Coverage for **tests/unit/test_diffusion2d_functions.py**: 98%

60 statements | 59 run | 1 missing | 0 excluded

```
1  """
2  Tests for functions in class SolveDiffusion2D
3  """
4  import numpy as np
5  from diffusion2d import SolveDiffusion2D
6  from unittest import TestCase
7
8  class TestDiffusion2D(TestCase):
9      """
10     Test suite for TestDiffusion2D operations functions.
11     """
12     def setUp(self):
13         # plate size, mm
14         self.w = 12.
15         self.h = 15.
16         # intervals in x-, y- directions, mm
17         self.dx = 0.2
18         self.dy = 0.15
19         # Thermal diffusivity of steel, mm^2/s
20         self.D = 8.
21         # Initial cold temperature of square domain
22         self.T_cold = 250.
23         # Initial hot temperature of circular disc at the center
24         self.T_hot = 650.
25
26         # Timestep
27         self.dt = None
28
29
30     def test_initialize_domain(self):
31         """
32         Check function SolveDiffusion2D.initialize_domain
33         """
34         solver = SolveDiffusion2D()
35
36         # Expected
37         expected_nx = 60
38         expected_ny = 100
39
40         # Actual
41         solver.initialize_domain(w=self.w, h=self.h, dx=self.dx, dy=self.dy)
42
43         actual_nx = solver.nx
44         actual_ny = solver.ny
45
46         #Test
47         self.assertEqual(actual_nx,expected_nx)
48         self.assertEqual(actual_ny,expected_ny)
49
50     def test_initialize_physical_parameters(self):
51         """
52         Checks function SolveDiffusion2D.initialize_domain
53         """
54         solver = SolveDiffusion2D()
55
56         # Expected
57         expected_T_cold = 250
58         expected_T_hot = 650
59         expected_dt = 0.0009
60         #expected_dt = 0.0009000000000000002
61
62         # Actual
63         solver.dx = 0.2
64         solver.dy = 0.15
65         solver.initialize_physical_parameters(d=self.D, T_cold=self.T_cold, T_hot=self.T_hot)
66         actual_T_cold = solver.T_cold
67         actual_T_hot = solver.T_hot
68         actual_dt = solver.dt
69
70         # Test
71         self.assertEqual(expected_T_cold, actual_T_cold)
72         self.assertEqual(expected_T_hot, actual_T_hot)
73         #self.assertEqual(expected_dt, actual_dt)
74         self.assertAlmostEqual(expected_dt, actual_dt)
75
76
```

```
77      def test_set_initial_condition(self):
78          """
79          Checks function SolveDiffusion2D.get_initial_function
80          """
81          solver = SolveDiffusion2D()
82
83          # Paramters
84          T_cold = 250
85          T_hot = 650
86          nx = 50
87          ny = 70
88          dx = 0.2
89          dy = 0.15
90
91          # Expected
92          expected_u = T_cold * np.ones((nx, ny))
93
94          # Initial conditions - circle of radius r centred at (cx,cy) (mm)
95          r, cx, cy = 2, 5, 5
96          r2 = r ** 2
97          for i in range(nx):
98              for j in range(ny):
99                  p2 = (i * dx - cx) ** 2 + (j * dy - cy) ** 2
100                 if p2 < r2:
101                     expected_u[i, j] = T_hot
102
103         # Actual
104         solver.T_cold = T_cold
105         solver.T_hot = T_hot
106         solver.nx = nx
107         solver.ny = ny
108         solver.dx = dx
109         solver.dy = dy
110         actual_u = solver.set_initial_condition()
111
112         #self.assertTrue(np.testing.assert_array_equal(actual_u, expected_u))
113         self.assertTrue((actual_u == expected_u).all())
114         #self.np.testing.assert_array_equal(actual_u, expected_u)
115         #python3 -m unittest /home/sabri/UNI/Sem4/SimulationSoftwareEngineering/git-repos/SSE_Exercises/testing-python-
116
117
118
```