



**Instituto de
Computação**

UNIVERSIDADE ESTADUAL DE CAMPINAS



MC102 – Aula 06

Strings

Algoritmos e Programação de Computadores

Zanoni Dias

2023

Instituto de Computação

Strings

Formatação de Strings

Operações, Funções e Métodos

Exercícios

Strings

Strings

- Uma string é uma sequência de caracteres.
- Em Python, strings são representadas como listas imutáveis de caracteres.
- Podemos representar uma string como uma sequência de caracteres entre aspas simples (') ou aspas duplas (").
- Exemplo:

```
1 msg = "hello world"
2 print(msg)
3 # hello world
```

Caracteres Especiais

- O seguinte trecho de código apresenta erros.

```
1 print("Você respondeu "SIM".")
2 # SyntaxError: invalid syntax
3 print("\")
4 # SyntaxError: EOL while scanning string literal
```

- Isso acontece porque " e \ são caracteres reservados da linguagem.
- Para representar os caracteres " e \, precisamos utilizar o seguinte código.

```
1 print("Você respondeu \"SIM\".")
2 # Você respondeu "SIM".
3 print("\\")
4 # \
```

- Outros exemplos:

```
1 print("Joe\'s Car")
2 # Joe's Car
3 print("Quebra de\nLinha")
4 # Quebra de
5 # Linha
6 print("\tTabulação")
7 #         Tabulação
```

- Por padrão, um <tab> (`\t`) tem tamanho equivalente a oito espaços em branco.

Acessando Elementos de uma String

- Podemos acessar elementos de uma string como acessamos elementos de uma lista.

```
1 msg = "hello world"
2 print(msg[0])
3 # h
4 print(msg[1])
5 # e
6 print(msg[-1])
7 # d
8 print(msg[-5])
9 # w
10 print(msg[12])
11 # IndexError: string index out of range
```

Acessando Elementos de uma String

- Como strings são listas imutáveis (assim como as tuplas), não é possível alterar uma posição da string.

```
1 msg = "hello world"
2 msg[0] = "y"
3 # TypeError: 'str' object does not support item assignment
```


Acessando Elementos de uma String

- Também podemos selecionar um trecho de uma string utilizando `string[start:stop:step]`.
- O trecho inicia na posição `start` (inclusive) e vai até a posição `stop` (exclusive), selecionando de `step` em `step` caracteres.
- Caso o parâmetro `step` não seja especificado, Python utilizará o valor 1 como padrão (assim como em listas ou tuplas).

```
1 msg = "hello world"
2 print(msg[3:8])
3 # lo wo
4 print(msg[:5])
5 # hello
6 print(msg[6:])
7 # world
8 print(msg[::-2])
9 # hlowrd
10 print(msg[::-1])
11 # dlrow olleh
```

Formatação de Strings

Formatação de Strings

- Uma das formas de formatar uma string é utilizando a função `format`.
- A função `format` recebe como parâmetros um valor e uma string com a formatação desejada.
- Como resposta a função retorna uma string formatada.
- A string de formatação possui uma especificação para cada tipo de dados.
- Especificação completa em:
<https://docs.python.org/3/library/string.html#formatspec>
- Iremos focar na formatação de valores do tipo inteiro (`int`) e real (`float`) em strings.

Formatação de Strings

- Formatando um número inteiro:

```
1 print(format(10, "d"))  
2 # 10
```

- Formatando um número inteiro com sinais:

```
1 print(format(13, "+d"))  
2 # +13  
3 print(format(-7, "+d"))  
4 # -7
```

Formatação de Strings

- Formatando um número real:

```
1 print(format(3.14159265359, "f"))  
2 # 3.141593
```

- Formatando um número real com sinais:

```
1 print(format(3.14159265359, "+f"))  
2 # +3.141593
```

- Formatando um número real com sinais e precisão:

```
1 print(format(3.14159265359, "+.10f"))  
2 # +3.1415926536
```

- Outra forma de formatar uma string é utilizando o método `format`.
- O método `format` gera uma nova string como resposta.
- O método recebe como parâmetros uma sequência de valores que são utilizados para criar a string no formato desejado.
- O método `format` também segue uma especificação para formação de strings.
- Especificação completa em:
<https://docs.python.org/3/library/string.html#formatstrings>

