



[estudar.com.vc](https://estudar.com.vc)

# Python

## Resumo e Exercícios P2

### Parte 1





## Resuminho Teórico

### Listas

*Estrutura sequencial indexada.*

```
lista = [] # Lista vazia => []
lista.append(3) # Adiciona o numero 3 a lista => [3]
lista.append(10) # Adiciona o numero 3 a lista => [3, 10]
len(lista) # retorna o tamanho da lista => 2
```

Podem haver listas de vários tipos, não só números, inclusive de tipos diferentes.

```
lista2 = ["TR", 14, True, 1.4]
val = lista2[1] # val = 14, listas são indexadas em 0
```

Fatias de listas: pode-se usar a notação lista[ini:fim] para acessar parte da lista, retornando os valores entre os índices ini e fim-1.

```
primos = [2, 3, 5, 7, 11]
primos[1:2] # [3]
primos[2:4] # [5, 7]
primos[:3] # Sem ini, retorna desde o primeiro valor => [2, 3, 5]
primos[3:] # Sem fim, retorna até o último valor => [7, 11]
primos[:] # Sem ambos, retorna a lista toda => [2, 3, 5, 7, 11]
```

Clones de lista: Cuidado ao igualar uma lista a outra.

```
p = primos # Aqui p é uma referencia a primos
p[2] = 6 # Troco o valor de p[2] para 6
print(primos[2]) # Imprime 6, ou seja, também troquei o valor de primos[2]
# Voltando para primos = [2, 3, 5, 7, 11]
p = primos[:] # Agora p é um clone de primos
p[2] = 6 # Troco o valor de p[2] para 6
print(primos[2]) # Imprime 5, não mudou o valor
```

Isso é especialmente importante quando for passar uma lista para uma função

### For e range()

For: loop, assim como o while, serve para iterar em listas.

```
for elem in lista2: # Para cada elemento "elem" na lista "lista2" faça:
    print(elem) # Imprime o elemento na tela
```

Range(inicio, fim): função que retorna uma lista de inteiros [inicio, inicio+1, ..., fim-1], pode-se omitir "inicio", que valerá 0.

```
for i in range(len(lista)): # range(len(lista)) = range(2) = [0, 1]
    print(lista[i]) # imprime cada elemento da lista
```



## Listas de listas (matrizes)

```
# Cria uma matriz de m linhas e n colunas, inicializa com v
m, n, v = 5, 6, ' ' # Poderia ser o parametro de uma funcao ou pedido para o usuario
matriz = [] # Lista vazia
for i in range(n): # Itera nas linhas
    linha = []
    for j in range(m): # Cada elemento da linha é v
        linha.append(v)
    matriz.append(linha) # Cada elemento da lista "matriz" é uma lista
matriz[2][3] = 4 # matriz[2] -> acessa a terceira linha; matriz[2][3] -> 4º elemento da linha
```

## Strings

### “Listas de caracteres”

Podem ser tratadas como listas, porém são imutáveis, ou seja, não é possível modificar seus caracteres individualmente, sendo necessário criar uma nova string para isso, além disso, não possuem o método `append()`

```
texto = "Ola! A ThundeRatz e muito legal!"
len(texto) # 32
texto[7:15] # "ThundeRa"
texto[4] = "p" # ERRO
for c in texto: # Itera em cada caractere
    print(c, end='*') # O*L*a*!* *A* *T*h*u*n*d*e*R*a*t*z* *e* *m*u*i*t*o* *L*e*g*a*L*!*
```

## **Exercício de fixação**

### **1. Campo Minado**

*Internet*

Campo Minado é um jogo que se tornou muito popular por acompanhar o sistema operacional Microsoft Windows.

Nesse jogo, o campo minado pode ser representado por uma matriz retangular. O jogador deve revelar todas as posições livres (sem bomba) da matriz, clicando em uma posição com conteúdo desconhecido. O jogo acaba quando o jogador clicar em uma posição com bomba, ou quando todas as posições livres forem abertas.

Nesse exercício, você deve implementar algumas funções que podem ser utilizadas na implementação desse jogo.



- a) Escreva uma função que recebe como parâmetros uma matriz inteira A e uma posição (lin, col) da matriz, e conta quantas posições ao redor da posição (lin, col) contém o valor -1 (valor adotado para representar uma bomba)
- b) Escreva um programa que lê uma matriz A de 0's (posições livres) e -1's (bomba). Utilizando a função do item anterior, o programa deve computar e imprimir a quantidade de bombas ao redor de cada posição livre da matriz.

## Exercício de Prova

### 1.

P2 2014 – Questão 1

Escreva uma função que recebe três sequências de caracteres (strings) não vazias, sendo que a terceira é um caractere coringa. A função deve devolver uma sequência que é uma cópia da primeira sequência, porém substituindo-se as ocorrências dos caracteres da segunda sequência pelo caractere coringa, obedecendo a ordem de ocorrência em ambas as sequências.

## Gabarito Exercício de Fixação

### 1.

```
# a
def conta_bomba(A, lin, col):
    posicoes_com_bomba = 0
    if lin > 0 and A[lin-1][col] == -1:
        posicoes_com_bomba += 1
    if lin < len(A) - 1 and A[lin+1][col] == -1:
        posicoes_com_bomba += 1
    if col > 0 and A[lin][col-1] == -1:
        posicoes_com_bomba += 1
    if col < len(A[0]) - 1 and A[lin][col+1] == -1:
        posicoes_com_bomba += 1

    return posicoes_com_bomba
```



```
# b
def main():
    m = int(input("Numero de linhas: "))
    n = int(input("Numero de colunas: "))
    A = []
    for i in range(m):
        linha = []
        for j in range(n):
            linha.append(int(input("Digite A[%d][%d]: " % (i, j))))
        A.append(linha)

    for i in range(m):
        for j in range(n):
            print("Bombas ao redor de A[%d][%d]: %d" % (i, j, conta_bomba(A, i, j)))

main()
```

## Gabarito Exercício de Prova

### 1.

```
def substitui(str1, str2, coringa):
    nova_str = "" # string vazia
    j = 0
    for i in range(len(str1)):
        # determine próximo caractere de nova_str
        if j < len(str2) and str1[i] == str2[j]:
            caractere = coringa
            j += 1
        else:
            caractere = str1[i]
        # concatena com nova_str
        nova_str += caractere
    return nova_str
```