Introdução ao jogo Sudoku

Sudoku é um jogo que consiste na colocação de números de 1 a 9 em cada uma das células vazias numa grade 9 x 9, constituída por 3 x 3 subgrades chamadas regiões. O objetivo do jogo é colocar os números de modo a não se encontrarem repetições nas linhas, colunas e regiões.

Arquitetura do programa

O programa foi desenvolvido em Python 3.12.7, pelo Visual Studio Code, utilizando uma estrutura procedural, ou seja, suas funcionalidades foram segmentadas em diversas funções menores. Esse estilo de organização simplifica o gerenciamento e a manutenção do código.

Funções do programa

Logo quando iniciei o código descobri que Python não permite iniciar uma variável como uma matriz diretamente, apenas como um array. Então tive que achar um jeito de corrigir isso. Para isso eu iniciei a variável matriz como um array então fiz um lopping que percorre esse array e coloca um array com 9 itens dentro, os itens colocados foram ‘ ‘, apenas para criar o tamanho da matriz que eu ia usar.

```
# Cria a matriz
matriz = []
repet = 0
valido = True

# Adiciona 9 arrays na matriz
for i in range(9):
    matriz = matriz + [[' ']*9]
```

invalido()

É uma função simples que printa na tela que o tabuleiro não é possível e sai do código para que o programa não gaste tempo com as outras operações.

```
8
9   def invalido():
10      print('Tabuleiro inválido')
11      exit()
12
```

recebeLinhas()

Primeiramente, precisamos obter os dados do tabuleiro fornecidos pelo usuário. Para isso, a função **recebeLinhas()** cria um array a partir da entrada recebida, que inicialmente está no formato de uma string, já que a função **input()**, própria do Python, retorna esse tipo de dado. É necessário percorrer essa string para remover os espaços que separam os valores. Durante esse processo, também verifico se o comprimento da string excede o limite permitido pelo tabuleiro, que tem apenas 9 colunas. Essa é a primeira validação. Se o tamanho estiver dentro do permitido, os valores são inseridos no tabuleiro (no programa é chamado de matriz) na ordem definida pelo usuário.

```
25
26  # Recebe linhas do tabuleiro
27  def recebeLinhas():
28      # Recebe uma entrada para cada linha
29      for i in range(9):
30          linha = input()
31          # Indice da coluna do valor recebido
32          coluna = 0
33          # Percorre a input
34          for ii in linha:
35              # Se tamanho maior que o suportado
36              if len(linha) > 18:
37                  # Entrada Inválida
38                  invalido()
39              # Se o valor for diferente de espaço
40              elif ii != ' ':
41                  # Coloque-o no tabuleiro
42                  matriz[i][coluna] = ii
43                  coluna += 1
44
```

verificaEntrada()

Deve ter sido a última função em que pensei, mas parece algo muito óbvio. O sudoku só permite valores inteiros maiores que 1 e menores que 10, fora os espaços que ainda vão receber valores. No input os valores vazios eram ‘.’, o que torna esse valor possível também como entrada. Então depois que todos os dados foram colocados no tabuleiro (matriz) eu faço um looping, que percorre todas as linhas e colunas, para verificar se todos os dados estão entre os valores possíveis, “123456789’.’”. Se houver um valor impossível **invalido()** é chamado e o código se encerra.

```
45
46  def verificaEntrada():
47      # Para cada linha do array
48      for i in range(9):
49          # Pegue o valor na coluna ii
50          for ii in range(9):
51              # Verifique se o valor é válido
52              if matriz[i][ii] not in '123456789.':
53                  invalido()
54
```

compara(parâmetro)

Essa função foi criada devido a repetição que tinha nas próximas 3 funções, que vão usar essa para fazer as comparações. É uma função que recebe como parâmetro um array, nesse array ela vai iterar sobre os itens e verificar se o valor é diferente de '.', se for igual ele vai ignorar, se for diferente a função vai pegar esse valor e procurar quantas vezes ele repete no array através da função **count()**, própria do Python, se o item aparecer mais de uma vez o programa vai printar que o tabuleiro é inválido no terminal e encerrar o código.

```
def compara(array):
    # Para cada valor do array recebido
    for x in array:
        # Se for diferente de '.' (vazio)
        if x != '.':
            # Conte quantas vezes ele se repete dentro do array
            repet = array.count(x)
            # Se houver repetição
            if repet > 1:
                invalido()
```

comparaLinhas()