Formatação de Strings

- Formatação com strings:

```
1 frutas = "Frutas: {0}, {1} e {2}"
2 print(frutas.format("abacaxi", "banana", "caqui"))
3 # Frutas: abacaxi, banana e caqui
4 pets = "Quem é mais inteligente: {1} ou {0}?"
5 print(pets.format("gato", "cachorro"))
6 # Quem é mais inteligente: cachorro ou gato?
```

- Formatação com números inteiros:

```
1 soma = "{0} + {1} = {2}"
2 print(soma.format(3, 4, 3 + 4))
3 # 3 + 4 = 7
4 valores = "Valor mínimo/médio/máximo: {0}/{1}/{2}"
5 print(valores.format(10, 35, 100))
6 # Valor mínimo/médio/máximo: 10/35/100
```


- Formatação com números reais:

```
1 pi = "O valor de pi é: {0:.4f}"
2 print(pi.format(3.14159265359))
3 # O valor de pi é: 3.1416
4 notas = "A média das notas da turma foi {0:.2f}."
5 print(notas.format(8.7525))
6 # A média das notas da turma foi 8.75.
```

- Formatação com vários tipos de dados:

```
1  cabeçalho = "{0}, {1} de {2} de {3}"
2  print(cabeçalho.format("Campinas", 7, "maio", 2020))
3  # Campinas, 7 de maio de 2020
4  temperatura = "{0:02d}/{1:02d}/{2}: {3:.1f}C"
5  print(temperatura.format(7, 5, 2020, 28.765))
6  # 07/05/2020: 28.8C
```

Operações, Funções e Métodos

Concatenação de Strings

- O operador + concatena duas strings.

```
1 msg = "hello"  
2 msg2 = "y" + msg[1:] + "w"  
3 print(msg2)  
4 # yellow
```

- O operador * replica uma string.

```
1 s = "abc"  
2 print(s * 3)  
3 # abcabcabc
```

Tamanho de uma String

- A função `len` retorna o tamanho (quantidade de caracteres) de uma string.

```
1 msg = "hello"
2 print(len(msg))
3 # 5
4 msg2 = "Hello World"
5 print(len(msg2))
6 # 11
7 msg3 = "Hello\nWorld"
8 print(len(msg3))
9 # 11
```

- Observe que qualquer tipo de caractere é contado pela função `len`, inclusive espaços, quebra de linhas ou tabulações.

Comparação de Strings

- O operador `==` verifica se duas strings são iguais.
- O operador `!=` verifica se duas strings são diferentes.

```
1 a = "Python"
2 b = "Py" + "thon"
3 c = "p" + "ython"
4 print(a == b)
5 # True
6 print(a == c)
7 # False
8 print(b != c)
9 # True
```

- O operador `in` verifica se uma string é parte de outra string.

```
1 print("thon" in "Python")
2 # True
3 print("thor" in "Python")
4 # False
```

- O método `startswith` verifica se a string recebida como parâmetro é um prefixo da string base.

```
1 msg = "Hello World"
2 print(msg.startswith("Hello"))
3 # True
4 print(msg.startswith("World"))
5 # False
```

Buscando uma String

- O método `index` retorna a primeira posição em que uma string fornecida como parâmetro ocorre na string base.
- Se a string fornecida como parâmetro não está contida na string base, então é gerado um erro (similar ao que ocorre com listas).

```
1 bond = "My name is Bond, James Bond"
2 print(bond.index("Bond"))
3 # 11
4 msg = "Hello World"
5 print(msg.index("World"))
6 # 6
7 print(msg.index("Bond"))
8 # ValueError: substring not found
```


Buscando uma String

- O método `find` também retorna a primeira posição em que uma string fornecida como parâmetro ocorre na string base.
- Se a string fornecida como parâmetro não está contida na string base, então é retornado o valor `-1`.

```
1 bond = "My name is Bond, James Bond"
2 print(bond.find("Bond"))
3 # 11
4 msg = "Hello World"
5 print(msg.find("World"))
6 # 6
7 print(msg.find("Bond"))
8 # -1
```

- O método `strip` remove todos os espaços em branco (incluindo quebras de linhas e tabulações) no início e no fim da string.

```
1 msg = " \n Hello World \t"  
2 print(msg.strip())  
3 # Hello World
```

