

RACIOCÍNIO ALGORÍTMICO

LISTAS

ALGORITMO

LISTAS

Listas em Python são estruturas de dados heterogêneas (podem armazenar quaisquer tipos de dados ao mesmo tempo) que permitem o armazenamento de informações de forma indexada, tendo como principais características:

- Estruturas mutáveis
- Simula o comportamento de vetor de outras linguagens de programação
- Simula o comportamento de matriz, podendo ser um vetor de vetores

PYTHON

LISTAS

- Criação:

```
myList = [3, 5, 'test', 0.2]  
myList = list((3, 5, 'test', 0.2))  
myList = []
```

- Acessar elemento da lista:

```
print(myList[1])
```

- Adicionar e remover elementos da lista

```
myList.append(10)  
del myList[0]
```

PYTHON

LISTAS

- Iterando a lista - for:

```
for element in myList:  
    print(element)
```

- Iterando a lista - while:

```
while count < len(myList):  
    print(myList[count])  
    count += 1
```

- Se um elemento existe na lista:

```
If element in myList:  
    print("Elemento encontrado")
```

ALGORITMO

LISTAS

Exemplo de aplicação 1: Elaborar um programa que solicita ao usuário 5 números, armazena-os em uma lista, exibindo como resultado os dados obtidos.

```
01. nums = [0, 0, 0, 0, 0]
02. for i in range(5):
03.     num = int(input("Digite um número: "))
04.     nums[i] = num
05. print(nums)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic01.py
Digite um número: 10
Digite um número: 20
Digite um número: 30
Digite um número: 40
Digite um número: 50
[10, 20, 30, 40, 50]

Process finished with exit code 0
```

ALGORITMO

LISTAS

Neste exemplo de aplicação, na linha 01, é criada uma lista com 5 elementos zerados, os quais são substituídos dentro do laço em todas as iterações da linha 04. Porém, o que aconteceria se no laço fossem solicitados 6 números ao invés de 5?

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic01.py
Digite um número: 10
Digite um número: 20
Digite um número: 30
Digite um número: 40
Digite um número: 50
Digite um número: 60
Traceback (most recent call last):
  File "/Users/user/projects/untitled1/First.py", line 4, in <module>
    nums[i] = num
IndexError: list assignment index out of range

Process finished with exit code 1
```

Este é o comportamento padrão de vetor, onde os valores devem ser inicializados para que sejam usados. Porém, neste ponto, pode ser usado o comportamento de listas para não ocorrer este tipo de problema, fazendo uso do método `append` da lista para adicionar elementos à mesma.

```
01. nums = []
02. for i in range(5):
03.     num = int(input("Digite um número: "))
04.     nums.append(num)
05. print(nums)
```

ALGORITMO

LISTAS

Exemplo de aplicação 2: *Elaborar um programa que solicita ao usuário 5 números e exibe duas listas, uma de números pares, e outra de números ímpares.*

```
01. pares = []
02. impares = []
03. for i in range(5):
04.     num = int(input("Digite um número: "))
05.     if num % 2 == 0:
06.         pares.append(num)
07.     else:
08.         impares.append(num)
09. print("Números pares:", pares, "- Números ímpares:", impares)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic02.py
Digite um número: 3
Digite um número: 8
Digite um número: 6
Digite um número: 11
Digite um número: 22
Números pares: [8, 6, 22] - Números ímpares: [3, 11]

Process finished with exit code 0
```

ALGORITMO

LISTAS

Exemplo de aplicação 3: *Dada a lista de dados `nums = [1, 4, 23, 11, 8]`, correr a lista usando um objeto `Range` e imprimir cada elemento em uma linha.*

```
01. nums = [1, 4, 23, 11, 8]
02. for i in range(len(nums)):
03.     print(nums[i])
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic03.py
```

```
1
4
23
11
8
```

```
Process finished with exit code 0
```

ALGORITMO

LISTAS

Exemplo de aplicação 4: *Dada a lista de dados $nums = [17, 33, 23, 11, 8, 15, 9]$, correr a lista e identificar qual o maior e menor número da mesma.*

```
01. nums = [17, 33, 23, 11, 8, 15, 9]
02. maior = nums[0]
03. menor = nums[0]
04. for num in nums:
05.     if num > maior:
06.         maior = num;
07.     if num < menor:
08.         menor = num
09. print("O maior número da lista é", maior, "e o menor é", menor)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic04.py
0 maior número da lista é 33 e o menor é 8

