

Gerando Placas Mecosul

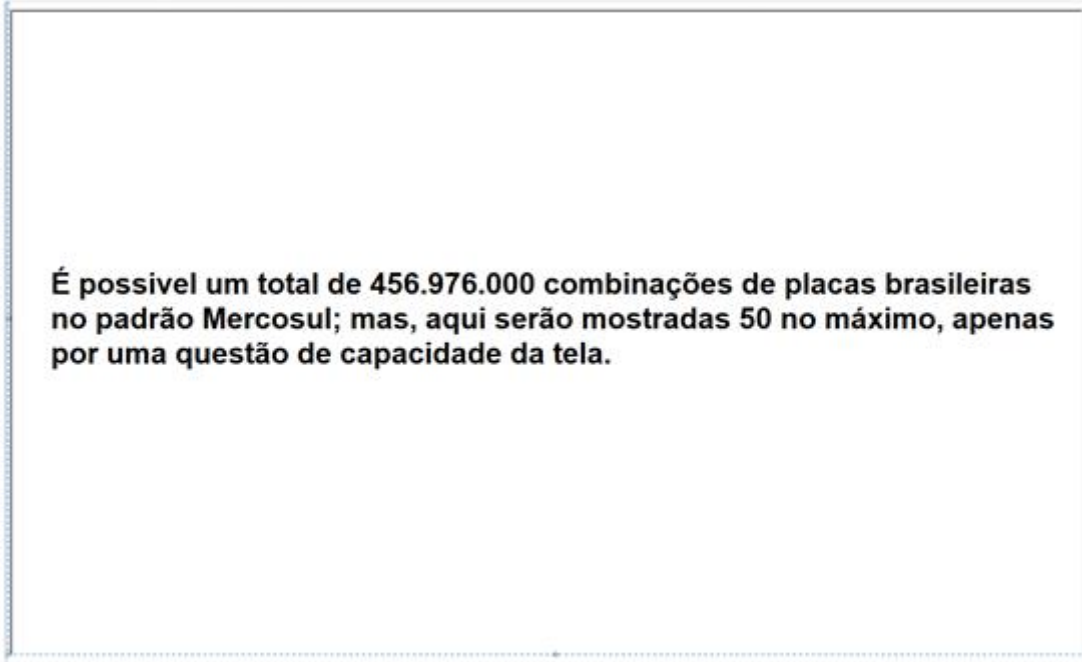
Mário Leite

...

O Brasil adotou o “Novo Sistema de Placas de Identificação Veicular” com o objetivo de se integrar ao Mercosul, criando um sistema único de identificação dos veículos automotores no âmbito dos países deste bloco. Estas placas são formatadas no seguinte modelo: três letras maiúsculas + um dígito + uma letra maiúscula + mais dois dígitos: por exemplo: **ANA1N81** e **ANA2N87**. E mais: com o nome “Brasil” centralizada acima destes caracteres, e mais a bandeira do Brasil no canto superior direito; e no canto inferior esquerdo a sigla “BR”. Esta nova configuração permite 456.976.000 placas diferentes, aumentando o número de combinações possíveis. Eu não conheço a lógica do programa que gera as placas oficiais no Brasil mas, o programa **"GeraPlacasMercosul"**, em Python, pode ser uma solução. A **figura 1a** mostra um aviso inicial (durando 9 segundos) para o usuário, e a **figura 1.b** mostra o *input* do programa para entrar com o número de placas desejado. A **figura 2** mostra as placas geradas (45 solicitadas). As cores das placas indicam os seguintes tipos: **vermelha** (veículo de aluguel), **verde** (veículo especial), **magenta** (veículo de colecionador), **azul** (veículo oficial), **amarela** (veículo do corpo diplomático) e **preta** (veículo particular). Estas cores são definidas de acordo com o tipo de número formado pelos três dígitos da placa: *primo*, *capicua*, *tetraédrico*, *triangular*, *hexagonal* e *normal*, respectivamente. Os cinco primeiros tipos primeiros são gerados em função de funções específicas; e o sexto tipo (números normais) não são gerados por funções.

Finalmente; que me desculpem os *Pythoneiros* mais puros (e em particular meu parceiro **Clésio Matias** que foi o responsável pela maior parte do código-fonte), mas tive que criar terminadores para as estruturas pois, eu (particularmente) não consigo codificar em Python sem terminadores; me passa a impressão de que o código fica “manco”, tal como o Saci Pererê...

