A.R.A.L.L.

Descrizione gioco

Il gioco creato è A.R.A.L.L (Advance Ruzzle and Letters Labyrinth).
Il nome del gioco è formato dall'acronimo delle iniziali dei nomi degli autori del gioco **(A)**ntonio.**(R)**enato.**(A)**ndrea.**(L)**uca.**(L)**orenzo.
Lo scopo è quello di trovare il maggior numero di parole possibili, così da raggiungere il punteggio maggiore. Il gioco è stato sviluppato con il software Python, che utilizza il linguaggio di programmazione C. Grazie ad esso è stato possibile realizzare la classe calcolo combinatorio, che funge da base per il funzionamento del gioco, e per l'idealizzazione della parte grafica. Essa si suddivide in due interfacce che, facendo riferimento al diagramma allegato, corrispondono rispettivamente al blocco "inizializzazione 1" "inizializzazione 2".
Per inizializzazione si intende un blocco contenente varie funzioni. Nel gioco sono presenti due tipi di inizializzazioni, le quali svolgono compiti diversi:
inizializzazione 1: consiste nella comparsa della schermata principale dove è possibile registrarsi o accedere al proprio account.
inizializzazione 2: consiste nel passaggio dalla schermata principale alla schermata di gioco, creazione dell'interfaccia e generazione delle parole.

INTERFACCIA DI LOGIN

Prima di iniziare a giocare, nella schermata principale vi è l'obbligo di creare un proprio account o di effettuare il login in un account già esistente. La creazione si basa sull'inserimento di un username ed una password, nel caso in cui l'username dovesse già essere associato ad un account bisogna necessariamente utilizzarne un altro. Grazie ad un sistema di salvataggio in cartelle dati, gli account creati vengono memorizzati e salvati nel blocco "*salva account*" del diagramma

INTERFACCIA DEL GIOCO

Il blocco "inizializzazione 2" del diagramma corrisponde alla generazione dell'interfaccia di gioco, costituita da:

- una griglia 3x3 in cui vengono generate le lettere.
- una sezione situata in alto a sinistra in cui compare il tempo di gioco rimanente.
- una tabella di testo in cui vengono inserite le parole trovate dal giocatore.
- una sezione situata sul centro-destra in cui, attraverso l'uso di una spunta verde o una croce rossa, si indica se la parola inserita dal giocatore è giusta o sbagliata.
- una sezione situata in alto a destra in cui viene indicato il punteggio del giocatore.
- una sezione situata in alto al centro in cui compare la scritta "exit", che permette al giocatore di terminare la partita e uscire dal gioco.
- una sezione situata in basso a destra in cui compare la scritta "next", che permette al giocatore di avanzare al livello successivo.

Bisogna specificare che la generazione delle lettere deve considerare il calcolo della probabilità del calcolo combinatorio, perché nel caso contrario uno dei 2 giocatori verrebbe svantaggiato.

REGOLE GIOCO

Una volta effettuato l'accesso all'account inizia il gioco.
Vi è una scelta da fare: modalità "giocatore singolo" o la modalità "più giocatori"

- Giocatore singolo: per arrivare alla vittoria bisogna superare il punteggio necessario per avanzare al livello successivo. Ci sono delle differenze nelle varie difficoltà:
    - principiante: non bisogna trovare le parole bonus;
    - intermedio: bisogna trovare un minimo di 3 parole bonus;
    - difficile: bisogna trovare un minimo di 6 parole bonus.
- Contro n giocatori: per arrivare alla vittoria bisogna trovare quante più parole possibili prima che si esaurisca il tempo con l'obiettivo di superare il punteggio degli avversari.

Le parole possono essere formate da qualsiasi tipo di combinazione tra lettere a patto che debbano essere prese una sola volta.

REGOLE PER IL CALCOLO DEL PUNTEGGIO

Il punteggio aumenta in base a determinate regole:

- la lunghezza della parola ne determina il punteggio:
    - dalle 2 alle 4 lettere = 2 punti
    - dalle 5 alle 8 = 5 punti
    - 9 lettere = 10 punti
- Esistono delle parole bonus che valgono il doppio. Esse sono scelte in base ad una categoria che viene specificata prima dell'inizio del tempo.
- Una serie di 5 parole giuste concede un bonus di 5 punti alla fine del livello.

Descrizione della classe

Per la realizzazione del gioco, è stata creata una classe a oggetti relativa a (?) utilizzando il linguaggio Python, che tiene conto delle regole del calcolo combinatorio. Nella classe sono state create varie funzioni che servono per realizzare il gioco A.R.A.L.L. (Advance Ruzzle and Letters Labyrinth).
Le funzioni principali sono:

- anagrammi: questa funzione attraverso un ciclo di for realizzerà una lista di anagrammi a partire da una stringa di caratteri data. Il metodo permutazioni genera una lista di tuple, ogni tupla è una permutazione e scorrendo la lista attraverso un ciclo, si possono visualizzare gli elementi della tupla per ricostruire la stringa.

```
def anagrammi(self):
    lettere = list(parola)

    permutazioni = list(permutations(lettere))

    temp = ''

    anagrammi = []

    for i in permutazioni:

        for carattere in i:
            temp += carattere

        anagrammi.append(temp)

        temp = ''
```

