/VETORES E MATRIZES

[Arranjos dimensionais e multidimensionais]

01

/VETORES

/ARRANJOS UNIDIMENCIONAIS /VETORES

"Quando uma determinada Estrutura de dados é composta de variáveis com o mesmo tipo primitivo, temos um conjunto homogêneo de dados"

ÍNDICE	0	1	2	3	4
VALOR	10	20	54	81	2

/ARRANJOS UNIDIMENCIONAIS /VETORES

/IMPLEMENTAÇÃO DE VETORES EM PYTHON

```
# Implementamos uma lista em Python

lista_de_jogos = ['Super Mario',
                  'Sonic',
                  'Pokemon Go']
```

02

/MATRIZES

/VARIÁVEIS COMPOSTAS HOMOGÊNEAS MULTIDIMENSIONAIS

Matrizes:

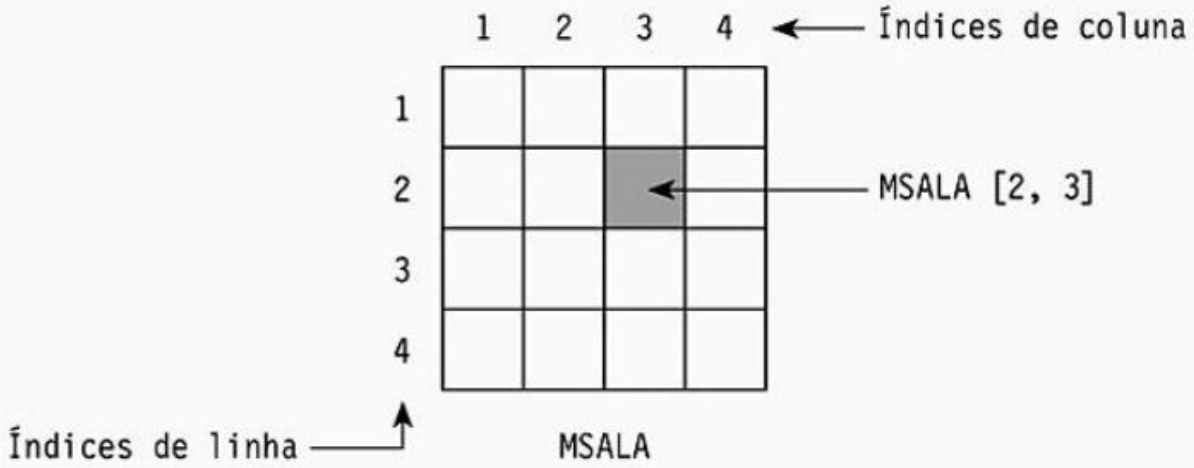
"Uma estrutura que precisasse de mais de um índice, [illegible]a denominada estrutura composta multidimensional"

/VETORES X /MATRIZES

- São estruturas complementares;
- Os vetores representam uma caixa de dados com apenas uma dimensão;
- A Matriz representa duas ou mais dimensões;

/EXEMPLO DE MATRIZ

FIGURA 4.4 Matriz MSALA



/IMPLEMENTAÇÃO DE MATRIZES EM PYTHON

```
# Implementamos uma lista de listas em Python

matriz_de_numeros = [[1, 2, 3],
                     [4, 5, 6],
                     [7, 8, 9]]
```

02

/LISTAS

Mais sobre listas em Python

/LISTA EM PYTHON

- Tem um conceito similar ao vetor, mas ela é uma estrutura de dados;

- Lista tem tamanho variável, podendo aumentar ou diminuir de tamanho através de funções;

/DECLARAÇÃO DE UMA LISTA EM PYTHON

```python
# Identificador nome e uma lista de nomes
nomes = ["Messias", "Emanuel", "Miguel", "João"]

# Imprimir a lista
print(nomes)

# Verificar o tipo de dado
type(nomes)

# Acessando um elemento da lista
nome[0]
```

01.1

EXEMPLOS PARA CONSULTA

Quando não lembrar, acesse esta parte!
Ir para tuplas

/ACESSO A UM ELEMENTO DE UMA LISTA EM PYTHON

```
# Lista de frutas

frutas = ["pêra", "uva", "maçã", "kiwi"]



# Alterando o elemento que está na posição 1

frutas[1] = "abacate"
```

