

# Prática de Programação em Sistemas de Informação

Listas





## Desenvolvimento do material

Alexandre Louzada

### 1<sup>a</sup> Edição

Copyright © 2022, Unigranrio

Nenhuma parte deste material poderá ser reproduzida, transmitida e gravada, por qualquer meio eletrônico, mecânico, por fotocópia e outros, sem a prévia autorização, por escrito, da Unigranrio.

# Sumário

## Listas

Para Início de Conversa...	3
Objetivo	3
1. Criação e Uso de Listas	4
2. Listas de Listas	8
3. Compreensão de Listas	9
Referências	10



## Para Início de Conversa...

Para resolver um problema com programação, talvez precisemos utilizar uma quantidade considerável de dados. Nesse caso, utilizar apenas variáveis pode ser bem trabalhoso. Imagine, por exemplo, que um programador precise armazenar o nome de vários produtos e, para isso, utilize variáveis. Não é difícil perceber que, quanto maior a quantidade de produtos, maior será a quantidade de variáveis necessárias para armazenar e manipular os dados. Felizmente, a linguagem Python possui algumas maneiras de lidar melhor com essa situação. Uma delas é agrupar os dados em uma coleção de dados, denominada lista.

## Objetivo

Utilizar listas para armazenar e manipular dados.



# 1. Criação e Uso de Listas

Uma lista é uma coleção de valores que pode ser modificada. Cada valor na lista é identificado por um índice ou **posição**. Os valores que formam uma lista são chamados **elementos** ou **itens**.

Na linguagem Python, as listas são escritas com colchetes. Vamos descrever, por meio do exemplo de uma lista de produtos, como criar e utilizar listas:

```
produtos = ["mouse", "teclado", "impressora"]
print(produtos)
```



No exemplo, foi criada uma lista de strings representando três produtos. Uma lista é uma sequência de objetos, em que esses objetos podem ser de qualquer tipo: números, strings, ou até mesmo outras listas.

O Python tem um conjunto de métodos internos que podem ser utilizados para manipular as listas.

Método	Descrição
append()	Adiciona um elemento no final da lista
clear()	Remove todos os elementos da lista
copy()	Retorna uma cópia da lista
count()	Retorna o número de elementos com o valor especificado
extend()	Adiciona os elementos de uma lista (ou qualquer iterável) ao final da lista atual
index()	Retorna o índice do primeiro elemento com o valor especificado
insert()	Adiciona um elemento na posição especificada
pop()	Remove o elemento na posição especificada
remove()	Remove o item com o valor especificado
reverse()	Inverte a ordem da lista
sort()	Ordena a lista

Vamos utilizar novamente o exemplo da lista de produtos para entender mais sobre o uso de índices e métodos para manipular uma lista.

#Você acessa os itens da lista produtos referindo-se ao número do índice:

```
produtos = ["mouse", "teclado", "impressora"]
print(produtos)
print(f'Segundo produto da lista: {produtos[1]}')
print()
```

```
#Para alterar o valor de um item específico, utilizamos o número do índice:  
produtos[1] = "notebook"  
print('Lista atualizada com ajuda do índice!')  
print(produtos)  
print()
```

```
#Você pode percorrer os itens da lista produtos usando um comando for:  
print('Relação de produtos:')  
print('*'*20)  
for x in produtos:  
    print(x)  
print('*'*20)  
print()
```

```
#Para determinar se um item especificado está presente em uma produtos, use a palavra-chave  
in:  
if "mouse" in produtos:  
    print("Sim, 'mouse' está na lista de produtos")
```

```
#Para determinar quantos itens a lista produtos possui, use o método len():  
print(f'Tamanho da lista de produtos: {len(produtos)}')  
print()
```

```
#Para adicionar um item ao final da lista produtos, use o método append ():  
produtos.append("monitor")  
print('Lista atualizada com monitor!')  
print(produtos)  
print()
```

```
#Para adicionar um item no índice especificado, use o método insert ():  
produtos.insert(1, "modem")  
print('Lista atualizada com modem!')  
print(produtos)  
print()
```

```
#Existem vários métodos para remover itens de uma lista produtos:  
#O método remove() remove o item especificado:  
produtos.remove("modem")  
print('Lista atualizada sem modem!')  
print(produtos)  
print()
```

```
#O método pop() remove o índice especificado (ou o último item, se o índice não for  
especificado):  
produtos.pop()  
print('Lista atualizada sem monitor!')  
print(produtos)  
print()
```

