

Manipulação de Strings e Arquivos

Paulo Almeida
André Grégio



JUSTIÇA 4.0: INOVAÇÃO E EFETIVIDADE NA REALIZAÇÃO DA JUSTIÇA PARA TODOS
PROJETO DE EXECUÇÃO NACIONAL BRA/20/015

1. Strings



Strings

Das aulas passadas

Uma string é uma cadeia de caracteres

Em Python, precisamos colocar a string entre aspas duplas " ou simples '

```
nome = "Maria Silva"
```

Algumas funções Úteis

Assim como para listas convencionais, a função `len()` vai retornar o tamanho da string (o número de caracteres)

```
frase = "Curso de Python"  
print("Num caracs", len(frase))
```

Acessando itens

De maneira similar a vetores, podemos acessar um caractere ou um conjunto de caracteres por índice

Internamente uma string é um vetor, onde o primeiro caractere está na posição zero

Acessando itens

De maneira similar a vetores, podemos acessar um caractere ou um conjunto de caracteres por índice

Internamente uma string é um vetor, onde o primeiro caractere está na posição zero

```
frase = "Curso de Python"
linguagem = frase[9:len(frase)]
print(linguagem)
print("Primeira Letra:", frase[0])
```

Replace

É possível substituir um trecho de uma frase por outro via `replace("texto antigo", "novo")`

A função retorna uma nova string alterada

A string original não é alterada

```
frase = "Esse é o curso de linguagem C++. Você pode criar programas em C++."  
mudanca = frase.replace("C++", "Python")
```

```
print(frase)  
print(mudanca)
```

Replace

É possível especificar o número limite de substituições no replace

```
frase = "Esse é o curso de linguagem C++. Você pode criar programas em C++."
```

```
somente_prim = frase.replace("C++", "Python", 1)  
print(somente_prim)
```


Concatenando

É possível concatenar strings utilizando o operador +

```
nome_curso = "Python"
ola = "Bom dia."
frase = ola + " Esse é o curso de " + nome_curso + "."
print(frase)
```

Caracteres de controle

Caracteres de controle “fazem algo”, sem necessariamente imprimir algo na tela

Em Python, um caractere de controle começa com uma barra invertida \

Exemplos:

"\n" insere uma quebra de linha

"\t" insere uma tabulação

Caracteres de controle

Exemplo

```
frase = "Esse é um texto\nde\texemplo"  
print(frase)
```

Caixa

É possível transformar a caixa dos caracteres de uma string para:

- Minúsculo usando: `lower()`

- Maiúsculo usando: `upper()`

- Primeira maiúscula e demais minúsculas: `capitalize()`

Caixa

Exemplos

```
frase1 = "CURSO de Python"  
print(frase1.upper())  
print(frase1.lower())  
print(frase1.capitalize())
```

Split

Utilize `split(string_de_quebra)` para “quebrar” uma string em uma lista de strings

```
frase = "Esse é o curso de Python - Foco em iniciantes"
lista1 = frase.split(" ")
for item in lista1:
    print(item)

lista2 = frase.split("-")
for item in lista2:
    print(item)
```

Índice

Para encontrar o índice de uma palavra ou caractere, use a função `find()`

```
frase = "Olá mundo teste isso vai ficar teste final"
inicio = frase.find("teste") + len("teste")
fim = frase.find("teste", inicio)
print(frase[inicio:fim])
```

Lista de Funções

<code>count()</code>	Quantas vezes a palavra aparece
<code>endswith()</code>	Verifica se a string termina com determinada palavra
<code>startswith()</code>	Verifica se a string começa com determinada palavra
<code>islower()</code>	Verifica se todos caracteres estão em caixa baixa
<code>isnumeric()</code>	Verifica se todos caracteres são numéricos
<code>isspace()</code>	Verifica se todos os caracteres são espaços em branco
<code>strip()</code>	Remove caracteres em branco do começo da string
<code>isascii()</code>	Verifica se todos os caracteres são ASCII

2. Formatando Strings



Formatando Strings

Existem várias formas para formatar uma string

Uma delas é usar a função *format()*

Usando Format

A forma mais simples de usar o format, é colocando {} na string.

