```
"""
Cap 9
"""
from Cell2D import Cell2D, Cell2DViewer
from scipy.signal import correlate2d
import numpy as np
import matplotlib.pyplot as plt
import thinkplot

class Percolation(Cell2D):
    """AC di percolazione."""

    kernel = np.array([[0, 1, 0],
                       [1, 0, 1],
                       [0, 1, 0]])

    def __init__(self, n, q=0.5, seed=None):
        """Inizializza gli attributi.

        n: numero di righe
        q: probabilità di porosità
        seed: seme casuale
        """
        self.q = q
        if seed is not None:
            np.random.seed(seed)
        self.array = np.random.choice([1, 0], (n, n), p=[q, 1-q])

        # riempe la primariga con celle bagnate
        self.array[0] = 5

    def step(self):
        """Esegue un passo nel tempo."""
        a = self.array
        c = correlate2d(a, self.kernel, mode='same')
        self.array[(a==1) & (c>=5)] = 5

    def num_wet(self):
        """Numero totale di celle bagnate."""
        return np.sum(self.array == 5)

    def bottom_row_wet(self):
        """Numero di celle bagnate nella prima riga."""
        return np.sum(self.array[-1] == 5)

class PercolationViewer(Cell2DViewer):
    """Disegna e anima l'oggetto della percolazione."""
    cmap = plt.get_cmap('Blues')
    options = dict(alpha=0.6,
                   interpolation='nearest',
                   vmin=0, vmax=5)
n = 10
q=float(input('Immetti il valore della probabilità q = '))
seed = 18
perc = Percolation(n, q, seed)
viewer = PercolationViewer(perc)

def test_perc(perc):
    """Esegue il modello di percolazione.
    Gira finché l'acqua non giunge nell'ultima riga o non vi è alcuna
variazione.
    returns: booleano se c'è un gruppo di percolazione
    """
    num_wet = perc.num_wet()
```

```
    while True:
        perc.step()
        if perc.bottom_row_wet():
           return True
        new_num_wet = perc.num_wet()
        if new_num_wet == num_wet:
           return False
        num_wet = new_num_wet

test_perc(perc)


viewer = PercolationViewer(perc)
anim = viewer.animate(frames=12)
anim

def estimate_prob_percolating(n=100, q=q, iters=100):
    """Stima la probabilità di percolazione.

    n: numero intero delle righe e delle colonne
    q: probabilità che una cella sia permeabile
    iters: numero di arrays da provare

    ritorni: probabilità di formare un gruppo di percolazione
    """
    t = [test_perc(Percolation(n, q)) for i in range(iters)]
    return np.mean(t)

fraction = estimate_prob_percolating(q=q)
print()
print('Frazione di percolazione ',fraction)
```