

Melhorando a apresentação da força

Transcrição

Com a melhor organização do nosso código, vamos melhorar a exibição, a apresentação da força, deixando o jogo mais amigável.

Novas mensagens de vencedor e perdedor

Vamos começar com a mensagem de perdedor, alterando a função `imprime_mensagem_perdedor`. Ela ficará assim:

```
def imprime_mensagem_perdedor(palavra_secreta):
    print("Puxa, você foi enforcado!")
    print("A palavra era {}".format(palavra_secreta))
    print(" ")
    print(" /-----\ ")
    print(" /           \ ")
    print(" /             \ ")
    print("//             \\ ")
    print("\\|    XXXX    XXXX | / ")
    print(" |    XXXX    XXXX | / ")
    print(" |    XXX      XXX | ")
    print(" |                   | ")
    print(" \\_        XXX      _/ ")
    print("   |\\      XXX      /| ")
    print("   ||              || ")
    print("   | I I I I I I | ")
    print("   | I I I I I I | ")
    print("   \\_            _/ ")
    print("   \\_          _/ ")
    print("   \\_____ / ")
```

Agora ela recebe a `palavra_secreta` por parâmetro, então não podemos esquecer de passá-la no momento que chamarmos a função:

```
def jogar():

    # restante do código omitido

    if(acertou):
        imprime_mensagem_vencedor()
    else:
        imprime_mensagem_perdedor(palavra_secreta)
```

Do mesmo jeito, vamos refazer a mensagem de vencedor, na função `imprime_mensagem_vencedor`:

```
def imprime_mensagem_vencedor():
    print("Parabéns, você ganhou!")
    print(" ")
    print("  _==_==_==_ ")
```

```

print("      .-\\:      /-.  ")
print("      | (|:.      |) |  ")
print("      '-|:.      |- '   ")
print("      \\\\:..      /      ")
print("      ':.. .'      ")
print("      ) (          ")
print("      _.' '._      ")
print("      '-----'      ")

```

Desenhando a força

Por fim, o jogo da força não seria o jogo da força se não mostrássemos a força, juntamente com o seu personagem.

Vamos criar a função `desenha_forca`, que recebe os `erros` por parâmetro. Para cada valor de `erros`, a função imprime um desenho diferente:

```

def desenha_forca(erros):
    print("  _____  ")
    print("  |/          |  ")

    if(erros == 1):
        print("  |          ( )  ")
        print("  |          ")
        print("  |          ")
        print("  |          ")

    if(erros == 2):
        print("  |          ( )  ")
        print("  |          \     ")
        print("  |          ")
        print("  |          ")

    if(erros == 3):
        print("  |          ( )  ")
        print("  |          \|    ")
        print("  |          ")
        print("  |          ")

    if(erros == 4):
        print("  |          ( )  ")
        print("  |          \|/   ")
        print("  |          ")
        print("  |          ")

    if(erros == 5):
        print("  |          ( )  ")
        print("  |          \|/   ")
        print("  |          |     ")
        print("  |          ")

    if(erros == 6):
        print("  |          ( )  ")
        print("  |          \|/   ")
        print("  |          |     ")
        print("  |          /     ")

```

```
if (erros == 7):
    print(" |      ( _ )  ")
    print(" |      \  /  ")
    print(" |      |  ")
    print(" |      / \  ")

    print(" |      ")
    print(" _|___  ")
    print()
```

Para finalizar, devemos chamar essa função quando o jogador erra, dentro do `else` e aumentar o limite de erros para 7:

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

    chute = pede_chute()

    if (chute in palavra_secreta):
        marca_chute_correto(chute, letras_acertadas, palavra_secreta)
    else:
        erros += 1
        desenha_forca(erros)

    enforcou = erros == 7
    acertou = "_" not in letras_acertadas
    print(letras_acertadas)

# restante do código omitido
```

Com isso, chegamos ao final da implementação do nosso jogo da forca!