/ADICIONANDO ELEMENTO A UMA LISTA EM PYTHON

```
'''O método insert() ajuda você a adicionar
um elemento em qualquer posição desejada.'''


frutas.insert(2, "morango")
```

/REMOVENDO ELEMENTO DE UMA LISTA EM PYTHON

```
'''A instrução del() pode remover um item da lista
passando como parâmetro sua posição. Lembre-se, para
isso você deve conhecer a posição do item na lista.
Você pode pesquisar o índice (posição) de um item da
lista com a função index()'''


del frutas[10]
```

/REMOVENDO ELEMENTO DE UMA LISTA EM PYTHON

```
# Vamos descobrir o índice da fruta

indice_fruta = frutas.index("melancia")



# Com o valor do índice, a gente deleta

del frutas[indice_fruta]
```

/REMOVENDO ELEMENTO DE UMA LISTA EM PYTHON

```
'''O método remove() é utilizando quando se deseja remover um item da lista pelo seu valor.'''


frutas.remove("banana")
```

/REMOVENDO ELEMENTO DE UMA LISTA EM PYTHON

```
'''O método pop() também pode ser utilizado para remover
qualquer elemento da lista. Desde que seja passado como
parâmetro o índice do item que deseja remover.'''


indice_fruta = frutas.index("abacaxi")


pop_fruta = frutas.pop(indice_fruta)
```

/02

/TUPLAS

/DEFININDO UMA TUPLA

```
# Definição de uma tupla utilize parênteses
dimensoes = (200, 50)

# Imprimindo os valores da tupla
print(dimensoes[0])
print(dimensoes[1])
```

PERCORRENDO OS VALORES COM UM LAÇO DE REPETIÇÃO

```python
# Definição de uma tupla
dimensoes = (200, 50)

# Utilizamos um For
for dimensao in dimensoes:
    print(dimensao)
```

TUPLAS TEM VALORES CONSTANTES (NÃO SE ALTERA)

```
# Definição de uma tupla
dimensoes = (200, 50)

dimensoes[0] = 250


Traceback (most recent call last):
  File "c:\Aula_13\teste.py", line 3, in <module>
    dimensoes[0] = 250
TypeError: 'tuple' object does not support item assignment
```

/SOBRESCREVENDO TUPLAS

```python
# Alterando toda a tupla
dimensoes = (400, 100)
print("\nDimensao Modificada")

for dimensao in dimensoes:
    print(dimensao)
```

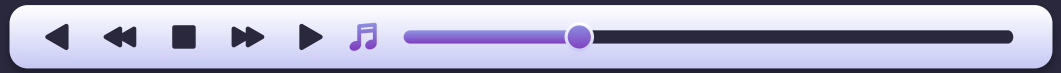
```python
# Definição de uma tupla
dimensoes = (200, 50)
print("Dimensões originais")

for dimensao in dimensoes:
    print(dimensao)
```

/03

/DICIONÁRIOS

You can enter a subtitle here if you need it



Um dicionário em Python é uma coleção de *pares chave-valor*. Cada *chave* é conectada a uma valor, e você pode usar uma chave para acessar o valor associado a ela.

MATTHES, Eric. Curso Intensivo de Python: Uma introdução prática e baseada em projetos à programação. Novatec Editora, 2017.

DICIONÁRIO SIMPLES

```python
# Definição de um dicionário
professor = {'nome': 'Messi', 'idade': 25}

# Imprimindo os valores
print(professor['nome'])
print(professor['idade'])
```

ACESSANDO VALORES

```
# Definição de um dicionário
professor = {'nome': 'Messias', 'idade': 23}

# Acessando os valores
professor['nome'] = ‘Messi’
professor['idade'] = '35'
```

ADICIONANDO NOVOS PARES CHAVE-VALOR

```
# Adicionando novos valores
professor['email'] = messi@gmail.com'
professor['cidade'] = 'João Pessoa'
professor['cpf'] = '000.000.000-00'
```

REMOVENDO PARES CHAVE-VALOR

```
# Removendo valores
del professor['cpf']
```

DICIONÁRIO DE OBJETOS SEMELHANTES

```
linguagens_preferidas = {
    'Messias': 'Python',
    'Arthur' : 'JavaScript',
    'Samuel' : 'Java',
    'Maria Eduarda' : 'Python',
    'Alirio' : 'SQL'
    }
```

/DÚVIDAS?

/VETORES E MATRIZES

[Arranjos dimensionais e multidimensionais]