

Design de Software

Aula: Arquivos em Python


Objetivos de Aprendizado

- Métodos e funções para strings
- Trabalhar com arquivos

Alguns métodos para Strings


str.strip() - remove espaços em branco do início e fim de uma string. Por exemplo:

```
print(" palavra ".strip())
```

 "palavra"

str.find() - Retorna a posição de um texto dentro de uma string, ou -1 se não achou. Por exemplo:

```
p = 'Insper'  
print(p.find('sp'))
```

 2

str.replace() - substitui um texto dentro de uma string. Exemplo:

```
p = 'Insper'  
print(p.replace('Ins', 'Su'))
```

 "Super"

Alguns métodos para Strings

str.split() - separa uma string pelo seu delimitador em uma lista de strings. Exemplo:

```
print(' 1 2 3 '.split())  
print('1,2,3'.split(","))
```

→ ['1', '2', '3']

str.join() - junta novamente os elementos de uma lista em uma string usando o delimitador. Exemplo:

```
print(' '.join(['1', '2', '3']))  
print(', '.join(['1', '2', '3']))
```

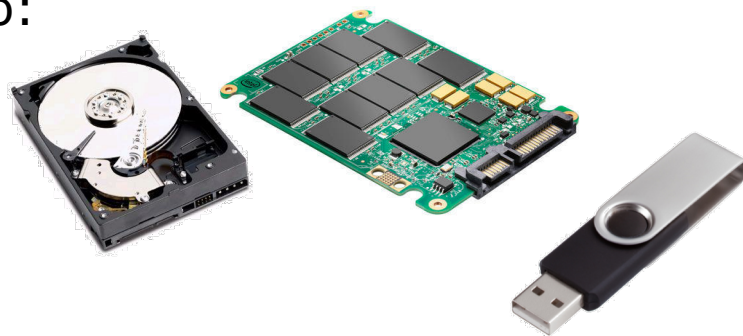
→ '1 2 3'

→ '1,2,3'

Arquivos

Arquivos são estruturas de dados que normalmente são armazenados em dispositivos secundários de memória. Os principais dispositivos de armazenamento atualmente são:

- Discos Rígidos (HDs)
- Solid-State Drive (SSD)
- Pen drives



Os arquivos armazenados nestes dispositivos possuem sempre um identificação (nome) e sua localização (normalmente em uma estrutura hierárquica de diretórios). Outros atributos como data e permissões de acesso também são normalmente usados no índice.

Acessando Arquivos

Arquivos são lidos como uma sequência de bytes

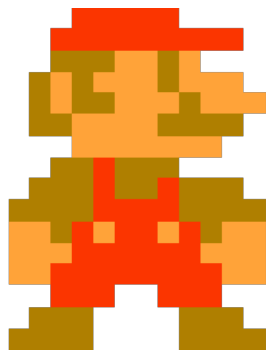
- Estes bytes são organizados de forma que um programa específico entenda;
- Estes bytes podem ser somente caracteres (ASCII ou Unicode).
Programas de edição de texto puro trabalham com estes arquivos.

```
<html>
  <header>
    <title>Título da Página</title>
  </header>
  <body> Hello world </body>
</html>
```

Arquivos Binários e Textos

Os arquivos podem ser divididos assim em dois sub-grupos:

- Texto
- Binários



Retro-Mario-icon.png

```
$ hexdump Retro-Mario-icon.png
00000000 89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52
00000010 00 00 00 10 00 00 00 10 08 06 00 00 00 1f f3 ff
00000020 61 00 00 01 d6 49 44 41 54 78 da 7d 93 cf 2b 44
00000030 51 14 c7 bf 77 bc 37 58 e8 46 16 46 cc 9b 28 0a
00000040 3b 0b 85 cc 34 8c 29 4a 59 b1 f0 63 61 a7 2c 65
00000050 25 21 f2 0f d8 88 f2 63 67 63 65 31 d2 30 93 50
00000060 4a 29 45 36 4c f6 e3 3d 3f 32 cd 8c ae 7b df bc
00000070 cb bc 99 c7 d9 bc fb ee 3d f7 73 cf f9 9e 73 08
00000080 0a 2c d6 0a 6f 47 05 12 05 db 46 fc 15 da cd 87
00000090 b9 66 b3 09 bc ca 03 92 ef 35 e7 03 5d ac 81 8e
000000a0 ff 4d 2f bb 44 a5 23 60 6d 8c d0 e6 46 66 02 54
000000b0 97 82 c1 86 5e 64 be 32 38 4a c4 31 e0 0b ea ae
000000c0 c9 a3 ca 42 da 0f 60 c4 0f 3a 1e 54 f4 7e ad 07
000000d0 11 7e 21 cc bf 6a 89 6a 02 a2 cf e7 08 d6 77 9a
000000e0 ff 22 9d d8 5d 52 0b ac 5e 19 45 80 31 0e 10 8e
000000f0 27 fc 82 c2 23 08 fb fc 60 8c e1 33 9b 32 21 02
00000100 2a 34 e0 67 86 6b 22 17 8d 0d 30 1a 80 2e 43 97
00000110 26 22 38 7c 8c da 34 18 5e fc 43 03 61 a7 2b d5
00000120 d4 ef 6d 77 14 92 47 a3 cb 97 9d 45 9c 82 b7 b9
00000130 0e 09 25 03 04 e2 c0 b1 08 84 71 41 b3 30 3c d4
00000140 a3 a9 aa 1b b1 fb 24 9b d9 78 2b 2e a3 bc 2c d6
00000150 12 70 ca 53 56 b2 40 f7 b9 05 b3 7a e2 f6 09 da
00000160 fc 0e ec 22 1e 2c 88 b7 72 26 00 a1 bc b4 33 8a
00000170 0d 60 1a d7 81 d8 00 6c 2f cc 84 60 91 87 28 42
00000180 67 0a dc 13 3d 48 6f 72 8a 3b b7 fe d8 8e 22 16
00000190 fa 15 98 8c 47 8a 01 a2 64 78 4f 21 bd 1b 2f 56
000001a0 d0 02 91 52 f5 4f c0 0b 4b a5 49 7a eb 84 ca 2d
000001b0 91 af e5 43 2d 08 73 4f 05 0d e2 22 02 60 ef 03
000001c0 61 d7 d3 55 b4 e5 3a 29 4b 68 f6 fc 7e 13 e8 50
000001d0 55 6e 3e 18 81 5e 7e 01 e7 32 2e 4f 82 b6 f9 ac
000001e0 41 62 bc 59 96 7e 1d d7 fb 40 6b bb 7e 86 cc b9
000001f0 91 f2 ab 50 e8 24 bb d4 a9 0a df 33 de ba 11 4f
00000200 ea 60 6c 00 00 00 00 49 45 4e 44 ae 42 60 82
```

