# O jogo "sopa de letras"

O jogo da sopa de letras é um passatempo que se baseia num tabuleiro de letras colocadas aleatoriamente, onde são inscritas palavras. O objetivo do jogo é encontrar e circundar as palavras escondidas no tabuleiro. As palavras podem estar escondidas verticalmente, horizontalmente ou diagonalmente no tabuleiro, tanto na sequência normal como invertida.

Observe o seguinte exemplo.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Y | L | C | Q | T | S | T |
| Y | P | O | R | T | O | O |
| M | L | I | P | X | O | X |
| Z | R | M | A | I | A | X |
| S | S | B | G | X | M | L |
| T | B | R | A | Z | V | N |
| D | F | A | R | O | F | W |
| R | F | R | B | E | A | H |

Neste tutorial iremos desenvolver uma versão mais simples, em que as palavras são colocadas apenas na horizontal.

# Estruturação do jogo

Este jogo já apresenta alguma complexidade, pelo que será útil utilizar a decomposição do problema para estruturar melhor o jogo em subproblemas:

1. Criar o tabuleiro: neste subproblema, constrói-se uma matriz, onde, de forma aleatória, são colocadas as palavras de uma determinada lista predefinida. Neste caso, apenas colocaremos as palavras na horizontal, podendo aleatoriamente ser colocadas normalmente ou de forma invertida. No final de colocar as palavras, é necessário preencher os espaços vagos com letras, de forma aleatória.
2. Mostrar o tabuleiro: Neste subproblema, é visualizada a matriz da sopa de letras.
3. Jogar: Este subproblema irá focar-se no ciclo de jogo, iterando ao longo de várias jogadas até que o jogador adivinhe todas as palavras.

# Criar o tabuleiro

O tabuleiro é constituído por uma matriz, que tipicamente é formada por um vetor de vetores. Em Pyhon, optámos por implementá-la através de uma lista de linhas, cada linha sendo uma cadeia de caracteres, uma vez que é um jogo de letras.

A função cria\_tabuleiro(linhas, colunas, lista\_palavras) recebe três parâmetros:

* **linhas**, o número de linhas do tabuleiro (inteiro);
* **colunas**, o número de colunas do tabuleiro (inteiro);
* **lista\_palavras**, uma lista de cadeias de caracteres (strings) com as palavras do jogo.

O algoritmo é composto por 3 passos:

1. Criar tabuleiro vazio;
2. Colocar no tabuleiro as palavras da lista;
3. Preencher os espaços em branco do tabuleiro com letras aleatórias.

## Criar tabuleiro vazio

Neste passo do algoritmo, é criada uma lista vazia que vai sendo preenchida com cadeias de caracteres, que por sua vez são preenchidas com espaços, em número igual ao número de colunas.

def cria\_tabuleiro(linhas, colunas, lista\_palavras):

# criar tabuleiro vazio

tabuleiro = []

for i in range(linhas):

linha = ""

for j in range (colunas):

linha += " "

tabuleiro += [linha]

|  |  |
| --- | --- |
| **Eye** | Neste código, o operador += faz a concatenação de cadeias de caracteres, juntando os espaços. Também faz a concatenação de listas, de forma a adicionar uma nova linha em cada iteração do ciclo for com variável i. |

## Colocar no tabuleiro as palavras da lista;

Neste passo do algoritmo, sendo aleatório, pode não ser possível colocar todas as palavras no tabuleiro, poisdepende da “arrumação” destas. Para esta colocação teremos um máximo de 50 tentativas. Caso não seja possível criar o tabuleiro, a função retorna False.

# coloca palavras de forma aleatória

i = 0

tentativas = 50 # pode não haver solução possível...

# ciclo de preenchimento do tabuleiro com a lista\_palavras de palavras

while i < len(lista\_palavras):

# palavra a colocar na sopa de letras: em maiusculas e pode ser invertida

lista\_palavras[i] = lista\_palavras[i].upper()

palavra = lista\_palavras[i]

# aleatoriamente a palavra poderá ser invertida

if random.choice([True, False]):

palavra = palavra[::-1] # inverte palavra

# determina localização horizontal aleatória

l = random.randint(0, linhas-1)

linha = tabuleiro[l] # linha aleatoria

col = random.randint(0, colunas-len(palavra)) # o comprimento da palavra limita o início da coluna

# verifica se tem espaço para a palavra

if linha.count(" ", col, col+len(palavra)) == len(palavra):

# coloca palavra

tabuleiro[l] = linha[:col] + palavra + linha[col+len(palavra):]

i += 1

else:

#procura outra posicao para a palavra

if tentativas > 0:

tentativas -= 1

else:

print("Nao foi possivel construir tabuleiro")

return False

## Preencher os espaços vazios do tabuleiro

Neste passo final, percorre-se todo o tabuleiro e, caso se encontre um espaço, coloca-se uma letra aleatória.

