

# PyTetris 解答集

Swimmy 高田馬場校 有志

2024

## 1 chapter6

### 1.1 6.2

ブロックを表示/移動するところまでのコードです。

ソースコード 1: main.py

---

```
1 import pygame
2 from tetris import Board
3 # chapter 6 ブロックを動かす
4 # OBlockをインポートします
5 from tetris import OBlock
6
7 def main():
8     # Create the board
9     board = Board(tile_size=30)
10
11     # Initialize pygame
12     pygame.init()
13     screen = pygame.display.set_mode(board.window_size())
14     pygame.display.set_caption("Tetris")
15     clock = pygame.time.Clock()
16
17     # chapter 6 ブロックを動かす
18     # とりあえず0ブロックを作っておきます
19     # 将来はランダムに作ります
20     board.moving_block = OBlock()
21
22     while True:
```

```

23     # Draw the board
24     board.draw(screen)
25
26     # Handle events
27     for event in pygame.event.get():
28         if event.type == pygame.QUIT:
29             return
30     # key handling
31     keys = pygame.key.get_pressed()
32     if keys[pygame.K_LEFT]:
33         board.cursor.move_left()
34         pygame.time.wait(100)
35     if keys[pygame.K_RIGHT]:
36         board.cursor.move_right()
37         pygame.time.wait(100)
38     if keys[pygame.K_DOWN]:
39         board.cursor.move_down()
40         pygame.time.wait(100)
41     # Update the display
42     pygame.display.update()
43
44     # Tick
45     clock.tick(60)
46
47 if __name__ == "__main__":
48     main()

```

---

## ソースコード 2: tetris.py

---

```

1 # tetris using pygame
2 import pygame
3
4 # Constants
5 WIDTH = 10
6 HEIGHT = 22
7 TILE_SIZE = 30
8
9 # Colors
10 WHITE = (255, 255, 255)
11 BLACK = (0, 0, 0)
12 GRAY = (128, 128, 128)
13 CURSOR_COLOR = (60, 60, 60)
14 CYAN = (0, 255, 255)

```

```

15 BLUE = (0, 0, 255)
16 ORANGE = (255, 165, 0)
17 YELLOW = (255, 255, 0)
18 GREEN = (0, 128, 0)
19 PURPLE = (128, 0, 128)
20 RED = (255, 0, 0)
21
22 class Board:
23     def __init__(self, tile_size):
24         # 作る時にタイルサイズを指定する
25         self.board = [[BLACK for _ in range(WIDTH)] for _
26                        in range(HEIGHT)]
27         self.TILE_SIZE = tile_size
28
29         # Cursor
30         self.cursor = Cursor()
31
32         # chapter6 ブロックを動かす
33         # 新たにmoving_blockという変数を追加します
34         # 最初は何も動かしていないので
35         # Noneを入れておきます
36         # わからない時は3.3.1を見てみましょう
37         self.moving_block = None
38
39     def draw(self, screen):
40         for y in range(HEIGHT):
41             for x in range(WIDTH):
42                 pygame.draw.rect(screen, self.board[y][x],
43                                (x * self.TILE_SIZE, y * self.
44                                 TILE_SIZE, self.TILE_SIZE, self.
45                                 TILE_SIZE))
46
47         # Draw the cursor
48         for y in range(HEIGHT):
49             if self.board[y][self.cursor.x] == BLACK:
50                 pygame.draw.rect(screen, CURSOR_COLOR, (
51                     self.cursor.x * self.TILE_SIZE, y *
52                     self.TILE_SIZE, self.TILE_SIZE, self.
53                     TILE_SIZE))
54
55         # chapter6 ブロックを動かす
56         # ブロックを動かす処理を追加します

```

```

49     # 動かすブロックがNoneではない時
50     if self.moving_block is not None:
51         # ブロックの情報を取得します
52         # 座標がリストになって帰ってくるはずなので変数
           に入れておきます
53         datas = self.moving_block.block_info(self.
           cursor)
54
55         # 何色で塗るかは
           Block の color という変数に入っています
56         # 一旦変数にコピーしておきます
57         color = self.moving_block.color
58
59         # ブロックからもらった座標リストを一つずつ見て
           いきます
60         # リストを一つずつ見ていくループは
           for 文を使います
61         for data in datas:
62             #
           for 文で決めた変数に座標が一つずつ入ってループされます
63
64         # それぞれの座標の場所を塗ることが目標です
65
66         # その 0 番目に
           x 座標が入っているのでそれを x に入れます
67         # 1 番目に
           y 座標が入っているのでそれを y に入れます
68         x = data[0]
69         y = data[1]
70
71         # マスを塗りつぶします。この行はわからなく
           て大丈夫ですが
72         # 色を入れた変数の名前を置き換えてください
           pygame.draw.rect(screen, color, (x * self.
           TILE_SIZE, y * self.TILE_SIZE, self.
           TILE_SIZE, self.TILE_SIZE))
73
74     # あとは前回と同じです
75     # 順番を間違えるとブロックが線を上書きして表示され
           てしまうので
76     # この順番です。書き足す位置を間違えないように気を
           つけてください

