFT_MARGIN, SCREEN_HEIGHT / 2 - 2 * PRINT_LINE_SIZE])
520     centre_text_with_object(message, LEFT_MARGIN, message_box_width, (SCREEN_HEIGHT - PRINT_LINE_SIZE) / 2, BLACK)
521
522
523 # Display the main board
524 def display_board(opponent, player, opponent_items, player_items, items):
525     player_card_image = player.get('image')
526     opponent_card_image = opponent.get('image')
527     empty_item_image = items.get('empty').get('image')
528     item_width = get_image_width(empty_item_image)
529     item_height = get_image_height(empty_item_image)
530
531     # display opponent
532     opponent_card_y = 292
533     game_screen.blit(opponent_card_image, [CARD_X, opponent_card_y])
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

 

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
534
535     # display player
536     player_card_y = CARD_PADDING
537     game_screen.blit(player_card_image, [CARD_X, player_card_y])
538
539     # Draw display boxes
540     box_left = LEFT_MARGIN
541
542     game_screen.blit(stats_box_image, [box_left, opponent_card_y])
543     game_screen.blit(stats_box_image, [box_left, player_card_y])
544
545     # Display stats
546     opponent_name = opponent.get('name')
547     display_stats(opponent_name, opponent_card_y, 0)
548
549     opponent_hea_text = 'Health: ' + str(opponent.get('health'))
550     display_stats(opponent_hea_text, opponent_card_y, 2)
551
552     opponent_str_text = 'Strength: ' + str(opponent.get('strength'))
553     display_stats(opponent_str_text, opponent_card_y, 3)
554
555     opponent_dex_text = 'Dexterity: ' + str(opponent.get('dexterity'))
556     display_stats(opponent_dex_text, opponent_card_y, 4)
557
558     opponent_mag_text = 'Magic: ' + str(opponent.get('magic'))
559     display_stats(opponent_mag_text, opponent_card_y, 5)
560
561     opponent_gold_text = 'Gold: ' + str(opponent.get('gold'))
562     display_stats(opponent_gold_text, opponent_card_y, 6)
563
564     display_stats('You', player_card_y, 0)
565
566     player_hea_text = 'Health: ' + str(player.get('health'))
567     display_stats(player_hea_text, player_card_y, 2)
568
569     player_str_text = 'Strength: ' + str(player.get('strength'))
570     display_stats(player_str_text, player_card_y, 3)
571
572     player_dex_text = 'Dexterity: ' + str(player.get('dexterity'))
573     display_stats(player_dex_text, player_card_y, 4)
574
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
575     player_mag_text = 'Magic: ' + str(player.get('magic'))
576     display_stats(player_mag_text, player_card_y, 5)
577
578     player_gold_text = 'Gold: ' + str(player.get('gold'))
579     display_stats(player_gold_text, player_card_y, 6)
580
581     # Display player items
582     for item_no, player_item in enumerate(player_items):
583         item = items.get(player_item)
584         item_image = item.get('image')
585         item_type = item.get('type')
586
587         item_value = 0
588         if item_type == 'weapon':
589             item_value = item.get('attack')
590         elif item_type == 'shield' or item_type == 'armour':
591             item_value = item.get('defence')
592
593         # If item_no is 0 - 2 then display on first line
594         if item_no < 3:
595             item_x = ITEM_X[item_no]
596             item_y = player_card_y
597
598         # If item no is 3 - 5 then display on second line
599         else:
600             item_x = ITEM_X[item_no - 3]
601             item_y = player_card_y + item_height + PRINT_LINE_SIZE
602
603         game_screen.blit(item_image, [item_x, item_y])
604
605         # Display the attack / defence value of the item if it has one
606         if item_value > 0:
607             display_item_value(item_x + item_width, item_y + item_height, item_value)
608
609         # If item_no is between 0 & 4 it's a weapon so display the item number (+1 to make it 1-5)
610         if item_no < 5:
611             centre_text_with_object(str(item_no + 1), item_x, item_width, item_y + item_height, BLACK)
612
613     # Display opponent items
614     for opponent_item_no, opponent_item in enumerate(opponent_items):
615         opp_item = items.get(opponent_item)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
            opp_item_image = opp_item.get('image')
            opp_item_type = opp_item.get('type')

            opp_item_value = 0
            if opp_item_type == 'weapon':
                opp_item_value = opp_item.get('attack')
            elif opp_item_type == 'shield' or opp_item_type == 'armour':
                opp_item_value = opp_item.get('defence')

            # If item_no is 0 - 2 then display on first line
            if opponent_item_no < 3:
                item_x = ITEM_X[opponent_item_no]
                item_y = opponent_card_y