- charRipetuti: questa funzione serve per trovare i caratteri ripetuti e anche essa sfrutta il ciclo di for, se trova il carattere nel dictionary, incrementa il suo valore; se non lo trova lo aggiunge.

```
def charRipetuti(self):

    word = list(parola)
    caratteriripetuti={}
    nCaratteri = 0
    count = 0

    for i in word:
        if (i in caratteriripetuti):
            caratteriripetuti[str(i)] += 1
    else:
        caratteriripetuti[str(i)] = 1
    for i in caratteriripetuti:
        if caratteriripetuti[i]>1:
            count+=1
            nCaratteri += caratteriripetuti[i]
```

- combUtil: questa funzione controlla la lista delle parole e verifica se esse esistono nel vocabolario italiano

```
def combUtil(self):
    words = 'words.italian.txt'
    f = open(words, 'r')
    for riga in f:

       p=f.readline()

       if self.__stringa in p:
           return("è una parola italiana")
       if parola == p[:-1]:
           return("vero")
```

- fattoriale: questa funzione serve semplicemente per realizzare il fattoriale di un numero: in matematica, il fattoriale di un numero intero positivo n, è il prodotto dei numeri interi da 1 a n

```
def fattoriale(n):
    if n==0:
        return 1
    else:
        return n*fattoriale(n-1)
```

- coeffBinom: questa funzione, invece, serve per realizzare il coefficiente binomiale: ilcoefficiente binomiale è un numero naturale definito a partire da una coppia di numeri naturali, solitamente indicati con n e k. In particolare il coefficiente binomiale n su k rappresenta il numero di sottoinsiemi di k elementi che si possono estrarre da un insieme di n elementi.

```
def coeffBinom(n, k):
    x = len(parola)
    y = int(input("Enter a value for y: "))
    if y == 1 or y == x:
       return(1)
    if y > x:
        return(0)
    else:
        a = fattoriale(x)
        b = fattoriale(y)
        div = a // (b*(x-y))
    return(div)
```

Dopo aver creato queste funzioni della classe calcComb (calcolo combinatorio), si procede con la creazione di ulteriori funzioni: permutazioni, disposizioni e combinazioni.

PERMUTAZIONI

Una permutazione è il risultato di uno scambio dell'ordine degli elementi di una sequenza, ossia è uno dei possibili modi per ordinare elementi di qualsiasi tipo. Si distinguono due tipi di permutazioni: le permutazioni semplici, le permutazioni con ripetizione.

- Permutazioni semplici: le permutazioni semplici di n elementi distinti sono tutti i gruppi formati dagli n elementi, che differiscono per il loro ordine $P_n = n!$
- Permutazioni con ripetizioni: Le permutazioni con ripetizioni di n elementi di cui h, k,... ripetuti, sono tutti i gruppi formati dagli n elementi, che differiscono per l'ordine in cui si presentano gli elementi distinti e la posizione che occupano gli elementi ripetuti. $P_n^{(h,k...)} = n! / h! * k! *…$

DISPOSIZIONI

Come per le permutazioni anche le disposizioni sono con e senza ripetizioni:

- DIsposizioni semplici: Le disposizioni semplici di n elementi distinti di classe k (con n,k appartenenti a N e k≤n) sono tutti i gruppi che si possono formare con k elementi, presi fra gli n, tali che ognuno è diverso dagli altri per gli elementi contenuti o l'ordine. $D_{n,k}= n* (n-1) * (n-2)*...*(n-k+1)=n!/(n-k)!$
- Disposizioni con ripetizioni: Le disposizioni con ripetizioni di n elementi distinti di classe k(con n,k appartenenti a N) sono tutti i gruppi che si possono formare con k elementi, anche ripetuti, presi fra gli n, tali che ogni gruppo è diverso dagli altri per gli elementi contenuti o per il loro ordine. $D'_{n,k} = n^k$

COMBINAZIONI

Una combinazione è un raggruppamento di k elementi, presi in qualsiasi ordine, formato a partire da n elementi distinti. In termini più rigorosi si dice combinazione ogni sequenza di k elementi estratti tra n elementi distinti, nell'ipotesi che l'ordine di estrazione sia ininfluente.
Pure in questi caso esistono combinazioni semplici e con ripetizioni:

- Combinazioni semplici: Le combinazioni semplici di n elementi distinti di classe k (con $0<k\leq n$) sono tutti i gruppi che si possono formare con k elementi, presi fra gli n, e tali che ogni gruppo è diverso dagli altri per almeno un elemento contenuto. $C_{n,k}=D_{n,k}/ P_k=n!/k!(n-k)!$
- Combinazioni con ripetizioni: Le combinazioni con ripetizioni di n elementi distinti di classe k, sono tutti i gruppi che si possono formare con k elementi, presi fra gli n; ogni elemento di un gruppo può essere ripetuto fino a k volte, non interessa l'ordine in cui gli elementi si presentano e in ciascun gruppo è diverso il numero delle volte in cui un elemento compare. $C'_{n,k}=C_{n+k-1,k}=(n+k-1)*(n+k-2)*.../k!$

PROBABILITÀ

La probabilità di un evento (E) è il rapporto fra il numero dei casi favorevoli (f) e quello dei casi possibili (u) quando sono tutti ugualmente possibili. $p(E)= f/u$.
Poiché il numero dei casi favorevoli è sempre minore o uguale al numero dei casi possibili la probabilità di evento è sempre compresa tra 0 e 1. $0 \leq p(E) \leq 1$
Poi abbiamo dei casi particolari:

- Quando il numero dei casi favorevoli è uguale al numero dei casi possibili l'evento sarà certo cioè uguale a 1. $p(E)=1$
- Quando il numero dei casi favorevoli è uguale a 0 cioè nullo l'evento sarà impossibile. $p(E)=0$

Interfaccia login

Interfaccia gioco