```

```

77         # Draw the grid
78         for y in range(HEIGHT):
79             pygame.draw.line(screen, GRAY, (0, y * self.
                TILE_SIZE), (WIDTH * self.TILE_SIZE, y *
                self.TILE_SIZE))
80         for x in range(WIDTH):
81             pygame.draw.line(screen, GRAY, (x * self.
                TILE_SIZE, 0), (x * self.TILE_SIZE, HEIGHT
                * self.TILE_SIZE))
82
83     def window_size(self):
84         return (WIDTH * TILE_SIZE, HEIGHT * TILE_SIZE)
85
86     class OBlock:
87         def __init__(self):
88             self.color = YELLOW
89
90         def block_info(self, cursor):
91             result = []
92
93             x = cursor.x
94             y = cursor.y
95
96             result.append((x, y))
97             result.append((x + 1, y))
98             result.append((x, y + 1))
99             result.append((x + 1, y + 1))
100
101             return result
102
103     def rotate(self):
104         pass
105
106     class Cursor:
107         def __init__(self):
108             self.x = WIDTH // 2
109             self.y = 0
110         def move_left(self):
111             self.x = max(0, self.x - 1)
112         def move_right(self):
113             self.x = min(WIDTH - 1, self.x + 1)
114         def move_down(self):

```

```
115         self.y = min(HEIGHT - 1, self.y + 1)
116     def move_up(self):
117         self.y = max(0, self.y - 1)
```

---

## 1.2 6.3

ブロックの動きを制限するところまでのコードです。

ソースコード 3: main.py

---

```
1 import pygame
2 from tetris import Board
3 # chapter 6 ブロックを動かす
4 # OBlockをインポートします
5 from tetris import OBlock
6
7 def main():
8     # Create the board
9     board = Board(tile_size=30)
10
11     # Initialize pygame
12     pygame.init()
13     screen = pygame.display.set_mode(board.window_size())
14     pygame.display.set_caption("Tetris")
15     clock = pygame.time.Clock()
16
17     # chapter 6 ブロックを動かす
18     # とりあえず0ブロックを作っておきます
19     # 将来はランダムに作ります
20     board.moving_block = OBlock()
21
22     while True:
23         # Draw the board
24         board.draw(screen)
25
26         # Handle events
27         for event in pygame.event.get():
28             if event.type == pygame.QUIT:
29                 return
30         # key handling
31         keys = pygame.key.get_pressed()
32         if keys[pygame.K_LEFT]:
```

```

33         board.cursor_move_left()
34         pygame.time.wait(100)
35     if keys[pygame.K_RIGHT]:
36         board.cursor_move_right()
37         pygame.time.wait(100)
38     if keys[pygame.K_DOWN]:
39         board.cursor_move_down()
40         pygame.time.wait(100)
41     # Update the display
42     pygame.display.update()
43
44     # Tick
45     clock.tick(60)
46
47 if __name__ == "__main__":
48     main()

```

---

#### ソースコード 4: tetris.py

---

```

1  # tetris using pygame
2  import pygame
3
4  # Constants
5  WIDTH = 10
6  HEIGHT = 22
7  TILE_SIZE = 30
8
9  # Colors
10 WHITE = (255, 255, 255)
11 BLACK = (0, 0, 0)
12 GRAY = (128, 128, 128)
13 CURSOR_COLOR = (60, 60, 60)
14 CYAN = (0, 255, 255)
15 BLUE = (0, 0, 255)
16 ORANGE = (255, 165, 0)
17 YELLOW = (255, 255, 0)
18 GREEN = (0, 128, 0)
19 PURPLE = (128, 0, 128)
20 RED = (255, 0, 0)
21
22 class Board:
23     def __init__(self, tile_size):
24         # 作る時にタイルサイズを指定する

```

```

25     self.board = [[BLACK for _ in range(WIDTH)] for _
26                     in range(HEIGHT)]
27     self.TILE_SIZE = tile_size
28
29     # Cursor
30     self.cursor = Cursor()
31
32     # chapter6 ブロックを動かす
33     # 新たにmoving_blockという変数を追加します
34     # 最初は何も動かしていないので
35     # Noneを入れておきます
36     # わからない時は3.3.1を見てみましょう
37     self.moving_block = None
38
39     def draw(self, screen):
40         for y in range(HEIGHT):
41             for x in range(WIDTH):
42                 pygame.draw.rect(screen, self.board[y][x],
43                                 (x * self.TILE_SIZE, y * self.
44                                 TILE_SIZE, self.TILE_SIZE, self.
45                                 TILE_SIZE))
46
47         # Draw the cursor
48         for y in range(HEIGHT):
49             if self.board[y][self.cursor.x] == BLACK:
50                 pygame.draw.rect(screen, CURSOR_COLOR, (
51                     self.cursor.x * self.TILE_SIZE, y *
52                     self.TILE_SIZE, self.TILE_SIZE, self.
53                     TILE_SIZE))
54
55         # chapter6 ブロックを動かす
56         # ブロックを動かす処理を追加します
57         # 動かすブロックがNoneではない時
58         if self.moving_block is not None:
59             # ブロックの情報を取得します
60             # 座標がリストになって帰ってくるはずなので変数
61             # に入れておきます
62             datas = self.moving_block.block_info(self.
63                 cursor)
64
65         # 何色で塗るかは
66         # Blockのcolorという変数に入っています