Process finished with exit code 0
```

ALGORITMO

LISTAS

Exemplo de aplicação 5: *Dada a lista de dados $nums = [17, 33, 23, 11, 8, 15, 9]$, ordená-la e mostrar o resultado ao usuário.*

```
01. nums = [17, 33, 23, 11, 8, 15, 9]
02. aux = 0
03. for _ in range(len(nums) - 1):
04.     for i in range(len(nums) - 1):
05.         if nums[i] > nums[i+1]:
06.             aux = nums[i]
07.             nums[i] = nums[i+1]
08.             nums[i+1] = aux
09. print(nums)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic05.py
[8, 9, 11, 15, 17, 23, 33]
```

```
Process finished with exit code 0
```

ALGORITMO

LISTAS COM COMPORTAMENTO DE MATRIZ

Matrizes possuem comportamento análogo aos vetores, porém com a capacidade de armazenamento multidimensional.

O vetor possui vários elementos, sendo que cada elemento ocupa uma posição do vetor. Analogamente, uma matriz bidimensional possui vários elementos dispostos como um vetor, porém cada posição do vetor possui um vetor de elementos.

Disposição em vetor:

```
v = [17, 23, 11, 8, 15, 9, 33]
```

Disposição em matriz:

```
m = [[3, 6, 8], [11, 5, 2], [1, 1, 21]]
```

No caso acima, basicamente, tem-se um vetor m com os seguintes elementos:

```
m[0] = [3, 6, 8]      m[1] = [11, 5, 2]      m[2] = [1, 1, 21]
```

ALGORITMO

LISTAS - COMPORTAMENTO DE MATRIZ

Exemplo de aplicação 6: *Solicitar ao usuário que digite 3 coordenadas (x, y), armazenando-as em uma matriz bidimensional.*

```
01. coordenadas = []
02. for i in range(3):
03.     x = int(input("Insira um valor de x: "))
04.     y = int(input("insira um valor de y: "))
05.     coordenadas.append([x, y])
06. print(coordenadas)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic06.py
Insira um valor de x: 1
insira um valor de y: 1
Insira um valor de x: 2
insira um valor de y: 2
Insira um valor de x: 3
insira um valor de y: 3
[[1, 1], [2, 2], [3, 3]]

Process finished with exit code 0
```

ALGORITMO

LISTAS - COMPORTAMENTO DE MATRIZ

Exemplo de aplicação 7: *Solicitar ao usuário que entre com diversos nomes, informando de forma separada nome e último sobrenome. Ao final, mostrar uma lista de nomes no formato: sobrenome, nome. Para encerrar a entrada de dados basta o usuário inserir no nome o texto sair.*

```
01. nomes = []
02. while True:
03.     nome = input("Digite o nome: ")
04.     if nome == "sair":
05.         break
06.     sobrenome = input("Digite o sobrenome: ")
07.     nome_completo = [nome, sobrenome]
08.     nomes.append(nome_completo)
09.
10. for nome in nomes:
11.     print(nome[1] + ", " + nome[0])
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic07.py
Digite o nome: Alcir
Digite o sobrenome: Damaceno
Digite o nome: Maria da Silva
Digite o sobrenome: Chaves
Digite o nome: Henrique
Digite o sobrenome: Maciel
Digite o nome: sair
Damaceno, Alcir
Chaves, Maria da Silva
Maciel, Henrique

Process finished with exit code 0
```

ALGORITMO

LISTAS - COMPORTAMENTO DE MATRIZ

Exemplo de aplicação 08. *Data a matriz $m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]$ efetuar a soma de todos os seus elementos e exibir o resultado.*

```
01. m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
02. soma = 0
03. for i in range(3):
04.     for j in range(3):
05.         soma += m[i][j]
06. print("A soma dos elementos da matriz m é igual a", soma)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic08.py
A soma dos elementos da matriz m é igual a 45
```

```
Process finished with exit code 0
```

PYTHON

LISTAS - MÉTODOS

Uma lista é um objeto em Python, ou seja, não faz parte dos tipos primitivos. Em sendo um objeto, possui métodos, que são ações que podem realizar. No caso, pode-se adicionar elementos, excluir elementos, entre outros. Segue a lista de métodos que podem ser aplicados sobre listas em Python:

Listas - Métodos	
Método	Descrição
append()	Adiciona um elemento no final da lista
extend()	Adiciona todos os elementos de uma lista para outra
insert()	Insere um elemento em um determinado índice da lista
remove()	Remove um elemento da lista
pop()	Remove e retorna um determinando elemento de um índice da lista
clear()	Remove todos os elementos da lista
index()	Retorna o índice do primeiro elemento localizado na lista
count()	Retorna a quantidade de um elemento na lista passado como argumento
sort()	Ordena os elementos de uma lista em ordem ascendente
reverse()	Inverte a ordem dos elementos de uma lista
copy()	Retorna uma cópia da lista

ALGORITMO

LISTAS - MÉTODOS

