



FACULDADE DE TECNOLOGIA SENAC PELOTAS

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Algoritmos e Estruturas de Dados I

FACULDADE DE TECNOLOGIA SENAC PELOTAS
CURSOS SUPERIORES: ESCOLA DE TECNOLOGIA
PROF. EDÉCIO FERNANDO IEPSSEN

Manipulação de Strings

- ▶ Diversas são as operações, em programação, que necessitam manipular cadeias de caracteres (strings).
- ▶ São úteis, por exemplo, para validar uma senha, gerar uma sugestão de e-mail, extrair palavras de um texto, dividir uma linha em partes.
- ▶ Todas as linguagens de programação dispõe de métodos específicos para trabalhar com strings, o que muda de uma para a outra, é a sintaxe.
- ▶ Uma cadeia de caracteres é uma sequência de letras, algarismos ou símbolos (símbolos de pontuação, parênteses, etc.).

Funções Python para Manipulação de Strings

```
cidade="Pelotas"      # exemplo de variável string com o conteúdo "Pelotas"
len(cidade)           # retorna o tamanho da string: 7
cidade.upper()        # converte para maiúsculas: "PELOTAS"
cidade.lower()        # converte para minúsculas: "pelotas"
cidade.find("t")       # encontra a posição de "t" em "Pelotas": 4. Obs.: se não existir retorna -1
cidade.rfind("e")      # pesquisa do final para o início da string
cidade.count("t")      # conta o número de ocorrências do caracter (ou caracteres) na string: 1
split()               # cria um vetor a partir de uma string, utilizando um caracter de separação.
cidade="Santa Vitória do Palmar"
partes=cidade.split(" ") # partes[0] = "Santa"; partes[1] = "Vitória"; partes[2] = "do"; ...
```

Obs.: As strings em Python são “imutáveis”, ou seja, não é possível alterar parte do seu conteúdo. Desta forma, operações como replace ou lower não podem ser realizadas sobre a string, apenas exibidas. A solução é atribuir o novo conteúdo a uma outra variável.

Operações sobre Strings

```
a = "Redes"
b = a * 4
print(b)                                # RedesRedesRedesRedes
print("-" * 10)                          # -----
print(a.center(20,"*"))                  # *****Redes*****
print(a.ljust(20,"*"))                   # Redes*****
print(a.rjust(20,"*"))                   # *****Redes
```

Alinhamentos de Strings no print(f" ")

O significado das várias opções de alinhamento é o seguinte:

Opção	Significado
'<'	Força o alinhamento à esquerda do campo dentro do espaço disponível (este é o padrão para a maioria dos objetos).
'>'	Força o alinhamento à direita do campo dentro do espaço disponível (este é o padrão para números).
'='	Força o preenchimento a ser colocado após o sinal (se houver), mas antes dos dígitos. É usado para imprimir campos na forma "+000000120". Esta opção de alinhamento só é válida para tipos numéricos. Torna-se o padrão para números quando "0" precede imediatamente a largura do campo.
'^'	Força a centralização do campo no espaço disponível.

Alinhamentos de Strings no print(f" ")

```
C:\Users\edeci>python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35)
Type "help", "copyright", "credits" or "license" for more ir
>>> a = "oi"
>>> print(f"{a:10}x")
oi          x
>>> print(f"{a:>10}x")
          oix
>>> print(f"{a:^10}x")
      oi      x
>>> print(f"{a:*^10}x")
****oi****x
```

Indexação e fracionamento

Em Python (e na maioria das linguagens de programação), uma string pode ser manipulada como um vetor de caracteres. Assim, caso a variável **cidade = "Pelotas"** se quisermos obter o primeiro caracter desta string podemos referenciar:

```
print(cidade[0])    # irá apresentar "P"
```

Fracionamento:

```
print(cidade[0:3])  # irá apresentar "Pel".
```

```
print(cidade[2:4])  # irá apresentar "lo"
```

```
print(cidade[2:])    # da posição 2 até o final. Irá apresentar "lotas"
```

```
print(cidade[:2])    # os 2 primeiros caracteres. Irá apresentar "Pe"
```

```
print(cidade[-1])    # do final para o início. Irá apresentar o último caracter
```

Verificação dos caracteres de uma String

Exemplos:

```
letra = "a"
```

```
letra.isalpha()      # verifica se é alfabética
```

```
letra.isdigit()      # verifica se é dígito numérico
```

```
letra.islower()      # verifica se é minúscula
```

```
letra.isupper()      # verifica se é maiúscula
```

```
letra.isspace()      # verifica se é espaço
```

<https://realpython.com/python-strings/>

Arquivos Texto

- ▶ A programação em arquivos de dados é uma habilidade essencial no desenvolvimento bem-sucedido de aplicações.
- ▶ Operações relacionadas a gravação e recuperação de dados armazenados em arquivos estão entre as mais importantes de qualquer linguagem de programação.
- ▶ Arquivos texto são utilizados para armazenar diversos tipos de informações, desde logs de ações realizadas em sistemas, até dados complexos formatados para a transferência de dados entre bancos.
- ▶ Um arquivo texto está projetado para ser lido do início até o fim toda a vez que for aberto.