As chaves servem como *placeholders*, e serão substituídas pelo `format()`

A função `format()` **retorna uma nova string** como conteúdo substituído

Exemplo

```
anterior = 5000
atual = 5500
diferenca = atual - anterior
pct = diferenca/anterior * 100
```

```
frase = "A diferença de salário é de R${}, ou seja, {}%"
```

```
formatado = frase.format(diferenca, pct)
print(formatado)
```

Exemplo

```
anterior = 5000
atual = 5500
diferenca = atual - anterior
pct = diferenca/anterior * 100
```

```
frase = "A diferença de salário é de R${}, ou seja, {}%"
```

```
formatado = frase.format(diferenca, pct)
print(formatado)
```

Formatando diretamente

É possível formatar diretamente no print.

Veja um exemplo:

```
salario = 7500  
print("0 seu salário é {}".format(salario))
```

Format e tipos

O format infere automaticamente o tipo da variável

Mas é possível forçar um tipo específico

Exemplo: use `{:f}` para imprimir como um float

```
salario = 7500  
print("O seu salário é {:f}.".format(salario))
```

Format e tipos

É possível ainda limitar o número de casas decimais

Exemplo: `{:.2f}` para duas casas

```
salario = 7500  
print("O seu salário é {:.2f}.".format(salario))
```


Format e tipos

É possível ainda limitar o número de casas decimais
Exemplo: `{:.2f}` para duas casas

```
salario = 7500  
print("O seu salário é {:.2f}.".format(salario))
```

Existem diversos outros especificadores de formato.

Veja uma lista aqui: www.w3schools.com/python/ref_string_format.asp

3.

Arquivos e Leitura



Tipos de arquivo

Existem basicamente dois tipos de arquivo

Arquivo texto: texto puro e pode ser lido por um humano

Tipos de arquivo

Existem basicamente dois tipos de arquivo

Arquivo texto: texto puro e pode ser lido por um humano

Arquivo binário: dados binários que usam a mesma forma da representação interna da máquina. Não podem ser lidos por um humano (normal)

Tipos de arquivo

Existem basicamente dois tipos de arquivo

Arquivo texto: texto puro e pode ser lido por um humano

Na disciplina vamos focar em arquivos texto.

Arquivo binário: dados binários que usam a mesma forma da representação interna da máquina. Não podem ser lidos por um humano (normal)

Abrindo e fechando

Para abrir um arquivo texto em modo **leitura**, use a função

```
f = open(nomeArquivo, "r")
```

`f` é uma variável que vai representar o arquivo aberto

"r" de *read* indica que o arquivo vai ser de leitura

Abrindo e fechando

Para abrir um arquivo texto em modo **leitura**, use a função

```
arq = open(nomeArquivo, "r")
```

arq é uma variável que vai representar o arquivo aberto
"r" de *read* indica que o arquivo vai ser de leitura

É obrigatório que o arquivo seja fechado depois de usado

```
arq.close()
```

Exemplo

```
arq = open("documento.txt", "r")  
#Faz algo com o arquivo  
arq.close()
```


Read

A função `read()` lê todo o conteúdo do arquivo

```
arq = open("documento.txt", "r")
conteudo = arq.read()
print(conteudo)
arq.close()
```

Readline

Para ler uma única linha do arquivo, utilize `readline()`

A função retorna a próxima linha do arquivo, ou uma string vazia se o arquivo terminou

```
arq = open("documento.txt", "r")
conteudo = arq.readline()
while(conteudo != ""):
    print("Linha lida:", conteudo)
    conteudo = arq.readline()
arq.close()
```

Iterando

É possível iterar linha a linha em um arquivo

```
arq = open("documento.txt", "r")
for linha in arq:
    print("Linha lida:", linha)
arq.close()
```

Exemplo

Assumindo que documento.txt possui em cada linha um nome seguido de uma idade.
Exemplo:

Maria 30
João 40
José 25

```
arq = open("documento.txt", "r")
for linha in arq:
    quebra = linha.split(" ")
    nome = quebra[0]
    idade = int(quebra[1])

    print(nome, "tem", idade, "anos de idade")
arq.close()
```

4.