Arquivos em Python

Arquivos podem ser acessados em Python com:

abrir para leitura: **open()**

terminar leitura: **close()**

```
# Modo tradicional de ler um arquivo
arquivo = open('arquivo_texto.txt', 'r')
conteudo = arquivo.read()
arquivo.close() # O que acontece se não fechar?

# Imprime o conteúdo
print(conteudo)
```

Idealmente abrir com o **with** para simplificar fechamento do arquivo.

```
# Abre um arquivo para a leitura.
with open('arquivo_texto.txt', 'r') as arquivo:
    conteudo = arquivo.read()
# Quando sai do bloco do 'with', fecha o arquivo
# automaticamente.

# Imprime o conteúdo.
print(conteudo)
```


Modos de abrir arquivos

- r** - somente leitura em modo texto
- rb** - somente leitura em modo binário

```
# Abre um arquivo para a leitura.  
with open('arquivo_texto.txt', 'r') as arquivo:  
    conteudo = arquivo.read()  
# Quando sai do bloco do 'with', fecha o arquivo  
# automaticamente.  
  
# Imprime o conteúdo.  
print(conteudo)
```

Arquivos

Para escrever no arquivo:

```
# Cria o arquivo para escrita (limpa o antigo se já existir)
with open('arquivo_texto.txt', 'w') as arquivo:
    # Escrevendo um texto
    arquivo.write("algum dado\n")

# Abre/Cria o arquivo para escrita SEM apagar o que tinha antes.
with open('arquivo_texto.txt', 'a') as arquivo:
    # Escrevendo um texto
    arquivo.write("novo dado\n")
```

Arquivos – Outras funções:

Vários comandos permitem ler e gravar conteúdos no arquivo:

- `read()`
- `write()`
- `readline()`
- `readlines()`
- `writelines()`

Maiores detalhes:

<https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files>

JSON

Forma de trocar informação de modo estruturado, simples e rápido entre sistemas. É muito parecido com um dicionário do Python, contudo o JSON é um texto (string) e não uma estrutura de dados.

Exemplo:

```
{ "Alunos": [
    { "nome": "João", "notas": [ 8, 9, 5 ] },
    { "nome": "Maria", "notas": [ 8, 10, 7 ] },
    { "nome": "José", "notas": [ 10, 10, 9 ] }
]}
```

JSON – Manipulando JSON em Python

Exemplo:

```
import json

texto = '''
    {"Alunos":[
        { "nome": "João", "notas": [ 8, 9, 5 ] },
        { "nome": "Maria", "notas": [ 8, 10, 7 ] },
        { "nome": "José", "notas": [ 10, 10, 9 ] }
    ]}
'''

# Criar um dicionário do texto JSON.
dicionario = json.loads(texto)

# Transformando de volta para json.
original = json.dumps(dicionario)
```

Exercícios

Fazer uma função que abre um arquivo de texto e conta a quantidade de palavras (não contar os espaços em branco).

Dica: usar `.split()`

Solução

```
def conta_palavras(nome):  
    with open(nome, 'r') as arquivo:  
        conteudo = arquivo.read()  
        palavras = conteudo.split()  
        return len(palavras)  
  
print(conta_palavras("lorem.txt"))
```

Exercícios

Fazer um programa que lê um arquivo com o estoque de uma empresa no formato JSON (conforme exemplo abaixo) e informe qual o valor de produtos desse estoque.

```
{"produtos": [
  { "produto": "açúcar", "quantidade": 4, "valor": 3.22},
  { "produto": "arroz", "quantidade": 1, "valor": 10.34},
  { "produto": "feijão", "quantidade": 2, "valor": 9.55}
]}
```


Solução

```
{"produtos": [
    { "produto": "açúcar", "quantidade": 4, "valor": 3.22},
    { "produto": "arroz", "quantidade": 1, "valor": 10.34},
    { "produto": "feijão", "quantidade": 2, "valor": 9.55}
]}
```

```
import json

with open('estoque.json', 'r') as arquivo:
    conteudo = arquivo.read()
    estoque = json.loads(conteudo)

total = 0

for p in estoque["produtos"]:
    total += p["quantidade"] * p["valor"]

print(total)
```

Insper

www.insper.edu.br