            # If item no is 3 - 5 then display on second line
            else:
                item_x = ITEM_X[opponent_item_no - 3]
                item_y = opponent_card_y + item_height + PRINT_LINE_SIZE

            game_screen.blit(opp_item_image, [item_x, item_y])

            # Display the attack / defence value of the item if it has one
            if opp_item_value > 0:
                display_item_value(item_x + item_width, item_y + item_height, opp_item_value)


    # Select a random weapon for the opponent to attack with
    def get_opponent_weapon(opponent_items, items):

        # Select a random item from the opponent's list of items
        random_item_name = random.choice(opponent_items)
        random_item = items.get(random_item_name)
        random_item_type = random_item.get('type')

        # Make sure it is a weapon, if not keep selecting another random item until a weapon is chosen
        while random_item_type != 'weapon':
            random_item_name = random.choice(opponent_items)
            random_item = items.get(random_item_name)
            random_item_type = random_item.get('type')

        return random_item
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
657
658
659 # Get a specific item type e.g. weapon, shield etc
660 def get_specific_item(character_items, all_items, search_item_type):
661
662     specific_item = 'Not found'
663     for item_name in character_items:
664         item = all_items.get(item_name)
665         item_type = item.get('type')
666         if item_type == search_item_type:
667             specific_item = item
668
669     return specific_item
670
671
672 # Work out the number on the dice that needs to be rolled if the attack is to be successful
673 def get_attack_dice(attacker, attack_weapon, defender, defence_shield):
674
675     # Calculate an attack score
676     # Max is 120
677     # Divide by 6 to get an attack score out of 20
678     attacker_strength = attacker.get('strength')
679     attacker_dexterity = attacker.get('dexterity')
680     attacker_weapon_attack = attack_weapon.get('attack')
681
682     attack_score = (attacker_strength * STRENGTH_ATTACK_MULTIPLIER +
683                     attacker_dexterity * DEXTERITY_ATTACK_MULTIPLIER +
684                     attacker_weapon_attack * WEAPON_ATTACK_MULTIPLIER)
685
686     attack_score = int(attack_score / 6)
687
688     # Calculate a defence score
689     # Max is 90
690     # Divide by 6 to get a defence score out of 15
691     defender_strength = defender.get('strength')
692     defender_dexterity = defender.get('dexterity')
693
694     if defence_shield == 'Not found':
695         armour_defence = 0
696     else:
697         armour_defence = defence_shield.get('defence')
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
698
699     defence_score = (defender_strength * STRENGTH_DEFENCE_MULTIPLIER +
700                      defender_dexterity * DEXTERITY_DEFENCE_MULTIPLIER +
701                      armour_defence * ARMOUR_DEFENCE_MULTIPLIER)
702
703     defence_score = int(defence_score / 6)
704
705     # If attack and defence are the same, looking to roll a 10 or more
706     # If attack is much stronger than defence, then a lower (than 10) score required
707     # If defence is much stronger than attack, then a higher (than 10) score required
708     dice_score = 10 - int((attack_score - defence_score) / 2)
709     if dice_score > 20:
710         dice_score = 20
711     elif dice_score < 1:
712         dice_score = 1
713
714     return dice_score
715
716
717 # Format the display of the board game stats, aligned with the relevant card (horizontally)
718 def display_stats(display_text, box_top, line_no):
719     text_x_coord = LEFT_MARGIN + CARD_PADDING
720     text_y_coord = box_top + 2 * CARD_PADDING + PRINT_LINE_SIZE * line_no
721
722     text = font.render(display_text, True, BLACK)
723     game_screen.blit(text, [text_x_coord, text_y_coord])
724
725
726 # Display the item attack / defence value in the bottom right of each item box
727 def display_item_value(item_box_right, item_box_bottom, item_score):
728     box_size = 14
729
730     # Draw a box
731     item_score_rect = (item_box_right - box_size, item_box_bottom - box_size, box_size, box_size)
732     pygame.draw.rect(game_screen, GREEN, item_score_rect)
733
734     # Display the text, centred in the box
735     text = small_font.render(str(item_score), True, WHITE)
736     text_rect = text.get_rect()
737     text_width = text_rect.width
738     text_height = text_rect.height
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

 

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
739
740         text_x_coord = item_box_right - (box_size + text_width) / 2
741         text_y_coord = item_box_bottom - (box_size + text_height) / 2
742
743         game_screen.blit(text, [text_x_coord, text_y_coord])
744
745
746     # Calculate the width of an image
747     def get_image_width(image):
748         image_width = image.get_rect().width
749         return image_width
750
751
752     # Calculate the height of an image
753     def get_image_height(image):
754         image_height = image.get_rect().height
755         return image_height
756
757
758     if __name__ == '__main__':
759         main()
```

|0 |1 |2 |3 |4 |5 |6 |7 |8

 