STRINGS

As strings em python são colocadas entre aspas simples ou aspas duplas.

Você pode exibir uma dado do tipo string com a função `print()`:

Exemplo:

```
print("Olá")
print('Olá')

#Resultado:
Olá
Olá
```

Atribuir String a uma Variável

A atribuição de uma string a uma variável é feita com o nome da variável seguido por um sinal de igual e a string:

Exemplo:

```
a = "Olá"
print(a)

#Resultado:
Olá
```

Strings Multilinha

Você pode atribuir uma string multilinha a uma variável usando três aspas:

Exemplo

```
#Você pode usar três aspas duplas:
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
#Ou três aspas simples
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

> 💡 **Nota:** no resultado, as quebras de linha são inseridas na mesma posição que no código.

Strings são matrizes

Como muitas outras linguagens de programação populares, as strings em Python são matrizes de bytes que representam caracteres Unicode. No entanto, Python não tem um tipo de dados de caractere, um único caractere é simplesmente uma string com o comprimento 1. Os colchetes podem ser usados para acessar os elementos da string.

Exemplo:

```
a = "Olá, Mundo!"
print(a[1])

#Resultado:
l
```

Looping através de uma string

Como as strings são arrays, podemos percorrer os caracteres de uma string, com um loop `for`.

Exemplo:

```
for x in "banana":
 print(x)

#Resultado:
banana
```

Comprimento da string

Para obter o comprimento de uma string, use a função `len().`

Exemplo:

```
a = "Comprimento"
print(len(a))

#Resultado:
11
```

Verificar String

Para verificar se uma determinada frase ou caractere está presente em uma string, podemos usar a palavra-chave `in`.

Exemplo:

```
texto = "Verificando a string..."
print("string" in texto)

#Resultado:
True
```

Verifique se NÃO

Para verificar se uma determinada frase ou caractere NÃO está presente em uma string, podemos usar a palavra-chave `not in`.

Exemplo:

```
texto = "Verificando a string..."
print("estar" not in texto)

#Resultado:
True
```

Fatiando Strings

Você pode retornar um intervalo de caracteres usando a sintaxe de fatia(slice).

Especifique o índice inicial e o índice final, separados por dois pontos, para retornar uma parte da string.

Exemplo:

```
b = "Olá, Mundo!"
print(b[2:5])
#Ao omitir o índice inicial, o intervalo começará no primeiro caractere:
b = "Olá, Mundo!"
print(b[:5])
#Ao omitir o índice final , o intervalo irá para o final:
b = "Olá, Mundo!"
print(b[2:])
#É possível utilizar indexação negativa também, começando do fim da string.
b = "Olá, Mundo!"
print(b[-11:])
```

💡 **Nota:** O primeiro caractere tem índice 0.

Modificar String

Python tem um conjunto de métodos embutidos que você pode usar em strings. Confira nas referências todos os métodos de string.

Exemplo:

```
a = "Modificando uma string de maneira simples"
print(a.upper()) #O método upper() retorna a string em maiúsculas
print(a.lower()) #O método lower() retorna a string em letras minúsculas

#Resultado:
MODIFICANDO UMA STRING DE MANEIRA SIMPLES
modificando uma string de maneira simples
```

Remover espaço em branco

O espaço em branco é o espaço antes e/ou depois do texto real e, muitas vezes, você deseja remover esse espaço.

Exemplo:

```
a = "     Retirando o espaço em branco     "
print(a.strip()) #O método strip() remove qualquer espaço em branco do início e/ou do final de uma string.

#Resultado:
Retirando o espaço em branco
#Observe que o método funciona apenas nos espaços anteriores ao primeiro caractér e após o último.
```

Dividir String

O método `split()` retorna uma lista onde o texto entre o separador especificado se torna os itens da lista.

Exemplo:

```
a = "Divida a string aqui, exato!"
print(a.split(",")) #O separador é a vírgula

#Resultado:
['Divida a string aqui', ' exato!']
```

Concatenação de String

Para concatenar ou combinar duas strings, você pode usar o operador +.

Exemplo:

```
a = "Concatenando"
b = "duas strings"
c = a + " " + b
print(c)

#Resultado:
Concatenando duas strings
```

Caractere de fuga

Para inserir caracteres ilegais em uma string, use um caractere de escape.

Um caractere de escape é uma barra invertida `\` seguida pelo caractere que você deseja inserir.

Um exemplo de caractere ilegal é uma aspa dupla dentro de uma string que está entre aspas duplas:

Exemplo:

```
texto = "Usando o "caractere de fuga" você evita erros."
#Você obterá um erro se usar aspas duplas dentro de uma string entre aspas duplas.
#Para corrigir esse problema, use o caractere de escape \"
texto = "Usando o \"caractere de fuga\" você evita erros."
```

Outros caracteres de escape usados em Python:

Código	Resultado
\'	Aspas Simples
\\	Barra invertida
\n	Nova linha
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Valor Octal
\xhh	Valor Hexadecimal

Formatação de String

Para ter certeza de que uma string será exibida conforme o esperado, podemos formatar o resultado com o método `format()` que permite que você formate partes selecionadas de uma string. Às vezes, há partes de um texto que você não controla, talvez venham de um banco de dados ou da entrada do usuário da qual você não tem acesso. Para controlar esses valores, adicione espaços reservados (chaves `{}`) no texto e execute os valores por meio do método `format()`:

Exemplo:

```
#A sintaxe padrão
preço = float(input("Digite o valor: R$"))
msg ="O preço é {} reais"
print(msg.format(preço))

#A sintaxe aninhada
```

```
preço = float(input("Digite o valor: R$"))
print("O preço é {} reais".format(preço))

#Você pode adicionar parâmetros dentro das chaves para especificar datalhes do valor:
preço = float(input("Digite o valor: R$"))
print("O preço é {:.2f} reais".format(preço)) #flutuante com duas casas decimais
```

Valores Múltiplos

Se você quiser usar mais valores, basta adicionar mais valores.

Exemplo:

```
preço = float(input("Digite o valor: R$"))
modelo= str(input("Digite o modelo: "))
print("O preço é {} reais e o modelo é {}".format(preço))
```

Números de Índice

Você pode usar números de índice para garantir que os valores sejam colocados nos marcadores de posição corretos.

Exemplo:

```
unidade = 3
item = 567
preço = 49
meupedido = "E quero {0} unidades do item número {1} que custa {2:.2f} reais.".format(unidade, item, pr
eço))
```

É possível utilizar os valores através dos índices.

Exemplo:

```
idade = 22
nome = "João"
print("Meu nome é {1}. {1} tenho {0} anos.".format(idade, nome))
```

Índices Nomeados

Você também pode usar índices nomeados inserindo um nome entre chaves `{nome}`, mas deve usar nomes ao passar os valores dos parâmetros `.format(nome = ")`.

Exemplo:

```
print("Eu tenho um {nome} modelo {modelo}.".format(nome = "Monza", modelo = "1998"))
```