Pigs and Bulls

Ruben Teimas, nº39868

Problema/Objetivo: Este trabalho tinha como objetivo criar um programa que permitisse emular o jogo Pigs and Bulls. Neste caso, o computador gera um numero aleatório inteiro de 4 algarismos e o jogador deverá tentar acertar nesse mesmo número.

Solução ao problema

Para criar o programa pretendido foram utilizadas 2 funções:

- random_result(), que tem como objetivo gerar um número aleatório inteiro de 4 algarismos;
- jogo(), onde está incluído todo o corpo do programa.

Lista de variáveis:

- n(int) - esta variável servirá como contador do primeiro loop na função random_result() até encontrar 4 números inteiros diferentes;
- y (int) – número inteiro gerado aleatoriamente;
- vazio(list) – a lista começa vazia pois será nela que serão guardados os algarismos diferentes gerados de forma aleatória;
- w (str) – transforma os números inteiros(y) guardados na lista(vazio) em strings;
- result(str) – permite formar um número inteiro de 4 algarismos utilizado os 4 números inteiros previamente guardados na lista(vazio);
- tenta(str) – palpite do jogador;
- nmr_tentativas(int) – contador do loop que irá incrementando ao final de cada tentativa dando o número final de tentativas;
- touro, porco (int) – dão o número de touros (algarismos na posição certa) e porcos (algarismos presentes mas na posição errada), respetivamente.
- x(str) – guarda a estatística do jogador nessa ronda;
- final(list) – acumula as estatísticas das tentativas(x) que serão apresentadas no final do jogo.

O código começa com uma breve introdução ao jogo para que o utilizador perceba facilmente o que deve fazer e as regras do mesmo.

De seguida o computador gera um numero aleatório, utilizando um ciclo while, que irá correr até que sejam gerados 4 números inteiros de 0 a 9, todos diferentes entre si. Para assegurar que sejam diferentes o primeiro número gerado(y) é guardado na lista(vazio) e n incrementa mais 1. Os próximos números serão comparados aos existentes na lista e **só caso sejam diferentes**, serão adicionados e n incrementará novamente. Quando n=4 já terão sido gerados 4 números diferentes e como tal o ciclo terminará. De seguida inicia-se um ciclo for que permite aceder a cada elemento da lista e converte-se esse mesmo elemento numa string guardando essa string numa variável que irá receber todos os elementos da lista. É essencial que cada elemento seja convertido para string, pois caso fosse inteiro o computador iria proceder a uma soma de inteiros, quando não é isso que se quer. A função random_result(), da Fig.1, irá assim retornar o número aleatório de 4 algarismos.

```
def random_result():
    n = 0
    vazio = [ ]
    result = ""
    while n<4:
        y = int(10*random.random())
        #Loop para que não haja repetição de valores
        if y not in vazio:
            vazio.append(y)
            n = n+1
    for i in range(len(vazio)):
        #permite construir um numero de 4 digitos, pois o type é str e não int ou float
        w = str(vazio[i])
        result = result + w
    return result
```

Fig. 1

Com o número já gerado, o jogo irá começar, entrando na função jogo(), fig.2, que irá pedir ao utilizador um palpite. Após o utilizador introduzir um palpite irá iniciar-se um ciclo que só acabará quando o palpite(tenta) = número aleatório (result).

Para isso o programa irá comparar os números na totalidade, caso sejam iguais o ciclo termina, senão irá entrar num ciclo for que irá aceder ao índice de cada um dos números (convertidos em strings para que seja possível aceder ao mesmo). Caso o elemento de índice i de ambos seja igual a variável **touro** irá incrementar o número de vezes que existirem elementos de índice i iguais em ambas para que possa depois dar uma pista ao jogador de quantos números estão no sitio certo.

De seguida o programa irá verificar se o elemento de índice i se encontra em **result** e se não pertence ao mesmo índice em ambos, a variável **porco** irá incrementar de forma semelhante à variável **touro,** mas com as condições acima referidas. Se alguma destas condições não se verificar a variável **porco** não incrementar.

Após este processo o programa guarda a tentativa e os seus dados numa variável **x** que será adicionada a uma lista **final** e será apresentada, após o jogador acertar, utilizando um ciclo for que irá aceder a cada elemento da lista dando print dos mesmos.

As restrições não mencionadas, Fig.2, servem um único propósito de estética no display do programa. Pois caso, por exemplo, não existam Touros não faz sentido ter "0T".

```
if touro>0 and porco>0:
    print(touro,"T", " ", porco,"P")
    x = "T{a}: {b}, {c}T {d}P".format(a=nmr_tentativas, b=tenta, c=touro, d = porco)
elif touro>0:
    print(touro,"T")
    x = "T{a}: {b}, {c}T".format(a=nmr_tentativas, b=tenta, c=touro)
elif porco>0:
    print(porco,"P")
    x = "T{a}: {b}, {d}P".format(a=nmr_tentativas, b=tenta, d = porco)
else:
    x = "T{a}: {b}".format(a=nmr_tentativas, b=tenta)
```

Fig.2

Bugs

Em todos os testes realizados não foram encontrados quaisquer problemas, sendo que o programa executa todas as funcionalidades pretendidas.

Se o utilizador introduzir algarismos iguais, o resultado estará errado e ausente de qualquer lógica, contudo, faz sentido que assim o seja pois esse caso não está previsto nas regras do jogo.