```
 1
 2 import math
 3
 4 class Board:
 5     """Grid board class
 6     Represent a two dimensional grid of items
 7     """
 8     def __init__(self, rows, cols, displaycol=9, empty_symbol='.'):
 9         """construct a board with specified rows and cols
10         displaytab can be set to display the board with a specified
11         number of columns so that items line up.
12         empty_symbol is the string that is displayed when a board
13         space is empty."""
14         self.rows = rows
15         self.cols = cols
16         self.displaycol = displaycol
17         self.empty_symbol = empty_symbol
18         # Generated 2D list representing an empty board
19         # in row-major order (rows indexed first)
20         self.board = \
21             [[None for c in range(cols)] for r in range(rows)]
22
23     def place(self, row, col, item):
24         "place an item"
25         self.board[row][col] = item
26
27     def get(self, row, col):
28         "get an item"
29         return self.board[row][col]
30
31     def get_rows(self):
32         "get_rows - return number of rows"
33         return self.rows
34
35     def get_cols(self):
36         "get_cols - return number of columns"
37         return self.cols
38
39     def __repr__(self):
40         "return a representation of the board"
41
42         lines = []
43         # NOTE:  This section uses Python's format strings (see string
44         # operations in the standard library).  Most times these are overkill
45         # and Python's string substitution can be used (e.g. "x=%d"%(result))
46         # but for centering the formatter is a bit easier.
47         # Basic format syntax
48         #    { } specifies something to be replaced
49         #
50
51         # Generate format strings such that:
52         # columns:
53         #  digits will be converted to string (!s)
54         #  column numbers will be centered (^)
55         colheader = "{!s:^%d}"%(self.displaycol)
56         # number of digits needed for rows
57         rowheadersz = int(math.ceil(self.rows / 10.0))
58         # rows labels are right justified (>) with a trailing space
59         rowheader  = "{:>%dd} "%(rowheadersz) # right justified digit
```

```
60
61          # force conversion to string !s and center in a field of
62          # displaycol spaces.
63          colentry = "{!s:^%d}"%(self.displaycol)
64          # Generate column labels
65          lines.append(
66              # leave space for row labels in subsequent rows
67              # one space for each digit + space before row content
68              "".join([" " for _ in range(rowheadersz+1)]) +
69              # column labels
70              "".join([colheader.format(idx) for idx in range(self.cols)]))
71          # Generate board string
72          r = 0
73          for row in self.board:
74              lines.append(
75                  # row label
76                  rowheader.format(r) +
77                  # row content
78                  "".join([colentry.format(entry if entry else self.empty_symbol)
79                           for entry in row]))
80              r = r + 1
81          # concatenate list into a string
82          return "\n".join(lines)
83
84
85
86
87
```