Battaglia Navale

February 23, 2017

1 Battaglia navale

Primo tentativo: con una lista di liste

Vogliamo scrivere un programma che giochi a battaglia navale.

Per cominciare con cose semplici, saremo noi a giocare contro il programma ma non il contrario. Questo ci risparmia il dover realizzare una strategia di gioco.

In pratica, il programma deve fare questo:

1. INIZIALIZZAZIONE
    0. creare due tavole di gioco 5x5
        1. una tavola "NAVI" dove posiziona le sue navi (che noi non vediamo)
        2. una tavola "TIRI" dove annota i notri tiri (che vediamo)
    1. schermata di benvenuto e visualizzazione della tavola TIRI dove compaiono i nostri tiri (inizialmente vuota)
    2. posiziona a caso 5 navi da 1 sulla tavola NAVI (che noi non vediamo mai)
    3. porta il numero di navi rimaste da colpire a 5 e porta il numero di colpi sparati a 0
2. CICLO fino a quando il numero di navi rimaste da colpire diventa 0
    1. mostra la tavola con i nostri colpi
    2. ci chiede la nostra mossa (ad es., C5)
    3. aumenta di 1 il numero di colpi sparati
    4. se il nostro colpo è andato a segno:
        1. dice "Affondato"
        2. segna la nave affondata sulla tavola visibile
        3. diminuisce di 1 il numero di navi rimaste da colpire
    5. altrimenti:
        1. dice "colpo a vuoto"
        2. segna il colpo a vuoto
3. Dice "hai vinto con" il numero di colpi sparati

1.1 La struttura dati

La struttura dati, come sappiamo, è il cuore di ogni programma perché influenza tutto il codice che dobbiamo scrivere.

Poiché il gioco ruota attorno a una tavola di caselle (ovvero: una matrice o, se vogliamo,un piano cartesiano discreto e limitato) il problema principale è come rappresentare la tavola.

La tavola dovrebbe essere una cosa così:

```
  | 1 | 2 | 3| 4 | 5 |
----------------------
A |   |   |  |   |   |
----------------------
B |   |   |  |   |   |
----------------------
C |   |   |  |   |   |
----------------------
D |   |   |  |   |   |
----------------------
E |   |   |  |   |   |
----------------------
```

1.1.1 primo metodo: una lista di liste

```
In [1]: #ricordiamoci che per ripetere un valore in una lista
        #possiamo usare l'operatore *
        fila = [0] * 10

        #quindi per creare una lista di liste possiamo usare sia il *
        #che  una "list comprehension"
        matrice =[ [0]*10 for i in range(10)]

        matrice
```

```
Out[1]: [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

```
In [2]: #con una matrice fatta in questo modo, accedere alle singole caselle
        #è intuitivo.
        # prviamo a mettere 6 dentro la terza colonna della terza riga
        #per come abbiamo fatto la matrice:
        #il primo indice determina la RIGA
        #il secondo indice determina la COLONNA
        #ricordiamoci che il PRIMO elemento di una lista ha indice 0
        #quindi l'elemento 2 è il TERZO...
```

```
matrice[2][3] = 6


matrice
```

```
Out[2]: [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 6, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

1.1.2 secondo metodo: un dizionario

```
In [3]: #altrimenti possiamo fare la stessa struttura come dizionario.
        #il vantaggio è di usar direttamente la lettera delle coordinate.

        #creiamo un dizionario vuoto, o l'assegnamento nel for dà errore
        matrice2 = {}

        for i in ['A','B','C','D','E','F','G','H','I','J']:
            matrice2[i] = [0] * 10

        matrice2
```

```
Out[3]: {'A': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'B': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'C': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'D': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'E': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'F': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'G': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'H': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'I': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'J': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}
```

```
In [4]: matrice2['C'][2] = '3'
        matrice2
```

```
Out[4]: {'A': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'B': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'C': [0, 0, '3', 0, 0, 0, 0, 0, 0, 0],
         'D': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'E': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         'F': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
'G': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
'H': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
'I': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
'J': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}
```

e ora avanti, a programmare!