Manipulação de Strings

- O método `split` divide uma string em uma lista de acordo com um padrão de caracteres (separador).
- Por padrão, o separador é igual a qualquer sequência de espaços em branco (incluindo quebras de linhas e tabulações).

```
1 str1 = "  MC102  Algoritmos\t\tProgramação\nComputadores  "  
2 dados = str1.split()  
3 print(dados)  
4 # ['MC102', 'Algoritmos', 'Programação', 'Computadores']  
5 str2 = "abacaxi, banana, caqui, damasco"  
6 frutas = str2.split(", ")  
7 print(frutas)  
8 # ['abacaxi', 'banana', 'caqui', 'damasco']
```

Lendo Múltiplos Valores

- Por exemplo, o método `split` pode ser usado para separar múltiplos valores lidos numa única linha.

```
1 # Lendo duas strings separadas por um espaço
2 s1, s2 = input().split()
3 # Lendo três números inteiros separados por espaços
4 a, b, c = [int(i) for i in input().split()]
5 # Lendo múltiplos números separados por espaços
6 numeros = [float(i) for i in input().split()]
```

Manipulação de Strings

- O método `join` junta uma lista de strings usando a string base como concatenador.

```
1 frutas = ['abacaxi', 'banana', 'caqui', 'damasco']
2 txt = ", ".join(frutas)
3 print(txt)
4 # abacaxi, banana, caqui, damasco
```

- A função `list()` pode ser utilizada para transformar uma string em uma lista de caracteres.

```
1 string = "aeiou"
2 lista = list(string)
3 print(lista)
4 # ['a', 'e', 'i', 'o', 'u']
```

Manipulação de Strings

- O método `replace` cria uma nova string onde todas as ocorrências de um padrão de caracteres numa string dada são trocadas por outro.

```
1 x = "Algoritmos e Programação de Computadores"
2 y = x.replace("a", "_")
3 print(y)
4 # Algoritmos e Progr_m_ção de Comput_dores
5 y = x.replace("Algoritmos", "$" * len("Algoritmos"))
6 print(y)
7 # $$$$$$$$$$ e Programação de Computadores
8 x = "a,b,c,d,e"
9 y = x.replace(",", "")
10 print(y)
11 # abcde
```

- `capitalize()`: converte o primeiro caractere para maiúsculo.

```
1 print("meu teste".capitalize())  
2 # Meu teste
```

- `lower()`: converte a string para letras minúsculas.

```
1 print("Meu TESTE".lower())  
2 # meu teste
```

- `upper()`: converte a string para letras maiúsculas.

```
1 print("mEU tESte".upper())  
2 # MEU TESTE
```

Outros Métodos

- `isnumeric()`: testa se todos os caracteres são dígitos.

```
1 print("1234".isnumeric())
2 # True
3 print("teste123".isnumeric())
4 # False
```

- `isalpha()`: testa se todos os caracteres são letras.

```
1 print("MeuTeste".isalpha())
2 # True
3 print("teste123".isalpha())
4 # False
```

- `isalnum()`: testa se todos os caracteres são letras ou dígitos.

```
1 print("teste123".isalnum())
2 # True
3 print("Meu teste".isalnum())
4 # False
```


- Exemplo:

```
1 sc = input("Entre com uma sequência de caracteres: ")
2
3 if sc.isalpha():
4     print(sc, "possui apenas letras")
5 elif sc.isnumeric():
6     print(sc, "possui apenas dígitos")
7 elif sc.isalnum():
8     print(sc, "possui letras e dígitos")
9 else:
10    print(sc, "não possui apenas letras e dígitos")
```

Laços e Strings

- Podemos utilizar o comando for para percorrer uma string.
- Exemplo:

```
1 s = "abc"
2 for c in s:
3     print(c)
4 # a
5 # b
6 # c
```

- Outro exemplo:

```
1 for c in "Algoritmos":
2     if c in "AEIOUaeiou":
3         print("A string possui a vogal:", c)
4 # A string possui a vogal: A
5 # A string possui a vogal: o
6 # A string possui a vogal: i
7 # A string possui a vogal: o
```

Exercícios

1. Escreva um programa que, dada uma sequência de números inteiros (todos fornecidos na mesma linha, separados por espaços), imprima a média desses números.
2. Escreva um programa que, dada uma string representando um texto, imprima o número de palavras existentes. Observação: você deve remover os sinais de pontuação (".", ",", ":", ";", "!" e "?") antes de realizar a contagem das palavras.