Operações sobre Arquivos em Python

```
arq = open("acessos.txt", "r") # abre o arquivo e associa ele a variável arq.
                                # O 2º parâmetro indica o modo de abertura:
                                # "w": criação; "r": leitura; "a": adição de dados

arq.write("usuário 2")          # escreve o texto no arquivo

tudo = arq.read(n)              # lê 'n' caracteres do arquivos.
                                # Sem 'n' todo o conteúdo de arq é lido

linha = arq.readline( )         # lê uma linha do arquivo e posiciona na linha seguinte

linhas = arq.readlines( )       # lê todo o conteúdo do arquivo e joga em um vetor (linhas)

arq.close( )                   # fecha o arquivo

os.path.isfile("nomearq.txt")   # verifica se o arquivo existe (necessita de import os)
```

A segunda maneira de fechar um arquivo é usar a `with` instrução :

Python

```
with open('dog_breeds.txt') as reader:  
    # Further file processing goes here
```

A `with` instrução se encarrega automaticamente de fechar o arquivo assim que ele sai do `with` bloco, mesmo em casos de erro. Eu recomendo fortemente que você use a `with` instrução o máximo possível, pois ela permite um código mais limpo e facilita o tratamento de erros inesperados.

Provavelmente, você também desejará usar o segundo argumento posicional, `mode`. Este argumento é uma `string` que contém vários caracteres para representar como você deseja abrir o arquivo. O padrão e mais comum é `'r'`, que representa a abertura do arquivo no modo somente leitura como um arquivo de texto:

Python

```
with open('dog_breeds.txt', 'r') as reader:  
    # Further file processing goes here
```

Arquivos CSV (Comma-Separated-Values)

Entenda o que é o formato CSV e saiba como importar e exportar esses arquivos

O formato CSV é um tipo de arquivo de texto fundamental para transferência de informações entre aplicativos diferentes, como uma plataforma CRM e o Microsoft Excel. Para fazer esse tipo de troca de informações com sucesso, você precisa saber como importar e exportar esses arquivos. Tem dúvidas de como fazer isso? Então continue lendo este post!

Se você trabalha com qualquer gestão de dados, seja ela a mais complexa ou a mais básica (como usar [planilhas](#) no dia a dia), você provavelmente já ouviu falar no **formato CSV**.

Em linhas gerais, o CSV é um formato usado para armazenar dados e que pode ser importado e exportado em programas como **Microsoft Excel, Google Sheets, Apple Numbers, OpenOffice Calc e outros aplicativos**.

<https://rockcontent.com/br/blog/csv/>

Arquivos CSV (Comma-Separated-Values)

Estrutura do CSV

Um arquivo CSV nada mais é do que uma planilha em texto puro. Cada linha do arquivo é uma linha da planilha, e as colunas de uma linha são separadas por um **delimitador** que normalmente é a vírgula ou o ponto-e-vírgula.

Segue um exemplo de conteúdo de um pequeno arquivo CSV (cidades.csv):



```
nome;populacao;pais  
Xangai;17836133;China  
Lagos;16060307;Nigéria  
Karachi;13969284;Paquistão  
Istambul;13907015;Turquia  
Mumbai;12478447;Índia
```



[Central de ajuda da Shopify](#) > [Tópicos](#) > [Produtos](#) > [Importação e exportação](#) > Usar arquivos CSV

Introdução à Shopify

Migrar para a Shopify

Admin da Shopify

Sua conta

Loja Virtual

PDV

B2B [SHOPIFY PLUS](#)

Usar arquivos CSV para importar e exportar produtos

Use um arquivo CSV (valores separados por vírgulas) para importar e exportar produtos da loja da Shopify. Assim, é possível fazer esses procedimentos de uma só vez com uma grande quantidade de produtos e as respectivas informações. Isso é útil para trocar informações entre a Shopify e outras plataformas.