01

Estipulando tentativas de erros

Transcrição

Agora vamos dar continuidade ao nosso jogo, pois ele ainda não está totalmente funcional, pois ao acertar a palavra secreta, o jogo ainda fica pedindo que o usuário digite uma letra:

```
*********************************
***Bem vindo ao jogo da Forca!***
*********************************
['_', '_', '_', '_', '_', '_']
Qual letra? b
['b', '_', '_', '_', '_', '_']
Qual letra? a
['b', 'a', '_', 'a', '_', 'a']
Qual letra? n
['b', 'a', 'n', 'a', 'n', 'a']
Qual letra?
```

O contrário também acontece, podemos errar várias vezes, até digitar quase o alfabeto inteiro, se quisermos. Ou seja, o usuário possui infinitas tentativas para descobrir a palavra, já que não há um limite de tentativas.

Encerrando o jogo quando o usuário errar seis vezes

Como os valores das variáveis `enforcou` e `acertou` nunca mudam, o loop é infinito. Logo, precisamos mudar os seus valores de acordo com situações do jogo.

Vamos começar com a variável `enforcou`. Quando é que o usuário se enforca? Vamos definir que é quando o jogador errar 6 vezes, então precisamos contar esses erros. Logo, vamos declarar uma variável `erros` e inicializá-la com o valor `0`:

```
enforcou = False
acertou = False
erros = 0
```

Agora, se o usuário errou, nós incrementamos a variável. Para tal, precisamos realizar um teste. Vamos testar se o chute está na palavra secreta, se estiver nós executamos o código que posiciona a letra na lista, etc. Mas se o chute não estiver na palavra secreta, incrementamos a variável `erros`:

```
while (not acertou and not enforcou):

    chute = input("Qual letra? ")
    chute = chute.strip()

    if (chute in palavra_secreta):
        index = 0
        for letra in palavra_secreta:
            if (chute.upper() == letra.upper()):
```

```
                letras_acertadas[index] = letra
            index = index + 1
    else:
        erros = erros + 1

    print(letras_acertadas)
```

Ao final do `while`, vamos verificar se a variável `erros` é igual a **6** e atribuir esse booleano (`true` ou `false`) à variável `enforcou`:

```
while (not acertou and not enforcou):

    chute = input("Qual letra? ")
    chute = chute.strip()

    if (chute in palavra_secreta):
        index = 0
        for letra in palavra_secreta:
            if (chute.upper() == letra.upper()):
                letras_acertadas[index] = letra
            index = index + 1
    else:
        erros = erros + 1

    enforcou = erros == 6
    print(letras_acertadas)
```

Agora, ao executarmos o jogo e errar 6 vezes, o jogo é encerrado! Para melhorar ainda mais o nosso código, podemos seguir a dica do PyCharm e realizar o incremento das variáveis `index` e `erros` da seguinte forma:

```
while (not acertou and not enforcou):

    chute = input("Qual letra? ")
    chute = chute.strip()

    if (chute in palavra_secreta):
        index = 0
        for letra in palavra_secreta:
            if (chute.upper() == letra.upper()):
                letras_acertadas[index] = letra
            index += 1
    else:
        erros += 1

    enforcou = erros == 6
    print(letras_acertadas)
```

O resultado é o mesmo.

Palavra secreta e chute sempre em caixa alta

Mas agora, se testarmos um chute com letra maiúscula?

```
*********************************
***Bem vindo ao jogo da Forca!***
*********************************
['_', '_', '_', '_', '_', '_']
Qual letra? B
['_', '_', '_', '_', '_', '_']
Qual letra?
```

Ué, já não tínhamos resolvido isso? Acontece que no `if` que acabamos de escrever, nós não fazemos a comparação do chute com as letras em caixa alta. Para não termos que sempre nos lembrar disso, vamos já definir a palavra secreta e o chute em letra maiúsculas. Assim, nas comparações não precisamos mais deixar todas as letras em caixa alta. O código ficará assim:

```
def jogar():

    print("*********************************")
    print("***Bem vindo ao jogo da Forca!***")
    print("*********************************")

    palavra_secreta = "banana".upper()
    letras_acertadas = ["_", "_", "_", "_", "_", "_"]

    enforcou = False
    acertou = False
    erros = 0

    print(letras_acertadas)

    while (not acertou and not enforcou):

        chute = input("Qual letra? ")
        chute = chute.strip().upper()

        if (chute in palavra_secreta):
            index = 0
            for letra in palavra_secreta:
                if (chute == letra):
                    letras_acertadas[index] = letra
                index += 1
        else:
            erros += 1

        enforcou = erros == 6
        print(letras_acertadas)

    print("Fim do jogo")
```

Por fim, falta encerrarmos o jogo quando o jogador acertar a palavra.

Encerrando o jogo quando o usuário acertar a palavra

Uma das maneiras de implementar essa funcionalidade é aproveitar a nossa lista de letras acertadas, já que enquanto o jogador não acertar a palavra, o caractere `_` vai existir na lista.

Logo, enquanto a lista não possui o `_`, significa que a palavra ainda não foi totalmente preenchida, já que os `_` são substituídos a cada letra acertada. Então vamos adicionar essa verificação abaixo da verificação da variável **`enforcou`**:

```
enforcou = erros == 6
acertou = "_" not in letras_acertadas
```

Ao testar, e acertar a palavra, temos o seguinte resultado:

```
**********
***Bem vindo ao jogo da Forca!***
**********
['_', '_', '_', '_', '_', '_']
Qual letra? b
['B', '_', '_', '_', '_', '_']
Qual letra? a
['B', 'A', '_', 'A', '_', 'A']
Qual letra? n
['B', 'A', 'N', 'A', 'N', 'A']
Fim do jogo
```

O jogo é encerrado ao acertarmos a palavra! Ótimo, funcionalidade concluída. Falta apenas exibir uma mensagem ao usuário, dizendo se ele acertou ou não a palavra. Após o **`while`**, adicionamos:

```
if (acertou):
    print("Você ganhou!")
else:
    print("Você perdeu!")
```

No próximo vídeo iremos melhorar a declaração da quantidade de `_` na lista de palavras acertadas, já que por enquanto a mesma está fixa.