# preencher o resto do tabuleiro com letras aleatorias

for i in range(linhas):

for j in range(colunas):

if tabuleiro[i][j] == " ":

tabuleiro[i] = tabuleiro[i][:j] \

+ random.choice("ABCDEFGHIJKLMNOPQRSTUVWXYZ") \

+ tabuleiro[i][j+1:]

return tabuleiro

|  |  |
| --- | --- |
| **Eye** | Observe que a quarta linha de instruções (if) é dividida em três linhas através do caracter "\".  Isto permite maior legibilidade do código. |

# Mostrar o tabuleiro

A função mostra\_tabuleiro(tabuleiro) percorre as várias linhas do tabuleiro e faz print da cadeia de caracteres da linha.

def mostra\_tabuleiro(tabuleiro):

for linha in tabuleiro:

print(linha)

# Iterando através das jogadas – o ciclo de jogo

A função sopa\_de\_letras() implementa o ciclo de jogo e tem os seguintes parâmetros:

* linhas - o número de linhas do tabuleiro;
* colunas - o número de colunas do tabuleiro;
* lista\_palavras - a lista com as palavras a colocar no tabuleiro;
* tentativas - o número máximo de tentativas erradas.

A função inicia-se com a criação do tabuleiro:

def sopa\_letras(linhas, colunas, lista\_palavras, tentativas):

# criacao do tabuleiro

tabuleiro = cria\_tabuleiro(linhas, colunas, lista\_palavras)

if type(tabuleiro) != list:

print ("Nao foi possivel construir o tabuleiro...")

return

# numero de palavras a adivinhar

palavras\_adivinhar = len(lista\_palavras)

O ciclo de jogo é executado através de um ciclo condicional (while) cuja condição avalia se o jogador ainda tem palavras para adivinhar, e se ainda tem tentativas para jogar.

while tentativas > 0 and palavras\_adivinhar > 0:

O algoritmo de cada jogada é o seguinte:

1. Interação com o jogador: mostrar tabuleiro e pedir uma palavra.

*Obs.: converte-se a palavra para maiúsculas de forma que seja mais simples a comparação.*

# Jogada do jogador

mostra\_tabuleiro(tabuleiro)

palavra = input("Palavra: ").upper()

1. Verificar se a palavra está correta (pertence à lista de palavras).

No caso de a palavra estar correta elimina-se da lista de palavras de forma a não ser validada novamente numa próxima jogada.

No caso de a palavra estar errada decrementa-se o número de tentativas restantes.

# verificar se a palavra esta correta

if palavra in lista\_palavras:

# palavra certa

palavras\_adivinhar -= 1

print("Certo! Palavras a encontrar: " + str(palavras\_adivinhar))

if palavras\_adivinhar >= 0:

#apaga palavra da lista\_palavras

for i in range (len(lista\_palavras)):

if palavra == lista\_palavras[i]:

lista\_palavras[i] = ""

else:

# palavra errada

print("Errado!")

tentativas -= 1

Após o fim do ciclo de jogo é verificado se o jogador ganhou ou perdeu...

# fim do jogo

if tentativas == 0 and palavras\_adivinhar > 0:

print ("Perdeu. Faltaram " + str(palavras\_adivinhar) + "palavras")

else:

print ("Parabens! Encontrou todas as palavras.")

|  |  |
| --- | --- |
| **Balloon animal** | Teste o jogo com a seguida lista:  cidades = ["Porto", "Lisboa", "Braga", "Faro", "Bragança"]  sopa\_letras(10,30, cidades, 5)  Experimente depois com várias listas de palavras de diferentes temas. |

# Desafio final

Na solução apresentada, fez-se a restrição de as palavras estarem apenas na horizontal. Acrescente a possibilidade de o tabuleiro ter palavras na vertical, ou até na diagonal.

|  |  |
| --- | --- |
| **Share with person** | Partilhe o **seu** jogo! |