LÓGICA DE PROGRAMAÇÃO

STRING

Variáveis String

Variáveis do tipo ***string*** armazenam cadeias de caracteres como nomes e textos em geral. Chamamos **cadeia de caracteres** uma sequência de símbolos como ***letras, números, sinais de pontuação etc***. Exemplo: ***João e Maria comem pão***

Representação de uma ***String*** em Python

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	**Índice**
J	o	ã	o		e		M	a	r	i	a		c	o	m	e	m		p	ã	o	**Conteúdo**

Para delimitar o texto de uma ***string***, é utilizado o caractere **(")** no início e no fim de uma frase. Exemplo: ***"João e Maria comem pão"***

LÓGICA DE PROGRAMAÇÃO

STRING

Função len()

Uma ***String*** em Python tem um tamanho associado, o tamanho de uma ***String*** pode ser obtido utilizando-se a função ***len()***

```
nome = "João da Silva"

print(len(nome))
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método count()

Conta quantas vezes a letra aparece no texto.

```
nome = "João da Silva"

print(nome.count("o"))
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método find(), rfind() e operador in

Indica em qual posição a palavra/letra foi encontrada a primeira vez e última. Caso não encontre, retorna -1 em ambos os casos; In verifica se palavra está na frase.

```
nome = "João da Silva"
print(nome.find("João"))
print(nome.rfind("a"))
print("Silva" in nome)
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método capitalize()

Retorna uma cópia da ***String*** com o ***primeiro caractere maiúsculo***

```
nome = "python"

print(nome.capitalize())
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método title()

Retorna a **primeira letra** de cada palavra em ***maiúsculo***

```
nome = "python é legal"

print(nome.title())
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método upper()

Retorna uma cópia da ***String*** com o ***todos os caracteres maiúsculo***

```
nome = "python"

print(nome.upper())
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método lower()

Retorna uma cópia da ***String*** com o ***todos os caracteres minúsculos***

```
nome = "PYTHON"

print(nome.lower())
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método strip()

Retorna uma cópia da ***String*** com o ***todos os espaços em branco, no início e no final removidos***

```
nome = " PYTHON "

print(nome.strip())
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método rstrip()

Retorna uma cópia da ***String*** com o ***todos os espaços a direita removidos***

```
nome = " PYTHON "

print(nome.rstrip())
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método lstrip()

Retorna uma cópia da ***String*** com o ***todos os espaços a esquerda removidos***

```
nome = " PYTHON "

print(nome.lstrip())
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método replace()

Troca a palavra por outra digitada/escolhida

```
nome = "PYTHON"

print(nome.replace("PYTHON", "JAVASCRIPT"))
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método startswith()

Retorna ***True*** se a ***String*** começar com o ***argumento passado por parâmetro***, caso contrário retorna ***False***

```
nome = "Python"

print(nome.startswith("py"))
```

LÓGICA DE PROGRAMAÇÃO

STRING

Método join() e split()

Join une os caracteres e separa por um ***delimitador***

Split divide a palavra ou frase e armazena dentro de uma ***lista***

```
nome = "Python é legal"
print(','.join(nome))
print(nome.split(" "))
```

LÓGICA DE PROGRAMAÇÃO

STRING

Fatiamento

O ***fatiamento*** em Python é muito poderoso, o fatiamento funciona com a utilização de dois pontos no ***índice*** da ***String***.

```
nome = "João da Silva"
print(nome[0:2])
```

Podemos também utilizar valores negativos para indicar posições a partir da direita. Assim **-1** é o último caractere, **-2** o penúltimo.

LÓGICA DE PROGRAMAÇÃO

STRING

Exemplos de Fatiamento

```
# alguns exemplos de fatiamento
s = "ABCDEFGHI"
print(s[:2])
print(s[1:])
print(s[0:-2])
print(s[:])
print(s[-1:])
print(s[-5:7])
print(s[-2:-1])
print(s[::-1])
```

VAMOS PRATICAR.

EXEMPLOS E ATIVIDADES DE STRINGS