Arquivos e Escrita



Modo escrita

Para abrir um arquivo texto em modo **escrita**, existem dois modos básicos

Modo escrita (*write*). Se o arquivo já existir, **será substituído**

```
f = open(nomeArquivo, "w")
```

Modo escrita

Para abrir um arquivo texto em modo **escrita**, existem dois modos básicos

Modo escrita (*write*). Se o arquivo já existir, **será substituído**

```
f = open(nomeArquivo, "w")
```

Modo anexagem (*append*). O conteúdo é anexado no final do arquivo, caso o arquivo já exista

```
f = open(nomeArquivo, "a")
```

Abrindo e fechando

Da mesma forma que para arquivos em modo leitura, é obrigatório que o arquivo seja fechado depois de usado `arq.close()`

Write

Para escrever no arquivo, use a função `write()`

Dicas:

`write()` **não** adiciona uma quebra de linha automaticamente no texto

`write()` só aceita strings. Converta para string usando, por exemplo, `str()` o conteúdo a ser escrito quando necessário

Exemplo

```
valor = 10  
pi = 3.14  
nome = "Maria da Silva"
```

```
arq = open("documento.txt", "w")  
arq.write(str(valor))  
arq.write(';')  
arq.write(str(pi))  
arq.write(';')  
arq.write(nome)  
arq.write('\n')  
arq.close()
```

Exemplo

```
valor = 10  
pi = 3.14  
nome = "Maria da Silva"
```

```
arq = open("documento.txt", "w")  
arq.write(str(valor))  
arq.write(';')  
arq.write(str(pi))  
arq.write(';')  
arq.write(nome)  
arq.write('\n')  
arq.close()
```

Inserindo um ; após cada item para tornar o arquivo legível posteriormente.

Atenção

É responsabilidade do programador armazenar os dados no arquivo de forma que seja possível a leitura posterior

O formato do arquivo pode ser “proprietário” do seu programa

Atenção

É responsabilidade do programador armazenar os dados no arquivo de forma que seja possível a leitura posterior

O formato do arquivo pode ser “proprietário” do seu programa

Exemplo: se você salvar no arquivo um inteiro, seguido de um float, seguido de uma string, precisa lembrar dessa ordem ao ler o arquivo

5.

Arquivos Delimitados



Formatos padrão

Você pode criar seu próprio padrão de arquivos

Formatos padrão

Você pode criar seu próprio padrão de arquivos

Mas em sistemas simples é comum a utilização de padrões já conhecidos e aceitos, como csv, xml, Json, ...

CSV

CSV - comma-separated values

Tipo de arquivo de **texto** delimitado

Os valores são separados por vírgula

CSV

CSV - comma-separated values

Tipo de arquivo de **texto** delimitado

Os valores são separados por vírgula

Muitas vezes usamos outros separadores, como ;

Exemplo de CSV

```
Nome;Idade;Salário  
Maria Silva;45;8500,85  
José Oliveira;30;5500,4  
Marcos Santos;35;8000
```

Exemplo de CSV

Um cabeçalho para ajudar a identificar as colunas

Nome;Idade;Salário

Maria Silva;45;8500,85

José Oliveira;30;5500,4

Marcos Santos;35;8000

Utilizando ; como separador para não confundir com a vírgula dos salários

Lendo um CSV

Utilizando o que aprendemos sobre arquivos, é fácil ler/escrever um CSV “no braço”

Exemplo

```
Nome;Idade;Salário
Maria Silva;45;8500,85
José Oliveira;30;5500,4
Marcos Santos;35;8000
```

```
arq = open("exemplo.csv", "r")
arq.readline()#descartando primeira linha
```

```
idades = 0
salarios = 0
qtde = 0
for linha in arq:
    quebra = linha.split(';')
    idades = idades + int(quebra[1])
    salarios = salarios + float(quebra[2].replace(',', '.'))
    qtde = qtde + 1
arq.close()

print("A idade media é", idades/qtde)
print("O salário médio é", salarios/qtde)
```

6. Biblioteca padrão para CSVs



Biblioteca

O Python conta com uma biblioteca padrão para leitura de CSVs

Exemplo

```
import csv
```

```
arq = open("exemplo.csv", "r")  
csv_reader = csv.reader(arq, delimiter = ';')
```

```
next(csv_reader)#pular a primeira linha
```

```
idades = 0
```

```
salarios = 0
```

```
qtde = 0
```

```
for linha in csv_reader:  
    idades = idades + int(linha[1])  
    salarios = salarios + float(linha[2].replace(',', '.'))  
    qtde = qtde + 1  
arq.close()
```

```
print("A idade media é", idades/qtde)
```

```
print("O salário médio é", salarios/qtde)
```