Para adquirir o *pdf/e-book* de alguns livros meus sobre programação, entre em contato pelo *e-mail*: **marleite@gmail.com**



É possível um total de 456.976.000 combinações de placas brasileiras no padrão Mercosul; mas, aqui serão mostradas 50 no máximo, apenas por uma questão de capacidade da tela.

Figura 1a - Aviso de abertura do programa sobre a criação das placas



Figura 1b - Tela do menu principal do programa

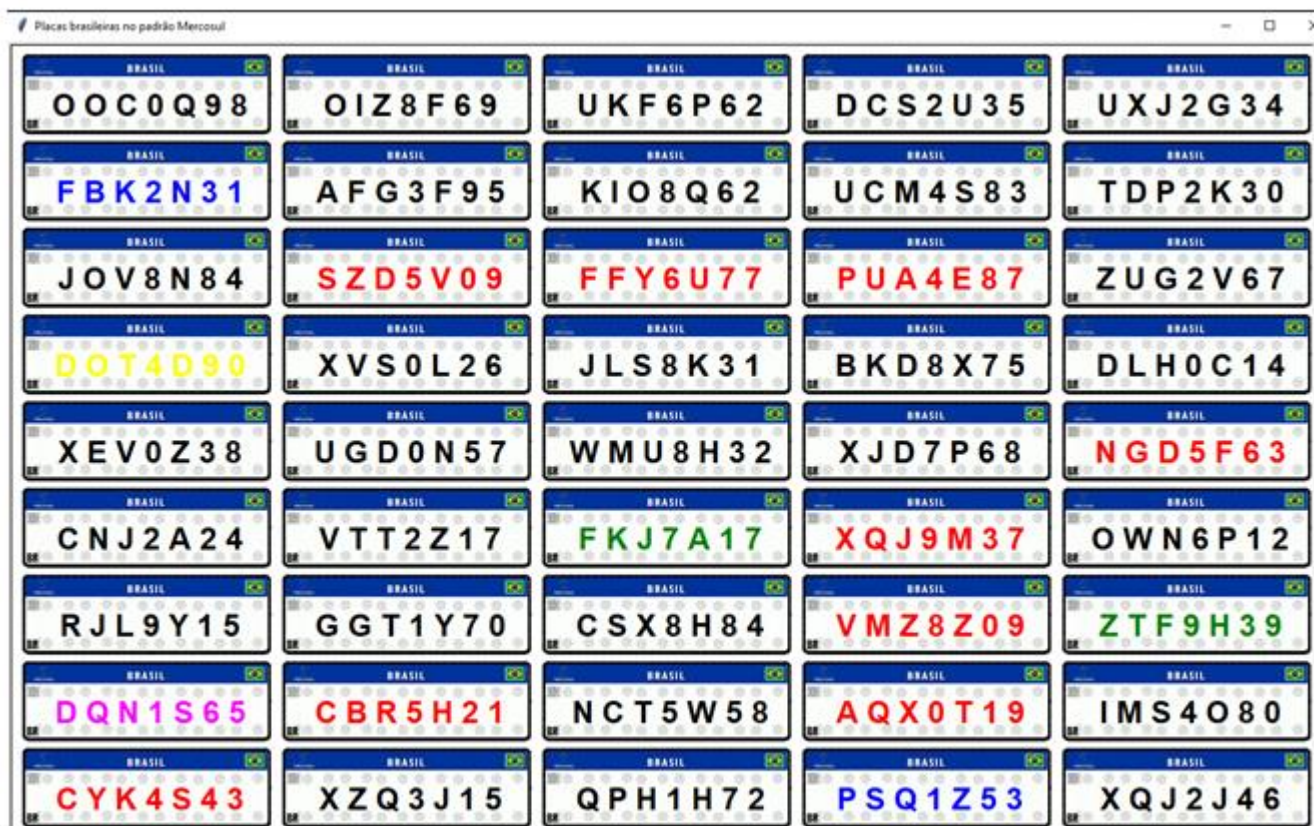


Figura 2 - Exemplo de saída do programa gerando 45 placas

```

'''
Gera placas brasileiras no padrão Mercosul, de maneira randômica.
Em Python
Autores: Clésio Matias e Mário Leite
'''

# -----
# Escopo de importações e definições
import turtle as t
from random import choice
from time import sleep
import math

#Definições de terminadores
endif = "endif"
endfor = "endfor"
enddef = "enddef"
endwhile = "endwhile"
# -----

# Escopo de declarações globais
MAXPLC = "456.976.000" # número máximo de placas que podem ser geradas
novoGeração = True # controla o loop de geração de placas ou saída do programa
listaDeLetras = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
'O', 'P', 'Q', 'R', 'S', 'T',
'U', 'V', 'W', 'X', 'Y', 'Z']
listaDeNumeros = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
listaTetraedicos = []
listaTriangulares = []
listaHexagonais = []
t.register_shape('bg.gif')
posXInicial = -540
posYInicial = 404
variacaoX = 270
variacaoY = -90
variantePosX = 0
variantePosY = 0
placaGerada = " "
# -----

# Escopo de funções
def criaTextoDaPlaca():
    #Cria os caracteres da placa
    global placaGerada
    texto = []
    letras = []
    num = []
    # sorteia as letras
    for l in range(4):
        letras.append(choice(listaDeLetras))
    endfor
    # sorteia os números
    for n in range(3):
        num.append(choice(listaDeNumeros))
    endfor
    # formatando os valores:
    texto.append(letras[0])
    texto.append(letras[1])

```

```

    texto.append(letras[2])
    texto.append(str(num[0]))
    texto.append(letras[3])
    texto.append(str(num[1]))
    texto.append(str(num[2]))
    textoFormatado = ''
    for i in texto:
        textoFormatado += f'{i} '
        placaGerada = textoFormatado
    endfor
    return textoFormatado
enddef

# -----
def geraPlaca(cordX, cordY):
    # Gera e imprime a placa na cor correta
    global numero
    global priDig, segDig, terDig
    global letra4
    texto = criaTextoDaPlaca()
    placa = t.Turtle()
    placa.speed(0)
    placa.pu()
    placa.setpos(cordX, cordY)
    placa.shape('bg.gif')
    escrita = t.Turtle()
    escrita.speed(0)
    escrita.pu()
    escrita.ht()
    ''' Analisa a quarta letra os dígitos da placa para definir a cor'''
    letra4 = placaGerada[8:9] #quarta letra da placa
