Using dictionaries

Announcements
● Homework #7 due Thursday

Outline
● Lists vs dictionaries
● Using dictionaries

Background: Battleship
● A board game where players try to guess the location of each others’ ships
● Each player has a grid on which they place ships of varying lengths (all ships are 1 unit wide) --- we’ll assume a 5x5 grid for simplicity; the original game had a 10x10 grid
● Each player’s grid of is hidden from the other player
● On their turn, a player names a grid coordinate--e.g., (1, 3)--which they think may be occupied by one of the other player’s ship
   ○ If there is not a ship at that coordinate, then the guess is a miss and it is the other player’s turn
   ○ If there is a ship at that coordinate, then the guess is a hit
      ■ The same player gets to guess again
      ■ When a player has guessed all of the coordinates occupied by one of the other player’s ships, then the ship is sunk

Lists vs Dictionaries
● Represent the grid as a list of lists
   ○ Each sublist corresponds to one row in the grid
   ○ Each value in a sublist is either a space (for empty) or a letter (for the corresponding ship)
● Write Python code to represent the following grid of ships as a list of lists.

```
grid = [['A',' ',' ',' ',' '],
        ['A',' ','B','B',' '],
        ['A',' ',' ',' ',' '],
        [' ',' ',' ',' ',' '],
        [' ','C','C','C','C']]
```

● What do you notice about most of the values? --- they are spaces; larger grid would have even more
● More efficient approach: only store the grid locations with ships
● Representing the grid as a dictionary
   ○ Each key is a tuple that corresponds to a grid coordinate -- e.g., (1,0)
- Write python code to represent the same grid of ships as a dictionary.
  
  ```python
  grid = {(0,0) : 'A', (1,0) : 'A', (2,0) : 'A', (1,2) : 'B', (1,3) : 'B',
          (4,1) : 'C', (4,2) : 'C', (4,3) : 'C', (4,4) : 'C'}
  ```

- Write a function called `gridL_to_gridD` that converts a list-of-lists representation of a Battleship grid to a dictionary representation of a Battleship grid. It should work with a grid of any size.

  ```python
  def gridL_to_gridD(gridL):
    gridD = {}
    for x in range(len(gridL)):
      row = gridL[x]
      for y in range(len(row)):
        cell = row[y]
        if cell != ' ':
          gridD[(x,y)] = cell
    return gridD
  ```

Using dictionaries

- Write a function called `grid_to_ships` that takes a Battleship grid and returns a dictionary whose keys are ship identifiers and whose values are the coordinates occupied by each ship. For example, given the grid above, the function should return:

  ```python
  {'A':[(0,0), (1,0), (2,0)], 'B':[(1,2), (1,3)], 'C':[(4,1), (4,2), (4,3), (4,4)]}
  ```

  ```python
  def grid_to_ships(grid):
    ships = {}
    for k in grid:
      ship = grid[k]
      if ship not in ships:
        ships[ship] = []
      ships[ship].append(k)
    return ships
  ```

- Write a function called `get_ship` that takes a Battleship grid, an x-coordinate, and a y-coordinate and returns the letter of the ship located at that coordinate, or None if no ship is located at that coordinate.

  ```python
  def get_ship(grid, x, y):
    if (x,y) not in grid:
      return None
    else:
      return grid[(x,y)]
  ```
• Write a function called attack that takes a Battleship grid, a list of ships, an x-coordinate, and a y-coordinate and returns 'Hit X' if ship X was hit, 'Sunk X' if ship X was sunk, and 'Miss' if no ship is located at that coordinate. (Hint: when a ship is hit, remove the hit coordinate from the ship's list of coordinates in the ship dictionary.)

```python
def attack(grid, ships, x, y):
    if (x, y) not in grid:
        return 'Miss'
    ship = get_ship(grid, x, y)
    ships[ship].remove((x, y))
    if len(ships[ship]) == 0:
        return 'Sunk ' + ship
    else:
        return 'Hit ' + ship
```

• Write a main function that repeatedly asks the user for x and y coordinates until all ships have been sunk. You can assume the user will not enter the same coordinates twice.

```python
def main():
    ships = grid_to_ships(gridD)
    while len(ships) > 0:
        x = int(input("X-coordinate: "))
        y = int(input("Y-coordinate: "))
        result = attack(gridD, ships, x, y)
        if 'Sunk' in result:
            ship = result[-1]
            del ships[ship]
        print(result)
```