#O comando del remove o índice especificado:

```
del produtos[0]
print('Lista atualizada!')
print(produtos)
print()
```

#O comando del também pode excluir completamente a lista:

```
del produtos
```

#O método clear() esvazia a lista de produtos:

```
produtos = ["mouse", "teclado", "impressora"]
print('Lista cheia!')
print(produtos)
produtos.clear()
print('Lista vazia...')
print(produtos)
print()
```

#Também é possível usar o construtor list () para criar uma produtos.

```
produtos = list(("mouse", "teclado", "impressora"))
print('Lista final!')
print(produtos)
```

Vejamos, agora, alguns exemplos de resolução de problemas típicos de introdução à programação, com a aplicação de listas. Teste cada programa e analise os resultados:

"""

Faça um programa que verifique se foi digitado um valor inteiro de zero a dez.

"""

```
valores = ['zero', 'um', 'dois', 'três', 'quatro',
           'cinco', 'seis', 'sete', 'oito', 'nove',
           'dez']
```

while True:

```
    numero = int(input('Digite um número entre 0 e 10: '))
```

```
    if 0 <= numero <= 10:
```

#verifica se o número é maior ou igual a zero e menor ou igual a dez

```
        break #força a saída do while
```

```
    print("Tente novamente. ; end=")
```

```
    print(f'Você digitou o número {valores[numero]}'")
```

"""

Um programa que armazena em uma lista 20 times de futebol e ao final informa a lista completa, os 5 primeiros, os 4 últimos, e lista ordenada e a posição de um determinado time de acordo com o seu nome

"""

```

times = ('Corinthians', 'Palmeiras', 'Santos', 'Grêmio',
         'Cruzeiro', 'Flamengo', 'Vasco', 'Chapecoense',
         'Atlético', 'Botafogo', 'Atlético-PR', 'Bahia',
         'São Paulo', 'Fluminense', 'Sport Recife',
         'EC Vitória', 'Coritiba', 'Avaí', 'Ponte Preta',
         'Atlético-GO')

print('=' * 15)
print(f'Lista de times do Brasileirão: {times}')
print('=' * 15)
print(f'Os 5 primeiros são {times[0:5]}')
print('=' * 15)
print(f'Os 4 últimos são {times[-4:]}')
print('=' * 15)
print(f'Times em ordem alfabética: {sorted(times)}')
print('=' * 15)
print(f'O Chapecoense está na {times.index("Chapecoense") + 1}ª posição')

```

"""

Faça um programa que armazene quatro valores inteiros digitados pelo usuário em uma lista.

Ao final, o programa deve informar:

\* os valores da lista

\* a quantidade de ocorrências do número 9

\* a posição (ou posições) da lista em que o número 3 aparece

\* os valores pares da lista

"""

```

numero = (int(input('Digite um número: ')),
          int(input('Digite outro número: ')),
          int(input('Digite mais um número: ')),
          int(input('Digite o último número: ')))
print(f'Você digitou os valores {numero}')
print(f'O valor 9 apareceu {numero.count(9)} vezes')
if 3 in numero:
    print(f'O valor 3 apareceu na {numero.index(3)+1}ª posição')
else:
    print('O valor 3 não foi digitado em nenhuma posição')
print('Os valores pares digitados foram', end="")
for n in numero:
    if n % 2 == 0:
        print(n, end=' ')

```

"""

*Em uma lista de palavras, selecionar e exibir as vogais encontradas em cada uma*

"""

```

palavras = ('aprender', 'programar', 'linguagem', 'python',
            'curso', 'leitura', 'estudar', 'praticar',
            'trabalhar', 'mercado', 'programador', 'futuro')
for p in palavras:
    print(f'\nNa palavra {p.upper()} temos ', end="")
    for letra in p:
        if letra.lower() in 'aeiou':
            print(letra, end=' ')

```

....

Faça um programa que armazene em uma lista 5 números inteiros e, ao final, informe os valores da lista, o maior e o menor valor digitado

....

```
listanum = []
maior = 0
menor = 0
for c in range(0, 5):
    listanum.append(int(input(f'Digite um valor para a Posição {c}: ')))
    if c == 0:
        maior = menor = listanum[c]
    else:
        if listanum[c] > maior:
            maior = listanum[c]
        if listanum[c] < menor:
            menor = listanum[c]
print('=-' * 30)
print(f'Você digitou os valores {listanum}')
print(f'O maior valor digitado foi {maior}', end="")
print()
print(f'O menor valor digitado foi {menor}', end="")
```

## 2. Listas de Listas

Matrizes são estruturas bidimensionais (tabelas) formadas por linhas e colunas. Elas são muito importantes na matemática, utilizadas, por exemplo, para a resolução de sistemas de equações e transformações lineares.