    priDig = placaGerada[6:7] #primeiro dígito da placa
    segDig = placaGerada[10:11] #segundo dígito da placa
    terDig = placaGerada[12:13] #terceiro dígito da placa
    numero = int(priDig + segDig + terDig)

    # Chama funções para definir a cor da placa
    ehPrimo = VerifPrimo(numero)
    ehCapicua = VerifCapicua()
    ehTetraedico = VerifTetraedico(numero)
    ehTriangular = VerifTriangular(numero)
    ehHexagonal = VerifHexagonal(numero)

    # Define a cor da placa
    if (ehPrimo):
        cor = "red" # cor vermelha: placa de veículo "Aluguel"
    elif (ehCapicua):
        cor = "green" # cor verde: placa de veículo "Especial"
    elif (ehTetraedico):
        cor = "magenta" # cor magenta placa de veículo "Colecionador"
    elif ((ehTriangular) and not(ehPrimo)):
        cor = "blue" # cor azul placa de veículo "Oficial"
    elif (ehHexagonal):
        cor = "yellow" # cor amarela placa de veículo "Diplomáticos"
    else:
        cor = "black" # cor preta placa de veículo "Particular"
    endif

    # Imprime a placa na cor correta
    escrita.color(cor)
    escrita.setpos(placa.xcor() + 10, placa.ycor() - 30)

```

```

    escrita.write(texto, False, align='center', font=('FE', 24, 'bold'))
    sleep(1)
    # retornando o valor para não haver repetição
    return texto
enddef

# -----
def avisoInicial():
    # Emite o aviso inicial na tela
    aviso = t.Turtle()
    aviso.speed(0)
    aviso.pu()
    aviso.ht()
    aviso.write(
        f' É possível um total de {MAXPLC} combinações de placas brasileiras\n no padrão
Mercosul; mas, '
        f' aqui serão mostradas 50 no máximo, apenas\n por uma questão de capacidade da
tela.', False, align='center',
        font=('Arial', 28, 'bold'))
    sleep(9)
    aviso.clear()
enddef

# -----
def VerifPrimo(num):
    # Verifica se os três números da placa é um primo
    if(num==1):
        return False
    else:
        if((num % 2 == 0) and (num != 2)):
            return False
        else: #Calcula os divisores da Raiz(Int(Num))
            IntRaiz = int(math.sqrt(num))
            TemDiv = False
            for j in range(2, (IntRaiz+1)):
                RDiv = num % j
                if(RDiv==0):
                    TemDiv = True
                    break #abandona (encontrou algum divisor (num não é primo)
            endif
        endfor
        if(TemDiv==False): #não encontrou nenhum divisor
            return True
        else: #encontrou algum divisor
            return False
        endif
    endif
enddef

# -----
def VerifCapicua():
    # Verifica um número "Capicua" com os três dígitos da placa gerada
    numCap = (priDig + segDig + terDig)
    paCnum = (terDig + segDig + priDig)
    if (numCap == paCnum):
        return True
    else:
        return False
    endif
enddef