3. Escreva um programa que, dada uma string `texto` e uma string `palavra`, ache todas as posições de ocorrência da palavra no texto. O seu programa deve desconsiderar se as letras são maiúsculas ou minúsculas.
4. Um palíndromo é uma palavra ou frase que pode ser lida da mesma forma tanto da esquerda para a direita como da direita para a esquerda (desconsiderando os espaços em branco). Considere que a entrada não possui sinais de pontuação ou acentos. Escreva um programa que, dada uma string, verifique se ela é um palíndromo.

Exercício 1

1. Escreva um programa que, dada uma sequência de números inteiros (todos fornecidos na mesma linha, separados por espaços), imprima a média desses números.

```
1 texto = input("Entre com uma sequência de números: ")
2 numeros = texto.split()
3 soma = 0
4
5 for n in numeros:
6     soma = soma + int(n)
7 media = soma / len(numeros)
8
9 print("A média é: ", format(media, ".2f"))
```

Exercício 2

2. Escreva um programa que, dada uma string representando um texto, imprima o número de palavras existentes. Observação: você deve remover os sinais de pontuação (".", ",", ":", ";", "!" e "?") antes de realizar a contagem das palavras.

```
1 texto = input("Entre com um texto: ")
2 pontuacao = [".", ",", ":", ";", "!", "?"]
3
4 # remove os sinais de pontuação
5 for p in pontuacao:
6     texto = texto.replace(p, " ")
7
8 numPalavras = len(texto.split())
9 print("Número de palavras:", numPalavras)
```

Exercício 3

3. Escreva um programa que, dada uma string `texto` e uma string `palavra`, ache todas as posições de ocorrência da palavra no texto. O seu programa deve desconsiderar se as letras são maiúsculas ou minúsculas.

```
1 texto = input("Entre com um texto: ")
2 palavra = input("Entre com uma palavra: ")
3 texto = texto.lower()
4 palavra = palavra.lower()
5
6 removido = 0
7 while palavra in texto:
8     posicao = texto.find(palavra)
9     print(removido + posicao)
10    texto = texto[posicao + 1:]
11    removido = removido + (posicao + 1)
```


Exercício 4

4. Um palíndromo é uma palavra ou frase que pode ser lida da mesma forma tanto da esquerda para a direita como da direita para a esquerda (desconsiderando os espaços em branco). Considere que a entrada não possui sinais de pontuação ou acentos. Escreva um programa que, dada uma string, verifique se ela é um palíndromo.

```
1 txt = input("Entre com uma palavra ou frase: ")
2 txt = txt.lower()
3 txt = txt.replace(" ", "")
4
5 i = 0
6 j = len(txt) - 1
7
8 palindromo = True
9
10 ...
```

Exercício 4 - Continuação

```
1 ...
2
3 while i < j:
4     if txt[i] != txt[j]:
5         palindromo = False
6         break
7     else:
8         i = i + 1
9         j = j - 1
10
11 if palindromo:
12     print("Palíndromo")
13 else:
14     print("Não palíndromo")
```

Exercício 4 - Continuação

```
1 ...
2
3 while i < j and palindromo:
4     if txt[i] != txt[j]:
5         palindromo = False
6
7     else:
8         i = i + 1
9         j = j - 1
10
11 if palindromo:
12     print("Palíndromo")
13 else:
14     print("Não palíndromo")
```

Exercício 4 - Continuação

```
1 ...
2
3 while i < j:
4     if txt[i] != txt[j]:
5         palindromo = False
6         break
7
8     i = i + 1
9     j = j - 1
10
11 if palindromo:
12     print("Palíndromo")
13 else:
14     print("Não palíndromo")
```

Exercício 4

4. Um palíndromo é uma palavra ou frase que pode ser lida da mesma forma tanto da esquerda para a direita como da direita para a esquerda (desconsiderando os espaços em branco). Considere que a entrada não possui sinais de pontuação ou acentos. Escreva um programa que, dada uma string, verifique se ela é um palíndromo.

```
1 txt = input("Entre com uma palavra ou frase: ")
2 txt = txt.lower()
3 txt = txt.replace(" ", "")
4
5 if txt == txt[::-1]:
6     print("Palindromo")
7 else:
8     print("Nao ´ palindromo")
```