```



```

56         # 一旦変数にコピーしておきます
57         color = self.moving_block.color
58
59         # ブロックからもらった座標リストを一つずつ見て
           いきます
60         # リストを一つずつ見ていくループは
           for 文を使います
61         for data in datas:
62             #
               for 文で決めた変数に座標が一つずつ入ってループされます
63
64             # それぞれの座標の場所を塗ることが目標です
65
66             # その0番目に
               x 座標が入っているのでそれを x に入れます
67             # 1番目に
               y 座標が入っているのでそれを y に入れます
68             x = data[0]
69             y = data[1]
70
71             # マスを塗りつぶします。この行はわからなく
               て大丈夫ですが
72             # 色を入れた変数の名前を置き換えてください
           pygame.draw.rect(screen, color, (x * self.
               TILE_SIZE, y * self.TILE_SIZE, self.
               TILE_SIZE, self.TILE_SIZE))
73
74         # あとは前回と同じです
75         # 順番を間違えるとブロックが線を上書きして表示され
           てしまうので
76         # この順番です。書き足す位置を間違えないように気を
           つけてください
77         # Draw the grid
78         for y in range(HEIGHT):
79             pygame.draw.line(screen, GRAY, (0, y * self.
               TILE_SIZE), (WIDTH * self.TILE_SIZE, y *
               self.TILE_SIZE))
80         for x in range(WIDTH):
81             pygame.draw.line(screen, GRAY, (x * self.
               TILE_SIZE, 0), (x * self.TILE_SIZE, HEIGHT
               * self.TILE_SIZE))
82

```

```

83     def window_size(self):
84         return (WIDTH * TILE_SIZE, HEIGHT * TILE_SIZE)
85
86     def cursor_move_left(self):
87         self.cursor.move_left(self.moving_block, self)
88
89     def cursor_move_right(self):
90         self.cursor.move_right(self.moving_block, self)
91
92     def cursor_move_down(self):
93         self.cursor.move_down(self.moving_block, self)
94
95     def cursor_move_up(self):
96         self.cursor.move_up(self.moving_block, self)
97
98     class OBlock:
99         def __init__(self):
100             self.color = YELLOW
101
102         def block_info(self, cursor):
103             result = []
104
105             x = cursor.x
106             y = cursor.y
107
108             result.append((x, y))
109             result.append((x + 1, y))
110             result.append((x, y + 1))
111             result.append((x + 1, y + 1))
112
113             return result
114
115     def rotate(self):
116         pass
117
118     def can_go_left(self, cursor, board_data):
119         if cursor.x == 0:
120             return False
121         if board_data[cursor.y][cursor.x - 1] != BLACK:
122             return False
123         if board_data[cursor.y + 1][cursor.x - 1] !=
            BLACK:

```

```

124         return False
125     return True
126
127     def can_go_right(self, cursor, board_data):
128         if cursor.x == WIDTH - 2:
129             return False
130         if board_data[cursor.y][cursor.x + 2] != BLACK:
131             return False
132         if board_data[cursor.y + 1][cursor.x + 2] !=
            BLACK:
133             return False
134         return True
135
136     def can_go_down(self, cursor, board_data):
137         if cursor.y == HEIGHT - 2:
138             return False
139         if board_data[cursor.y + 2][cursor.x] != BLACK:
140             return False
141         if board_data[cursor.y + 2][cursor.x+1] != BLACK:
142             return False
143         return True
144
145     def can_go_up(self, cursor, board_data):
146         if cursor.y == 0:
147             return False
148         if board_data[cursor.y - 1][cursor.x] != BLACK:
149             return False
150         if board_data[cursor.y - 1][cursor.x + 1] !=
            BLACK:
151             return False
152         return True
153
154     class Cursor:
155         def __init__(self):
156             self.x = WIDTH // 2
157             self.y = 0
158         def move_left(self, block, board):
159             if block.can_go_left(self, board.board):
160                 self.x = max(0, self.x - 1)
161         def move_right(self, block, board):
162             if block.can_go_right(self, board.board):
163                 self.x = min(WIDTH - 1, self.x + 1)

```

```
164     def move_down(self, block, board):
165         if block.can_go_down(self, board.board):
166             self.y = min(HEIGHT - 1, self.y + 1)
167     def move_up(self, block, board):
168         if block.can_go_up(self, board.board):
169             self.y = max(0, self.y - 1)
```

---