```
01. nums = [17, 33, 8, 11, 8, 15, 9]
02. print(nums)
03. # adiciona o elemento 10 ao final da
     lista
04. nums.append(10)
05. print(nums)
06. # extend nums com os elementos de nums2
07. num2 = [3, 7]
08. nums.extend(num2)
09. print(nums)
10. # insere o elemento 0 na posição 2 da
     lista
11. nums.insert(2, 0)
12. print(nums)
13. # remove o elemento 11 da lista
14. nums.remove(11)
15. print(nums)
16. # remove e retorna o elemento na
     posição 7 da lista
|
17. print(nums.pop(7))
18. print(nums)
19. # cria uma cópia da nums em numsCpy
20. numsCpy = nums.copy()
21. print(numsCpy)
22. # remove todos os elementos de nums
23. nums.clear()
24. print(nums)
25. # conta quantos elementos 8 existem na
     lista
26. nums = numsCpy.copy()
27. print(nums.count(8))
28. # ordena nums em ordem ascendente
29. nums.sort()
30. print(nums)
31. # inverte os elementos de nums
32. nums.reverse()
33. print(nums)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/listas_metodos.py
[17, 33, 8, 11, 8, 15, 9]
[17, 33, 8, 11, 8, 15, 9, 10]
[17, 33, 8, 11, 8, 15, 9, 10, 3, 7]
[17, 33, 0, 8, 11, 8, 15, 9, 10, 3, 7]
[17, 33, 0, 8, 8, 15, 9, 10, 3, 7]
10
[17, 33, 0, 8, 8, 15, 9, 3, 7]
[17, 33, 0, 8, 8, 15, 9, 3, 7]
[]
2
[0, 3, 7, 8, 8, 9, 15, 17, 33]
[33, 17, 15, 9, 8, 8, 7, 3, 0]
```

```
Process finished with exit code 0
```

PYTHON

LISTAS - FUNÇÕES BUILT IN

Funções Built in são funções internas da linguagem que podem ser utilizadas sobre um determinado objeto. No caso do Python, existem algumas funções que podem ser aplicadas sobre listas, facilitando a execução de alguns algoritmos que teriam que ser desenvolvidos pelo programador para realizar as mesmas funções.

PYTHON

LISTAS - FUNÇÕES BUILT IN

Python len()

Conforme já visto anteriormente, a função **len** retorna a quantidade de elementos de uma lista.

```
nums = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
len(nums) => 10
```

A função **len** também é muito usada para saber quantos caracteres possui uma string.

```
nome = "Python"
len(python) => 6
```

Pyhton sum()

A função **sum** retorna a soma de todos os elementos de uma lista.

```
nums = [17, 33, 8, 11, 8, 15, 9]
sums(nums) => 101
```

PYTHON

LISTAS - FUNÇÕES BUILT IN

Python max()

A função **max** retorna o maior valor dentro de uma lista.

```
nums = [17, 33, 8, 11, 8, 15, 9]
max(nums) => 33
```

Caso os elementos da lista sejam strings, a comparação é feita tomando por base a ordem alfabética.

```
heróis = ["Zorro", "Capitão América", "Hulk", "Super Homem"]
max(heróis) => "Zorro"
```

Python min()

A função **min** retorna o menor valor dentro de uma lista.

```
nums = [17, 33, 8, 11, 8, 15, 9]
min(nums) => 8
```

Da mesma forma que a função **max**, caso os elementos da lista sejam strings, a função **min** efetua a comparação tomando por base a ordem alfabética.

```
heróis = ["Zorro", "Capitão América", "Hulk", "Super Homem"]
max(heróis) => "Capitão América"
```

PYTHON

LISTAS - FUNÇÕES BUILT IN

Python sorted()

A função **sorted** retorna a lista passada em ordem ascendente.

```
nums = [17, 33, 8, 11, 8, 15, 9]
sorted(nums) => [8, 8, 9, 11, 15, 17, 33]
```

Também é possível fazer a ordenação em ordem descendente, fazendo uso do parâmetro **reverse**.