Em Python, uma matriz pode ser representada como uma lista de listas, em que um elemento da lista contém uma linha da matriz, que, por sua vez, corresponde a uma lista com os elementos da coluna da matriz.

Observe um exemplo de uso de listas para construir uma matriz que armazena quatro números inteiros e, ao final, exibe a soma dos elementos da matriz.

```
matriz = [[0, 0], [0, 0]] # matriz com duas linhas e duas colunas
soma = 0
for l in range(0, 2): # controla a linha da matriz
    for c in range(0, 2): # controla a coluna da matriz
        matriz[l][c] = int(input(f'Digite um valor para [{l}, {c}]: '))
        soma += matriz[l][c]
print('=-' * 20)
for l in range(0, 2):
    for c in range(0, 2):
        print(f'[{matriz[l][c]}]', end="")
    print()
print('=-' * 20)
print(f'Soma dos valores da matriz: {soma}', end="")
```

### 3. Compreensão de Listas

Uma compreensão de lista é, na verdade, uma lista criada, em tempo real, durante a execução do programa, e não é descrita estaticamente.

A compreensão de lista itera sobre todos os elementos de uma lista e executa uma ação para cada item encontrado, de acordo com o filtro que utilizamos. Em outras palavras, é descrita com a seguinte estrutura:

[ação a ser tomada, itens iterados, filtro]

**ação a ser tomada:** Ação desejada para cada item.

**itens iterados:** lista que queremos iterar e extrair seus itens.

Filtro (opcional): Onde aplicamos as condições para devolver os itens de uma lista.



Digamos que um programador precise armazenar, em uma lista, oito elementos, contendo as oito primeiras potências de dois (1, 2, 4, 8, 16, 32, 64, 128), e exibi-los na tela. Com o uso de compreensão de listas, esse problema pode ser resolvido da seguinte forma:

```
lista = [2 ** i for i in range(8)]
```

```
print(lista)
```

"""

*Nessa versão, foi inserido um filtro para exibir os valores a partir de 2*

"""

```
lista = [2 ** i for i in range(8) if i > 0]
```

```
print(lista)
```

Ao utilizar compreensão de listas, podemos encurtar o código, o que pode facilitar o trabalho do programador. Para compreender isso de forma mais clara, analise os dois trechos de código:

"""

*Cria uma matriz 2x2 e inicializa a matriz com zeros*

"""

```
matriz = []
for i in range(2):
    linha = [0 for i in range (2)]
    matriz.append(linha)
print(matriz)
```

'''

Cria uma matriz 2x2 e inicializa  
a matriz com zeros usando compreensão de listas  
'''

```
matriz = [[0 for i in range(2)] for j in range(2)]  
print(matriz)
```

Podemos, também, aproveitar o conceito de compreensão de listas para criar matrizes em que seus valores sejam informados a partir da digitação do usuário. Observando cada colchete, note que a parte interna cria uma linha e a parte externa cria uma lista de linhas.

'''

Cria uma matriz 2x2 e armazena e exibe  
valores inteiros digitados pelo usuário e  
A soma dos elementos da matriz

'''

```
soma=0  
matriz = [[int(input('Valor: ')) for i in range(2)] for j in range(2)]  
print('=-'* 20)  
print('Exibindo os valores da matriz')  
print('=-'* 20)  
for l in range(0, 2):  
    for c in range(0, 2):  
        print(f'{[matriz[l][c]:^5]}', end="")  
        soma+=matriz[l][c]  
    print()  
print('=-'* 20)  
print(f'Soma dos valores da matriz: {soma}', end="")
```

Embora o uso de compreensão de listas apresente vantagens, como a redução do número de linhas de um programa, é importante avaliar com cuidado sua aplicação, para não prejudicar o entendimento do código.

Neste capítulo, vimos que uma lista é uma coleção de dados composta por dados organizados de forma linear, na qual cada um pode ser acessado a partir de um índice que representa sua posição na coleção (iniciando em zero).

Aprendemos, também, a trabalhar com listas em Python, linguagem na qual essa estrutura de dados pode armazenar, inclusive simultaneamente, objetos de diferentes tipos, como strings, números ou outras listas.

## Referências

PERKOVIC, L. **Introdução à computação usando Python**: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.