Exemplo

Importar a biblioteca de manipulação normalmente

```
import csv
```

```
arq = open("exemplo.csv", "r")  
csv_reader = csv.reader(arq, delimiter = ';')
```

```
next(csv_reader) #pular a primeira linha
```

```
idades = 0  
salarios = 0  
qtde = 0  
for linha in csv_reader:  
    idades = idades + int(linha[1])  
    salarios = salarios + float(linha[2].replace(',', '.', ''))  
    qtde = qtde + 1  
arq.close()
```

```
print("A idade media é", idades/qtde)  
print("O salário médio é", salarios/qtde)
```

Exemplo

Abra o arquivo, e depois passe esse arquivo para criar um leitor de csv. É possível especificar o delimitador.

```
import csv
```

```
arq = open("exemplo.csv", "r")  
csv_reader = csv.reader(arq, delimiter = ';')
```

```
next(csv_reader) #pular a primeira linha
```

```
idades = 0
```

```
salarios = 0
```

```
qtde = 0
```

```
for linha in csv_reader:  
    idades = idades + int(linha[1])  
    salarios = salarios + float(linha[2].replace(',', '.', ''))  
    qtde = qtde + 1  
arq.close()
```

```
print("A idade media é", idades/qtde)
```

```
print("O salário médio é", salarios/qtde)
```

Exemplo

A função next lê uma linha. Aqui está sendo usada para descartar a linha de cabeçalho.

```
import csv
```

```
arq = open("exemplo.csv", "r")  
csv_reader = csv.reader(arq, delimiter = ';')
```

```
next(csv_reader) #pular a primeira linha
```

```
idades = 0  
salarios = 0  
qtde = 0  
for linha in csv_reader:  
    idades = idades + int(linha[1])  
    salarios = salarios + float(linha[2].replace(',', '.', ''))  
    qtde = qtde + 1  
arq.close()  
  
print("A idade media é", idades/qtde)  
print("O salário médio é", salarios/qtde)
```

Exemplo

O loop passa por cada linha usando o csv_reader.
O csv reader se encarrega de quebrar a linha de acordo com o delimitador. Dessa forma linha é uma lista, e linha[0] contém a primeira coluna da linha, linha[1] a segunda coluna, ...

```
import csv
```

```
arq = open("exemplo.csv", "r")  
csv_reader = csv.reader(arq, delimiter = ';')
```

```
next(csv_reader) #pular a primeira linha
```

```
idades = 0  
salarios = 0  
qtde = 0
```

```
for linha in csv_reader:  
    idades = idades + int(linha[1])  
    salarios = salarios + float(linha[2].replace(',', '.'))  
    qtde = qtde + 1  
arq.close()
```

```
print("A idade media é", idades/qtde)  
print("O salário médio é", salarios/qtde)
```

Um exemplo do CNJ

No arquivo CSV disponibilizado pelo Datajud contém diversas colunas referentes a processos de diferentes localidades

A coluna *assuntos_count* contém a quantidade de assuntos para os processos

Vamos criar um programa que mostra a quantidade de assuntos total em determinado arquivo

Um exemplo do CNJ

assuntos_count é a 13a coluna (iniciando a contagem do zero)

Sendo assim, o programa pode ser uma versão similar ao anterior.

Um exemplo do CNJ

```
import csv
```

```
arq = open("ArquivoCNJ.csv", "r")  
csv_reader = csv.reader(arq, delimiter = ';')  
next(csv_reader)#pular a primeira linha
```

```
qtde_assuntos = 0  
for linha in csv_reader:  
    qtde_assuntos = qtde_assuntos + int(linha[13])  
arq.close()  
print("O total de assuntos no arquivo é", qtde_assuntos)
```


Mais

Existem diversas outras configurações e facilidades dadas pela biblioteca csv do Python

Veja em docs.python.org/pt-br/3/library/csv.html

6.

Teste seus conhecimentos



Executando o script

1. Replique tudo que foi ensinado durante as aulas no seu computador para fixar os conhecimentos.
2. Escreva novamente o programa de lista de compras de aulas passadas. Mas agora o seu programa deve salvar a lista de compras em um arquivo quando o usuário indicar que deseja fazer isso. Também deve ser possível carregar uma lista de compras de um arquivo, editá-la, e salvá-la novamente em arquivo. Atualize também o programa para que seja armazenado o item, e a quantidade do item a ser comprada (e.g., 5 maçãs).



Obrigado!

Bons Estudos!!!