

# Tuplas

**Guilherme Arthur de Carvalho**

Analista de sistemas

**@decarvalhogui**

# Objetivo Geral

Entender o funcionamento da estrutura de dados tupla.

# Pré-requisitos

- Python 3
- VSCode

# Percurso

## **Etapa 1**

**Criação e acesso aos dados**

## **Etapa 2**

**Métodos da classe tuple**

## Etapa 1

# Criação e acesso aos dados

# Criando tuplas

Tuplas são estruturas de dados muito parecidas com as listas, a principal diferença é que tuplas são imutáveis enquanto listas são mutáveis. Podemos criar tuplas através da classe **tuple**, ou colocando valores separados por vírgula de parenteses.

# Exemplo

```
frutas = ("laranja", "pera", "uva",)
```

```
letras = tuple("python")
```

```
numeros = tuple([1, 2, 3, 4])
```

```
pais = ("Brasil",)
```

# Acesso direto

A tupla é uma sequência, portanto podemos acessar seus dados utilizando índices. Contamos o índice de determinada sequência a partir do zero.



# Exemplo

```
frutas = ("maçã", "laranja", "uva", "pera",)  
frutas[0] # maçã  
frutas[2] # uva
```

# Índices negativos

Sequências suportam indexação negativa. A contagem começa em -1.

# Exemplo

```
frutas = ("maçã", "laranja", "uva", "pera",)  
frutas[-1] # pera  
frutas[-3] # laranja
```

# Tuplas aninhadas

Tuplas podem armazenar todos os tipos de objetos Python, portanto podemos ter tuplas que armazenam outras tuplas. Com isso podemos criar estruturas bidimensionais (tabelas), e acessar informando os índices de linha e coluna.

# Exemplo

```
matriz = (  
    (1, "a", 2),  
    ("b", 3, 4),  
    (6, 5, "c"),  
)  
  
matriz[0]    # (1, "a", 2)  
matriz[0][0] # 1  
matriz[0][-1] # 2  
matriz[-1][-1] # "c"
```

# Fatiamento

Além de acessar elementos diretamente, podemos extrair um conjunto de valores de uma sequência. Para isso basta passar o índice inicial e/ou final para acessar o conjunto. Podemos ainda informar quantas posições o cursor deve "pular" no acesso.

# Exemplo

```
tupla = ("p", "y", "t", "h", "o", "n",)  
  
tupla[2:] # ("t", "h", "o", "n")  
tupla[:2] # ("p", "y")  
tupla[1:3] # ("y", "t")  
tupla[0:3:2] # ("p", "t")  
tupla[:] # ("p", "y", "t", "h", "o", "n")  
tupla[::-1] # ("n", "o", "h", "t", "y", "p")
```

# Iterar tuplas

A forma mais comum para percorrer os dados de uma tupla é utilizando o comando **for**.



# Exemplo

```
carros = ("gol", "celta", "palio",)  
  
for carro in carros:  
    print(carro)
```

# Função enumerate

Às vezes é necessário saber qual o índice do objeto dentro do laço **for**. Para isso podemos usar a função **enumerate**.

# Exemplo

```
carros = ("gol", "celta", "palio",)

for indice, carro in enumerate(carros):
    print(f"{indice}: {carro}")
```

# Percurso

~~Etapa 1~~

~~Criação e acesso aos dados~~

**Etapa 2**

**Métodos da classe tuple**

## Etapa 2

# Métodos da classe tuple

# ()count

```
cores = ("vermelho", "azul", "verde", "azul",)
```

```
cores.count("vermelho") # 1
```

```
cores.count("azul") # 2
```

```
cores.count("verde") # 1
```

# ()index

```
linguagens = ("python", "js", "c", "java", "csharp",)
```

```
linguagens.index("java") # 3
```

```
linguagens.index("python") # 0
```

# len

```
linguagens = ("python", "js", "c", "java", "csharp",)  
len(linguagens) # 5
```



# Percurso

~~Etapa 1~~

~~Criação e acesso aos dados~~

~~Etapa 2~~

~~Métodos da classe tuple~~

# Links Úteis

- <https://github.com/digitalinnovationone/trilha-python-dio>

# Dúvidas?

- > Fórum/Artigos
- > Comunidade Online (Discord)