```

```
# -----
def VerifTetraedico(numero):
    # Verifica se os dígitos da placa formam um "Número Tetraédico"
    '''Gera os primeiros 17 números Tetraédicos'''
    for j in range(1, 18):
        numTetraedico = (j * (j + 1) * (j + 2)) / 6 # gera um Número Tetraédico
        numTetraedico = round(numTetraedico)
        listaTetraedicos.append(numTetraedico)
    endfor
    # Verifica se os dígitos da placa está em "listaTetraedicos"
    if (numero in listaTetraedicos):
        return True
    else:
        return False
    endif
enddef
```

```
# -----
def VerifTriangular(numero):
    #Gera números Triangulares na faixa [1-999]
    for j in range(1,1000):
        '''{Resolve a equação  $j = n(n+1)/2 \Rightarrow n^2 + n - 2*j = 0$  }'''
        delta = 1 - 4*1*(-2*j)
        raizDelta = math.sqrt(delta)
        if(int(raizDelta)==raizDelta): # número j é triangular
            listaTriangulares.append(j)
        endif
    endfor
    # Verifica se os dígitos da placa está em "listaTriangulares"
    if(numero in listaTriangulares):
        return True
    else:
        return False
    endif
enddef
```

```
# -----
def VerifHexagonal(numero):
    # Gera primeiros 23 números hexagonais
    for j in range(1,23):
        numHexagonal = j*(2*j -1) #expressão de um número Hexagonal
        listaHexagonais.append(numHexagonal)
    endfor
    # Verifica se os dígitos da placa está em "listaHexagonais"
    cond1 = numero in listaHexagonais
    cond2 = VerifTriangular(numero)==False
    cond3 = VerifTetraedico(numero) == False
    cond4 = VerifCapicua() == False
    if((cond1) and ((cond2) or (cond3) or (cond4))):
        return True
    else:
        return False
    endif
enddef#
```

```
=====

# Corpo do programa principal
janela = t.Screen()
janela.setup(1380, 930)
janela.title('Placas brasileiras no padrão Mercosul')
```

```

while novoGeração:
    # reset da tela para nova busca
    janela.clear()
    janela.screensize(1380, 930)
    variantePosY = 0
    variantePosX = 0
    avisoInicial()

    #Trata o botão [Cancel] na entrada
    valor = None
    while(valor == None):
        valor = janela.numinput(f'Digite quantas placas', f'[min-1 , max-50]', 1,
                                minval=1, maxval=50)
    endwhile

    valor = int(valor)
    if valor >= 1:
        if valor > 5:
            cont = valor
            for linhas in range(valor - 4):
                variantePosX = 0
                if cont > 5:
                    for colunas in range(5):
                        geraPlaca(posXInicial + variantePosX, posYInicial + variantePosY)
                        variantePosX += variacaoX
                    endfor
                    cont -= 5
                    variantePosY += variacaoY
                else:
                    variantePosX = 0
                    for colunas in range(cont):
                        geraPlaca(posXInicial + variantePosX, posYInicial + variantePosY)
                        variantePosX += variacaoX
                    endfor
                    break
                endif
            endfor
        else:
            for colunas in range(valor):
                geraPlaca(posXInicial + variantePosX, posYInicial + variantePosY) #normal
                variantePosX += variacaoX
            endfor
        endif
        continuar = janela.textinput('Nova busca?', '[S/N]')

        # Trata o botão [Cancel] no encerramento
        if(continuar == None):
            janela.clear()
            janela.title('Encerrando o programa...')
            sleep(2)
            exit()
        endif
    endif

    while True:
        if continuar not in 'SsNn':
            continuar = janela.textinput('Escolha Inválida!\nNova busca?', '[S/N]')
        elif continuar in 'Ss':
            break
        elif continuar in 'Nn':
            novoGeração = False
            janela.clear()

```

```
        janela.title('Encerrando o programa...')
        sleep(2)
        exit() # encerra o programa
    enddef
endwhile
endwhile
```

```
janela.mainloop() # mantém a janela gráfica aberta
```

```
# ===== Fim do programa =====
```