Depois de verificar se os dados são possíveis começam as comparações do sudoku de fato. Nessa função está a comparação de repetição entre as linhas. Ela se trata de um looping que passa por todas as linhas e pega todos os valores dessas linhas e armazena em um array chamado linha[]. Com os dados no array ela inicia a verificação de repetições através da função **compara()**, se não houver repetições ele segue para a próxima linha e repete.

```
def comparaLinhas():
    # Para cada linha
    for i in range(9):
        # Transforme essa linha num novo array
        linha = matriz[i]
        # Verifica repetição
        compara(linha)
```

comparaColunas()

Após verificar as repetições nas linhas, caso o tabuleiro ainda seja válido, então o programa vai verificar se há repetição nas colunas. A lógica por trás é a mesma da **comparaLinhas()**, mas é um pouco mais complexo. A função vai iterar pelas colunas, dentro dessa iteração ela vai percorrer, através de um for, todos os itens da coluna e jogar dentro de um array, através da função **append()**, chamado coluna[], quando terminar essa coluna ele vai chamar a função **compara()** e vai fazer a comparação dos itens da coluna[], caso não haja repetição ele vai para a próxima coluna e repete até acabar as colunas.

```python
def comparaColunas():
    # Para cada coluna
    for ii in range(9):
        coluna = []
        # Pegue todos os valores da coluna
        for i in range(9):
            # Coloque eles num array
            coluna.append(matriz[i][ii])
        # Verifica repetição
        compara(coluna)
```

comparaBlocos()

Após as linhas e colunas, falta verificar as regiões. Possivelmente essa é a verificação mais complexa por pegar linhas e colunas em cada verificação. Nessa função há dois for que vai de 3 em 3 começando do 0, para pegar o inicio de cada região. Então inicia-se o array bloco[] que vai receber os valores que serão comparados, depois disso começa outros dois for que vão até o 3 número após o início, tanto da linha quanto da coluna, nessa iteração os valores vão sendo colocados, através da função **append()**, no bloco[], assim que a iteração pelo bloco acaba ele chama **compara()** e põe bloco[] como parâmetro, fazendo a verificação das repetições, se o bloco for válido ele passa para o bloco do lado até não haver mais, então ele passa para o bloco inferior da esquerda e repete o processo.

```python
80
81  def comparaBlocos():
82      # Vá de 3 em 3 linhas
83      for i in range(0, 9, 3):
84          # E de 3 em 3 colunas
85          for ii in range(0, 9, 3):
86              bloco = []
87              # Para cada linha dessas 3 linhas
88              for iii in range(3):
89                  # E cada valor dessas linhas
90                  for iiii in range(3):
91                      # Coloque eles numa matriz
92                      bloco.append(matriz[i+iii][ii+iiii])
93              # Verifica repetição
94              compara(bloco)
95
```

desenhaMatriz()

Se o código não foi interrompido por erros, ao chegar ao fim, essa função é chamada para exibir o estado final do tabuleiro, que já foi verificado. Ela percorre os dados do tabuleiro linha por linha, e, no início e fim de cada linha, imprime o caractere '|', representando as bordas laterais do tabuleiro. Durante essa iteração, há uma verificação que identifica se o valor atual é igual a '.', que representa uma célula vazia. Se for, o caractere '.' é substituído por um espaço em branco ao exibir na tela, para que o vazio seja representado visualmente. Caso contrário, o valor original é impresso. Quando uma

linha é concluída, a função passa para a próxima e repete o processo, formando assim a representação completa do tabuleiro.

```python
# Desenha a matriz
def desenhaMatriz():
    # Para cada linha
    for i in range(9):
        # Inicie ela com | sem quebrar a linha
        print('|', end=' ')
        # Para cada valor da linha
        for ii in range(9):
            # Se for equivalente a vazio
            if matriz[i][ii] == '.':
                # Digite vazio sem quebrar linha
                print(' ', end='  ')
            # Senão
            else:
                # Digite o valor e não quebre linha
                print(matriz[i][ii], end='  ')
        # Finalize a linha com |
        print('|\n')
```

Obs: na função print foi usado end = ‘ ’ para não quebrar as linhas na hora de digitar os valores da linha