```
nums = [17, 33, 8, 11, 8, 15, 9]
sorted(nums, reverse=True) => [33, 17, 15, 11, 9, 8, 8]
```

Também é possível aplicar a função sorted sobre uma string, sendo retornada uma lista não editável, no caso, uma tupla, como será visto na próxima unidade.

```
nome = "José dos Santos"
sorted(nome) => [' ', ' ', 'J', 'S', 'a', 'd', 'n', 'o', 'o', 'o', 's', 's', 't', 'é']
```

PYTHON

LISTAS - FUNÇÕES BUILT IN

Exemplo de aplicação 9. Em uma competição de saltos ornamentais são obtidas dos jurados 7 notas, onde são eliminadas a nota maior e a nota menor, e a valoração do salto é feita pela soma das demais notas. Crie um programa que recebe as notas dos juízes, remove a maior e menor nota e soma as demais fazendo uso de métodos de listas e funções Built in.

```
01. notas = []
02. for _ in range(7):
03.     nota = float(input("Entre com uma das
notas: "))
04.     notas.append(nota)
05. menor = min(notas)
06. if notas.count(menor) == 1:
07.     notas.remove(menor)
08. else:
09.     indice = -1
10.    for i in range(len(notas)):
11.        if notas[i] == menor:
12.            indice = i
13.            break
14.    notas.pop(indice)
15.    maior = max(notas)
16.    if notas.count(maior) == 1:
17.        notas.remove(maior)
18.    else:
19.        indice = -1
20.        for i in range(len(notas)):
21.            if notas[i] == maior:
22.                indice = i
23.                break
24.        notas.pop(indice)
25. soma = sum(notas)
26. print(f"A potuação final do salto foi
{soma:.1f}")
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic9.py
Entre com uma das notas: 7.2
Entre com uma das notas: 7.6
Entre com uma das notas: 6.3
Entre com uma das notas: 9.2
Entre com uma das notas: 7.8
Entre com uma das notas: 9.2
Entre com uma das notas: 7.8
A potuação final do salto foi 39.6
```

```
Process finished with exit code 0
```

LISTAS

EXERCÍCIOS

Listas com Comportamento de Vetores

1. Criar um programa que solicita ao usuário 6 números, calculando sua média. Mostrar ao usuário uma lista com os números iguais ou acima da média e uma lista com os números abaixo da média.
2. Criar um programa que solicite ao usuário 2 listas com 5 elementos cada. Como resultado, criar uma terceira lista que intercala os elementos das duas listas.
3. Criar um programa que lê as temperaturas médias de cada mês do ano e as armazena em uma lista. Usar um outro vetor para guardar e exibir, quando necessário, os nomes dos meses do ano. Calcular a média anual de temperatura, e informar quais meses ficaram acima desta média anual.

LISTAS

EXERCÍCIOS

Listas com Comportamento de Matrizes

4. Criar um programa que efetua a soma de duas matrizes 3x3, sabendo que a soma das matrizes 3x3 é uma nova matriz 3x3 onde cada elemento é resultado da soma dos elementos das matrizes na mesma posição.
Exemplo:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 3 & 2 & 3 \\ 1 & 3 & 3 \\ 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 & 6 \\ 5 & 8 & 9 \\ 7 & 10 & 11 \end{bmatrix}$$

5. A matriz identidade é uma matriz de mesma quantidade de linhas e colunas que tem todos os elementos da diagonal principal (de cima para baixo, esquerda para direita) iguais a 1, e demais elementos iguais a 0. Criar um programa que solicita o tamanho da matriz desejada, que gera a matriz identidade. Exemplo:

$$\text{Matriz } I_3: \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Matriz } I_4: \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LISTAS

EXERCÍCIOS

Listas – Métodos e Funções Built in

6. Dado o vetor `nums = [3, 7, 2, 9, 5, 6]` criar um programa que, em uma linha, calcula a média dos seus elementos.

7. Dado o vetor `nums = [3, 11, 6, 32, 15, 22, 4, 10, 5]`, criar uma matriz 3x3 com os 3 maiores elementos na primeira linha, os 3 elementos intermediários na segunda linha, e os elementos menores na terceira linha.

LISTAS

EXERCÍCIOS

Desafio

9. Criar um programa que solicite o nome de 4 times de futebol. O programa deve simular partidas de forma que cada time jogue uma vez com os outros 3 times. Na partida deve perguntar quantos gols fez cada time, e somar as devidas pontuações. Ao final deve dizer qual ou quais times foram campeões, uma vez que pode haver empate em pontuação. Obs.: vitória vale 3 pontos para o vencedor, empate vale 1 ponto para cada time e derrota não soma pontos.